COLÉGIO PEDRO II

Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura

Mestrado Profissional em Matemática em Rede Nacional

VANESSA HENRIQUES BORGES

COMBINATÓRIA E PENSAMENTO COMPUTACIONAL:

CONEXÕES PARA A EDUCAÇÃO BÁSICA

NO SÉCULO XXI

Rio de Janeiro
2021

Vanessa Henriques Borges

COMBINATÓRIA E PENSAMENTO COMPUTACIONAL:

CONEXÕES PARA A EDUCAÇÃO BÁSICA NO SÉCULO XXI

Dissertação de Mestrado apresentada ao Programa de Mestrado Profissional em Matemática em Rede Nacional, vinculado à Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura do Colégio Pedro II, como requisito parcial para obtenção do título de Mestre em Matemática.

Orientador: Prof. Dr. Ivail Muniz Junior

Rio de Janeiro
2021

COLÉGIO PEDRO II

PRÓ-REITORIA DE PÓS-GRADUAÇÃO, PESQUISA, EXTENSÃO E CULTURA

BIBLIOTECA PROFESSORA SILVIA BECHER

CATALOGAÇÃO NA FONTE

B732 Borges, Vanessa Henriques

Combinatória e pensamento computacional: conexões para a educação básica no século XXI / Vanessa Henriques Borges. - Rio de Janeiro, 2021.

266 f.

Dissertação (Mestrado Profissional em Matemática em Rede Nacional) – Colégio Pedro II, Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura.

Orientador: Ivail Muniz Junior.

1. Matemática – Estudo e ensino. 2. Análise combinatória. 3. Algoritmos computacionais. 4. Resolução de problemas (Matemática). I. Muniz Junior, Ivail. II. Colégio Pedro II. III Título.

CDD 510

Ficha catalográfica elaborada pela Bibliotecária Simone Alves – CRB7 5692.

Vanessa Henriques Borges

COMBINATÓRIA E PENSAMENTO COMPUTACIONAL:

CONEXÕES PARA A EDUCAÇÃO BÁSICA NO SÉCULO XXI

Dissertação de Mestrado apresentada ao Programa de Mestrado Profissional em Matemática em Rede Nacional, vinculado à Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura do Colégio Pedro II, como requisito parcial para obtenção do título de Mestre em Matemática.

Aprovado em: _____/_____/_____.

Banca Examinadora:

_____
Prof. Dr. Ivail Muniz Junior (Orientador)
Colégio Pedro II - PROFMAT CPII

_____
Prof. Dr. Sergio Carrazedo Dantas (Externo)
Universidade Estadual do Paraná

_____
Prof. Dra. Andreia Carvalho Maciel Barbosa (Interno)
Colégio Pedro II - PROFMAT CPII

Rio de Janeiro
2021

Dedico esse trabalho a minha mãe, familiares, amigos, professores que sempre me ajudaram e incentivaram, ao meu orientador e amigo Prof. Ivail Muniz por todo aprendizado, paciência e orientação por todo esse tempo. Gratidão eterna.

AGRADECIMENTOS

Agradeço à minha mãe por todo apoio, suporte e parceria, sendo meu porto seguro de ontem, de hoje e de sempre. Agradeço ainda aos meus amigos e amigas que sempre estiveram comigo, me ajudando a caminhar e a não desistir. Gratidão imensa ao meu amado Colégio Pedro II pela oportunidade de poder voltar a estudar nessa Instituição de excelência e em outra etapa da vida adulta. Ao corpo docente do PROFMAT que tive a honra de conhecer e ser aluna, aos meus queridos amigos da turma de mestrado que criei e a meu querido orientador Prof. Dr. Ivail Muniz a quem tive a honra e alegria de ser orientada: obrigada por toda paciência, motivação e determinação que tem ao lecionar e orientar aos discentes; gratidão e admiração eternas. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

RESUMO

BORGES, Vanessa Henriques. **Combinatória e pensamento computacional: conexões para a Educação Básica no século XXI.** 2021. 265 f. Dissertação (Mestrado) – Colégio Pedro II, Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura, Programa de Mestrado Profissional em Matemática em Rede Nacional, Rio de Janeiro, 2021.

O ensino de combinatória tem sido objeto de muitos estudos na área de Educação Matemática nas últimas décadas, em diversos países, incluindo o Brasil, muito embora ainda sejam escassas as conexões entre problemas de combinatória, o uso de tecnologias digitais e os diferentes entendimentos sobre pensamento computacional na Educação Básica e suas conexões com as habilidades e competências, preconizadas pela Base Nacional Comum Curricular (BNCC) para o século XXI. Nosso estudo apresenta uma investigação de diferentes tipos de problemas de combinatória (contagem, existência, enumeração, classificação e otimização), em nível do Ensino Médio, ora abordados pelas usuais técnicas de contagem ensinadas na Educação Básica, ora abordados por meio de processos que caracterizam um tipo de pensamento computacional, mostrando conexões, potencialidades e limitações das duas abordagens, inclusive para o ensino de combinatória na Educação Básica. A metodologia utilizada foi a Pesquisa em Desenvolvimento. Para a investigação, foram selecionados 24 problemas de combinatória, cujas soluções foram analisadas em quatro etapas de investigação: (i) solução matemática; (ii) solução via pensamento computacional; (iii) comparação de soluções e diferentes representações (fluxogramas e algoritmos) e (iv) considerações e conexões para a sala de aula de matemática, articulada à BNCC. Apresentamos também uma implementação em linguagem C++ para cada um dos problemas investigados. Os resultados mostram variadas conexões entre combinatória e pensamento computacional, gerando formas alternativas de selecionar, categorizar, abordar, resolver e pensar problemas de combinatória, tanto pela sua natureza como pelas diferentes estratégias de solução, tais como: novas possibilidades de problemas de contagem, tais como os de existência e otimização, a partir do uso de processos recursivos; o uso de algoritmos, fluxogramas e programas criando oportunidades de novas percepções e produção de significados sobre as etapas de resolução de problemas de combinatória; a ampliação das estratégias enumerativas devido à natureza da estrutura algorítmica e do aumento e velocidade do poder computacional; possibilidades de novas leituras sobre a dinâmica dos limites e das intersecções dos tipos de problemas de combinatória; a ampliação das estratégias enumerativas devido à natureza da estrutura algorítmica e do aumento e velocidade do poder computacional; o dinâmica do papel das técnicas de contagem em processos computacionais, dentre outras.

Palavras-chave: Combinatória; Pensamento computacional; BNCC; Algoritmos e fluxogramas; Resolução de problemas.

ABSTRACT

BORGES, Vanessa Henriques. **Combinatorial and computational thinking: connections for Basic Education in the 21st century.** 2021. 265 f. Thesis (Master's Degree) – Pedro II College, Pro-Rectory of Graduate Studies, Research, Extension and Culture, Professional Master's Program in Mathematics in National Network Mathematics, Rio de Janeiro, 2021.

The combinatorics teaching has been the subject of many studies in the area of Mathematics Education in the last decades, in several countries, including Brazil, although the connections between combinatorics problems, the use of digital technologies and the different understandings about thought are still scarce. computing in Basic Education and its connections with skills and competences, recommended by the National Common Curricular Base (BNCC) for the 21st century. Our study presents an investigation of different types of combinatorial problems (counting, existence, enumeration, classification and optimization), at the level of High School, sometimes addressed by the usual counting techniques taught in Basic Education, sometimes addressed through processes that characterize a type of computational thinking, showing connections, potentialities and limitations of the two approaches, including for the teaching of combinatorics in Basic Education. The methodology used was Research in Development. For the investigation, 24 combinatorial problems were selected, whose solutions were analyzed in four stages of investigation: (i) mathematical solution; (ii) solution via computational thinking; (iii) comparison of solutions and different representations (flowcharts and algorithms) and (iv) considerations and connections for the mathematics classroom, articulated to the BNCC. We also present an implementation in C ++ language for each of the problems investigated. The results show varied connections between combinatorics and computational thinking, generating alternative ways to select, categorize, approach, solve and think combinatorial problems, both by their nature and by the different solution strategies, such as: new possibilities of counting problems, such as such as those of existence and optimization, based on the use of recursive processes; the use of algorithms, flowcharts and programs creating opportunities for new perceptions and production of meanings on the steps of solving combinatorial problems; the expansion of enumerative strategies due to the nature of the algorithmic structure and the increase and speed of computational power; possibilities of new readings on the dynamics of the limits and intersections of the types of combinatorial problems; the expansion of enumerative strategies due to the nature of the algorithmic structure and the increase and speed of computational power; the dynamics of the role of counting techniques in computational processes, among others.

Keywords: Combinatorial; Computational thinking; BNCC; Problem Solving.

LISTA DE TABELAS

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A abordagem de tecnologias digitais em ambientes escolares, de forma intencionalmente pedagógica para o ensino de matemática, é algo que já vem sendo estudado e em alguns casos implementado desde a década de 70 do século passado, e ampliado nas duas décadas do século XXI, conforme se pode ver em Papert (1972), Wing (2006; 2011), NRC (2011), CSTA (2015), Bairral (2013), Dantas (2019), dentre outros.

Com a Base Nacional Comum Curricular (BNCC), a discussão do uso das tecnologias se amplia. Além do documento reforçar o uso dos recursos tecnológicos computacionais no desenvolvimento de habilidades matemáticas, bem como na forma de abordar e ensinar objetos de conhecimento, noções matemáticas, as operações, suas lógicas e representações, a BNCC apresenta (ainda que de forma insatisfatória, imprecisa e omissa) uma discussão que vem sendo realizada em nível internacional sobre o papel da computação no desenvolvimento de habilidades para o cidadão do século XXI, onde as tecnologias são necessárias tanto para usuários quanto para desenvolvedores, desde processos básicos de comunicação entre pessoas e empresas, passando pelo consumo e entretenimento, chegando até processos de natureza profissional.

Nesse contexto, algo que diferencia a BNCC dos outros documentos é a ampliação do uso de tecnologias digitais de forma transversal aos componentes curriculares, incluindo uma menção explícita sobre a necessidade de se desenvolver um tipo de pensamento denominado pensamento computacional[1], com foco na resolução de problemas, fortemente ligado aos processos e etapas de resolução, associados a algoritmos e computação.

Apesar da BNCC não definir satisfatoriamente o pensamento computacional, e nem trazer conexões e orientações práticas de como articular tal pensamento às habilidades de matemática, o documento faz menção à tecnologia digital em quatro das dez competências gerais (1, 2, 4 e 5). Em especial, a quinta competência desejada para o estudante é a de

> Compreender, utilizar e criar tecnologias digitais de informação e comunicação de forma crítica, significativa, reflexiva e ética nas diversas práticas sociais (incluindo as escolares) para se comunicar, acessar e

[1] Há várias concepções do que seja pensamento computacional, cada uma delas influenciada por diferentes visões, experiências e intenções, conforme mostraremos no capítulo 1. Inicialmente, pontuamos que adotaremos aqui a ideia apresentada em Raabe (2017), onde pensamento computacional se refere à capacidade de resolver problemas considerando conhecimentos e práticas de computação, compreendendo sistematizar, representar, analisar e resolver problemas.

> disseminar informações, produzir conhecimentos, resolver problemas e exercer protagonismo e autoria na vida pessoal e coletiva. (BRASIL, 2018, p.9)

Contudo, diante dessas novas recomendações, várias questões emergem, tais como: o que é pensamento computacional? Quais as diferenças e similaridades entre pensamento combinatório e pensamento computacional? Como desenhar tarefas que favoreçam o desenvolvimento desse tipo de pensamento? Como usar tecnologias digitais e ensino de matemática para desenvolver habilidades que desenvolvam o pensamento computacional dos estudantes, considerando as demandas do século XXI? Como vamos trazer isso para dentro do currículo, e em especial, para dentro do currículo de matemática? Quais as habilidades da BNCC podem ser mais facilmente articuladas para o desenvolvimento do pensamento computacional? Quais as diferenças de resolver um problema usando pensamento combinatório e usando pensamento computacional? Como articular o pensamento computacional aos objetos de conhecimento dos programas de matemática presentes na BNCC, em especial a do Ensino Médio? Essas são apenas algumas perguntas que precisam ser respondidas diante das demandas da BNCC.

Nesse trabalho, apresentaremos algumas reflexões para tentar contribuir com reflexões e construção de soluções, certamente parciais, para uma dessas perguntas, qual seja: **Como articular o pensamento computacional às habilidades e objetos de conhecimento dos programas de matemática?**

Vemos o pensamento matemático como um conjunto de tarefas a serem exploradas. Dentre elas, destacamos: separar um problema em casos menores, identificar padrões similares que possam ajudar, identificar quais conteúdos e habilidades necessários para resolução do problema, identificar pressupostos, selecionar estratégias apropriadas, buscar por padrões ou possíveis conexões, generalizar exemplos, conjecturar e demonstrar e usar raciocínio lógico dedutivo para demonstrar afirmações ou exibir contraexemplo sobre as afirmações levantadas. Entendemos também que resolução matemática é o uso das ferramentas matemáticas e do raciocínio lógico para resolver problemas de matemática e, em particular, de combinatória.

Olhando para os objetos de conhecimento e habilidades da BNCC do Ensino Médio, identificamos várias oportunidades de se fazer tal articulação em cada um dos objetos já presentes, bem como com várias das habilidades apresentadas na BNCC. É possível, por exemplo, abordar problemas envolvendo funções das mais variadas por meio do pensamento computacional, usando programas como o Geogebra ou planilhas eletrônicas. Ao mesmo tempo

é possível usar simuladores, como o do Banco Central e da Receita Federal, para entender em que aspectos as mudanças previdenciárias, em especial na forma de cálculo do INSS e do tempo de aposentadoria, modeladas por tais funções, prejudicam ou beneficiam (será?) Diferentes extratos da população, incluindo os mais vulneráveis, aumentando a chance de problemas de proteção social num futuro próximo.

Temos também outras oportunidades, como por exemplo na compreensão de situações de natureza aleatória, na medida em que vários Problemas não determinísticos envolvendo probabilidade e estatística são naturalmente de natureza computacional e podem ser excelentes oportunidades para o desenvolvimento desse tipo de pensamento.

Diante do amplo espectro de articulações entre objetos matemáticos e o pensamento computacional, identificamos que a **Combinatória**[2], numa perspectiva mais ampla que a usual (contemplando problemas combinatoriais resolvidos de enumeração, classificação, (por técnicas de contagem, existência e otimização), oferece diversos problemas interessantes que podem ser resolvidos por estratégias e etapas envolvendo análise, abstração e automação, que caracterizam em certa perspectiva o pensamento computacional.

Diante do exposto levantamos a seguinte questão: Como conectar o pensamento computacional aos objetos de conhecimento dos programas de matemática presentes na BNCC, em especial, a combinatória do Ensino Médio?

Por isso o objetivo geral deste trabalho é **apresentar uma investigação de diferentes tipos de problemas de combinatória (contagem, existência, enumeração, classificação e otimização), em nível do Ensino Médio, ora abordados pelas usuais estratégias de enumeração ou técnicas de contagem ensinadas na Educação Básica (pensamento combinatório[3]), ora abordados por meio de processos que caracterizam um tipo de pensamento computacional, mostrando conexões, potencialidades e limitações das duas abordagens, inclusive para o ensino de combinatória na Educação Básica.**

[2] Usaremos cinco tipos de problemas de combinatória, ampliando a concepção dessa área, para além dos usuais problemas de contagem, conforme caracterizados por Batanero, Godino e Navarro-Pelayo (1996). Discutiremos mais detalhadamente essa classificação no capítulo 1, especificamente na seção 1.2., bem como as justificativas da nossa escolha.

[3] O termo pensamento combinatório foi usado neste trabalho no sentido de captar modos de resolver problemas de combinatória, baseados em enumerações, classificações (dividir o problema em casos) e técnicas de contagem, que usualmente são ensinados e apresentados em sala de aula, numa tentativa de trazer diferenciações do pensamento computacional.

Para atingir a esse objetivo geral, buscamos especificamente:

1. Identificar e selecionar problemas de combinatória de contagem, existência e otimização que podem ser explorados via estratégias do pensamento computacional e pelo pensamento combinatório.
2. Resolver os problemas selecionados usando as duas estratégias, analisando-as e comparando-as nas perspectivas combinatorial, computacional e educacional.
3. Apresentar considerações didáticas, incluindo conexões entre pensamento computacional, combinatória e habilidades da BNCC.

O texto está dividido em quatro capítulos. No capítulo 1 apresentamos diferentes concepções de pensamento computacional por meio de uma revisão da literatura sobre o tema, em especial, às questões relacionadas ao ensino.

O capítulo 2 trata de combinatória numa perspectiva mais ampla que a usual adotada na Educação Básica, no aspecto da resolução de problemas de combinatória para o Ensino Médio. Apresentamos e discutimos os cinco diferentes tipos de problema de combinatória que abordamos no trabalho, conforme Batanero, Godino e Navarro-Pelayo (1996), seguido de uma revisão da literatura sobre o ensino de combinatória nessa perspectiva mais ampla. Seguimos com uma síntese sobre noções matemáticas sobre técnicas de contagem, que subsidiaram a resolução e investigação dos problemas de enumeração, classificação e contagem. Finalizamos com noções básicas de teoria dos grafos, que dão suporte para a resolução e análise da maioria dos problemas de existência e otimização abordados nesse em nosso estudo, em boa parte adaptados de Muniz (2007) e Borges (2018).

No capítulo 3 apresentamos o processo metodológico adotado, caracterizado por ser uma pesquisa em desenvolvimento e detalhamos as características de tal metodologia, bem como as classificações realizadas em cada tipo de problema e seu respectivo referencial teórico. Nele mostramos como foi a identificação e escolha dos trinta e cinco problemas de combinatória, bem como as categorias analíticas referentes tanto aos tipos de problemas de combinatória como as quatro etapas de investigação nas quais apresentamos (i) a solução matemática; (ii) a Resolução via pensamento computacional, (iii) uma comparação da Resolução matemática com representações algorítmicas e finalizando com (iv) reflexões e conexões didáticas.

No capítulo 4 apresentamos a investigação propriamente dita de 35 problemas de combinatória[4] selecionados, conforme descrita na metodologia, considerando a Resolução matemática e a Resolução via pensamento computacional, usando categorias baseadas na perspectiva do CIEB. Seguiremos com uma análise e comparação de ambas as soluções e suas representações algorítmicas, incluindo fluxogramas e implementações em linguagem C ++. Finalizamos o capítulo com articulações e conexões para a sala de aula de matemática, no que chamamos de considerações didáticas.

Nas considerações finais concluímos as contribuições dessa dissertação, seguido das referências e do apêndice com o endereço eletrônico de cada uma das implementações em linguagem C++ das questões abordadas.

[4] Nesse texto, vamos considerar problema de combinatória e questão de combinatória como tendo o mesmo sentido.

2 PENSAMENTO COMPUTACIONAL: PESQUISAS E ASPECTOS TEÓRICOS

Neste capítulo apresentamos aspectos teóricos que fundamentam o trabalho de forma articulada com uma revisão da literatura sobre pensamento computacional. O capítulo tem duas seções: a primeira sobre pensamento computacional e a segunda sobre combinatória.

Dois pontos iniciais merecem ser destacados. **O primeiro** se refere ao resgate, nada original do "pensamento computacional" pela BNCC, que pode ser considerado, em certa medida, a um retorno ao ensino de computação articulado ao ensino de matemática iniciado na década de 70 pelo grupo do matemático Seymour Papert. Assim, o badalado e amedrontador pensamento computacional, que está na moda por conta da BNCC, não é algo novo e exclusivo do Século XXI, conforme mostraremos logo em seguida, uma vez que formas de resolver os problemas de combinatória, usando computação, são em parte já "antigas".

É importante situar, no âmbito deste trabalho, nossa visão crítica sobre a BNCC. Uma tentativa de explicar o que é pensamento computacional é de que "[...]envolve as capacidades de compreender, analisar, definir, modelar, resolver, comparar e automatizar problemas e suas soluções, de forma metódica e sistemática, por meio do desenvolvimento de algoritmos". (BRASIL, 2017, p.21).

Essa definição além de não ser clara, se confunde com a resolução de problemas sem apoio a computação, não destaca e nem explicita os processos que diferenciam o pensamento computacional de outras formas de pensar e/ou resolver problemas, conforme veremos na próxima seção. A BNCC nos oferece oportunidades, mas está longe de mostrar caminhos e estratégias, sequer iniciais para aproveitá-las. Essa talvez seja uma justificativa para essa dissertação, e um aumento da responsabilidade do papel que ela pode ter na formação de professores.

O segundo ponto é sobre a concepção de combinatória que estamos usando. Gostaríamos de reforçar (correndo o risco de ser repetitivos) que combinatória nesse texto abarca um conjunto mais amplo de problemas do que os usuais problemas de contagem, geralmente abordados na Educação Básica, em especial no Ensino Médio. Mas não é apenas uma ampliação de categorias. Apresentamos também conexões entre formas de resolver com e sem o apoio de computação, trazendo para a discussão etapas que caracterizam uma dada perspectiva de pensamento computacional. Além disso, alguns problemas, como o problema do amigo oculto, são resolvidos por meio de recorrências, que apesar de ser uma construção

matemática que antecede à criação do computador, por exemplo, tem conexões diretas com processos do pensamento computacional, e ganha uma importância que não teria sem a possibilidade do poder computacional de um computador. “Estão separados (em uma certa perspectiva) porém extremamente juntos (quando usamos para resolver problemas com n maior)”.

2.1 Pensamento Computacional: uma Revisão da Literatura

Ao longo das últimas cinco décadas, diferentes abordagens articulando o ensino de matemática com ideias de programação foram se constituindo em vários países, a partir de alguns pesquisadores. Veremos que nessas diferentes formar de ver, entender e praticar tal ensino, podemos perceber que um ponto em comum entre elas é que: aprender a programar um computador ajuda o estudante a pensar.

A **primeira abordagem**, chamada de construcionismo e letramento computacional, nasceu juntamente com a linguagem *Logo* criada pelos matemáticos Seymour Papert, Cyntia Solomon e Wally Feurzeig.

Papert (1972), ancorado nas ideias do construtivismo de Piaget, identificou nos grupos estudados em diversos experimentos que a programação possibilita que estudantes aprendam e resolvam problemas pela criação de modelos conceituais que são transformados em código. Segundo ele, para construir um algoritmo que soluciona um problema, é necessário conhecer o problema e criar um modelo de como ele funciona, levando a uma profunda compreensão de conceitos e relações envolvidos.

Papert defendia que a partir da construção de algoritmos para resolução de problemas o aluno teria autonomia e aprenderia através do fazer. O aluno a partir disso, teria motivação pessoal ao conseguir resolver um problema através do desenvolvimento do seu conhecimento em cima da questão a ser solucionada.

Ele também enfatizou que no processo de escrita e depuração do código, o desenvolvimento cognitivo dos alunos é visível de forma detalhada e precisa, oferecendo a pesquisadores uma oportunidade inédita de investigação. (RAABE; COUTO; BLIKSTEIN, 2020).

Através da Resolução de problemas com a construção de um algoritmo, o aluno consegue relacionar o seu pensamento concreto na busca de uma solução pratica com o pensamento abstrato ao tentar criar uma abstração ou uma Resolução computacional para um problema prático e real.

No Brasil, na década de 1980, foram realizados muitos projetos com a linguagem *Logo*, que envolviam escolas públicas e privadas, mas o impacto é considerado muito pequeno devido às condições tecnológicas e políticas de formação de professores. Em 1996, O livro *O professor no ambiente Logo* (VALENTE, 1996) foi uma das últimas obras relacionadas com o tema, segundo Raabe et al (2020).

Para Resnick e Brennan (2012), o *Scratch* por meio da sua linguagem em blocos, possibilita desenvolver conceitos de lógica de programação tais como: sequenciamento, repetições, eventos, paralelismo, condições, operadores e dados.

Tais autores apresentam características do processo de usar essa lógica de programação: (i) incrementare iterar; (ii) testar e depurar; (iii) reutilizar; (iv) usar abstração e modularização (usar coleções ou blocos com funções específicas para construir ou resolver algo maior). Essa estrutura pode caracterizar não apenas a lógica de programação, mas uma maneira de entender o que é pensamento computacional.

Na perspectiva destes autores, na etapa de incrementação e iteração acontece mudanças e os processos podem ser separados em menores casos. No teste e depuração precisamos testar se a solução é ótima. Na reutilização, podemos repetir a mesma função ao longo de um processo de programação, ou usar de problema já resolvido para chegar na solução de outro mais rebuscado. E a abstração e modularização é quando a partir de questões mais simples usamos para solução de questões maiores ou mais abstratas.

Resnick (2013) também defende que o ato de programar pode ajudar no processo de organização do pensamento, em que os alunos aprendem a organizar, refinar e refletir sobre as ideias. Ao serem capazes de codificar problemas, conseguimos escrever o mundo real através de processos recursivos ou pensamentos lógicos seguindo etapas estruturadas no pensamento computacional.

O letramento digital é o termo usado por Disessa (2001) para a técnica usada para leitura e escrita no processo de alfabetização. Uma pessoa letrada digitalmente é capaz de ler jornais, cartas, ler placas de trânsito, relatório, jornais digitais e impressos, dentre outros. Defende ainda uma mudança nas estruturas escolares, nas quais os alunos usariam o computador como parte integrante do processo de aprendizagem, e não apenas nas aulas computacionais. Esse processo

se tornou mais presente durante a pandemia, na qual a educação se viu em um momento emergencial em que precisava utilizá-lo como meio de aprendizagem o tempo todo e mudou a maneira como vemos e usamos tal ferramenta em sala de aula.

A maneira como aprendemos e o processo que usamos como ferramenta para esse ato mudam a visão que temos do mundo e como lidamos com ele. Assim, as soluções buscadas e apresentadas se modificam e são habilidades que são aplicáveis em diversas áreas de conhecimento. Ou seja, o letramento digital e o pensamento computacional podem ser aplicados em aprendizagem e solução de problemas em áreas como artística, exata, humanista, social, biológica, tecnológica, dentre outras. (DISESSA 2001).

Para Blikstein (2018), o letramento computacional é um conjunto de elementos materiais, cognitivos e sociais que geram novas formas de pensar e aprender, permitindo novos tipos de operações mentais e representações de conhecimento, criando novos tipos de "literaturas".

A **segunda abordagem**, segundo Raabe at al (2020) é a emergência do pensamento computacional, que acabou influenciando fortemente currículos em diversos países, incluindo a presença do termo na BNCC.

Em 2006, Jeanette Wing (2006) retoma o termo pensamento computacional no artigo Computational Thinking. Ela resgata a ideia do pensamento computacional, mas não com a questão do ideal construcionista antes defendido, mas como uma disciplina importante a ser aplicada em diferentes áreas de conhecimento e na vida cotidiana. Esse pensamento seria dividido em etapas tais como: reconhecimento de padrões, decomposição de problemas, abstração do problema e solução através de algoritmos.

Ela é a responsável pela difusão desse termo e sua definição, bem como de analisá-lo do ponto de vista de explorá-lo pedagogicamente. Wing (2011, p. 30) destaca que:

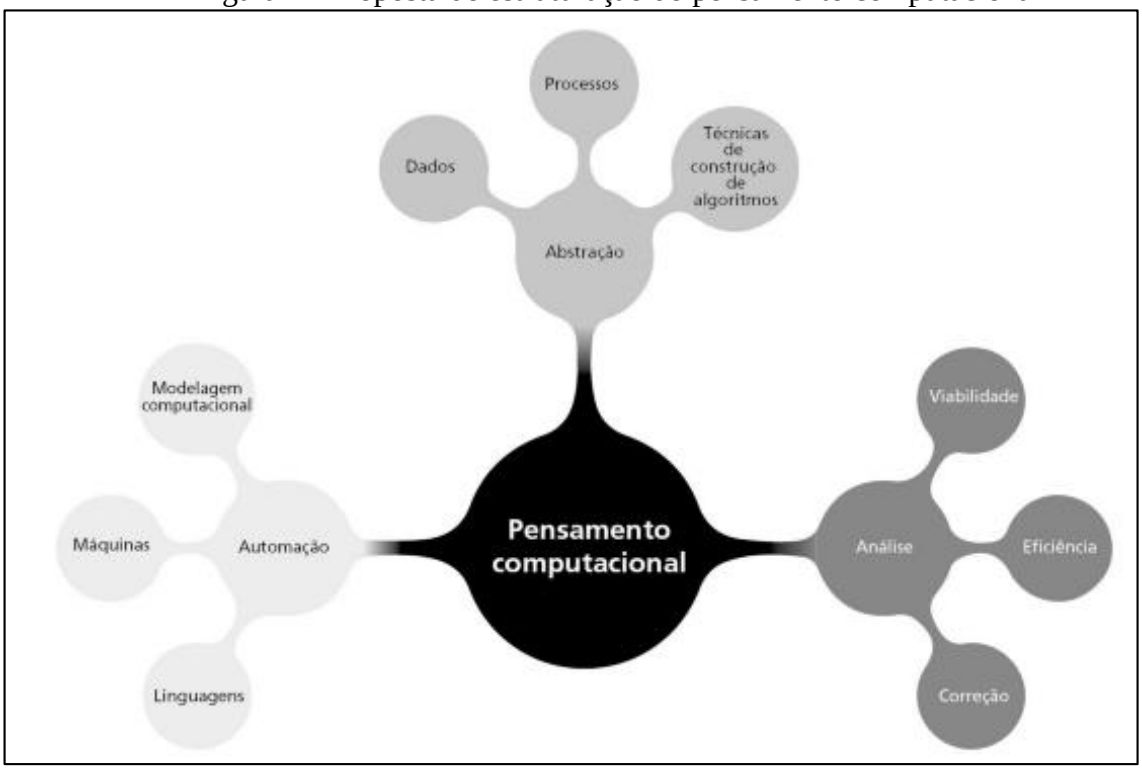
> Pensamento computacional é o processo de pensamento envolvidos na formulação de problemas e suas soluções, para que estas sejam representadas de uma maneira que possam ser efetivamente executadas por um agente de processamento de informações.

Wing propôs que essa forma de pensar e resolver problemas, geralmente usada por profissionais da computação, que inclui a capacidade de usar abstrações, reconhecer padrões para representar problemas de diversas maneiras, dividir problemas em partes menores e usar algoritmos para gerar soluções, não pode ser exclusiva dos profissionais de computação. E

chega a ponto de defender que o pensamento computacional deveria ser adicionado às habilidades de ler, escrever e operar com números, tal a sua importância desde a infância.

Em 2008, Wing apresenta uma proposta de estruturação para o pensamento computacional, representada por meio da figura a seguir.

Figura 1 - Proposta de estruturação do pensamento computacional



Fonte: Os pilares do pensamento computacional (WING, 2008).

No pilar de abstração se encontra as generalizações necessárias para a construção, análise e produção de algoritmos. Já no pilar de análises há uma busca por melhorias na eficiência de alguns algoritmos já implementados. E na automação buscamos a mecanização das soluções, de modo que as máquinas ajudem a solucionar problemas.

Nessa perspectiva, no pensamento computacional quando falamos em abstração relacionado a dados estamos nos referindo ao processo de representar informações do problema através de informações de entrada e saída no algoritmo. Dentro da computação os dados de maior relevância são os registros (coleção de informações, como um formulário), listas (listas de dados, baralho, pilha etc.) e grafos (estrutura usada para representar muitas estruturas tais como redes de modo geral, mapas etc.)

Os processos se referem a definição dos algoritmos utilizados para Resolução do problema de modo que seja compreensível ao leitor. Nestes podemos usar as seguintes

abstrações: composição (junção de passos no algoritmo), escolha (condicional ou tomada de decisão de acordo com os antecedentes passos do algoritmo) e repetição (laços ou recursão).

E a técnica de construção de algoritmos nos auxilia na construção de problemas mais complexos. Podemos dividi-la em: decomposição (divisão do problema em problemas menores), generalização (construção mais geral do problema) e transformação (utilização da solução de problema para resolver outro).

Em relação a automação, buscamos criar métodos que tornem a solução do problema automática pelo computador. Podemos usar as seguintes técnicas: máquina (escolha dela para trabalhar), linguagem (C, C++, python etc.) e modelagem computacional (modelos que simulam comportamento de sistemas reais tais como na área de biologia e médica).

Quando vamos implementar um problema, precisamos ter uma análise deste para saber se será possível ou não o solucionar. Para tal, vemos três questões: viabilidade (se existe a possibilidade de encontrar uma solução computacional), correção (verifica se o algoritmo realmente resolve o problema) e eficiência (avalia se o algoritmo é eficaz em relação ao tempo de execução, à margem de erro aceitável, se é compreensível ao leitor, dentre outros).

Assim, entendemos que enquanto a matemática busca soluções para problemas matemáticos ou de áreas afins, o algoritmo vai além disso, pois este pode ser usado para buscar soluções para questões cotidianas tais como: como devo ensinar a uma criança a arrumar as blusas no armário? Esse tipo de problema não pode ser resolvido através do pensamento combinatório. Ou seja, o algoritmo vai além disso. Então ele engloba as técnicas para a construção de algoritmos (técnicas de solução de problemas). O processo de sistematizar, representar e analisar a atividade de resolução de problemas é conhecido como pensamento computacional.

Em 2011, Wing (2011, p. 30) apresenta uma definição mais clara de pensamento computacional, expressa nos seguintes termos:

> Pensamento computacional é o processo envolvido na formulação de problemas e suas soluções, para que estas sejam representadas de uma maneira que possam ser efetivamente executadas por um agente de processamento de informações.

Assim, de maneira rotineira, o pensamento computacional descreve as atitudes mentais envolvidas na criação de problemas a fim de aceitar soluções computacionais e na proposta de soluções (WING, 2011).

A posição influente de Wing na National Science Foundation (NSF) ajudou a consolidar o termo pensamento computacional, ainda que não se tenha uma definição que não deixe lacunas na sua interpretação. Para Raabe (2020), não há atualmente nenhuma argumentação para esclarecer se de fato estamos falando de um novo tipo de pensamento ou de uma combinação de vários pensamentos existentes.

Uma definição que se apresenta com mais detalhamento e, consequentemente, com menos lacunas, sobre o que é pensamento computacional é a apresentada pelo International Society for Technology in Education (ISTE) juntamente com a Computer Science Teachers Association (CSTA), que concebem o termo como

> Um processo de resolução de problemas que inclui as seguintes características: formulação de problemas de forma que computadores e outras ferramentas possam ajudar a resolvê-los; organização lógica e análise de dados por meio de abstrações como modelos e simulações; automatização de soluções a partir de algoritmos; identificação, análise e implementação de soluções visando a combinação mais eficiente e eficaz de etapas e recursos; generalização e transferência de soluções para uma ampla gama de problemas (CSTA, 2015, p. 20).

Para finalizar, vale afirmar que essa segunda abordagem nasce da ciência da computação e atraca no cais da Educação Básica. Com a popularização do termo pensamento computacional, as ações sobre como difundir o ensino de computação, as discussões sobre como introduzi-lo na educação básica se ampliaram significativamente (Raabe, 2020, p.8)

Diversos trabalhos de mestrado, doutorado, pesquisas surgiram nesse movimento, muitas vezes atreladas a iniciativas de introdução ao pensamento computacional nas escolas, em parceria com pesquisadores e universidades (BARCELOS; SILVEIRA, 2012; ANDRADE et al, 2013; RAABE; ZEFERINO, 2014; DANTAS, 2019)

A Sociedade Brasileira de Computação (SBC) também busca formalizar esse pensamento e incentivá-la através da Olimpíada Brasileira de Informática no ensino básico. E o Referencial de Formação em Computação: Educação Básica. Este estabelece o que deve ser ensinado em relação a computação na educação básica. Outros ambientes e iniciativas também existem como CodeClubs, formada por voluntários que vão até às escolas a fim de realizar atividades de programação, ou a implantação de ambientes de programação como o Portugol Studio e em larga escala o incentivo de programação como o Programê.

A **terceira abordagem,** se refere a uma iniciativa da CODE.ORG, que vamos chamar de demanda de mercado.

A Code.org surgiu nos estados unidos devido a uma defasagem de programação nas escolas em 2013. Os idealizadores receberam patrocínio de grandes empresas para difundir a ideia. E nesse mesmo caminho surgiu a Hora do Código para incentivar a participação de escolas no ensino de programação. O Code.org apresenta problemas no estilo de jogos de com personagens e com programação no estilo de blocos. O foco dessa abordagem é nas oportunidades de trabalho e de carreira e focado nas empresas de tecnologia. Focado também que a programação traz novas oportunidades de trabalho no futuro.

Esta abordagem está de acordo com a ideia de que a maior demanda não será de programadores, mas sim de profissionais diferentes áreas que saibam programar e aplicar esse conhecimento em sua área de atuação. (BLIKSTEIN, 2018).

O foco dessa abordagem está nas oportunidades de trabalho e de carreira, e representa o interesse das empresas de tecnologia no aumento da oferta de mão de obra qualificada para continuar crescendo. É a programação como forma de atender à demanda do mercado.

Entendemos que aqui precisamos de um equilíbrio. Assim como uma população de estudantes pode ser beneficiar de uma formação que os leve a produzir novos significados, gerando conhecimentos que podem lhes ajudar em suas atividades profissionais, entendemos também que a forma de pensar por meio da computação não pode estar desvinculada da crítica, da ética, da solidariedade, ou seja, não pode ser limitada à geração de oportunidades com o risco de destruir algumas delas por não se pensar na coletividade e para a vida fora do trabalho.

A **quarta abordagem** se refere à equidade e inclusão. Basicamente, segundo Blikstein (2028), a preocupação aqui está em compreender o impacto da computação na sociedade e garantir a equidade e a diversidade de participação.

Para O´Neill (2016), a ciência da computação se tornará cada vez mais crucial para a participação cívica e a tomada de decisão informada.

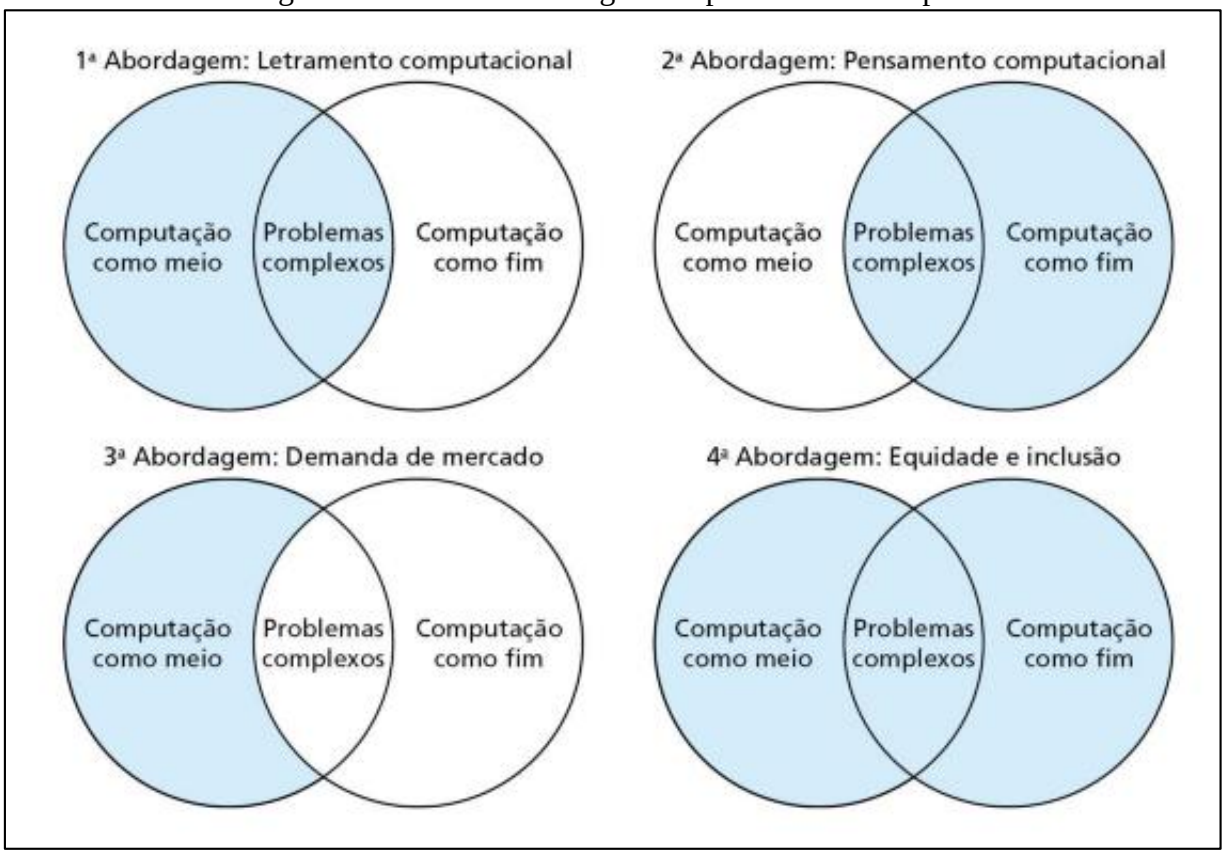
Nessa perspectiva, para exercerem plenamente sua cidadania, as pessoas vão precisar entender o que são algoritmos, como o trabalho vai mudar em função da automação e da inteligência artificial, como se posicionar e lutar politicamente para preservar a dignidade e alocar as pessoas substituídas por tal tecnologia, com usar dispositivos móveis para diagnósticos de doenças na busca por soluções de tratamento e assim por diante.

Em todas as abordagens vemos que a computação é vista como meio para produção de soluções, para construção de softwares e questões enriquecidas por tecnologia, programação básica de sistemas robotizados, jogos, materiais educacionais, etc; para solucionar problemas complexos por meio de modelagem matemática, por meio de aprendizagem de máquina,

estatística; estudar e classificar os problemas, a eficiência dos algoritmos e limites de computação.

A figura abaixo busca comparar as quatro abordagens apresentadas conforme as finalidades da computação em cada uma delas.

Figura 2 - Possíveis abordagens do pensamento computacional



Fonte: Raabe et al (2020).

Tendo em vista que cada abordagem parte de culturas e valores diferentes, elas têm potencial de impactar a educação também de formas diferentes, tanto na formação dos professores, quanto no design de currículos e nas escolhas de sua implementação.

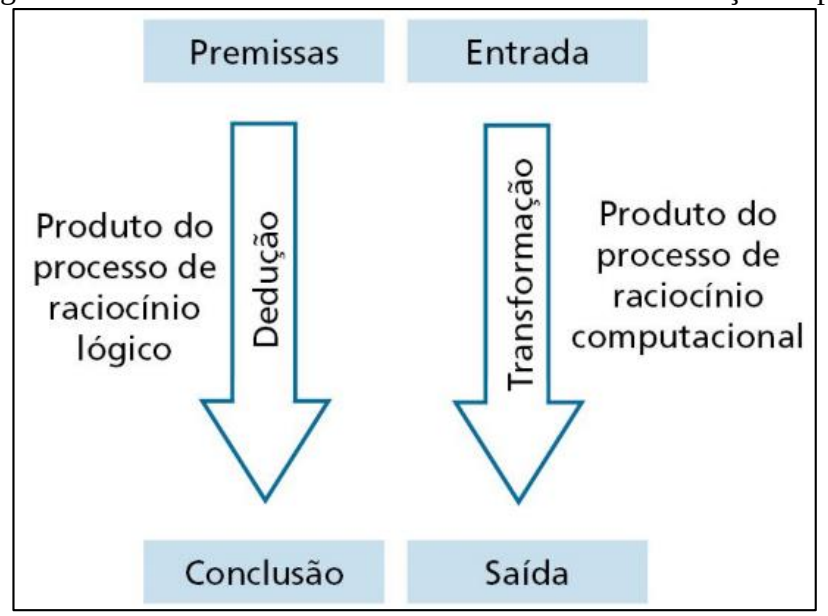
Algumas fontes importantes tais como o currículo australiano, o Currículo de Referência em Tecnologia e Computação do Centro de Inovação na Educação Brasileira (CIEB) e o Currículo da FabLearn da Columbia University dialogam entre si sobre o pensamento computacional como meio de ferramenta para diversas áreas de conhecimento na sua aprendizagem e desenvolvimento na aprendizagem.

Já a Sociedade Brasileira de Computação vai em direção ao uso do pensamento computacional ser trabalhado diretamente como uma disciplina voltada para a programação e aplicação desta.

Em relação a formação docente, podemos ter algumas perspectivas. Dentre elas, o pensamento computacional pode ser trabalhado com a aprendizagem de como programar com aplicações no ensino da matemática durante a graduação. Ou ainda através de formações continuadas os docentes possam ver esse tema. E também profissionais da área de ciência da computação atuarem como docentes.

O objetivo maior da computação é raciocinar sobre o pensamento lógico. Em outras palavras, ela busca formalizar o pensamento de modo a permitir sua automação e análise matemática. De uma maneira geral, o pensamento computacional pode ser visto como uma generalização do raciocínio matemático.

Figura 3 - Análise de entrada e saída de resultados e resolução de problemas



Fonte: Raabe et al (2020, p.59)

Já Alan Perlis defende a ideia de que todos devem programar em computadores com um desempenho universitário (PERLIS, 1962), pois isso poderia ser aplicado em todas as áreas de conhecimento.

Internacionalmente, a busca pela implementação do pensamento computacional nas escolas tem sido presente. Um dos motivos é que essa área traz benefícios na aprendizagem por prover reflexão, resolução de problemas e que a tecnologia digital está presente em todas as áreas de nossa sociedade e é interdisciplinar.

Com o levantamento realizado por Jones (2011), vamos ver alguns países e o processo de implantação do pensamento computacional. Na Alemanha a partir de 2004 as disciplinas de computação não eram mais obrigatórias e se tornaram eletivas.

Por outro lado, o fator motivacional para adoção dessa disciplina nas escolas foi pautado em: possibilitar que os discentes saibam lidar com questões cotidianas e tecnologias da informação no cotidiano de cada um e a promoção do ensino mais avançado de computação para os níveis técnico e superior. Dentre as disciplinas ofertadas destacamos algumas: orientação a objetos (codificação); modelagem entidade-relacionamento; autômatos; modelagem algorítmica; interação homem-máquina, privacidade e segurança; arquitetura de computadores; computabilidade, eficiência e questões sociais (BRINDA; PUHLMANN; SCHULTE, 2009). O maior foco no fundamental II é na modelagem e possuem uma grande quantidade de professores com formação adequada para lecionar nessa área. É exigido duas graduações, preparação pedagógica e estágio de dois anos.

Na Argentina em 2015 Consejo Federal de Educación (CFE) através da resolução nº 263/15, estabeleceu diversos elementos, nos quais destacamos o ensino de programação como integrante do currículo ou atividade extraclasse durante os anos obrigatórios, criação de Red de Esculas que programan (REP) de acordo com a disponibilidade de professores com formação para tal e propagação dessas escolas a ponto de atingir níveis estaduais com o tempo através de iniciativas de formação continuada dos docentes em serviço.

Na Austrália a partir de 2015 ocorreu uma reestruturação dos currículos e foi adotada a programação como uma das principais competências com objetivo de desenvolver recursos curriculares para o ensino da matemática, oferecer suporte para o conhecimento básico de programação em todos os níveis de escolaridade, criar escola-piloto pautada na estratégia P-TECH (ONG's com metodologia de ensino com foco na formação do aluno para o mercado de trabalho) e financiamento de escolas de verão para alunos da área de ciência, tecnologia, engenharia e matemática (STEM: Science, technology, engineering, and mathematics). No 5º e 6º ano os discentes aprendem a codificar (pensar computacionalmente) e no 7º aprendem a programar (escrever algoritmos) e as disciplinas de história e geografia deixam de ser obrigatórias e passam a ser optativas.

A Coréia do Sul já trabalha com computadores em sala de aula desde a década de 70. No fim da década de 80 já era trabalhada a alfabetização digital. Em 2007 houve um consenso entre docentes que o computador não deveria ser trabalhado em sala de aula apenas com esse fim, mas também para a resolução de diversos tipos de problemas e que são do mundo real, para aumentar a habilidade de raciocínio. A disciplina era ofertada desde o ensino fundamental como eletiva e ainda havia conteúdos como ética na informática e cibercrime (CHOI; AN; LEE, 2015). Em 2015 houve uma redução na oferta dessa disciplina devido à redução da carga horária

de disciplinas eletivas, ausência de uma regulamentação específica para essa disciplina e a ausência de avaliações institucionais relativa às essas aulas. E a partir 2017 iniciou uma busca de como mudar essa realidade.

A Escócia iniciou a discussão sobre o ensino do pensamento computacional a partir dos anos 80. Nessa época, a visão era voltada para a área de software e de banco de dados aplicada em questões práticas. Essa visão foi discutida por estudiosos se realmente era eficiente no ensino básico, pois o ensino computação era voltado para aplicativos de escritório, de acordo com pesquisa feita em 2004. Assim, em 2011, foi lançado um documento Curriculum for Excellence (EDUCATION SCOTLAND, 2009), que destaca três pontos esperados na educação (de 3 a 18 anos) a ser alcançados: TICs para melhorar a aprendizagem, ciência da computação contextualizada no desenvolvimento de competências tecnológicas.

Nos Estados Unidos em 2015, no documento Every Student Succeeds Act (ESSA) a disciplina ciência da computação é colocada em mesmo grau de importância tanto quanto matemática, história e inglês. Ainda não existe um documento legal que obriga o ensino dessa disciplina em todo o país, mas alguns estados adotaram tal disciplina como obrigatória e até mesmo as vezes substituindo disciplinas tidas como obrigatórias ou tradicionais. Alguns estados trocaram a disciplina de matemática por ciência da computação baseados no currículo proposto por Computer Science Teachers Association (CSTA), conhecido como A Model Curriculum for K-12 Computer Science (CSTA, 2011). Nele é apresentado como trabalhar ciência da computação em todos os anos escolares, bem como são propostos exercícios.

A Estônia é um país totalmente pautado na dependência da internet para seu desenvolvimento. Por isso, no final da década de 90 todas as escolas do país tinham computadores e o projeto levava em consideração o aluno aprender a programar desde os 7 anos. Atualmente esse ensino acontece nacionalmente e está presente em todos os níveis educacionais e técnico. Dentre as ações existentes, há ferramentas à disposição, tais como Oppematerjalid, para incentivar professores a formação continuada e ao ensino de programação nas escolas.

Na França, a partir de 2015, foi decidido que era necessário trabalhar programação e computação nos primeiros anos escolares, pois era uma maneira de seguir as novas demandas da atualidade. Então em 2017 foram estabelecidas as disciplinas de fundamentos das linguagens de programação e desenvolvimento de aplicativos com o uso de algoritmos simplificados (FLEURY; NEVEUX, 2017).

A Finlândia adotou o ensino de computação a partir de 2016 pautada no guia Koodiaapinen 2016 de Mykkänen e Liukas (2014). Neste guia é recomendado trabalhar no 1ª e 2º ano o pensamento computacional por atividades lúdicas, do 3º ao 6º ano desenvolver atividades no computador com programação visual e incentivando a perda do medo de errar e do 7º ao 9º ano familiarizar os alunos com uma linguagem de programação real como por exemplo o Python. Eles oferecem também uma formação gratuita para os professores através da plataforma massive open online course (MOOC).

Outros países como Grécia e Reino Unido também já adotam em seu currículo essa área de conhecimento. A tabela a seguir mostra alguns países e como a computação foi implementada:

Figura 4 - Tabela comparativa do ensino do pensamento computacional nos países

Países	Ano de adoção	Fundamental	Médio	Técnico/vocacional	Depende da região ou do currículo adotado	Possui disciplina específica	Modo de integração
Alemanha	2004	F	F				N
Argentina	2015	F	F		Sim	Varia	NR
Austrália	2015	C					N
Áustria	2009	F	F	F	Sim	Sim	N
Bélgica / Holanda	Varia	F	F				R
Bulgária	2006		C	C		Sim	N
Coreia do Sul	2007	F	F			Sim	N
Dinamarca	2014	C	F	F		Varia	N
Escócia	1987	F	F		Sim	Varia	R
Eslováquia	1990	C	C	C		Sim	NE
Espanha	2015	F	F		Sim	Varia	NR
Estados Unidos	2015	F	F		Sim	Varia	NR
Estônia	1996	F	F	F	Sim	Varia	NR
Finlândia	2016	C				Não	NRE
França	2016	FC	C		Sim	Não	N
Grécia	1993	C	C			Sim	N
Hungria	1995		C	C		Sim	N
Irlanda	2014	F			Sim	Sim	NE
Israel	1976	F	F	F		Sim	N
Lituânia	1986	F	F			Varia	NE
Malta	1997		F			Sim	N
Polônia	1985	F	F	F	Sim	Sim	N
Portugal	2012	C		C		Sim	N
Reino Unido	2014	FC	FC		Sim	Sim	N
República Tcheca	1990		C	F		Sim	E

F = facultativo C = compulsório N = nacional R = regional E = escolar

Fonte: Raabe, 2014.

No Brasil ainda não existe um reconhecimento oficial da importância ou implantação do pensamento computacional no currículo educacional. Mas a BNCC apresenta tecnologias

digitais como um tema integrador. Isso pode ser usado como uma ponte entre essa área e a educação. Em 2018 na tentativa de acelerar ou auxiliar a implantação dessa área nos currículos, o Centro de Inovação para a Educação Brasileira (CIEB) apresentou o Currículo de Referência em Tecnologia e Computação organizado em três eixos e 10 habilidades. Tais eixos se dividem em: cultural digital, tecnologia digital e pensamento computacional. O tema dessa dissertação se encaixa no eixo de pensamento computacional. Em relação aos conceitos, o pensamento computacional se divide em abstração, algoritmo, decomposição e reconhecimento de padrões. Esses são os conceitos que são utilizados para analisar as questões escolhidas para essa dissertação.

Figura 5 - Estrutura do Currículo de Referência em Tecnologia e Computação

Fonte: CIEB

O conceito de abstração é a filtragem e classificação dos dados ou formas de organizar as informações em estruturas que auxiliem na resolução de problemas. O algoritmo é o conjunto de passos a ser dado ou de processos a serem executados para a resolução do problema em si. A decomposição é a partição do problema em casos menores e de mais facilidade de solução. A representação de padrões trata-se na identificação de características comuns entre problemas e soluções.

Nesse mesmo rumo, a Sociedade Brasileira de computação buscou uma abordagem similar (SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 2019). O início da ideia de

computação surgiu com a linguagem logo. Até o ano de 96 no Brasil muitas pesquisas foram realizadas acerca dessa linguagem (VALENTE, 1996). Posteriormente, o interesse em robótica educacional aumentou e perpetua até os dias de hoje. A partir da linguagem logo que se desenvolveu ou se modificou com o tempo, surgiu o Scratch. Ele faz muito sucesso de início com os alunos que ficam interessados e motivados com a iteração da linguagem orientada a objetos. Mas com o passar do tempo muitos alunos perdem o interesse em tal linguagem ou acham infantis os temas propostos.

Várias resoluções foram criadas no intuito de dar abertura para a tecnologia na educação. Destacamos o Conselho Nacional de Educação (CNE) em 2015 incentiva o ensino de TICs para ampliação da formação cultural de professores e estudantes. Consequentemente, diversas pesquisas têm sido realizadas sobre o pensamento computacional. (BARCELOS; BRACKMANN, 2017; FRANÇA; AMARAL, 2013; ANDRADE et al., 2013; VIEL; RAABE; ZEFERINO, 2014; CAMPOS et al., 2014).

Dentre os ambientes de programação vale destaque o Portugol Studio e olimpíadas como a Olimpíada Brasileira de Matemática ou a Olimpíada de Raciocínio Lógico, com o intuito de despertar no aluno o interesse por uma ciência que é primordial para sociedade através da competição e curiosidade com desafios a resolver.

O pensamento computacional auxilia ao discente desenvolver outras áreas de pensamento, tais como o pensamento abstrato (uso de subjetividade em diferentes níveis de dificuldade), pensamento algorítmico (demonstração da solução através de diversos passos ou procedimentos), pensamento lógico (formulação e eliminação de hipóteses) e pensamento dimensionável (divisão de um problema em partes menores).

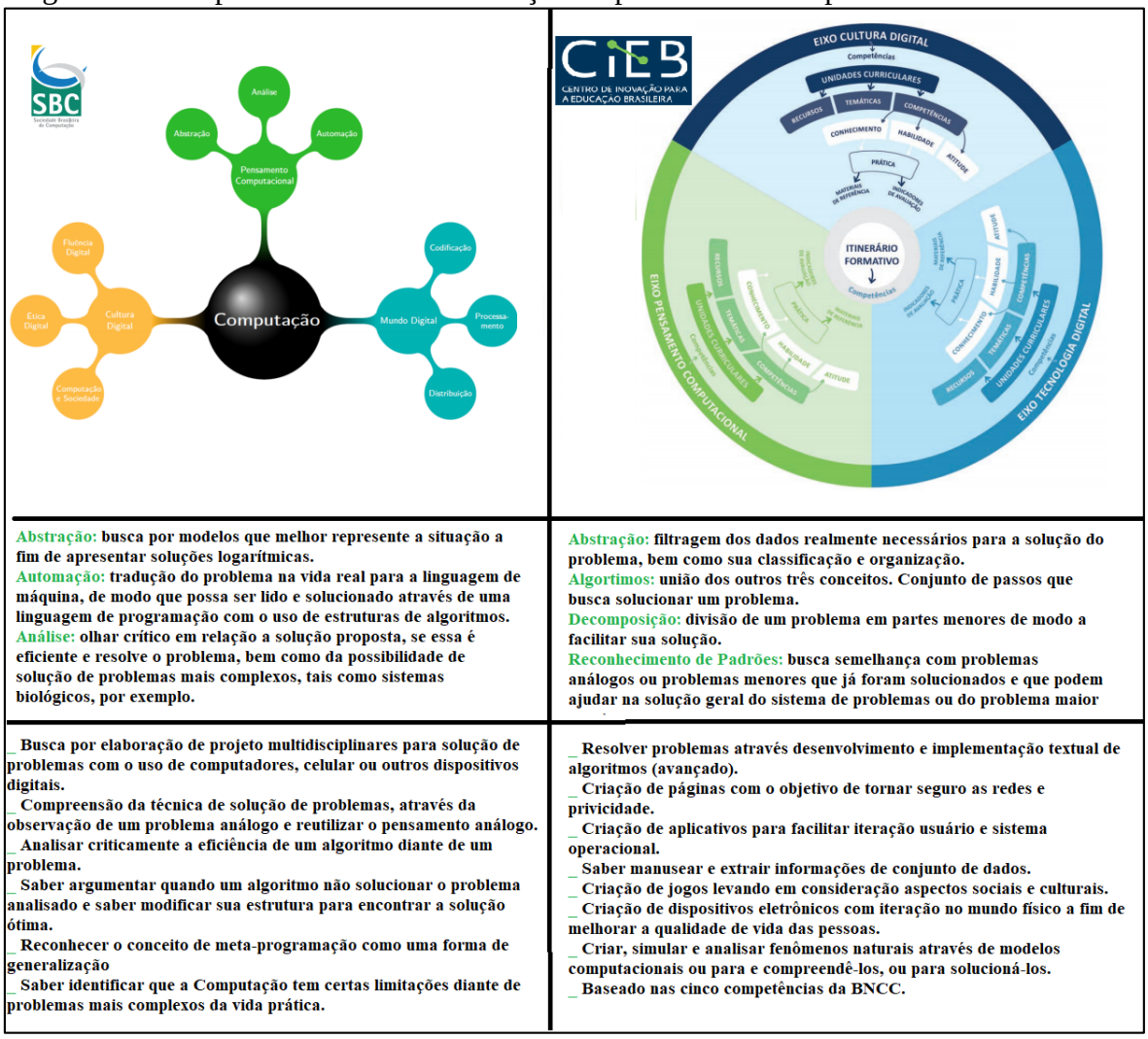
Outro aspecto a ser considerado é que a aprendizagem de Álgebra, como também aquelas relacionadas a Números, Geometria e Probabilidade e Estatística, podem contribuir para o desenvolvimento do pensamento computacional dos alunos (BRASIL, 2018, p. 271).

Ao trabalharmos ao longo dessa dissertação com a noção de pensamento computacional, usaremos certos termos que necessitam ser explicados. O termo algoritmo é um conjunto de passos dados a fim de resolver um problema ou uma classe (conjuntos) de problemas de mesma espécie. Já a linguagem de programação se trata de um conjunto de comandos que transcrevem para o computador o que foi pensado pelo humano que criou o algoritmo. O fluxograma é uma estrutura visual de apresentação de um algoritmo. A condicional é um conceito que engloba a ideia de que uma ação só pode ocorrer se antes outra tiver acontecido, ou seja, temos que tomar a decisão de acordo com os passos anteriores. A variável é um espaço na memória do computador onde é armazenado uma parte da informação por um determinado tempo. A

repetição é um recurso que está presente como uma estrutura na linguagem de programação, com o intuito de refazer os mesmos passos de acordo quando certa ação ocorrer no algoritmo. Barichello (2021).

Na BNCC encontramos o uso de fluxogramas proposto para o Ensino Fundamental (como nas habilidades EF06MA04, EF07MA07, EF08MA10, EF09MA15). Este quando bem definido como um conjunto de ações a serem realizadas também pode ser definido como um algoritmo ou uma representação algorítmica. Um algoritmo para ser de fácil entendimento, deve estar bem-organizado e bem estruturado com cada função, procedimento ou recursão bem escritos e claros para leitura.

Figura 6 - Comparativo entre a estruturação do pensamento computacional: SBC x CIEB



Abstração: busca por modelos que melhor represente a situação a fim de apresentar soluções logarítmicas.
Automação: tradução do problema na vida real para a linguagem de máquina, de modo que possa ser lido e solucionado através de uma linguagem de programação com o uso de estruturas de algoritmos.
Análise: olhar crítico em relação a solução proposta, se essa é eficiente e resolve o problema, bem como da possibilidade de solução de problemas mais complexos, tais como sistemas biológicos, por exemplo.

Abstração: filtragem dos dados realmente necessários para a solução do problema, bem como sua classificação e organização.
Algortimos: união dos outros três conceitos. Conjunto de passos que busca solucionar um problema.
Decomposição: divisão de um problema em partes menores de modo a facilitar sua solução.
Reconhecimento de Padrões: busca semelhança com problemas análogos ou problemas menores que já foram solucionados e que podem ajudar na solução geral do sistema de problemas ou do problema maior

_ Busca por elaboração de projeto multidisciplinares para solução de problemas com o uso de computadores, celular ou outros dispositivos digitais.
_ Compreensão da técnica de solução de problemas, através da observação de um problema análogo e reutilizar o pensamento análogo.
_ Analisar criticamente a eficiência de um algoritmo diante de um problema.
_ Saber argumentar quando um algoritmo não solucionar o problema analisado e saber modificar sua estrutura para encontrar a solução ótima.
_ Reconhecer o conceito de meta-programação como uma forma de generalização
_ Saber identificar que a Computação tem certas limitações diante de problemas mais complexos da vida prática.

_ Resolver problemas através desenvolvimento e implementação textual de algoritmos (avançado).
_ Criação de páginas com o objetivo de tornar seguro as redes e privicidade.
_ Criação de aplicativos para facilitar iteração usuário e sistema operacional.
_ Saber manusear e extrair informações de conjunto de dados.
_ Criação de jogos levando em consideração aspectos sociais e culturais.
_ Criação de dispositivos eletrônicos com iteração no mundo físico a fim de melhorar a qualidade de vida das pessoas.
_ Criar, simular e analisar fenômenos naturais através de modelos computacionais ou para e compreendê-los, ou para solucioná-los.
_ Baseado nas cinco competências da BNCC.

Fonte: O autor baseado no CIEB e SBC.

Realizamos um comparativo entre o currículo apresentado pela SBC e do CIEB. O do CIEB é pautado no currículo da SBC, bem como nas competências da BNCC e em referências internacionais curriculares, bem como em teóricos tais como Raabe (2017) e Valente (2002). No desenvolvimento desse trabalho utilizamos a definição do CIEB para pensamento computacional. Escolhemos essa uma vez que é mais usada no referencial teórico, bem como é um currículo mais detalhado. As etapas do pensamento computacional adotadas são mais familiares aos termos usados na resolução de problemas de combinatória, o que facilita a solução via pensamento computacional. Além disso, o próprio currículo do CIEB foi desenvolvido a partir do SBC, mas com mais detalhes e relacionado com a BNCC e suas respectivas habilidades e competências. Para quem queira saber mais sobre a relação entre matemática e pensamento computacional recomendamos Barcelos (2015), o qual apresenta uma revisão sistemática da literatura sobre tal área.

2.2 Pensamento Computacional: CIEB X BNCC

Nessa seção apresentamos mais detalhadamente a versão de pensamento computacional, apresentada pelo CIEB (2020), **que tomamos como referencial em nossa investigação,** compreendendo a resolução dos problemas de combinatória via pensamento computacional. Nesse documento, o CIEB considera que pensamento computacional se refere à capacidade de compreender, definir, modelar, comparar, solucionar, automatizar e analisar problemas (e soluções) de forma metódica e sistemática, considerando conhecimentos e práticas de computação, geralmente através da construção de algoritmos (CIEB, 2020, p. 8;21)

Essa concepção do CIEB considera quatro "conceitos" (os quais chamaremos de processos) presentes no pensamento computacional, que se conectam de forma dinâmica e não necessariamente na ordem que apresentamos abaixo. Os processos são:

1) reconhecimento de padrão - padrão no problema
2) decomposição - divisão de problemas complexos para problemas menores
3) algoritmos - construção de orientações claras para a resolução de problemas.
4) abstração - filtragem e classificação de dados para resolução de problemas

Para o CIEB (2020, p.21)

> ✓ **Abstração**: o conceito envolve a filtragem dos dados e sua classificação, ignorando elementos que não são necessários, visando os que são relevantes. Envolve também formas de organizar informações em estruturas que possam auxiliar na resolução de problemas.
>
> ✓ **Decomposição**: a decomposição trabalha o processo pelo qual os problemas são divididos em partes menores e mais fáceis de resolver. Compreende também a prática de analisar problemas a fim de identificar quais partes podem separadas, e de que forma podem ser reconstituídas para a solução de um problema global. Essa prática também possibilita aumentar a atenção aos detalhes.
>
> ✓ **Reconhecimento de padrões:** trabalha a identificação de características comuns entre os problemas e suas soluções. Resulta do fato de realizar a decomposição de um problema complexo para encontrar padrões entre os subproblemas gerados. Esses padrões são similaridades ou têm características que alguns dos problemas compartilham e que podem ser explorados para que sejam solucionados de forma mais eficiente
>
> ✓ **Algoritmos:** conceito que agrega todos os demais. O algoritmo é um plano, uma estratégia ou um conjunto de instruções claras e necessárias para a solução de um problema. Em um algoritmo, as instruções são descritas e ordenadas para que o objetivo seja atingido; podem ser escritas em formato de diagramas, pseudocódigo (linguagem humana) ou escritos em códigos, por meio de uma linguagem de programação

Tais processos são tão aplicáveis em nosso cotidiano quanto a abstração matemática por si só. E, diante disso, levantamos algumas reflexões. Dentre elas, numa aula de matemática, observamos a contextualização e a resolução de problemas partindo de um tema, mas que é focado na demonstração da resolução daquele problema. Já no Pensamento Computacional tendemos a pensar em automatizar a resolução de um problema para torná-la genérica para aquela classe de problemas (complexidade dos algoritmos, por exemplo, temos problemas NP-completo).

A noção de algoritmo em pensamento computacional pode ser direcionada para a resolução de problemas, enquanto no pensamento matemático nem sempre é problematizado, como por exemplo a aplicação do algoritmo de Euclides diretamente para resolver uma equação diamantina. Até podemos generalizar tal equação, mas não é fator condicional para que o algoritmo seja aplicado.

A comparação de respostas viabiliza a troca e conhecimento, aprendizagem, aperfeiçoamento, comparação de dados, algoritmos e tomada de decisão. O fluxograma serve para organizarmos e estruturarmos as condições do problema e os passos a serem seguidos em cada etapa do algoritmo. É uma visão do PC mais voltada para programação, que é uma subárea do PC (este é muito mais amplo).

Uma aula de matemática trabalha o pensamento dedutivo e lógico por exemplo. A oportunidade de testar mais rápido algumas conjecturas e realizar experimentos fica mais eficiente através do pensamento computacional aplicado ao computador como uma ferramenta quando programo alguma fórmula ou algo que queremos testar ser verídicos ou não.

Para professores, o Pensamento Computacional pode resolver problemas matemáticos e ser uma ótima ferramentas para resolver e expandir problemas. O algoritmo vira um objeto de interesse por si só (interesse que sai da matemática em direção à computação). Introduzir recursos computacionais como linguagens de programação e como ferramentas é uma tentativa de implementar a tecnologia em sala de aula ou recursos computacionais nas aulas de matemática. E, teoricamente, de acordo com a BNCC, não é o livro que vai dizer para que série é, é a escola que vai decidir isso, em relação a como, quando, porque, quantas vezes e por quanto tempo faremos essa aplicação do pensamento computacional aliado a matemática.

Ao propormos um problema ao aluno esse muitas vezes tem dificuldades em organizar o pensamento e talvez fosse necessário ajudá-los na organização do pensamento através da divisão do problema em casos a fim de que o aluno percebesse algum padrão. Tal característica é fator de intersecção entre o pensamento combinatório e o computacional.

A partir do momento que observamos problemas matemáticos e como podemos resolvê-los do ponto de vista do pensamento computacional, deixamos de nos focar na solução. A partir daí focamos no processo de resolução de problemas como um processo que acontece através de um processo algorítmico, o qual segue os passos com determinada ordem a ser seguida. Ou seja, discute o processo de resolução de problemas.

Porém também podemos resolver problemas matemáticos através do uso do computador como ferramenta auxiliar, analisar os resultados, otimizar o tempo e até mesmo expandir o problema.

Tanto no pensamento combinatório quanto no computacional podemos ter processos de experimentação. No pensamento combinatório buscamos uma conjectura que solucione nosso problema e no computacional buscamos procedimentos. Diante da observação de um procedimento empírico, podemos tentar chegar a conclusões que seja pela conjectura ou seja pela implementação não estamos demonstrando, mas podemos testar vários casos para ver se encontramos algum caso que mostre ser falsa a suposição.

Quando pensamos em casos pequenos está presente o pensamento combinatório, mas para maiores temos o pensamento computacional. Ao solucionarmos um problema de anagramas de uma palavra estabelecida, aplicamos o pensamento combinatório. Mas, ao pensarmos nesse mesmo problema de maneira generalizada, chegamos numa intersecção com pensamento computacional e matemático. Se pensarmos em simulações do corpo humano, em ordenação de procedimentos e automação, nós direcionamos ao pensamento computacional. Algumas funções matemáticas podem ser implementadas computacionalmente e no uso de simuladores o aluno tem a oportunidade de perceber o funcionamento e comportamento de certas funções e propriedades matemáticas. Alguns problemas matemáticos são imediatos de resolver através de fórmulas. Já para alguns problemas já são mais computacionais e no papel se torna exaustivo e nem sempre precisamos todo a bagagem matemática, mas sim a computacional e através da observação dos resultados podemos refletir sobre propriedades matemáticas.

Outra questão é a abordagem de pensamento computacional ao longo do ensino médio, a questão de seguir a ordem dos procedimentos não é fácil e tampouco o homem consegue ter um pensamento linear, somos complexos e holísticos. Ou seja, vemos algo como um grande bloco e não minuciosamente. O nosso pensamento é global e não local.

O pensamento computacional é mais abrangente e a matemática seria uma possível área de conhecimento de aplicação. De acordo com o DisEssa é uma grande habilidade e ferramenta que pode ser usada em todas as áreas de conhecimento.

Ainda que o computador possa ter aplicabilidade o aprendizado da matemática é importante para o desenvolvimento do raciocínio lógico e dedutivo, bem como até mesmo para identificar erros na programação que demonstram que aquela programação está equivocada. O pensamento abstrato e dedutivo é característico da matemática. E isso se diferente do pensamento indutivo.

Podemos fazer uma comparação entre o pensamento combinatório e computacional. Através do pensamento combinatório podemos chegar a uma conjectura, enquanto o computacional pode auxiliar-nos no teste se essa conjectura é válida através de testes calculados rapidamente pelo computador e encontrar possíveis erros. Quando pensamos na resolução de problemas, conseguimos resolver alguns casos limitados, mas para a generalização precisamos de um pensamento mais no caminho de algoritmo e no passo de abstração no pensamento computacional.

Quando vamos para testar certos resultados matemáticos através de cálculos, levamos mais tempo para resolver e testar os casos do que se fizéssemos os cálculos através de um programa que automatizasse essa testagem. Alguns problemas de combinatória podem ser resolvidos facilmente através do método de enumeração ou exaustão, mas por outro lado se tivéssemos que resolver para valores maiores, talvez dessa maneira seria ineficiente e o computador seria a solução por meio de ferramentas como recursividade, o algoritmo, abstração, reconhecimento de padrões e decomposição. E para resolver e expandir a resolução de problemas em que utilizamos o computador como uma ferramenta potente e capaz de calcular questões que sem ele seriam muito trabalhosas ou plausíveis de muito erro.

Dentre os problemas que podemos usar o computador como ferramenta estão as modelagens matemáticas voltadas para o entendimento do funcionamento do corpo humano. Sem ele, não seria plausível observarmos o que acontece no corpo humano.

Além disso, podemos pensar em expandir ou aumentar a quantidade analisada e que sem ser no computador seria impossível ou com alta probabilidade de erro nos cálculos e os cálculos manualmente exigiriam muito tempo. O computador otimiza o tempo e diminui a margem de erro em relação aos cálculos.

Podemos usar o algoritmo como ferramenta na resolução de problemas ou podemos ter o algoritmo como foco e termos atenção na sua construção, formulação, prática e comportamento. Nesse caso estamos focados no processo de resolução de problemas. Quando o algoritmo é uma ferramenta estamos focados na solução obtida através de um algoritmo sem darmos ênfase no processo em si.

Ao levarmos o foco para o pensamento computacional ganhamos o foco no algoritmo, no processo de resolução de problemas, o foco na construção da resolução do problema e não apenas o foco na solução final e como ordenar e organizar o pensamento do aluno a fim de que siga um padrão ou um a recursividade a fim de que solucione aquele problema ou que busque generalizar tal solução.

A modernização dos currículos e dos conceitos no currículo matemático é muito fraca e sutil, mas o site CSunpulged por exemplo, faz com que esse direcionamento para o computador e ao pensamento computacional de maneira direta e simples e objetiva. Isso acontece não só na matemática, mas em todas as áreas de conhecimento. Todas utilizam computação como ferramenta e o pensamento computacional está penetrando em todas as áreas

Então atualizar o currículo na direção da computação é interessante e muito importante. Tanto é necessário que vemos na BNCC já habilidades voltadas para essa área, tais como:

(EM13MAT315) Reconhecer um problema algorítmico, enunciá-lo, procurar uma solução e expressá-la por meio de um algoritmo, com o respectivo fluxograma

(EM13MAT405) Utilizar os conceitos básicos de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática.

Em combinatória ao esquematizarmos, descrevermos como um problema é resolvido, quais são os cálculos necessários para resolvermos a questão e a descrição do trajeto a ser percorrido é uma intersecção com o tópico de algoritmos em pensamento computacional. Ao resolvermos um problema de combinatória no qual o dividimos em casos ou buscamos aquilo que não queremos, estamos trazendo à tona uma questão também que intercepta com o Pensamento computacional na decomposição de tarefas, agrupamentos por semelhanças ou características em comum, dentre outros aspectos.

Aos nos depararmos com problemas matemáticos nos quais classificamos em: otimização, existência e enumeração, estamos identificando e reconhecendo padrões, nos quais separamos por métodos distintos de solução, o que facilita na resolução do problema. Analogamente, isso também acontece na parte de pensamento computacional, quando separamos por determinadas características.

Posteriormente, ao tentarmos generalizar uma solução, podemos usar o fluxograma ou o algoritmo para facilitar a compreensão e abstração do conhecimento, bem como em soluções de maiores níveis de dificuldades.

Dentre as competências da BNCC destacamos algumas: “valorizar e utilizar os conhecimentos historicamente construídos sobre o mundo físico, social, cultural e digital”. Tenhamos atenção ao termo digital que já é usado para contextualizar a época na qual vivemos.

Para utilizarmos conhecimentos digitais podemos naturalmente automatizar algumas atitudes, mas o pensamento computacional é uma característica que auxilia nessa automatização dos recursos matemáticos. Além de “utilizar diferentes linguagens para se expressar e partilhar informações, experiências, ideias e sentimentos”, em outras palavras, o algoritmo dentro do pensamento computacional nada mais é do que uma linguagem de máquina de alto. E em relação a comunicação: “exercitar a empatia, o diálogo, a resolução de conflitos e a cooperação”. A resolução de problemas e pensamento computacional podem ser trabalhados nessa direção. Ao tentarmos encontrar uma solução abstrata ou gera do problema, como por exemplo, pelo algoritmo, estamos lindando com a seguinte com “agir pessoal e coletivamente com autonomia, responsabilidade, flexibilidade, resiliência e determinação”

Por exemplo, quando trabalhamos com percursos, menor caminho ou solução ótima normalmente pensamos em usar um algoritmo eficiente e de rápida solução.

A importância e aplicação do pensamento computacional é reconhecida e trabalhada internacionalmente em países tais como França, Portugal, Espanha, Argentina, dentre outros. Tal reconhecimento no Brasil aparece com destaque a partir da BNCC.

Um dos pontos defendidos é que esta área de conhecimento deve ser trabalhada e explorada desde o início do ensino básico, a fim de que se torne algo natural e comum para as crianças tanto quanto é mexer no celular.

Por isso ao longo dessa dissertação apresentamos questões já existentes de vestibulares passados com o intuito de mostrar possíveis caminhos de trabalharmos o pensamento computacional na resolução de problemas de combinatória. Tais questões possuem uma sugestão de solução de como seria possível abordar tal tema em cada uma delas.

Além disso, há orientações de como poderia ser trabalhada, em particular no Ensino Médio. Ainda assim, é importante frisar que são sugestões e não imposições ou caminhos rígidos e únicos. Cada uma delas é passível e possível de ser adaptada e abordada de acordo com o público-alvo e seu nível de conhecimento, dificuldade e experiências.

Ou seja, damos possíveis caminhos de como desenvolver tais questões em sala de aula, mas cada um irá trilhar por si o tema de sua maneira, o que amplia e diversifica o tema e acervo sobre essa temática.

A Base Nacional Comum Curricular (BNCC) garante o direito de aprendizagem e desenvolvimento do aluno, assim como é defendido no Plano Nacional de Educação (PNE). Na BNCC (BRASIL, 2018, p. 16) destaca-se que: "Os currículos devem adequar as proposições da BNCC à realidade local, considerando a autonomia dos sistemas ou das redes de ensino e das instituições escolares, como também o contexto e as características dos alunos. "

Isso reforça a importância de trabalharmos o pensamento computacional desde o início da educação básica, uma vez que essa faz parte do cotidiano do cidadão desde pequeno, seja na tomada de decisão, seja nas novas demandas que a era digital requer, seja no letramento digital, dentre outros.

Destacamos que a " BNCC deve fundamentar a concepção, formulação, implementação, avaliação e revisão dos currículos, e consequentemente das propostas pedagógicas das instituições escolares" (BRASIL, ANO, p. 20). Ou seja, ela é a base para que cada instituição desenvolva de acordo com a sua realidade as habilidades e competências ali destacadas.

A BNCC é referência nacional e auxilia no gerenciamento de políticas e ações educacionais em todos os âmbitos, federal, estadual, distrital e municipal e na formação de professores em relação à definição de recursos didáticos e aos critérios definidores de infraestrutura adequada para o pleno desenvolvimento da oferta de educação de qualidade (Resolução CNE/CP 2/17, Art. 5, § 1º).

Dentre as competências gerais da Educação Básica no Ensino Fundamental, destacamos as seguintes:

2 - Exercitar a **curiosidade intelectual** e recorrer à abordagem própria das ciências, incluindo a **investigação, a reflexão, a análise crítica, a imaginação e a criatividade, para investigar causas, elaborar e testar hipóteses, formular e resolver problemas e criar soluções (inclusive tecnológicas)** com base nos conhecimentos das diferentes áreas. – Grifo nosso

4 - **Utilizar diferentes linguagens** – verbal (oral ou visual-motora, como Libras, e escrita), corporal, visual, sonora e **digital** –, bem como conhecimentos das **linguagens** artística, **matemática** e **científica**, para se **expressar** e **partilhar informações, experiências, ideias e sentimentos em diferentes contextos e produzir sentidos que levem ao entendimento mútuo**. – Grifo nosso

5 - **Compreender, utilizar e criar tecnologias digitais de informação e comunicação de forma crítica, significativa, reflexiva** e ética nas diversas práticas sociais (incluindo as escolares) para se comunicar, acessar e disseminar informações, produzir conhecimentos, **resolver problemas** e exercer protagonismo e autoria na vida pessoal e coletiva. – Grifo nosso

Diante dos destaques que realizamos fica explícito a importância do pensamento computacional no ensino dos alunos, na autonomia de resolver problemas, da presença de uma mudança nos conceitos de sociedade em que temos a parte digital e novos problemas cotidianos para resolver. Percebemos o protagonismo do aluno sempre defendido e que agora aparece destacado na BNCC e que pode ser alcançado através de metodologias ativas.

A ideia de resolução de problemas não é só para problemas de matemática, mas que o aluno tenha capacidade de resolver problemas e buscar soluções para a vida cotidiana, o que exigido do ser humano no século em que vivemos, em um mundo cada vez mais dinâmico e digital e com os avanços tecnológicos que trazem cada vez mais mudanças rápidas.

Quando ensinamos matemática, iniciamos com exemplos do cotidiano, de preferência experiências comuns dos alunos e depois vamos aumentando a complexidade das questões e do assunto. Alguns são precedentes a outros e tem pré-requisitos que se o aluno não aprender a matéria anterior, ele não pode entender a próxima. Essa construção do conhecimento e a necessidade de desenvolver habilidades cognitivas para o entendimento de cada vez mais conteúdos abstratos na matemática também se dá em todo o processo de aprendizagem e em níveis diferentes. A BNCC convida ao professor realizar esse processo de ensino-aprendizagem através de etapas a serem concluídas e novas etapas serem alcançadas. Tal característica é conhecida como progressão de aprendizagens. E esse tipo de procedimento é bem semelhante a etapa de decomposição no pensamento computacional e posteriormente para problemas mais complexos, a entrada da etapa de abstração para posteriormente generalizarmos pelos algoritmos.

No Ensino Médio a área de Matemática e suas Tecnologias têm por objetivo aumentar a noção de entendimento da realidade desde sua rotina até mesmo a área tecnológica e científica ou raciocínio lógico. Para que isso aconteça a matemática no Ensino Médio precisa auxiliar na obtenção de capacidades tais como abstração, generalização e argumentação

> [...] a área de Matemática e suas Tecnologias têm a responsabilidade de aproveitar todo o potencial já constituído por esses estudantes no Ensino Fundamental, para promover ações que ampliem o letramento matemático iniciado na etapa anterior. Isso significa que novos conhecimentos específicos devem estimular processos mais elaborados de reflexão e de abstração, que deem sustentação a modos de pensar que permitam aos estudantes formular e resolver problemas em diversos contextos com mais autonomia e recursos matemáticos. (BRASIL, 2018, p. 528-529).

Podemos ver que a matemática tem papel de levar a reflexão e posterior abstração. Tal característica é uma das etapas do desenvolvimento do pensamento computacional. Diante da importância desse tópico, vamos destacar algumas competências da área de Matemática e suas Tecnologias para o Ensino Médio que tem o foco que queremos dar

1 -**Utilizar estratégias, conceitos e procedimentos matemáticos** para **interpretar** situações em **diversos contextos**, sejam atividades **cotidianas**, sejam fatos das Ciências da

Natureza e Humanas, das questões socioeconômicas ou **tecnológicas**, divulgados por **diferentes meios**, de modo a contribuir para uma **formação geral**. Grifo nosso

Quando nos adaptamos a buscar estratégias de solução para diferentes situações ainda que fora da área da matemática, tal busca é aguçada através do pensamento computacional ao analisarmos um problema e buscarmos a melhor solução para tal questão. Isso nos auxilia a ter o hábito de questionar e generalizar o que foi encontrado.

A primeira competência específica pressupõe habilidades que favorecem a interpretação e a compreensão da realidade pelos estudantes. Essa competência traz também a utilização de conceitos matemáticos como suporte para julgamentos bem fundamentados.

2 - Propor ou participar de ações para investigar desafios do mundo contemporâneo e tomar decisões éticas e socialmente responsáveis, com base na análise de problemas sociais, como os voltados às situações de saúde, de sustentabilidade, das **implicações da tecnologia no mundo do trabalho**, entre outras, mobilizando e articulando conceitos, procedimentos e linguagens próprios da Matemática. Grifo nosso

Cada vez mais o mercado de trabalho surge com novas tecnologias e substituição de antigas operações ou profissões humanas que podem ser realizadas por computadores ou robôs. Por outro lado, novas profissões surgem ou a necessidade de se atualizar e se adequar a estas se torna constante e necessária.

Por isso, o pensamento computacional trabalhado desde cedo pode auxiliar ao discente crescer naturalizado com o poder de atitude e de buscar soluções eficientes para problemas colocados à sua frente.

As competências 1 e 2 estão voltadas para a interpretação e para a compreensão de aspectos da realidade.

3 - Utilizar **estratégias, conceitos, definições e procedimentos matemáticos** para **interpretar, construir modelos** e **resolver problemas** em diversos contextos, **analisando a plausibilidade dos resultados e a adequação das soluções propostas, de modo a construir argumentação consistente.**

Ao juntarmos a matemática com o pensamento computacional, temos uma grande ferramenta que pode auxiliar-nos em soluções de problemas mais simples, como um problema de combinatória, mas ele também pode ajudar a criarmos modelos matemáticos que buscam soluções ou melhor entendimentos de fenômenos que acontecem no nosso corpo como por exemplo como funciona o corpo humano, ou qual função ou sistema de funções melhor representa determinado sistema do corpo humano.

E no geral, o pensamento computacional pode ajudar a encontrar uma solução mais rápida para questões cotidianas do que quem nunca estudou essa área de conhecimento aplicada à matemática.

Isso porque quando buscamos soluções, tendemos a fazer comparações com questões que já vimos ou já aprendemos. Por isso, o problema analisado pode remeter a algum problema já visto anteriormente.

4 - **Compreender** e **utilizar**, com flexibilidade e precisão, diferentes registros de **representação matemáticos** (algébrico, geométrico, estatístico, **computacional** etc.) na **busca de solução** e comunicação de **resultados de problemas**. Grifo nosso

Quando buscamos a solução de um problema, queremos aquele que resolva a situação e que seja o melhor possível.

E pelo pensamento computacional, quando analisamos um problema podemos decompor este em etapas menores até chegarmos na solução que melhor se adequa ao nosso problema e que seja ótima, assim como nos problemas de combinatória em otimização.

5 - **Investigar e estabelecer conjecturas a respeito de diferentes conceitos e propriedades matemáticas, empregando estratégias e recursos, como observação de padrões, experimentações e diferentes tecnologias, identificando a necessidade, ou não, de uma demonstração cada vez mais formal na validação das referidas conjecturas.** Grifo nosso

Essa competência tem várias etapas do pensamento computacional existentes tais como a investigação de soluções, busca de estratégias, observação de padrões, experimentação e validação das conjecturas encontradas através de testes computacionais.

Diante dessas competências fica evidente que os alunos precisam desenvolver habilidades relativas aos processos de: investigação, construção de modelos e resolução de problemas.

Esses aspectos podem ser trabalhos em diversas direções e podem ser organizados em torno de um ou mais eixos, conforme a Resolução CNE/CEB nº 3/2018, Art. 12, § 2º:

Dentre as diversas maneiras de trabalharmos, podemos ir na direção da investigação científica. Esta aprofunda as definições científicas e como aplicá-las em problemas cotidianos. Ainda podemos ir na direção de processos criativos. Ela está pautada na produção de conhecimento através de criação de experimentos, modelos ou protótipos para criar processos ou produtos que atendam a demandas pela resolução de problemas de situações cotidianas.

Através de mediação e intervenção sociocultural em situações que demandem os conhecimentos de uma ou mais áreas para soluciona conflitos ou implementar soluções das

mais eficientes possíveis. Outra questão é a promoção do empreendedorismo. Esta demanda a intersecção de diversos conhecimentos para resolvermos problemas que envolvem conhecimentos interdisciplinares. Os eixos destacados anteriormente estão normatizados pela Portaria nº 1.432, de 28 de dezembro de 2018.

Além desses eixos, podemos incentivar a aprendizagem, em particular do pensamento computacional em questões de combinatória através de cursos, estágios, oficinas, trabalho supervisionado, atividades de extensão, pesquisa de campo, iniciação científica, aprendizagem profissional, participação em trabalhos voluntários e demais atividades com intencionalidade pedagógica orientadas pelos docentes. (BRASIL, 2005, p. 13 Artigo 17, parágrafo 13).

Ao trabalharmos o pensamento computacional em combinatória, temos diversos problemas que podem ser resolvidos através de computadores de maneira mais rápido como o problema das quatro cores. Este é classificado como de otimização e pode ser resolvido através da teoria de grafos. Uma resposta possível para esse problema foi proposta e 1852 e apenas através de processos computacionais muito longos e que foi possível chegar a uma resposta (alto nível de complexidade do algoritmo) e isso levou aos matemáticos questionarem se tal demonstração era válida ou não. Aqui podemos destacar qual é a margem de erro aceitável nesse tipo de solução de modo que sua resposta tenha consistência e confiabilidade.

Existe uma diferença entre o pensamento computacional e matemático. Enquanto os matemáticos resolvem no papel e estão limitados pelo tempo e por possíveis erros nos cálculos, no computador ao implementarmos esse algoritmo o tempo de execução e eficácia são muito maiores. Uma das dificuldades que sempre encontramos é de seguir os passos na ordem do algoritmo ou de organizar a ordem nos pensamentos e execuções em nossa mente. Um erro na ordenação do algoritmo, o resultado pode sair errado devido a essa falha humana.

Outra característica é que a condicional na matemática fica implícito, enquanto no pensamento computacional, fica explícito. Já na resolução de problemas por processos recursivos na matemática, temos algumas recursividades que no computador não seriam eficientes, tais como substituir a multiplicação por soma. Temos o ganho da automatização dos processos ao ser implementado no computador e que manualmente não seria possível. Tal processo é importante para observarmos comportamentos e padrões que apenas com o pensamento combinatório demoraríamos muito ou não alcançaríamos.

O aluno teria dificuldade em ordenar em o algoritmo. Ao fazer isso, o aluno tem a oportunidade de representar o pensamento através de uma linguagem a ser implementada computacionalmente e a chance de entender melhor o universo computacional.

3 COMBINATÓRIA: PESQUISAS E ASPECTOS TEÓRICOS

Apresentaremos nesse capítulo uma retomada na classificação dos diferentes tipos de problema de combinatória que abordamos no trabalho, conforme Batanero, Godino e Navarro-Pelayo (1996), seguido de uma revisão da literatura sobre o ensino de combinatória nessa perspectiva mais ampla, e finalizando com uma síntese sobre noções matemáticas sobre técnicas de contagem e grafos, que dão suporte para a resolução e análise dos problemas.

3.1 Categorizações e Revisão da Literatura

Vamos começar essa seção estabelecendo em que termos usamos o termo Combinatória em nossa pesquisa. A concepção de combinatória que usamos engloba cinco categorias de problemas, conforme incialmente classificadas por Batanero, Godino e Navarro-Pelayo (1996). Segundo as autoras, a solução de um problema de natureza combinatorial pode ser classificada em cinco categorias: contagem, existência, classificação, enumeração e otimização.

Antes de apresentarmos e trabalharmos com tais categorias, é importante dizer que um dado problema pode ser categorizado em mais uma delas. As categorias podem ser úteis para estabelecer diferenças. Não devem ser compreendidas como limitantes de produções de significados, mas sim como uma ferramenta para resolução de problemas combinatoriais e que enriquece nossa percepção de problemas de combinatória e como podemos analisá-lo e chegar a novos significados.

Assim, podemos resolver um problema aparentemente de contagem: “Quantos grupos existem? ”, listando todas as possibilidades – o que caracterizaria o processo de enumeração. Um problema de existência pode ser resolvido enumerando-se todas elas. O problema do caixeiro viajante, que ora pode ter uma boa solução por um algoritmo guloso, também pode ser resolvido, dependendo do poder computacional disponível, examinando-se cada caso.

Então, a classificação aqui considerada é usada em nossa investigação para ampliar a visão sobre diferentes problemas de combinatória, bem como, diferentes formas de resolver um mesmo problema. Ou seja, usamos as categorias para organizar e ressaltar características especiais dos problemas, sem ter a intenção de limitar possibilidades de estratégia, resolução e produção de significados, sendo, portanto, um convite à ampliação

da visão sobre a natureza dos problemas de combinatória, e da dinâmica da produção de significados.

A primeira categoria, a de contagem, ocorre quando buscamos a quantidade de soluções existentes de um problema, usando uma ou mais técnicas de contagem para resolver o problema. Uma boa parte da combinatória do Ensino Médio usa problemas dessa categoria, associados ao princípio fundamental da contagem, e aos seus desdobramentos na contagem de ordenamentos (permutações simples, com repetição, circulares, caóticas e os famigerados arranjos) e de agrupamentos (combinações simples, completas, lemas de kaplansky, etc).

A segunda é a de existência, na qual a pergunta "quantos têm? ", dá lugar à pergunta: "será que existe um? ", de modo que o objetivo é avaliar a existência de determinada solução? Os problemas mais famosos são aqueles envolvendo grafos, tais como os de caminhos eulerianos (é possível desenhar a casinha sem retirar o lápis do papel?) e caminhos hamiltonianos (é possível entregar comida para todos os clientes passando em frente à casa de cada um deles exatamente uma vez?).

A terceira é de classificação, quando uma solução encontrada para o problema apresenta lógicas envolvendo a divisão do problema em casos, como quando perguntamos quantos anagramas existem iniciados com uma vogal ou consoante específica e somamos a quantidade de cada caso, e de uma maneira geral quando a solução envolve a união de conjuntos, disjuntos ou não.

A quarta é de enumeração, quando mostramos todas as soluções possíveis, uma a uma para um problema. Nesse caso, um problema que pede que se liste todos os casos (possibilidades) ou uma solução que liste todos os casos para um problema que pede quantas existem, podem ser classificados na mesma categoria.

E a quinta é de otimização, ao apresentarmos a solução ótima, seja de minimização ou maximização de um resultado, tal como o algoritmo de Bellman-Ford realiza em uma das questões desse trabalho.

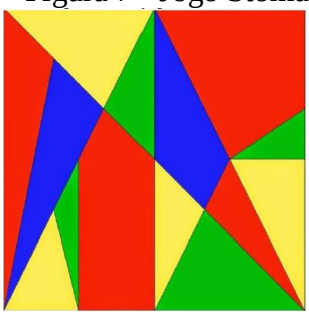
A combinatória foi pela primeira vez trabalhada pelo matemático Arquimedes Silva (2020). Para Pitombeira (1986, p 21) a combinatória é a arte de contar, já para Leibniz (1966, p.4) tinha como o estudo da colocação, ordenação e escolha de objetos. A contagem como objeto de estudo formal não temos precisamente sua origem (BIGGS, 1979, p. 79). Uma das possíveis origens que se tem registro é 1881 por Leon Rodet com um problema de sete casas, cada uma com sete gatos, cada gato que mata sete ratos cada camundongo teria comido sete

cabeças de trigo, cada uma das quais teria produzido sete medidas hekat de grãos. (BIGGS, 1979, p.111).

Já a parte de combinações, arranjos e permutações remontam a povos árabes, hindus e chineses. (WILSON, 2000; BIGGS, 1979).

Dentre os estudiosos, destacamos Reviel Netz, Fábio Acerbi e outros descobriram documentos antigos nos quais Arquimedes trabalhou com um jogo chamado Stomachion, ou também conhecido como jogo de Arquimedes. Tal jogo não foi criado por Arquimedes, mas ele foi quem estudou o formato do jogo e a questão da combinatória. O jogo é formado por várias peças e lembra o tangram. Pode ser jogado para formar quadrados ou formas livres. O jogo era usado por crianças e idosos. (NETZ et al.,2004). Por ter poucos registros sobre esse jogo, ele foi esquecido, mas com os escritos árabes pudemos ter acesso a essa informação. Ainda que combinatória não esteja tão presente nos relatos da história da matemática, sua importância significativa e está presente no PCN e na BNCC.

Figura 7 - Jogo Stomachion



Fonte: http://somainfinita.blogspot.com/2015/01/stomachion.html

Bastos (2020) trabalha com uma escola em Minas no Ensino Médio com uma aplicação de combinatória aplicada em problemas reais envolvendo a disciplina de educação física. Nesse trabalho há uma abordagem de modelagem matemática que é trabalhada de acordo com a realidade dos alunos. Essa etapa de investigação, busca por soluções e tentativa de solucionar através de modelagem matemática envolvendo combinatória é exemplo da aplicação do pensamento computacional seguindo as etapas de análise dos problemas, decomposição, abstração e solução dos problemas.

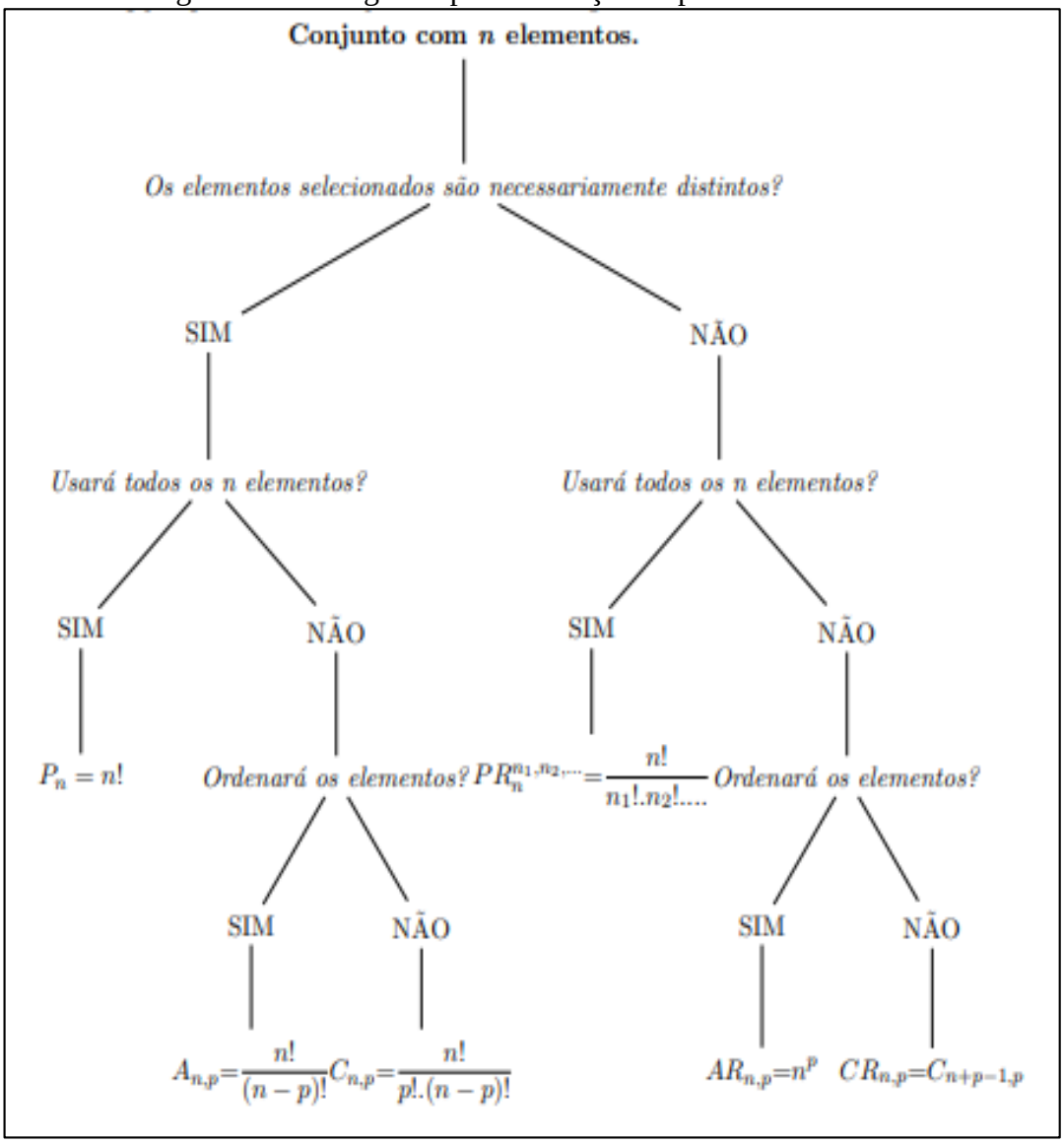
Ribas (2020) nos traz uma aplicação de grafos diretamente no ensino de combinatória. Ele trabalha com questões em que os alunos precisam representar questões de princípio multiplicativo, combinação e arranjo através da representação da árvore de possibilidades. Ele também traz uma aplicação de busca por rotas de menor caminho em torno de um mapa construído pelo autor. Tal atividade é classificada como otimização e foi aborda em Borges e

Muniz (2018).

Pires (2019) exemplifica a ideia da solução de problemas de otimização discreta através do uso do algoritmo do caixeiro viajante aplicado ao problema de colônia de formigas, ou busca da menor rota e compara com soluções tais como as encontradas com o uso do Excel. Essa questão também poderia ter sido abordada com uma visão computacional tal qual apresentamos em nossa dissertação.

Bezerra (2013) apresenta uma possível abordagem didática para o ensino da combinatória. Destacamos esse esquema de seu trabalho:

Figura 8 - Fluxograma para resolução de problemas combinatórios:



Fonte: Bezerra (2013)

Percebemos um esquema em forma de fluxograma para o aluno seguir e saber quando usar arranjo combinação, permutação ou permutação com repetição. Normalmente os docentes

ao abordar esse esquema não o desenha, mas explica que perguntas devem ser feitas. Esse fluxograma nos remete ao pensamento computacional na etapa de decomposição dos problemas em situações menores. Na dissertação de Cardoso (2013) a aplicação de lógica nas questões de combinatória podem auxiliar na implementação dos algoritmos nas linguagens de programação, por exemplo. Schmidke (2013) apresenta um software para auxiliar no aprendizado de combinatória de maneira discreta e finita. Nessa época percebemos a busca de pesquisas por implementar a tecnologia na aprendizagem da matemática. Na dissertação de Silva (2013) existe a busca da solução de problemas de combinatória através da análise dos casos pelo método da exaustão. Explicitamos a tentativa de explicar alguns problemas dessa natureza através da solução generalizando casos particulares. Se olharmos pelo pensamento computacional, ela se enquadra na etapa da abstração.

Trabalhos tais como Farias (2013) com uma sequência didática de questões de combinatória resolvidas, Gonçalves (2014), Maciel (2015) e Américo (2013) com resoluções de questões da obmep podem ser fontes de questões para serem classificadas e resolvidas através do pensamento computacional. Ou o trabalho de (2013) com contexto de telefonia. Miotto (2014) apresenta algumas resoluções de combinatória, as quais podemos considerar com características comuns entre esta área e o pensamento computacional: recursividade, abstração e decomposição de casos.

Assim como vimos atividades lúdicas sobre pensamento computacional, como as atividades disponibilizadas no site do CSunplugged, o trabalho do Jacinto (2015) apresenta no mesmo molde atividades lúdicas para o ensino de combinatória e que podem ser comparadas e adaptadas ao pensamento computacional. Oliveira (2015) trata a combinatória em seu processo recursivo ao longo de sua dissertação e este é um ponto em comum com o pensamento computacional e que mostra o potencial de analisarmos essa área de matemática devido à sua natureza à luz do pensamento computacional.

Batanero, Godino e Navarro-Pelayo (KAPUR, 1970) dentre as vantagens de ensinar combinatória no Ensino Médio, destacamos: ela não depende do cálculo e isso nos permite decidirmos quais problemas aplicar nos diversos níveis de ensino, podemos também conversar sobre problemas ainda sem solução e assim percebam a importância de aprender novos conceitos matemáticos; podemos trabalhar questões tais como enumeração, realização de conjecturas, generalização e otimização e sistemas de pensamentos (destacamos, enumeração e generalização e sistemas de pensamentos como etapas em comum com o pensamento computacional); auxilia o desenvolvimento de vários conceitos como por exemplo a aplicação, a ordem e as relações de equivalência, função, programa, conjunto, subconjunto e produto

cartesiano; possui aplicações em diversas áreas tai como Química, Biologia, Física, Comunicação, Probabilidade, Teoria dos números, gráficos e com o auxílio do pensamento computacional podem nos auxiliar a resolver problemas cotidianos.

Internacionalmente pesquisas na área de combinatória são mais focadas nas questões de combinatória de contagem (KENNETH (2008); NEVES; BOLINHAS (2016); BORBA; ROCHA; AZEVEDO (2015); PESSOA; BORBA (2012)). As pesquisas na área de otimização e existências ainda são poucas em relação as de contagem (MUNIZ (2007); MALTA (2008); DEGGERONI (2010); BORGES, MUNIZ (2018A, 2018B, 2020)).

A combinatória tem grande intersecção com o pensamento computacional devido à sua potencialidade. E para Muniz (2007), o avanço tecnológico, em particular de computadores e celulares, transformaram nossos hábitos de tal maneira que rotinas de processos sequenciais se tornaram padrão. Por isso, o objetivo desse trabalho é uma proposta de conjunto de tarefas didáticas na área de combinatória através de uma visão de pensamento computacional e implementada em linguagem c++ para o ensino médio. Isto propicia uma chance de relacionar o conteúdo matemático trabalhado no ensino médio com temas pertinentes da ciência e da tecnologia.

Além disso, nas Orientações Curriculares para o Ensino Médio de acordo com o MEC recomenda-se utilizar a matemática para o entendimento da tecnologia. Ao realizar-se esse trabalho, direciona-se para áreas da matemática que necessitam ser ensinadas para pessoas que precisam lidar com algoritmos, lidar com minimização de custos e maximização de recursos cada vez menores e relacionar problemas de combinatória via pensamento computacional.

Dentre as estruturas matemáticas que podemos relacionar com a área de combinatória, pensamento computacional e que até compõe uma das disciplinas de quem cursa Ciência da Computação e uma estrutura de algoritmo temos os Grafos. Ele está presente nos currículos de países tais como Portugal, Espanha e França e no Brasil temos habilidades dentre os documentos norteadores do Ensino Básico a demanda de construção de propostas de intervenção na realidade e a tomada de decisão em problemas envolvendo matemática, incluindo os de natureza combinatória em Parâmetros Curriculares Nacionais (PCN), as Orientações Curriculares para o Ensino Médio (OECM) e a Base Nacional Comum Curricular (BNCC). No ensino português destacamos Lovász, Pelikán e Vesztergombi (2003).

Identificamos que abordagens didáticas locais na Educação Básica por meio de problemas de combinatória de diferentes tipos, como os abordados aqui neste trabalho, vem sendo implementadas e pesquisadas no Brasil. Tais pesquisas têm mostrado diferentes formas

de abordar combinatória; diferentes estratégias apresentadas pelos estudantes; a importância de variados tipos de problemas de combinatória no desenvolvimento de habilidades matemáticas, tais como divisão do problema em casos, construção de estratégias para além das perguntas: “a ordem importa?”, “ é para somar ou multiplicar”, “é arranjo ou combinação?”. Tais pesquisas trazem evidências de que estudantes ampliam suas habilidades na resolução de problemas e tomada de decisão, como se pode ver em Muniz; Marinho et al (2005); Souza (2015); Jurkiewicz e Muniz (2008), Ferreira e Lozano (2009) e Guedes (2014).

Alguns trabalhos se destacam na busca de uma metodologia de ensino de combinatória no ensino médio Gualandi (2012); Búrigo (2012); Silva (2009); Francisco (2008). Me documentos norteadores da Educação Básica, tais como PCN + (BRASIL, 2006), BNCC (BRASIL, 2018) e Matriz de Referência (INEP 2010) existem orientações e sugestões sobre o ensino de combinatória, algumas das quais na direção de abordagens de problemas em grafos, ainda que muito aquém das suas potencialidades, como exploradas neste trabalho.

A combinatória como área de matemática discreta para solução de problemas e tomada de decisão e em particular atrelada a Teoria de Grafos e ao pensamento computacional pode ser encontrada em variados trabalhos, tais como (PEZETA (2013); SA (2016); TROCADO (2017); FERNANDES (2013); MARTIN (2013); SERRAZINA E OLIVEIRA (2001); PETERS (2015); WEINBERG, WIESNER E FUKAWA-CONNELLY (2016); MÜLLER (2015); MESQUITA (2015); TEIXEIRA (2015); SEIBERT (2014); SOUZA (2014); OLGIN (2015); SILVA (2016); MALTA (2008); GUALANDI (2012); FRANCE (2013); LUMIÈRE (FRANCE, 2014); FERREIRA (2009) e NOGUEIRA (2015)). Destacamos as Diretrizes Curriculares para o Ensino de Matemática proposto pela Sociedade Brasileira de Matemática, Batty (2004), Caleiro (2012) inspirado em Batty (2004) e que trabalha interdisciplinaridade já no caminho para o que vemos nos livros do PNLD 2021.

Dentre pesquisas internacionais destacamos Patiño Avendaño (2013); Fleury et al. (2004); Piaget e Inhelder (1951 *apud* ESPINOZA; ROA, 2014); Espinoza e Roa (2014) e currículos de outros países que já trabalham esse temática tal como em Diseño curricular nueva escuela secundaria de la Ciudad de Buenos Aires ciclo básico (BUENOS AIRES, 2015a); Diseño curricular nueva escuela secundaria de la Ciudad de Buenos Aires: ciclo orientado del bachillerato; formación general (BUENOS AIRES, 2015b); Cartier (2008); Modeste (2012); Quatre documents sur les grafes en terminale ES (FRANCE, 2013) e Notion de Graphe (FRANCE, 2002).

Dambros (2015) apresenta um conjunto de atividades de combinatória resolvidas de

diversas áreas e em particular com aplicações na Teoria de Grafos. Destacamos também o uso de um software em flash desenvolvido para o auxílio da aprendizagem do princípio fundamental da contagem. É trabalhado o termo de linguagem de programação: o flash. Isso nos mostra já a existência do pensamento computacional de maneira muito ativa nesse campo da matemática. Dentre os conteúdos abordados nesse trabalho, destacamos alguns aspectos que o uso da informática no ensino pode trazer como vantagem para a aprendizagem:

- ✓ Levar o estudante a experimentar o sucesso e aprender a lidar com o insucesso, refazendo até acertar, valorizando o erro como etapa necessária para o desenvolvimento humano;
- ✓ Permitir que o estudante construa seu próprio projeto refletindo suas vivências;
- ✓ Apoiar o desenvolvimento cognitivo;
- ✓ Promover o pensar sobre o próprio pensar;
- ✓ Ampliar a capacidade de aprendizagem;
- ✓ Facilitar a integração interdisciplinar;
- ✓ Desenvolver diversas e distintas linguagens;
- ✓ Fazer simulações;
- ✓ Realizar induções de conceitos abstratos, a partir de exemplos visuais e dinamicamente alteráveis;
- ✓ Apresenta maior motivação dos estudantes, pois representam desafios grandes e motivadores;
- ✓ Permitir melhor representação e manipulação de alguns objetos do que outras mídias;
- ✓ Permitir diferentes análises, tanto macroscópicas como microscópicas de todos os objetos e fatos;
- ✓ Permitir que o estudante desenvolva seu trabalho conforme seu ritmo;
- ✓ Permitir que o estudante se comunique e troque ideias com pessoas de toda parte do mundo, de diferentes línguas e costumes, inclusive instantaneamente;
- ✓ Tornar a aula mais agradável, divertida, dinâmica e produtiva;
- ✓ Fazer provas de teoremas e comprovação de resultados por meio do uso do computador, assim como testar conjeturas;
- ✓ Modernizar o ambiente de ensino-aprendizagem, desde a característica física com a presença dos computadores, como a mental.

Oliveira (2017) apresenta um recorte sobre uma experiência realizada em escola no Rio de Janeiro (Itaperuna) sobre as classificações possíveis aos problemas de combinatória e como alunos de uma escola se comportam diante de tal classificação ao responderem um questionário. Braga (2017) resolve um conjunto de questões de combinatória com as orientações de como resolver cada uma delas e os passos a seguir. Fica evidente o caráter de passos que a natureza desse tema possui e o potencial de trabalharmos o pensamento computacional nele. Silva (2020) também apresenta resolução de problemas dessa área da matemática e Taveira (2019) seleciona questões de problemas de otimização para apresentar como um conjunto de atividades, os quais demandam recursos do pensamento computacional, tais como a decomposição e a

recursividade.

Machado (2019) apresenta a importância do pensamento computacional na educação básica, analisa questões da BNCC em relação a essa temática, aborda currículos de outros países tais como Austrália e Estados Unidos e propõe um modelo de aula maker para o 6 ano pautado no pensamento computacional.

Polya (1995 *apud* SILVA, 2020) define que para resolver um problema precisamos buscar uma ação adequada para encontrar a resposta de um objetivo específico. Este pode ser que não tenha uma solução atingível e nesse caso teríamos um obstáculo a superar. Já para Van de Walle (2010 *apud* SILVA, 2020) um problema é definido como qualquer tarefa ou atividade para a qual não se tem **métodos** ou **regras prescritas** ou memorizadas, nem a percepção de que haja um método específico para chegar à solução correta." Em ambas as definições temos traços que o pensamento computacional pode auxiliar-nos na busca por soluções ou métodos de solução.

Gostaríamos de finalizar essa seção explorando um recente trabalho que faz conexões entre o ensino de combinatória e o pensamento computacional.

Dantas (2019) reflete sobre a diferença entre o pensamento combinatório e computacional, além de trazer à tona características em comum às duas áreas, bem como termos utilizados em ambas, tais como abstração e algoritmo, mas que possuem significados distintos. E também destaca que ambas as áreas podem se complementar das seguintes maneiras:

> ✓ possibilitar a produção de significados além daqueles que os materiais didáticos habituais possibilitam;
>
> ✓ integrar aritmética, álgebra, geometria e métodos computacionais na resolução de problemas;
>
> ✓ trazer a experimentação para o centro da atividade de produção de conhecimento matemático e de outras áreas do conhecimento humano (quando possível);
>
> ✓ relativizar a importância de algumas atividades matemáticas, por exemplo, tirar do cenário principal os processos puramente algébricos e pôr em cena a interação numérica, a visualização geométrica dinâmica e a produção de automações.

3.2 Combinatória e Técnicas de Contagem

Nessa seção trabalhamos algumas técnicas de contagem e noções de combinatória, na perspectiva usada em Morgado et al (2004). A primeira noção e contato com essa área é através da contagem, normalmente ainda quando jovens no processo natural e intuitivo de contagem. Nesse processo as primeiras técnicas que utilizamos é de agrupamentos de elementos que possuem uma propriedade em comum, daí surge a ideia de conjunto. E concomitantemente a noção de somar e posterior multiplicar elementos.

Antes de definirmos os Princípios da Adição e Multiplicação precisamos relembrar alguns conceitos da Teoria de Conjuntos. O conjunto intuitivamente nos remete a noção de unir objetos. Mas, para isso, na matemática, o conjunto representa a união de objetos, os quais definimos como **elementos**, que possuem certa característica em comum, a quem definimos de **propriedade**.

Assim, ao reunirmos elementos que possuem certa propriedade em comum, formamos um conjunto. Ou seja, um **conjunto** pode ser definido como a união de elementos que possuem determinada propriedade. Dessa maneira, representamos o conjunto pela letra maiúscula e seus elementos com letra minúscula e dentre chaves. Por exemplo, o conjunto das letras da palavra matemática pode ser representado por:

$$\mathrm{M} = \{m, a, t, e, a, i, c\}$$

Podemos formar grupos menores desse conjunto, como por exemplo o conjunto

$$\mathrm{L} = \{m, t, c\}$$

Nesse caso, L é definido como **subconjunto** de M.

Ao reunirmos todos os elementos de um ou mais conjuntos, trabalhamos a noção de **união** e que é representado pela notação ∪. Ao reunirmos elementos de dois ou mais conjuntos, o total de elementos dessa união é dado pela soma da quantidade de elementos em cada um deles. O total de elementos pertencentes a um conjunto é definido como **cardinalidade** e é representado pelo símbolo #. No caso do exemplo do conjunto M, sua cardinalidade é dada por #(M) = 7, ou seja, esse conjunto tem 7 elementos.

Passemos agora a dois princípios importantes a partir dos quais todas as ténicas de contagem usadas nesse trabalho derivam.

O primeiro é o Princípio Aditivo. O **Princípio da Adição** pode ser definido da seguinte maneira:

Se A e B são dois conjuntos disjuntos, com p e q elementos, respectivamente, então AUB possui p + q elementos.

Além deste, o Princípio **Fundamental da Enumeração** ou **Princípio da Multiplicação** nos diz que:

Se uma decisão d_1 pode ser tomada de x maneiras e se, uma vez tomada de y maneiras então o número de maneiras de se tomarem as decisões d_1 e d_2 é xy.

Se dentre essas possíveis enumerações pensarmos em ordená-las, temos a noção de permutação simples. Nesse caso, para permutarmos n objetos distintos teremos n.(n – 1).(n – 2). ... .1 = n!

Cada ordenação dos n objetos é chamada de permutação simples de n objetos e o número de permutações simples de n objetos distintos é o número de permutações simples de n objetos distintos é representado por P_n. Assim, $P_n = n!$ e $P_0 = 1$.

A noção de combinar vem da ação de escolher uma quantidade dentre um total. Em outras palavras, diante de n objetos, caso queiramos escolher uma quantidade p estamos diante de uma combinação de n escolhemos p. Representamos por $C_{n,p}$ e seu cálculo é feito através de:

$C_{n,p} = \frac{n!}{p!(n-p)!}$, com $0 \leq p \leq n$.

Além disso, além de combinarmos ou permutarmos de maneira simples, podemos ter problemas em que precisamos permutar em situação circular. Nesse caso, se tivermos n elementos temos n permutações que se configuram com uma única roda, pois podemos "girar" a mesma configuração circular n vezes, mantendo a posição relativa entre os temos. Assim, o número de permutações circulares, dispondo de n elementos distintos dados, chamada de PCn, é igual $PC_n = \frac{n!}{n} = (n - 1)!$.

Ainda podemos ter de resolver questões que envolvem letras repetidas em palavras e precisamos descontar os casos que aparecem duas vezes. Por exemplo, na palavra BATATA,

temos as letras A e T que se repetem. Se então trocarmos as letras T entre si e as letras A entre si, ainda teremos a mesma configuração de palíndromo. Para evitar que contemos ou enumeremos os mesmos casos, trabalhamos com a noção de permutação com repetição. E para contarmos a quantidade de elementos que se repetem utilizamos a seguinte fórmula: dado um conjunto com n elementos nos quais temos π elementos iguais a, α elementos iguais a b, μ elementos iguais a c, ..., Ω elementos iguais a a_i e β elementos iguais a a_j, tal que $\pi + \alpha + \mu + \Omega + \beta \leq n$ então teremos

$$C_n^{\pi,\alpha,\mu,\ldots,\Omega,\beta} = \frac{n!}{\pi!\,\alpha!\,\mu!\,\ldots\,\Omega!\,\beta!}$$

maneiras de permutarmos e denominamos de permutações com repetição.

Outro conceito importante é o de combinação completa. Nesse caso podemos usar a notação de CR_n^{p} e pode ser interpretado como o número de modos de selecionarmos p objetos, distintos ou não no total de n elementos dados**.** Ou ainda como sendo o número de soluções da equação: $x_1 + x_2 + \ldots + x_{n-1} + x_n = p$.

Ainda que não seja o foco vale ressaltarmos o Princípio da Inclusão e Exclusão. Nele, devemos levar em consideração os casos em que temos em comum entre dois ou mais conjuntos dados. Se tivermos dois conjuntos A e B e cada um com cardinalidade #A e #B, o total de elementos distintos se dá da seguinte maneira:

$$\#(A \cup B) = \#A + \#B - \#(A \cap B)$$

Ou seja trabalhamos com a mesma ideia de contagem de elementos de dois ou mais conjuntos. Precisamos retirar os casos em comum a fim de que não os contemos mais de uma vez.

Outro método de contagem importante e que está presente em uma das questões apresentadas é o da permutação caótica, ou desordenamento. Nela questionamos de quantas maneiras distintas podemos permutar n elementos de modo que cada elemento não ocupe seu local de origem.

Queremos calcular o número de elementos do conjunto Ω das permutações de (1, 2, .., n) que pertencem exatamente a zero dos conjuntos A_1, A_2, ..., A_{n-1}, A_n. Temos:

$S_0 = \#(\Omega) = n!$

$$S_1 = \sum_{i=1}^{n} \#(A_i) = i = 1n(n-1)! = n.(n-1)! = n!$$

$$S_2 = \sum_{1 \le i < j \le n} \#(A_i \cap A_j) = \sum_{1 \le i < j \le n} (n-2)! = C_n^2.(n-2)! = \frac{n!}{2!};$$

$$S_3 = \sum_{1 \le i < j < k \le n} \#(A_i \cap A_j \cap A_k) = \sum_{1 \le i < j < k \le n} (n-3)! = C_n^3.(n-3)! = \frac{n!}{3!};$$

...

$$S_n = C_n^n.(n-n)! = \frac{n!}{n!}$$

O número de elementos do conjunto **Ω** das permutações de (1, 2, .., n) que pertencem exatamento a zero dos conjuntos A_1, A_2, ..., A_{n-1}, A_n é:

$$a_0 = \sum_{k=0}^{n=0} (-1)^k C_{0+k}^k S_{0+k}$$

$$\sum_{k=0}^{n} (-1)^k S_k$$

$$= S_0 - S_1 + S_2 - S_3 + \ldots + (-1)^n S_n$$

$$= n! - n! + \frac{n!}{2!} - \frac{n!}{3!} + \ldots + (-1)^n \frac{n!}{n!}$$

$$= n!\left[\frac{1}{0!} - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \ldots + \frac{(-1)^n}{n!}\right]$$

Assim, o número de permutações caóticas de (1,2, ..., n) é

$$D_n = n!\left[\frac{1}{0!} - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \ldots + \frac{(-1)^n}{n!}\right]$$

Ao realizarmos as devidas aproximações chegamos a

$$D_n = \frac{n!}{e}$$

Para n = 1 e n = 2 a afirmação é verdadeira, mas para n > 2 temos que provar. Sabemso que

$$e^x = \frac{1}{0!} + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots.$$

Logo

$$e^{-1} = \frac{1}{0!} - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots.$$

$$\left| D_n - \frac{n!}{e} \right| < \frac{1}{2}$$

E portanto D_n é o inteiro mais próximo de $\frac{1}{2}$

Nesse tipo de questão trabalhamos diretamente com a noção de recursividade e que é um ponto de intersecção entre o pensamento combinatório e computacional. Apresentamos

A recursividade dessa questão poderá ser observada ao longo de uma das atividades resolvidas de maneira recursiva e que evidencia o potencial da natureza combinatorial para ser analisado via pensamento computacional.

Outros métodos de contagem podem ser vistos em Morgado (ANO), mas abordamos apenas os anteriores com foco nas questões trabalhadas ao longo dessa dissertação e qual tipo de teoria era necessária para resolver tais questões. Outros métodos e propriedades podem ser integrantes de futuros trabalhos e pesquisas.

3.3 Combinatória e Grafos

Nesta seção faremos uma revisão de definições de grafos necessários para o entendimento e desenvolvimento das questões de combinatória de existência e otimização, que envolvem a Teoria de Grafos, conforme apresentamos em Borges (2018)

Definição 1: Um grafo G = (V, E, Ψ_G) é uma estrutura Matemática composta por um conjunto V de vértices, um conjunto E de arestas e uma função de incidência Ψ_G: E → V x V que associa a cada aresta e ∈ E um par não ordenado {u,v} de vértices de V.

Definição 2: Se $\Psi_G(e) = \{u,v\}$, os vértices *u* e *v* são chamados os **extremos** da aresta *e*. Assim, dizemos que os vértices *u* e *v* são **adjacentes** ou **vizinhos** e que a aresta *e* é **incidente** sobre *u* e *v*. Dizemos também que a aresta *e* **liga** os vértices *u* e *v*. Duas arestas que repartem um extremo são ditas **adjacentes**.

Definição 3: Se a aresta *e* possui extremos iguais, é chamada **laço**, isto é, $\Psi_G(e) = \{u,v\}$ para algum $u \in V$.

Definição 4: Caso dois vértices sejam ligados por mais de uma aresta, essas arestas são chamadas **paralelas**, ou seja, e_1 e e_2 são arestas paralelas se, e somente se, $\Psi_G(e_1) = \Psi_G(e_2)$.

Definição 5: Um grafo $G = (V, E, \Psi_G)$ é dito **simples** quando não apresenta nem laços e nem arestas paralelas. Caso contrário, ele é denominado **multigrafo**.

Definição 6: Dado um grafo $G = (V, E)$, dizemos que o número de vértices n é a **ordem (grau)** de G representado por $d(G)$, enquanto o número de arestas m é o **tamanho** de G.

Definição 7: Grafo completo é um grafo simples com arestas entre quaisquer pares de vértices. Representamos um grafo completo com n vértices por K_n. O tamanho de um grafo completo K_n é $m = \frac{n(n-1)}{2}$.

Definição 8: Um grafo H é um **subgrafo** do grafo G se $V(H) \subset V(G)$ e $E(H) \subset E(G)$.

Definição 9: Um **caminho** ou **percurso** $P_n = (v_0, v_1, \cdots, v_{n-1})$, ou também representado por $v_0 \rightarrow v_1 \rightarrow, \cdots \rightarrow v_n$ é uma sequência de vértices distintos em que v_i é adjacente a v_{i+1} para todo $i \in \{0,1, \cdots, n-2\}$ .O comprimento do caminho P_n é dado pelo número de arestas de P_n, ou seja, $n - 1$. O número de arestas é o tamanho do percurso. Um percurso onde as arestas são distintas é uma **trilha** e quando os vértices são todos distintos um **caminho**. Uma trilha ou um caminho é fechado se $v_0 = v_k$ e um caminho fechado contendo pelo menos uma aresta é um ciclo.

Definição 10: Um **ciclo** C_n é um caminho em que $v_0 = v_{n-1}$. O comprimento do ciclo C_n é n.

Definição 11: Um grafo é **conexo** se dados $u, v \in V(G)$ quaisquer, existe um caminho ligando u a v.

Se existe um par de vértices que não possuem nenhum caminho em G ligando-os, então G é **desconexo**.

Definição 12: Dado um grafo $G = (V, E)$, um **emparelhamento** em G é um subconjunto $M \in E$ composto por arestas duas a duas não adjacentes. Os vértices incidentes às arestas de um emparelhamento M são ditos **M-saturados**, enquanto os demais vértices são ditos **M-não-saturados**. Um emparelhamento é dito **perfeito** quando satura todos os vértices do grafo.

Definição 13: Um grafo é k-**regular** se todos os seus vértices possuem o mesmo grau k.

Definição 14: Uma sequência de vértices $(v_0, v_1, v\_2, \cdots, v_{n-1})$ não necessariamente distintos, tal que v_i é adjacente à v_{n+1} para todo $i \in \{0, \cdots, n-2\}$ e $v_0 = v_{n-1}$ é chamada **circuito**.

Definição 15: Um circuito é dito **euleriano** se ele contém todas as arestas de um grafo.

Definição 16: Um grafo que contém um circuito euleriano é um **grafo euleriano**.

Definição 17: Uma **k-coloração** de vértices de um grafo $G = (V, E)$ é uma função $c: V \rightarrow S$, na qual $|S| = k$, ou seja, a cada vértice de G é atribuída uma cor dentre um conjunto de k cores. Entende-se por **cor** um elemento de S. Usualmente trabalhamos com o conjunto dos k primeiros números naturais positivos, isto é, $S = \{1,2, 3, ..., k\}$.

Definição 18: O conjunto de vértices assinalados com uma mesma cor é denominado uma **classe de cor**.

Definição 19: Uma **k-coloração** é dita **própria** se vértices adjacentes possuem cores distintas, isto é, $c(v) \neq c(w)$ se v e w são adjacentes.

Definição 20: O **número cromático** $X(G)$ é o menor valor de k tal que G é k-colorível. Um grafo G tal que $X(G) = k$ é dito **k-cromático**.

Definição 21: Uma **k-coloração** de um grafo **k-cromático** é dita uma **coloração ótima**.

Definição 22: Um grafo conexo G é **bipartido** se, e somente se, $X(G) = 2$.

Se o conjunto de vértices de um grafo G puder ser agrupado em dois conjuntos disjuntos de vértices, onde cada aresta liga o vértice de um conjunto ao vértice de outro conjunto, então o grafo é denominado **bipartido**.

Teorema 1: $X(C_n) = 2$ se n é par e $X(C_n) = 3$ se n é ímpar.

Definição 23: Uma **k-coloração** de arestas de um grafo $G = (V, E)$ é uma função $c: E \rightarrow S$, na qual $|S| = k$, ou seja, a cada aresta de G é atribuída uma cor dentre um conjunto de k cores. Entende-se por *cor* um elemento de S. Usualmente trabalhamos com o conjunto dos k primeiros números naturais positivos, isto é, $S = \{1,2,3 \cdots,k\}$.

Definição 24: O conjunto de arestas assinaladas com uma mesma cor é denominado uma **classe de cor.**

Definição 25: Uma k-aresta-coloração é dita **própria** se as arestas adjacentes possuem cores distintas, isto é, $c(e_i) \neq c(e_j)$ se e_i e e_j são adjacentes.

Definição 26: Um grafo G é **k-aresta-colorível** se admite uma **k-aresta-coloração própria**.

Definição 27: O **número cromático em arestas** $X'(G)$ é o menor valor de k tal que G é ***k-aresta-colorível***. O número $X'(G)$ também é conhecido como o **índice cromático de G**.

Definição 28: Um grafo G tal que $X'(G) = k$ é dito **k-aresta-cromático**. Uma k-aresta-coloração de um grafo k-aresta-cromático é dita uma **coloração de arestas ótima**.

Definição 29: Uma **ponte** de *G* é uma aresta cuja remoção desconecta *G*.

Definição 30: Uma **k-coloração total** de um grafo $G = (V, E)$ é uma função de modo que $c'': V \cup E \rightarrow \{1, 2, \cdots, k\}$ que atribui a cada vértice $v \in V$ e a cada aresta $e \in E$ uma cor representada no conjunto $\{1, 2, \cdots, k\}$ de modo que vértices ad jacentes, arestas adjacentes e vértices e arestas incidentes possuam cores distintas.

Definição 31: O **número cromático total** $X''(G)$ de um grafo é o número mínimo de cores de uma coloração total de *G*.

Lema 1: Se um grafo é 2-regular, então $X''(G) \leq 4$.

Lema 2: Se *G* é um grafo cúbico livre de pontes, então $X''(G) \leq 5$.

Teorema 2: Se *n* é par, então $X'(C_n) = 2$ e se *n* é ímpar, então $X'(C_n) = 3$.

Lema 3: Se *G* é um grafo que só apresenta vértices de grau dois ou três e não possui pontes, então $X''(G) \leq 5$.

Lema 4: Se um grafo tem no máximo grau 3 e não possui pontes, então $X''(G) \leq 5$.

Teorema 3: Se *G* é um grafo com no máximo grau três, então $X''(G) \leq 5$.

Teorema 4: Para todo grafo *G* com *n* vértices $v_1, \ldots, v_n$ e *m* arestas, $\sum_{i=1}^{n} d(v_i) = 2m$.

Prova: Na soma dos graus dos vértices de G, contamos cada aresta duas vezes. ■

Corolário 1: Todo grafo tem número par de vértices de grau ímpar.

A Teoria de Grafos surgiu através de Euler na tentativa de resolver um problema de transporte entre cidades de modo a não repetir pontes que interligavam tais cidades. E para resolver tal questão, Euler acabou formalizando uma nova área da matemática, que é a Teoria dos Grafos. Nela, um grafo conexo G é dito euleriano se existir uma trilha fechada contendo todas as arestas de G. Essa trilha é denominada trilha euleriana. Note que esta definição requer que cada aresta seja utilizada no percurso exatamente uma vez. Um grafo não euleriano G é dito semi-euleriano se existir uma trilha contendo cada aresta de G.

Problemas em grafos eulerianos aparecem quando se quer saber se um diagrama pode ser desenhado sem levantar o lápis do papel e sem repetir nenhuma linha. O nome “euleriano” vem do fato de Euler ter sido o primeiro a resolver o famoso problema das Pontes de Konigsberg, que perguntava se seria possível atravessar cada uma das sete pontes exatamente uma vez e retornar ao ponto de partida. Esses conceitos aparecerão nas atividades desenvolvidas nessa dissertação.

Lema 5: Se *G* é um grafo em que o grau de cada vértice é pelo menos 2 então G contém um ciclo.

Teorema 5: Euler (1736): Um grafo conexo *G* é euleriano se, e somente se, todos os vértices têm grau par.

Prova: Suponha que *P* é uma trilha euleriana de *G*. Sempre que *P* passa por um vértice, há uma contribuição de 2 do grau desse vértice. Como cada aresta aparece exatamente uma vez em *P*, cada vértice deve ter grau par. Reciprocamente, a prova é por indução no número de arestas de *G*. Suponha que o grau de cada vértice é par. Como *G* é conexo, cada vértice tem grau pelo menos 2 e pelo Lema 5, *G* tem um ciclo *C*. Se *C* contém todas as arestas de *G*, a prova está completa. Se não, vamos retirar de *G* as arestas de *C* para formar um novo grafo *H* com um número menor de arestas e tal que cada vértice ainda tem o grau par. Pela hipótese de indução, cada componente de *H* tem uma trilha euleriana. Como cada componente de *H* tem pelo menos um vértice em comum com *C*, pela conexidade, obtemos a trilha euleriana de *G*, seguindo as arestas de *C* até um vértice não isolado de *H* ser alcançado, percorrendo a trilha euleriana da componente de *H* que contém tal vértice e, em seguida, continuando ao longo das arestas de *C* até alcançarmos um vértice de outra componente de *H* e, assim por diante. O processo todo termina quando voltamos ao vértice inicial. ■

Corolário 2: Um grafo conexo é Euleriano se, e somente se, o seu conjunto de arestas pode ser dividido em ciclos disjuntos.

Outro conceito importante é o ciclo Hamiltoniano. O nome ciclo hamiltoniano veio de Sir William Hamilton investigar sua existência no grafo dodecaedro, apesar de um problema mais geral ter sido estudado anteriormente por T. P. Kirkman

Teorema 6: Ore (1960): Se *G* é um grafo simples com $n \geq 3$ vértices e $d(v)+d(w) \geq n$, para quaisquer pares de vértices não adjacentes *v* e *w*, então *G* é hamiltoniano.

Definição 32: Árvore é um grafo *G* de *n* vértices sem ciclos com $n - 1$ arestas.

Definição 33: Folhas são vértices de grau um.

Teorema 7: Seja T um grafo com n vértices. Então as seguintes proposições são equivalentes.

1. T é uma ´arvore;

2. T não contém ciclos e possui n − 1 arestas;

3. T é conexo e possui n − 1 arestas;

4. T é conexo e cada aresta é uma ponte;

5. Quaisquer dois vértices de T estão conectados por exatamente um caminho;

6. T não contém ciclos, mas a adição de qualquer aresta cria exatamente um ciclo.

Definição 34: Grafo dirigido (direcionado) é quando as arestas possuem sentido e direção.

Definição 35: Peso de uma aresta é o valor atribuído a uma aresta de um grafo *G*.

Colocamos nesse capítulo a definição de árvore, pois ela está presente na linguagem de programação e é uma estrutura algorítmica usada para resolução de problemas.

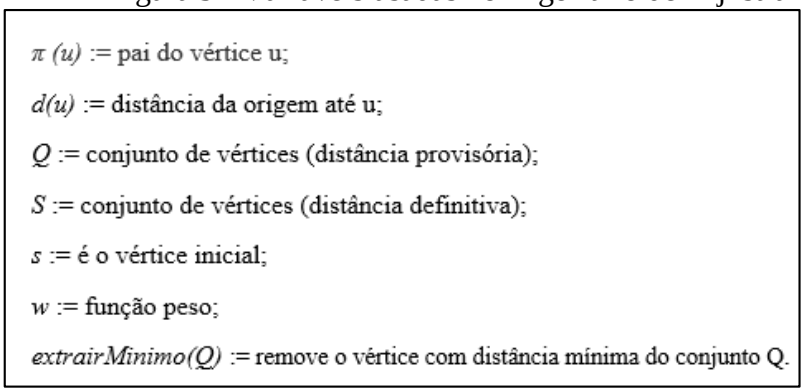
Com o avanço tecnológico, os cálculos antes cansativos e inviáveis de serem realizados manualmente receberam um grande aliado: a computação. O avanço computacional e de sua capacidade de cálculo auxilia muito em problemas que antes não eram possíveis de serem calculados ou estimados através de processos de cálculos matemáticos tradicionais.

3.4 Grafos e Algoritmos

Algoritmo é um conjunto de tarefas e ações a serem realizadas a fim de resolver um problema. Diante disso, apresentamos nesse capítulo alguns algoritmos de otimização que foram utilizados na elaboração e resolução de questões de combinatória, em particular de Borges, Muniz (2018).

Dentre os algoritmos trabalhados destacamos o de Dijkstra (1959). Destacamos na figura a seguir as notações utilizadas no algoritmo e em seguida o algoritmo em si escrito em pseudocódigo. Ele é utilizado para encontrar valores de máximo ou mínimo nas resoluções de problema.

Figura 9 - Variáveis usadas no Algoritmo de Dijkstra



Fonte: Adaptado de MACEDO, VITALI (2011).

Figura 10 – Algoritmo de Dijkstra

```
Dijkstra(G = (V,A), w, s) {
    para cada v ∈ V{
        d(v) ← ∞
        π (v) ← NULO
    }
    d(s) ← 0
    S ← {}
    Q ← V
    Enquanto |Q| ≠ 0{
        u ← extrairMinimo(Q)
        S ← S U {u}
        para cada v ∈ Adj(u)
            se d(v) > (d(u) + w(u,v)) então{
                d(v) ← d(u) + w(u,v)
                π (v) ← u
            }
        }
    }
    }
```

Fonte: Adaptado de MACEDO, VITALI (2011).

Utilizamos também o algoritmo de Bellman-Ford. Este algoritmo foi publicado em Bellman (1958). A diferença deste para o algoritmo de Dijkstra é que ele recebe arestas com peso negativo e por ser capaz de buscar o caminho mínimo em um dígrafo. Sua complexidade é de *O(v.e)*, no qual *v* é o número de vértices e é o número de arestas.

Figura 11 - Algoritmo de Bellman-Ford

```
Bellman-Ford(G, s){
        para todos os nós v de G faça {
                v.distancia ← ∞ ;
                s.distancia ← 0;
        }
        para i ← 1 até |V| - 1 faça {
                para todas as arestas (u,v) de G faça {
                        se u.distancia + peso(u,v) < v.distancia então {
                        v.distancia ← u.distancia + peso(u,v);
                        }
                }
        }
}
```

Fonte: Adaptado de RIBEIRO (2015).

Dentre as atividades há uma que trabalha com o algoritmo que foi publicado por FORD e FULKERSON (1956). De início foi desenvolvida para uso em rede ferroviárias na União Soviética nos

anos 30,40 e 50. O objetivo é encontrar o maior fluxo possível numa rede quando existir um ponto origem e um de chegada. A complexidade desse algoritmo é da ordem *O (m + n*) em que *m* é o número de arestas e *n* o número de vértices do grafo. É necessário colocarmos algumas definições antes de apresentarmos o algoritmo. Em um grafo com peso em duas arestas, com um nó de origem s e um nó de destino t queremos encontrar o fluxo máximo tendo em que *f(u,v)* representa o fluxo da aresta *(u,v)* e *c(u,v)* a sua capacidade, temos que obedecer a algumas restrições:

1) O fluxo não é negativo e tem de ser menor ou igual à sua capacidade:

   $0 \leq f(u,v) \leq c(u,v)$ para qualquer (u,v)

2) Em cada nó que não seja a origem e o destino, a soma do fluxo que entra é igual à soma do fluxo que sai

   Para cada $u \in V - \{s,t\} : \sum_{v \in V} f(v,u) = \sum_{v \in V} f(v,u) =$

3) O fluxo total é o que sai da origem menos o que entra na origem

   Queremos maximizar $|f|$, em que $|f| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s)$.

Caminho de aumento:= caminho pelo qual ainda é possível enviar fluxo

Grafo residual G_f := indica como podemos modificar o fluxo nas arestas de *G* depois de já aplicado o fluxo *f*.

Figura 12 - Algoritmo de Fluxo-Máximo

```
Ford-Fulkerson(G,t,s){
        Inicializar fluxo f a zero;
        Enquanto existir um caminho de aumento p no grafo residual Gf faça{
                aumentar fluxo f ao longo de p
        }
retornar f;
}
```

Fonte: Adaptado de RIBEIRO (2015).

Outro algoritmo abordado nas tarefas foi o Problema do Caixeiro Viajante. O objetivo deste algoritmo é encontrar um circuito que tenha a menor distância, de maneira que comece em qualquer vértice e visite cada um dos vértices uma vez e volte à origem. Ele é um problema de otimização classe NP-difícil. O algoritmo adotado na atividade foi o do vizinho mais próximo. A complexidade dele é *O(n.m),* em que *n* é o número de vértices e *m* é o número de arestas. Este algoritmo é uma heurística. Ou seja, as soluções não são ideais. E mostra a dificuldade que temos em traduzir problemas da vida real para a linguagem computacional.

Figura 13 - Algoritmo Do Vizinho Mais Próximo

1 – Escolha um vértice qualquer e marque-o.
2 – Veja a distância deste vértice para os outros e escolha a menor de todas as distâncias
3 – Vá para o vértice escolhido e marque-o
4 – Observe os vértices que ainda não foram visitados e escolha o que possui menor distância do vértice atual.
5 – Enquanto houver vértices não visitados, repita os passos 3 e 4.
6 – Após visitar todos os vértices, volte ao vértice de origem.

Fonte: Adaptado de SUTTON (2012).

Esses eram os pressupostos necessários do conhecimento matemático e computacional para entendimento da análise de dados apresentada a seguir.

4 METODOLOGIA

Em nosso estudo temos como objetivo geral **apresentar uma investigação de diferentes tipos de problemas de combinatória (contagem, existência, enumeração, classificação e otimização), em nível do Ensino Médio, ora abordados pelas usuais estratégias de enumeração ou técnicas de contagem ensinadas na Educação Básica (pensamento combinatório), ora abordados por meio de processos que caracterizam um tipo de pensamento computacional, mostrando conexões, potencialidades e limitações das duas abordagens, inclusive para o ensino de combinatória na Educação Básica.**

Especificamente buscamos 1) identificar e selecionar problemas de combinatória de contagem, existência e otimização que podem ser explorados via estratégias do pensamento computacional e pelo pensamento combinatório; 2) resolver os problemas selecionados usando as duas estratégias, analisando-as e comparando-as nas perspectivas combinatorial, computacional e educacional. 3) apresentar considerações didáticas, incluindo conexões entre pensamento computacional, combinatória e habilidades da BNCC.

Para atingir ao objetivo 1, foram selecionados 24 problemas de combinatória de modo a contemplar os cinco tipos de categorias aqui abordadas, quais sejam: enumeração, contagem, classificação, existência e otimização; as principais técnicas de contagem abordadas no ensino médio associadas aos problemas que podem ser resolvidos diretamente pelo princípio fundamental da contagem, bem como aqueles que decorrem dele tanto na contagem de ordenamentos (permutações simples, com repetição, circulares, caóticas e os famigerados arranjos) como de agrupamentos (combinações simples e combinações completas)

Para as questões das três primeiras categorias, buscamos problemas clássicos presentes nos livros didáticos de matemática da Educação Básica e no Exame Nacional do Ensino Médio, tais como o do número de apertos de mão, o número de caminhos em uma malha retangular, o número de agrupamentos, dentre outros.

Para as questões das duas últimas categorias: existência e otimização, trouxemos problemas relacionados aos grafos, dentre eles os de caminhos Eulerianos, Hamiltonianos, problema chinês do carteiro, problema do caixeiro viajante, problema de fluxo máximo, caminho mínimo, inspirados em pesquisas envolvendo a produção de significados de estudantes da Educação Básica para tais tipos de problemas, conforme Muniz (2007) e Borges (2018).

O quadro abaixo mostra a quantidade de questões para cada classificação (predominante), contendo 10 questões de combinatória de contagem, 8 questões de combinatória de existência, 5 de otimização 1 de enumeração.

Tabela 1 – Classificação dos problemas por categoria predominante

Categoria predominante	Número dos problemas	Quantidade total
Enumeração	3	1
Contagem	1, 2, 4, 5, 6, 7, 8, 18, 19, 20	10
Classificação	Presente no processo	0
Existência	9, 10, 11, 12, 13, 14, 15, 17	8
Otimização	16, 21, 22, 23, 24	5
Total	Todas	24

Fonte: elaborado pelos autores.

Para atingir os objetivos 2 e 3, resolver os problemas selecionados usando as duas estratégias, considerando uma Resolução matemática e outra computacional, buscamos estruturar tal Resolução computacional a partir das etapas do pensamento computacional apresentadas pelo CIEB.

Para cada um dos 24 problemas, teremos **quatro etapas de investigação**, nas quais apresentamos (i) a Resolução matemática; (ii) a Resolução via pensamento computacional, (iii) uma comparação da Resolução matemática com representações algorítmicas, e finalizando com (iv) reflexões e conexões didáticas, conforme mostra o quadro a seguir.

Tabela 2 – Etapas de investigação

Problema k Etapa j	Etapa de investigação	Caracterização da Etapa	Partes da Etapa
k.1	**Resolução Matemática**	Baseada no pensamento combinatório sem apoio de tecnologias digitais e/ou sem a preocupação com estruturas do pensamento computacional	Etapa única
k.2	**Resolução via pensamento computacional**	Orientada às etapas do pensamento computacional na perspectiva do CIEB, qual seja, estruturadas em quatro partes:	Decomposição
			Identificação de padrões
			Abstração
			Algoritmo
k.3	**Comparando soluções**	Comparação da Resolução matemática com três diferentes representações da Resolução computacional	Resolução matemática
			Fluxograma de processos
			Algoritmo com pseudocódigo
			Algoritmo em linguagem C ++
k.4	**Análises e conexões didáticas para a sala de aula considerando a BNCC**	Apresentar uma análise do problema buscando conexões para a sala de aula e para o professor de matemática considerando inclusive a BNCC	Etapa única

Fonte: elaborado pelos autores, 2021.

O terceiro objetivo: 3) apresentar considerações didáticas, incluindo conexões entre pensamento computacional, combinatória e habilidades da BNCC busca ser atingido na quarta etapa de investigação apresentada no quadro anterior, no qual trazemos comentários sobre cada um dos problemas selecionados, com conexões com a BNCC e algumas orientações para o professor.

Nessa quarta etapa de investigação, utilizamos as seguintes habilidades da BNCC, considerando tanto o Ensino Fundamental como o Ensino Médio, por possuírem conexões entre a combinatória ou pensamento computacional. Tais habilidades estão no Apêndice B.

Destacamos a seguir habilidades da BNCC que tem potencial de trabalhar o pensamento computacional na matemática, mas que não estão diretamente ligadas com a área de combinatória. Tal intersecção pode ser fonte de trabalhos futuros, bem como novas pesquisas entre pensamento computacional e pensamento combinatório. As habilidades estão no Apêndice C.

Dentre as vinte e quatro questões analisadas, notamos que problemas de combinatória que são classificados como de existência e de contagem são os que mais aparecem. Em contrapartida, os problemas de otimização continuam sendo minoria, ainda que tenha grande importância na realidade do discente.

Dentre as habilidades que mais aparece destacamos: EF01MA02, EF01MA20, EF04MA06, EF05MA09, EF06MA03, EF07MA05 e EF07MA06. Já as habilidades que menos apareceram foram: EF06MA33, EM13MAT310, EM13MAT315, EM13MAT507, EM13MAT508 e EM13MAT405.

Finalmente, no que tange à natureza da pesquisa, este trabalho está pautado na metodologia de Pesquisa de Desenvolvimento, conhecida também como *Design-Based Research.* Segundo Barab e Squire (2004 *apud* MATA; BOAVENTURA, 2014) a pesquisa em desenvolvimento pode ser entendida como um conjunto de ações investigativas aplicadas para o desenvolvimento de teorias, objetos e práticas pedagógicas de potencial aplicabilidade e utilidade nas situações de ensino e aprendizagem.

Essa metodologia ganhou muito espaço na área de práticas pedagógicas em ambiente digital ou que envolvam tecnologia no geral.

Destacamos algumas características dessa metodologia:

1) Teoricamente orientada.
2) Intervencionista
3) Colaborativa
4) Fundamentalmente dialogável
5) Iterativa.

Nosso trabalho tem a característica 1, que consiste na teoria ser o ponto de partida e chegada. Nessa dissertação nosso ponto de partida é a análise do potencial que a resolução de problemas de combinatória tem para desenvolver o pensamento computacional dos discentes.

E nosso ponto de chegada é justamente apresentar isso através da análise da resolução de 24 questões através da Resolução matemática e do pensamento computacional e consequente comparação entre os dois tipos de pensamento e solução.

A característica 2 é a proposta de mudança necessária no ensino de combinatória para que essa possa ser vista como um grande campo a explorar a ideia do pensamento computacional no ensino de matemática e na busca por soluções de problema. Neste trabalho vemos que ele também é colaborativo, uma vez que se pauta no referencial teórico já existente sobre o tema e servirá de base para futuros trabalhos de colegas docentes ou aprofundamentos de conhecimento dessa área em futuros trabalhos dos autores.

Este trabalho também é dialogável, uma vez que traz troca de experiências dos autores com o referencial teórico bem como sua experiência de prática docente e para futuros trabalhos a troca de conhecimento e prática. E finalmente é iterativa, uma vez que utiliza questões já conhecidas por muitos alunos e docentes em fase de vestibular, bem como aborda uma nova maneira de resolver essas questões através da Resolução do ponto de vista do pensamento computacional. Além de trazer no apêndice a implementação em linguagem C++ dessas questões.

Estabelecida a metodologia, partiremos para o próximo capítulo no qual investigaremos cada um dos 24 problemas de combinatória selecionados, apresentando a Resolução matemática, seguida da Resolução via pensamento computacional, o que inclui diferentes representações algorítmicas, sendo uma delas implementada em linguagem C ++, e finalizando com orientações e reflexões para a sala de aula de matemática.

5 INVESTIGAÇÃO DOS PROBLEMAS E ANÁLISE DOS RESULTADOS

O capítulo 5 traz a nossa investigação propriamente dita, procurando estabelecer conexões entre diferentes problemas de combinatória com o pensamento computacional, gerando tanto um convite a pensar os problemas de combinatória de forma ampliada, com a pensar em consequências e potencialidades do pensamento computacional na abordagem e ensino de combinatória na educação básica

Na seção 5.1, para cada um dos 2 problemas selecionados, teremos **quatro etapas de investigação**, nas quais apresentamos (i) a Resolução matemática; (ii) a Resolução via pensamento computacional, (iii) uma comparação da Resolução matemática com representações algorítmicas, e finalizando com (iv) reflexões e conexões dos problemas de combinatória com o pensamento computacional para a sala de aula de matemática, incluindo conexões com a BNCC e orientações para o professor.

Em seguida, apresentaremos um quadro síntese com todos os problemas e algumas possíveis conexões com as habilidades da BNCC.

5.1 Análise comparada dos problemas

Questão 1 - Problema do aperto de mão ou dos times de futebol - Em uma festa havia 21 pessoas presentes. Ao chegarem, cada pessoa cumprimentou todas as outras uma única vez, com um aperto de mão. Quantos apertos de mão ocorreram no total?

1.1 Resolução matemática

O problema se trata de uma questão de combinação simples. Se a pessoa P_1 cumprimentar P_2 ou o contrário, estamos abordando o mesmo aperto de mão. Dessa maneira podemos considerar que é combinação simples:

$$C_{21,2} = \frac{21!}{2!19!} = \frac{21.20.19!}{2!19!} = \frac{21.20}{2} = 210 \text{ apertos de mão possíveis}$$

1.2 Resolução Pensamento Computacional

Outra maneira de pensarmos nessa resolução é buscando um padrão para esse tipo de problema e ver como podemos generalizar esse pensamento através do pensamento computacional. Nesse tipo de resolução, baseada na classificação do CIEB, temos quatro etapas

- ✓ **decomposição**
- ✓ **padrões**
- ✓ **abstração**
- ✓ **algoritmo**

1A Decomposição:

Vamos pensar na primeira pessoa analisada. Ela deve cumprimentar as outras 19 pessoas que estão na festa. Descartamos ela cumprimentar ela mesma e contamos 19 apertos de mão.

A segunda pessoa vai cumprimentar todas as outras pessoas. Isso dará 19 apertos de mão. Mas ela já cumprimentou a pessoa que veio anteriormente a ela e nesse caso estaríamos contando o mesmo aperto de mão duas vezes. Isso porque se A cumprimenta B ou B cumprimenta A se trata do mesmo aperto de mão. A ordem nesse caso não importa, o que demonstra ser um problema de combinação simples. Ela tampouco vai cumprimentar a si mesma. Logo existem 18 pessoas para receber o cumprimento.

Esse padrão de retirar a pessoa anterior da contagem e a si mesma vai se repetindo até todos terem cumprimentado quem faltava na fila.

1B Padrões

Uma pessoa não cumprimenta a si mesma. Colocamos as pessoas numa espécie de fila para contarmos os apertos de mão. Se uma pessoa A cumprimenta outra pessoa B é o mesmo que B cumprimentar A, por isso temos que retirar da fila quem já cumprimentou para não a contar duas vezes.

1C Abstração

Dentre 20 pessoas, a primeira cumprimenta as 19 restantes;

A 19ª pessoa cumprimenta as 18 restantes da fila.

A 18ª cumprimenta as 17 restantes da fila.

Esse padrão se repete até que cheguemos na segunda pessoa da fila que cumprimenta a única que restou e a última que não cumprimenta mais ninguém. A ordem na qual dizemos o cumprimento, por exemplo, a 20ª pessoa cumprimenta a 19ª ou vice-versa não faz diferença na contagem. Porém, podemos acabar contanto duas vezes o mesmo cumprimento.

Para evitar isso, dividimos por 2. Um cumprimento é realizado por 2 pessoas. Para escolher a primeira, temos 20 pessoas para começar esse ato. Posteriormente, sobram 19

indivíduos para receber o aperto de mão. Assim, pelo Princípio Fundamental da Contagem existem $\frac{20.19}{2} = 210$ apertos de mão.

Outra maneira de pensar nessa Resolução é por combinação simples e que já foi resolvido na etapa de resolução matemática. Com o intuito de generalizar essa ideia, vamos pensar em n pessoas que vão dar o aperto de mão. Para o cálculo da quantidade de aperto de mãos, podemos pensar no princípio fundamental da contagem. Existirão $\frac{n(n-1)}{2} = \frac{n^2-1}{2}$ apertos de mão. Esse pensamento é combinatório e o usamos para construir a etapa da abstração.

1D Algoritmo

```
Inicio
n: natural
contador: inteiro
Leia (n)
Enfileire(n)
Enquanto (n <> 0) faça{
        Imprima: 1ª pessoa cumprimenta (n – 1) novas pessoas;
                 2ª pessoa cumprimenta (n – 2) novas pessoas;
            ....
            (n – 1)-estima pessoa cumprimenta 1 nova pessoa
        Fim imprime
        Contador++
        Desinfileire (n)
Fim do enquanto
Fim do programa
```

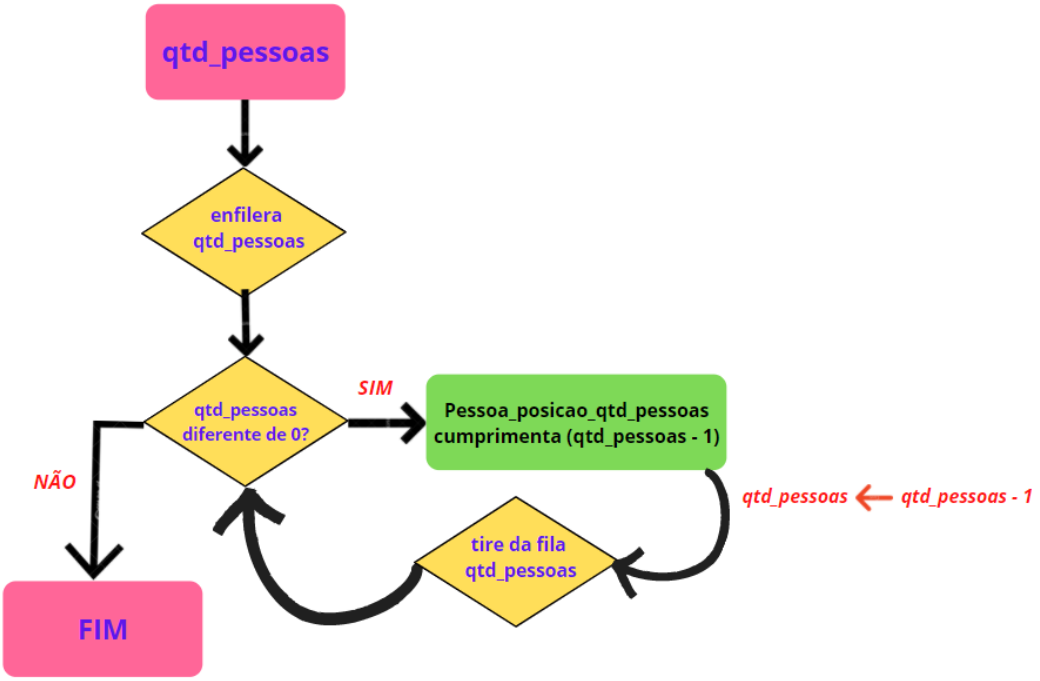
1.3 Comparando Soluções

Tabela 3 – Comparação de Resolução – Questão 1

A – Resolução Matemática

$C_{21,2} = \frac{21!}{2!19!} = \frac{21.20.19!}{2!19!} = \frac{21.20}{2} = 210$ apertos de mão possíveis

B – Fluxograma



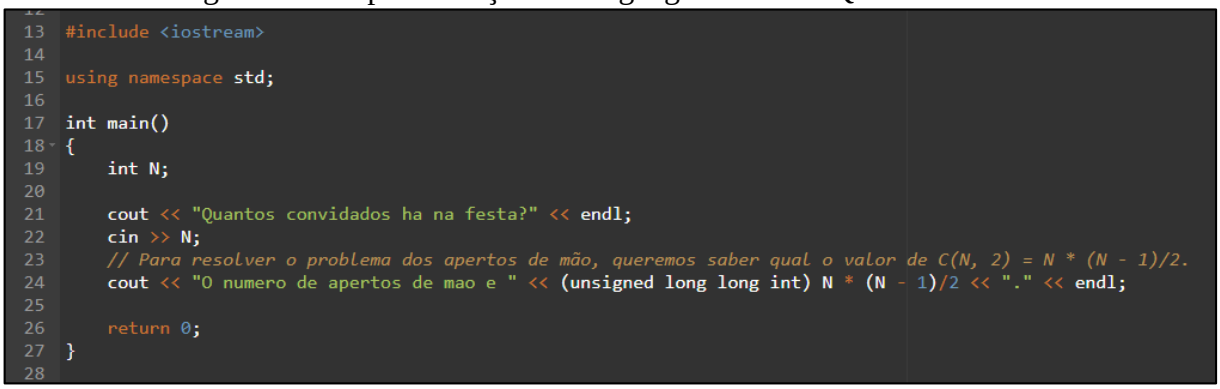
C – Abstração

```
Inicio( )
Leia (n)
Enfileire(n)
    Equanto (n <> 0) faça{
        Imprima: 1ª pessoa cumprimenta (n - 1) novas pessoas;
                 2ª pessoa cumprimenta (n - 2) novas pessoas;
                 ....
                 (n - 1)-ésima pessoa cumprimeira 1 nova pessoa
        Fim imprime
        Contador++
        Desinfileire (n)
Fim do enquanto
Fim do programa( )
```

Fonte: Os autores, 2021.

D – Implementação – linguagem C++

Figura 14 – Implementação em linguagem C++ da Questão 1



```cpp
#include <iostream>

using namespace std;

int main()
{
    int N;

    cout << "Quantos convidados ha na festa?" << endl;
    cin >> N;
    // Para resolver o problema dos apertos de mão, queremos saber qual o valor de C(N, 2) = N * (N - 1)/2.
    cout << "O numero de apertos de mao e " << (unsigned long long int) N * (N - 1)/2 << "." << endl;

    return 0;
}
```

Fonte: Os autores, 2021.

1.4 Considerações Didáticas:

A questão 1 pode ser classificada como de contagem, mas apresenta caráter enumerativo, uma vez que podemos nos questionar quais são as possíveis configurações de cumprimento entre as pessoas. Na etapa de abstração da Resolução matemática nos deparamos com uma equação do segundo grau e essa nos faz refletir sobre outras perguntas que podem surgir através dessa etapa. Ou seja, podemos nos questionar se, dado uma quantidade n de apertos de mão, se é possível ter tal configuração em que duas pessoas se cumprimentam umas às outras, sem que haja repetição. Dessa maneira nos deparamos com uma questão de existência e evidencia com a análise da generalização de soluções combinatorial, aliada ao pensamento computacional podem levar ao docente e o aluno a novas conclusões, reflexões e resolução de problemas de natureza distinta da originalmente proposta.

O pensamento computacional nesse caso foi usado para obter uma solução de maneira muito semelhante à solução em forma textual, com auxílio de notação matemática, usualmente abordada e ensinada nas aulas de matemática.

Questão 2 - (Problema do amigo oculto) Cinco pessoas de uma família farão um amigo oculto, sorteando como de praxe o nome entre eles, de modo que cada pessoa deve retirar apenas um nome, que obviamente não pode ser seu próprio nome. Quantas configurações diferentes podem ser formadas para que o sorteio não tenha que ser refeito?

2.1 Resolução Matemática:

Essa questão envolve o conceito de permutação caótica, ou desordenamento, ou seja, cada pessoa não ocupar o seu lugar primitivo. Se rotularmos cada pessoa dessa família como sendo A, B, C, D, E, temos ao permutar essas pessoas que a configuração B A D E C é um exemplo de permutação caótica. Já a configuração B A C E D não é caótica, uma vez que C ocupa sua posição de origem.

Vamos apresentar uma Resolução que utiliza o processo recursivo. Vamos iniciar de um caso menor para observarmos os padrões e a relação entre a quantidade de pessoas envolvidas e o número possível de permutações caóticas existentes. Se pensarmos em duas pessoas e a representarmos por A e B e representarmos a configuração inicial de posição por (A B), teremos apenas uma permutação caótica: BA. Vamos representar essa quantidade de desordenamentos por $D_2 = 1$.

Se pensarmos em três pessoas e representarmos a configuração inicial delas por (A B C), vamos apresentar as possibilidades através de enumeração: (B C A), (C A B). Logo, $D_3 = 2$. Caso calculemos para quatro pessoas (A B C D). Para isso, vamos dividir em dois casos, uma vez que este problema cresce consideravelmente em relação aos anteriores. O primeiro caso é quando pensamos nas possíveis posições em que o A pode estar e trocá-lo de posição ordenadamente em relação às outras posições existentes. Ou seja, vamos trocar $A \leftrightarrow B, A \leftrightarrow C, A \leftrightarrow D, A \leftrightarrow E$. Para esses casos encontramos as seguintes configurações: (B A _ _), (C _ A _) e (D _ _ A). Reparemos que não completamos duas posições dentre as representadas anteriormente, pois elas recaem no caso que vimos anteriormente, de permutação caótica com dois elementos. Quando temos duas posições e já sabemos quantas possibilidades existem de representação $D_2 = 1$. Logo nesse caso teríamos 3 casos de D_2, o que resulta em $3.1 = 3$ permutações caóticas (B A D C), (C D A B) e (D C B A).

No segundo caso vamos ver as permutações em que o A não troca de lugar com o elemento que ele substitui. Ou seja, se colocamos o A na posição B, então B não pode ocupar a posição A ($A \rightarrow B, mas\ B \nrightarrow A$). Nesse caso, recaímos numa permutação caótica de 3 elementos, uma vez que B não pode ocupar sua nova posição de origem, onde estava A, nem o

C pode ocupar sua posição primitiva e o mesmo para o D. Assim caímos no caso de desordenamento com 3 elementos. Mas já calculamos o valor de D_3 anteriormente.

Se colocamos o A na posição C, então C não pode ocupar a posição A ($A \rightarrow C, mas\ C \nrightarrow A$). Assim, recaímos numa permutação caótica de 3 elementos, uma vez que C não pode ocupar sua nova posição de origem, onde estava A, nem o B pode ocupar sua posição primitiva e o mesmo para o D. Logo temos o caso de desordenamento com 3 elementos (D_3).

Se colocamos o A na posição D, então D não pode ocupar a posição A ($A \rightarrow D, mas\ D \nrightarrow A$). Assim, recaímos numa permutação caótica de 3 elementos, uma vez que D não pode ocupar sua nova posição de origem, onde estava A, nem o B pode ocupar sua posição primitiva e o mesmo para o C. Logo temos o caso de desordenamento com 3 elementos (D_3).

Portanto chegamos à conclusão de que $D_4 = 3(D_2 + D_3) = 9$ casos.

Vamos resolver o caso original para 5 pessoas. Analogamente ao que fizemos para quatro pessoas, faremos com cinco. No primeiro caso vamos observar as configurações em que A troca de lugar com cada uma das outras pessoas e essa vai para a posição primitiva de A. Ou seja, $A \leftrightarrow B, A \leftrightarrow C, A \leftrightarrow D, A \leftrightarrow E$. Nesta situação, temos que B tem que ocupar a posição de A, mas C e D não podem estar nas suas posições primitivas. Logo, recaímos numa permutação caótica de 3 elementos. O mesmo acontece para $A \leftrightarrow C, A \leftrightarrow D, A \leftrightarrow E$. Temos então $4.D_3 = 4.3 = 12$.

Analogamente, se considerarmos que A troca de posição com B, mas que esse não pode ocupar a posição primitiva de A e repetirmos esse processo para C. D e, teremos quatro casos de permutação caótica de quatro elementos. Ou seja, $4.D_4 = 4.4 = 16$.

Logo a solução do problema é $D_5 = 4(D_3 + D_4) = 4.(2+9) = 44$.

Uma outra maneira de calcularmos essa questão é através da aproximação de permutação caótica dada por $D_n = \frac{n!}{e}$. No caso da questão para n = 5, $D_5 = \frac{5!}{e} \cong 44{,}28 \cong 44$ desordenamentos.

A solução acima pode ser vista em mais detalhes, bem como o processor recursivo para obtenção do número de permutações caóticas de n elementos distintos, no link:

https://www.youtube.com/watch?v=iLpsLwWpI4c&t=39s

2.2 Resolução Pensamento Computacional

2A – Decomposição

Para dois já calculamos que existem 1 permutação caótica. Ou seja, $D_2 = 1$.

Para três posições, através de enumeração notamos que existem 3 permutações caóticas. Logo $D_3 = 2$

Para quatro posições, dividimos em dois casos: trocamos A com cada uma das outras posições. Isso resulta em 3 situações possíveis. E a outra situação é colocar A em cada posição subsequente, mas não colocando a letra primitiva daquela posição no local de A. Nessa configuração existem 12 permutações possíveis.

Para cinco posições temos 44 desordenamentos.

Podemos resolver essa questão através do uso da fórmula da permutação caótica dada por $D_n = \frac{n!}{e}$. No caso da questão para n = 5, $D_5 = \frac{5!}{e} \cong 44{,}28 \cong 44$ desordenamentos.

2B - Padrão

Para que uma permutação seja caótica o número que está na posição primitiva não pode permanecer ali.

Dividimos em casos para cada problema. O primeiro caso é quando trocamos a primeira letra de posição com a letra subsequente e assim sucessivamente até esgotar todas essas mudanças.

O segundo caso é colocar a primeira letra nas posições subsequentes de modo que essa letra não ocupe a posição primitiva da letra A.

2C – Abstração

Vamos buscar um padrão recursivo diante das situações que já calculamos:

$D_2 = 1$

$D_3 = 2$

$D_4 = 3.(D_2 + D_3)$

$D_5 = 4.(D_3 + D_4)$

...

Parece que a recursão para esse tipo de problema seja

$D_n = (n - 1).(D_{n-1} + D_{n-2})$

Se temos n elementos, o primeiro caso seria retirar o A_1 de sua posição de origem e ir permutando para cada posição restante, n – 1, pois na posição de A_1 estaria o elemento no qual A_1 trocou de lugar. Assim sobrariam D_{n-2} posições para serem permutadas.

No segundo caso, A_1 permuta com n – 1 posições, mas o elemento no qual ele troca de posição não pode entrar na posição A_1 e com isso temos D_{n-1} posições possíveis.

Logo o total é de $D_n = (n-1)\,D_{n-2} + (n-1)\,D_{n-1} = (n-1)(D_{n-2} + D_{n-1})$

2D – Algoritmo

```
Inicio ( )
N: natural
Leia (n)
D_n : natural
D_2 = 1
D_3 = 3
Para 4 ≤ i ≤ n faça
        D_i = (n – 1)(D_{n-2} + D_{n-1})
        Imprime (D_i)
        i = i + 1
Fim para
Fim
```

2.3 Comparando Soluções

Tabela 4 - Comparando Soluções – Questão 3

A – Resolução Matemática

$D_2 = 1$

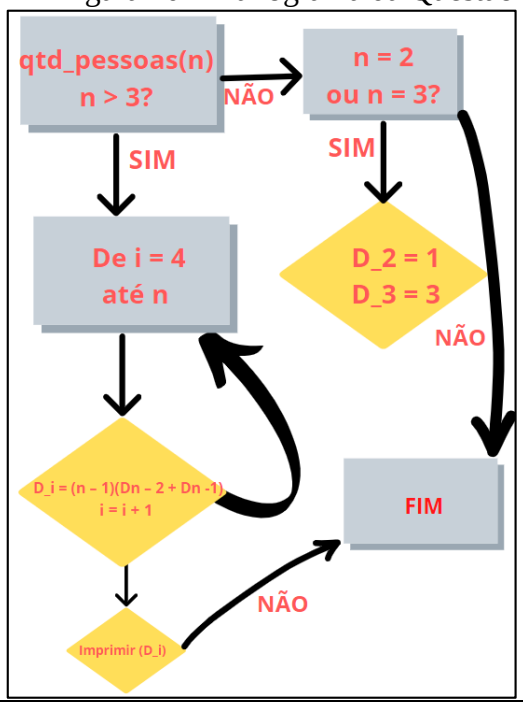
$D_3 = 2$

$D_4 = 3.(D_2 + D_3) = 16$

$D_5 = 4(D_3 + D_4) = 4.(2+9) = 44$

B – Fluxograma

Figura 20 - Fluxograma da Questão 3



C – Abstração

```
Inicio ( )
N: natural
Leia (n)
D_n : natural
D_2 = 1
D_3 = 3
    Para 4 ≤ i ≤ n faça
        D_i = (n – 1)(Dn – 2 + Dn -1)
        Imprime (D_i)
        i = i + 1
    Fim para
Fim
```

Fonte: Os autores, 2021.

D – Implementação – Linguagem C ++

Figura 15 - Implementação em linguagem C++ da Questão 2

```cpp
#include <iostream>

using namespace std;

unsigned long long int PermutacaoCaotica(int n){
    if (n == 1){
        return 0;
    } else if (n == 2){
        return 1;
    } else {
        return (n - 1) * (PermutacaoCaotica(n - 1) + PermutacaoCaotica(n - 2));
    }
}
// Usamos recorrência para facilitar o cálculo para o computador, lidando com variáveis
// menores do que n!. Com isso, calculamos a permutação caótica com n elementos para dois casos:
// 1- Quando o elemento n assume a posição de algum elemento k e o elemento k assume a posição de n (exemplo: [1]23[4] -> [4]32[1])
//      Neste caso, o número de permutações caóticas é (n - 1) * PermutacaoCaotica(n - 2)
// 2- Quando o elemento n assume a posição de algum elemento k e o elemento k NÃO assume a posição de n (exemplo: [1]23[4] -> [4]3[1]2)
//      Neste caso, o número de permutação caóticas é (n - 1) * PermutacaoCaotica(n - 1)

int main()
{
    int N;

    cout << "Quantas pessoas estao participando do amigo oculto?" << endl;
    cin >> N;

    cout << "O numero de sorteios validos e " << PermutacaoCaotica(N) << "." << endl;

    return 0;
}
```

Fonte: Os autores, 2021.

2.4 Considerações didáticas

A questão 2 é classificada como de contagem, porém ela pode ser resolvida de duas maneiras distintas: ou através do uso da fórmula da permutação caótica ou através de recursividade. Ao pensarmos em resolver tal questão via pensamento computacional, ao estruturar de maneira recursiva ganhamos duas características em tal solução. A primeira é que podemos exibir para cada etapa a quantidade de desordenamentos possíveis, ou seja podemos contar quantos desordenamento existem, bem como exibir enumerando tais casos para valores menores. Esse tipo de estratégia de enumeração auxilia no entendimento da recursividade existente no problema. Uma outra visão que podemos ter desse problema é se dada uma quantidade de desordenamento é possível termos uma quantidade de pessoas viável para o problema ter solução ou não. E, finalmente podemos também observar estratégia de classificação quando dividimos os desordenamentos em casos tais como para n = 1, n = 2 até n = 5 e levamos em consideração a regra imposta pelo problema.

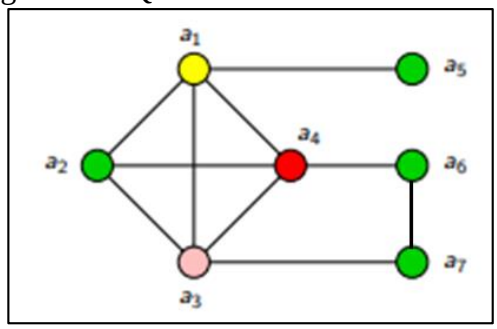
Através da análise podemos perceber que, ainda que uma questão seja de contagem, ela pode ser analisada pelo professor com potencial para criar questões de existência, bem como utilizar estratégias de enumeração e classificação para a solução do problema.

A partir da solução matemática vimos que um resultado pode ser encontrado através de um estudo de casos e posterior conjectura da solução através do processo recursivo,

Consequentemente obtemos os próximos resultados. Já com o pensamento computacional modificamos a construção da solução tanto na etapa da abstração quanto no fluxograma ao representar esse padrão através de uma estrutura de repetição que não temos na matemática. Porém, ao observarmos a implementação em C++ vimos que esta estrutura não foi usada, mas sim a ideia de recursão, também presente na solução matemática, indicando uma aproximação entre a solução matemática e o algoritmo, com estruturas iguais.

Questão 3 - ***É possível sair da cidade a_6 e chegar até a cidade a_5 sem passar pela cidade a_4? Liste tais possibilidades***

Figura 16 – Questão de combinatória de existência



Fonte: Os autores, 2021.

3.1 Resolução Matemática:

Vamos enumerar alguns possíveis caminhos para chegarmos de a_6 até a_5 sem repetirmos cidades:

$a_6 \rightarrow a_7 \rightarrow a_3 \rightarrow a_2 \rightarrow a_1 \rightarrow a_5$

$a_6 \rightarrow a_7 \rightarrow a_3 \rightarrow a_1 \rightarrow a_5$

$a_6 \rightarrow a_4 \rightarrow a_2 \rightarrow a_1 \rightarrow a_5$

$a_6 \rightarrow a_4 \rightarrow a_3 \rightarrow a_1 \rightarrow a_5$

$a_6 \rightarrow a_7 \rightarrow a_3 \rightarrow a_2 \rightarrow a_1 \rightarrow a_5 \rightarrow a_4$

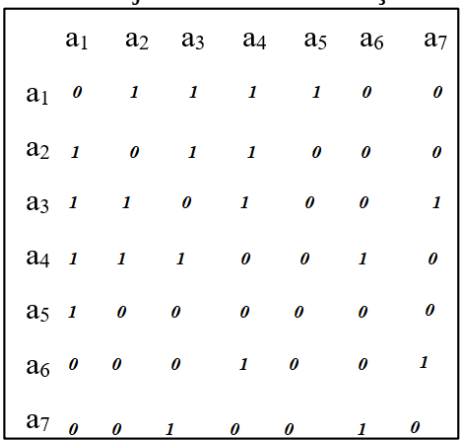
Dentre esses caminhos, existem aqueles que não passam por a_4:

$a_6 \rightarrow a_7 \rightarrow a_3 \rightarrow a_2 \rightarrow a_1 \rightarrow a_5$

$a_6 \rightarrow a_7 \rightarrow a_3 \rightarrow a_1 \rightarrow a_5$

Uma outra maneira de mostrar que existe um caminho que satisfaz a regra determinada é montarmos a matriz de adjacência dos vértices da imagem anterior:

Figura 17 – Matriz de Adjacência da Resolução matemática da Questão 3

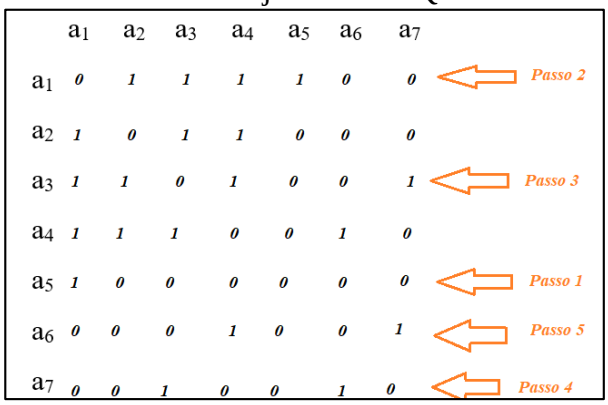


	a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_1	0	1	1	1	1	0	0
a_2	1	0	1	1	0	0	0
a_3	1	1	0	1	0	0	1
a_4	1	1	1	0	0	1	0
a_5	1	0	0	0	0	0	0
a_6	0	0	0	1	0	0	1
a_7	0	0	1	0	0	1	0

Fonte: Os autores, 2021.

Buscamos nessa matriz de adjacência linhas ou colunas em que estejam rotulados 1 para a_5, a_6 e 0 para a_4.

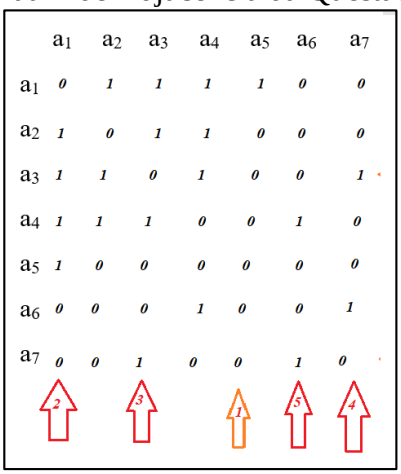
Figura 18 – Matriz de Adjacência da Questão 3 – Etapa 1



	a_1	a_2	a_3	a_4	a_5	a_6	a_7	
a_1	0	1	1	1	1	0	0	Passo 2
a_2	1	0	1	1	0	0	0	
a_3	1	1	0	1	0	0	1	Passo 3
a_4	1	1	1	0	0	1	0	
a_5	1	0	0	0	0	0	0	Passo 1
a_6	0	0	0	1	0	0	1	Passo 5
a_7	0	0	1	0	0	1	0	Passo 4

Fonte: Os autores, 2021.

Figura 19 - Matriz de Adjacência da Questão 3 – Etapa 2



	a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_1	0	1	1	1	1	0	0
a_2	1	0	1	1	0	0	0
a_3	1	1	0	1	0	0	1
a_4	1	1	1	0	0	1	0
a_5	1	0	0	0	0	0	0
a_6	0	0	0	1	0	0	1
a_7	0	0	1	0	0	1	0

2 3 1 5 4

Fonte: Os autores, 2021.

Essa é uma possível Resolução de caminho que satisfaz a regra do enunciado e mostra que existe pelo menos um caminho nas condições estabelecidas.

3.2 Resolução Pensamento Computacional

3A - Decomposição

Devemos olhar para todas as conexões que a_6 realiza

Escrevemos os possíveis caminhos de a_6 até a_5

Descartamos os caminhos que passam por a_4

Não repetimos vértices

3B - Padrões

Podemos representar cada caminho entre uma cidade e outra (vértice) através de uma matriz de adjacência, em que 0 representa que não existe ligação direta entre as duas cidades e 1 significa que existe um caminho direto entre os vértices analisados.

Não vamos repetir cidade e tampouco caminho entre cidades.

Pela enumeração conseguimos ver todos ou a maior parte dos caminhos possíveis entre a_5 e a_6

3C - Abstração

Podemos encontrar os caminhos possíveis através do desenho e enumerando as possibilidades de caminho ou podemos através da matriz de adjacência construir esse caminho, tanto analisando por linha ou coluna.

3D - Algoritmo

Inicio ( )

Leia (matriz de adjacência)

Enquanto não encontrar um caminho de a_i até a_j sem passar por a_k faça

Leia a linha i da matriz

Se $a_{ij} <> 0$ então imprima "$a_{ij} \rightarrow a_{lm}$" e pule para a linha seguinte

Se $a_{jm} <> 0$ então imprima "$a_{jm} \rightarrow a_{no}$" e pule para a próxima linha

Senão pula linha

Fim enquanto

Fim do programa ( )

3.3 Comparando Soluções

Tabela 5 - Comparando Soluções – Questão 3

A – Resolução Matemática

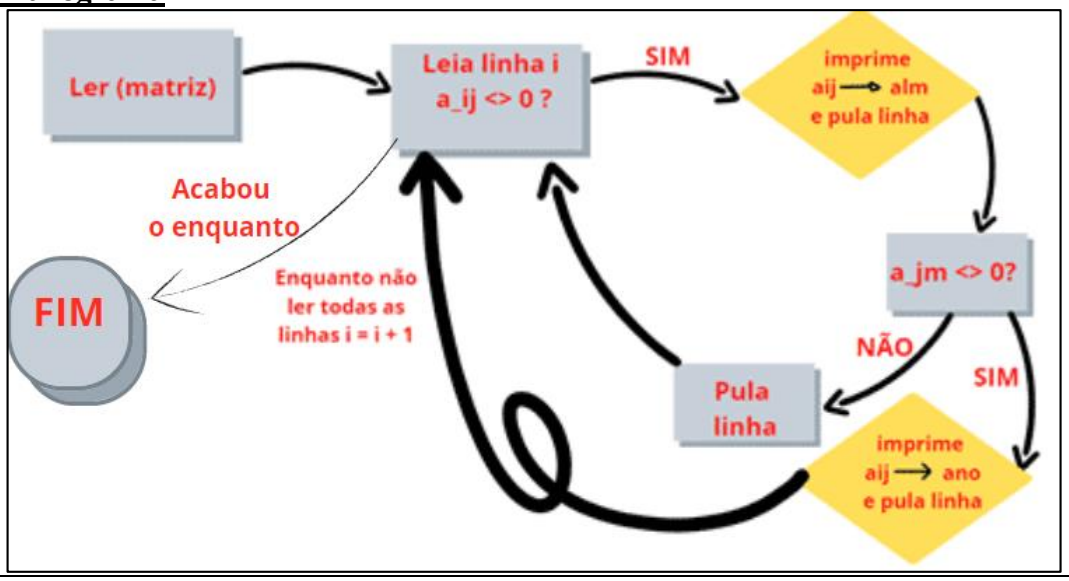
Vamos enumerar alguns possíveis caminhos para chegarmos de a_6 até a_5 sem repetirmos cidades :

$a_6 \rightarrow a_7 \rightarrow a_3 \rightarrow a_2 \rightarrow a_1 \rightarrow a_5$
$a_6 \rightarrow a_7 \rightarrow a_3 \rightarrow a_1 \rightarrow a_5$
$a_6 \rightarrow a_4 \rightarrow a_2 \rightarrow a_1 \rightarrow a_5$
$a_6 \rightarrow a_4 \rightarrow a_3 \rightarrow a_1 \rightarrow a_5$
$a_6 \rightarrow a_7 \rightarrow a_3 \rightarrow a_2 \rightarrow a_1 \rightarrow a_5 \rightarrow a_4$

Dentre esses caminhos, existem aqueles que não passam por a_4:

$a_6 \rightarrow a_7 \rightarrow a_3 \rightarrow a_2 \rightarrow a_1 \rightarrow a_5$
$a_6 \rightarrow a_7 \rightarrow a_3 \rightarrow a_1 \rightarrow a_5$

B – Fluxograma



C – Abstração

```
Inicio (  )
Leia (matriz de adjacência)
Enquanto não encontrar um caminho de ai até aj sem passar por ak faça
        Leia a linha i da matriz
        Se aij <> 0 então imprima "aij --> alm" e pule para a linha seguinte
            Se ajm <> 0 então imprima "ajm --> ano" e pule para a próxima linha
        Senão pula linha
Fim enquanto
Fim do programa (  )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 20 - Implementação em linguagem C++ da Questão 3

```cpp
#include <iostream>
#include <vector>

using namespace std;

vector<vector<int> > mapa;
// Usamos um vetor de vetores para representar nosso mapa.
// Cada cidade tem um vetor que indica quais são suas cidades vizinhas.
// Usamos essa representação por ser menos custosa para o computador.

vector<vector<int> > caminhos;

void EncontraCaminho(vector<int> caminho, vector<bool> visitados, int origem, int destino, int evitada){
    // Adiciona a cidade onde se está
    caminho.push_back(origem);

    // Verifica se já chegou
    if (origem == destino){
        caminhos.push_back(caminho);
    } else {
        visitados[origem] = true;

        // Verifica cada um dos vizinhos da cidade onde se esta
        for(int i = 0; i < mapa[origem].size(); ++i){
            int vizinho = mapa[origem][i];

            // Se for a cidade a ser evitada, ignore-a
            if (vizinho == evitada){
                continue;
            }

            if (!visitados[vizinho]){
                EncontraCaminho(caminho, visitados, vizinho, destino, evitada);
            }
        }
    }
}
// Esse é um algoritmo computacional conhecido chamado de Busca em Profundidade,
// onde basicamente visita-se cada cidade checando seus vizinhos e vendo se já chegou ao destino,
// ignorando a cidade a ser evitada.

int main()
{
    int n, m, u, v, origem, destino, evitada;
    vector<bool> visitados;

    cout << "Quantas cidades ha neste mapa?" << endl;
    cin >> n;

    cout << "Quantos caminhos ha entre cidades?" << endl;
    cin >> m;

   for(int i = 0; i <= n; ++i){
       mapa.push_back(vector<int>());
       visitados.push_back(false);
   }

   cout << "Agora descreva os " << m << " caminhos. Por exemplo, 1 2 indica um caminho entre as cidades a1 e a2." << endl;
   for(int i = 0; i < m; ++i){
       cin >> u >> v;
       mapa[u].push_back(v);
       mapa[v].push_back(u);
   }

   cout << "Qual a cidade de origem?" << endl;
   cin >> origem;

   cout << "Qual a cidade de destino?" << endl;
   cin >> destino;

   cout << "Qual a cidade que se deve evitar?" << endl;
   cin >> evitada;

   while (origem == evitada || destino == evitada){
       cout << "Nao e possivel evitar a cidade de origem ou de destino. Escolha outra cidade pra se evitar." << endl;
       cin >> evitada;
   }
   while (origem == evitada || destino == evitada){
       cout << "Nao e possivel evitar a cidade de origem ou de destino. Escolha outra cidade pra se evitar." << endl;
       cin >> evitada;
   }

   EncontraCaminho(vector<int>(), visitados, origem, destino, evitada);

   if (caminhos.size() > 0){
       cout << "Os possiveis caminhos de " << origem << " ate " << destino << " sem passar por " << evitada << " sao:" << endl;
       for(int i = 0; i < caminhos.size(); ++i){
           cout << caminhos[i][0];
           for(int j = 1; j < caminhos[i].size(); ++j){
               cout << " -> " << caminhos[i][j];
           }
           cout << endl;
       }
   } else {
       cout << "Nao e possivel ir de " << origem << " ate " << destino << " sem passar por " << evitada << "." << endl;
   }

   return 0;
```

Fonte: Os autores, 2021

3.4 Considerações didáticas

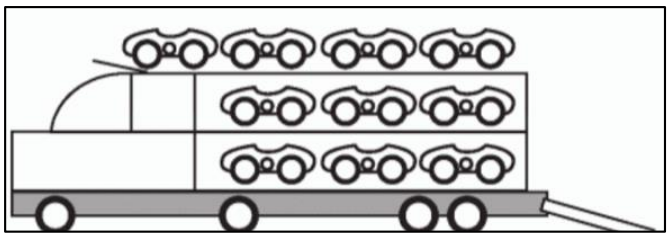
A questão 3 possui duas classificações e duas perguntas que podemos questionar. Ela pode ser classificada como de existência, uma vez que pergunta se é possível realizar um

caminho diante de algumas regras, mas também pode ser visto como um processo de enumeração, pois para que possamos encontrar caminho que satisfaça as regras estabelecidas é necessário enumerar as possíveis rotas assim como a enumeração na implementação do pensamento computacional pode ser vista como a estrutura de condição em que o comando se delimita as regras que devem ser seguidas a fim de encontrarmos a solução ótima. E ainda podemos classificar como estratégia de otimização uma vez que ao existir mais de um caminho podemos questionar qual o menor deles para se chegar ao destino. Ou seja, a partir da análise da classificação da natureza da questão e a conexão com pensamento computacional podemos perceber que a questão ganha novos questionamentos bem como novas demandas de pesquisa.

Podemos perceber que a resolução matemática se dá através processo enumerativo. Quando apresentamos a solução via pensamento computacional, esse processo enumerativo acontece através de uma estrutura de repetição de enquanto como solução e tal estrutura não está presente na solução matemática. O fluxograma seguiu a ideia de abstração. Já na implementação vemos que é usado algumas estruturas que não estão presentes em nenhum dos passos anteriores. Estruturas de lógica ou vetorial. Outra estrutura é a de função e rotina para que o programa fique mais organizado e rotinas de repetição e condicionais.

Questão 4 - (ENEM – 2017) - Um brinquedo infantil caminhão-cegonha é formado por uma carreta e dez carrinhos nela transportados, conforme a figura.

Figura 21 – Questão 4



Fonte: ENEM, 2017.

No setor de produção da empresa que fabrica esse brinquedo, é feita a pintura de todos os carrinhos para que o aspecto do brinquedo fique mais atraente. São utilizadas as cores amarelo, branco, laranja e verde, e cada carrinho é pintado apenas com uma cor. O caminhão cegonha tem uma cor fixa. A empresa determinou que em todo caminhão-cegonha deve haver pelo menos um carrinho de cada uma das quatro cores disponíveis. Mudança de posição dos carrinhos no caminhão-cegonha não gera um novo modelo do brinquedo. Com base nessas informações, quantos são os modelos distintos do brinquedo caminhão-cegonha que essa empresa poderá produzir?

a) $C_{6,4}$
b) $C_{9,3}$
c) $C_{10,4}$
d) 6^4
e) 4^6

4.1 Resolução Matemática:

Para resolver a questão vamos começar pintando o que é obrigatório. É pedido que exista pelo menos um carro de cada cor dentre as quatro existentes. Então vamos fixar um carro pintado de laranja, outro carro de cor amarela, outro com cor branca e um pintado de verde. Existem no caminhão 10 carros. Como já fixamos a cor de 4 deles, sobram 6 carros a serem pintados. Esses não têm restrição de pintura e tampouco de que cada andar tenha que ter um número mínimo de cores. Assim restam seis carros que trataremos como iguais para serem pintados com quatro cores distintas. A quantidade de carros pintada de cada cor é indiferente. Se quisermos pintar, por exemplo, todos os carros de laranja podemos e está dentro do que foi previsto. Vamos representar essa configuração da seguinte maneira:

C + C + CC+ C = 6 é um exemplo de pintura possível para os carros restantes. Vamos imaginar que as cores estão dispostas de acordo com uma determinada ordem: laranja, amarelo, branco e verde. Essa questão se trata de um problema de permutação com repetição. Temos quatro

cores, seis carros e três disposições distintas dentre a coloração desses carros. Portanto temos 9 elementos a serem permutados seis a seis e três a três. Ou seja:

$$P_9^{3,6} = \frac{9!}{6!3!}$$

Mas quando falamos de permutação com repetição é equivalente a dizermos sobre a combinação com repetição, uma vez que se trata de um problema em que a ordem dos carros não importa e existem carros pintados da mesma cor. No caso de resolvermos com combinação com repetição, contamos 9 elementos (carros e disposições das cores) tomados seis a seis: $C_{9,6}$.

Porém falarmos que se trata de $C_{9,6}$ é equivalente a sua combinação complementar, ou seja, $C_{9,3} = P_9^{3,6}$.

4.2 Resolução Pensamento Computacional

4A - Decomposição

Primeiro observamos que precisamos pintar um carro utilizando uma vez cada uma das quatro cores fornecidas no enunciado.

Estabelecemos uma ordem entre essas cores para colorir os carros restantes

Não somos obrigados a usar as quatro cores nos carros restantes depois de termos pintado pelo menos um carro de cada cor. Por isso tratamos os carros como sendo todos iguais e a disposição para pintá-los também

Calculamos a permutação com repetição que é equivalente a uma combinação simples complementar.

4B - Padrões

Temos dez carros, quatro cores com 4 restrições

Logo temos 10 – 4 = 6 carros para pintarmos bem como queiramos.

Temos 4 – 1 = 3 configurações possíveis para a coloração desses carros.

4C - Abstração

Temos n carros para e queremos colorir com k cores. Precisamos de k carros para colorir cada um com uma das k cores. Os n – k carros restantes podem ser coloridos em qualquer cor.

Ordenando essas cores obedecendo um certo critério numa fila, temos n – k – 1 maneiras de permutar essas cores e usá-las nos n – k carros restantes. Temos uma permutação com repetição no total de $P_{n+k-1}^{n-k,\ n-k-1} = \frac{(n-k-1)!}{k!(n-1)!}$ Maneiras de permutar

4D - Algoritmo

```
Inicio ( )
        qtd_carros: natural
        qtd_cores: natural
        aux: natural
        resultado: natural
        Leia(qtd_carros, qtd_cores)
        carros_pintados = qtd_cores
        fatorial = 1
        aux = 1
        para 1 ≤ aux ≤ qtd_carros − qtd_cores − 1
                fatorial = fatorial*aux
                aux = aux + 1
        fim para
        aux = 1
        fatorial2 = 1
        para 1 ≤ aux ≤ qtd_cores
                fatorial2 = fatorial2*aux
                aux = aux + 1
        fim para
        aux = 1
        fatorial3 = 1
        para 1 ≤ aux ≤ qtd_carros – 1
                fatorial3 = fatorial3*aux
                aux = aux + 1
        fim para
        resultado =(fatorial)/(fatorial2 * fatorial3)
        imprimir (“existem”, resultado, “possibilidades”)
Fim
```

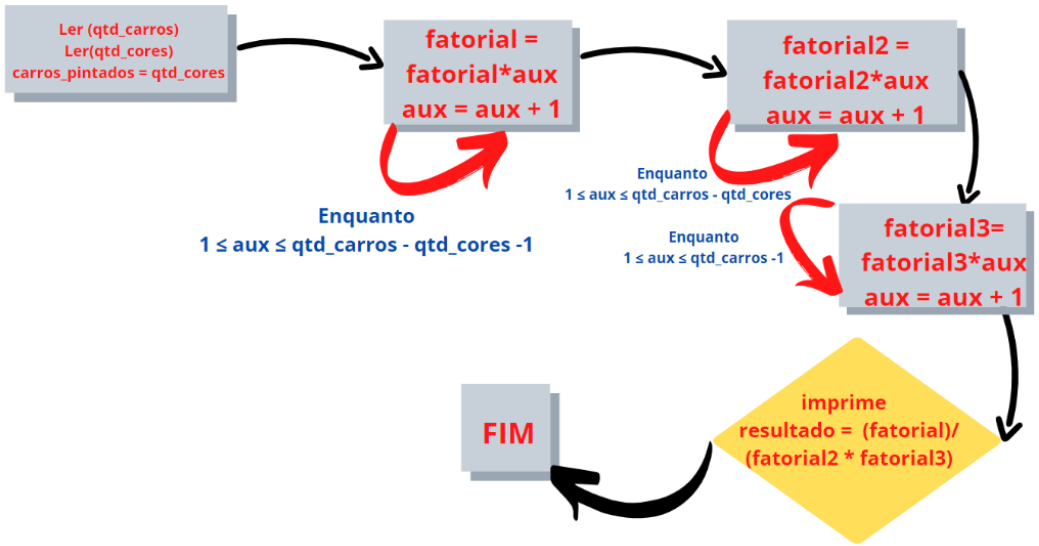
4.3 Comparando Soluções

Tabela 6 - Comparando Soluções – Questão 4

A – Resolução Matemática

$$P_9^{3,6} = \frac{9!}{6!3!}$$

B – Fluxograma



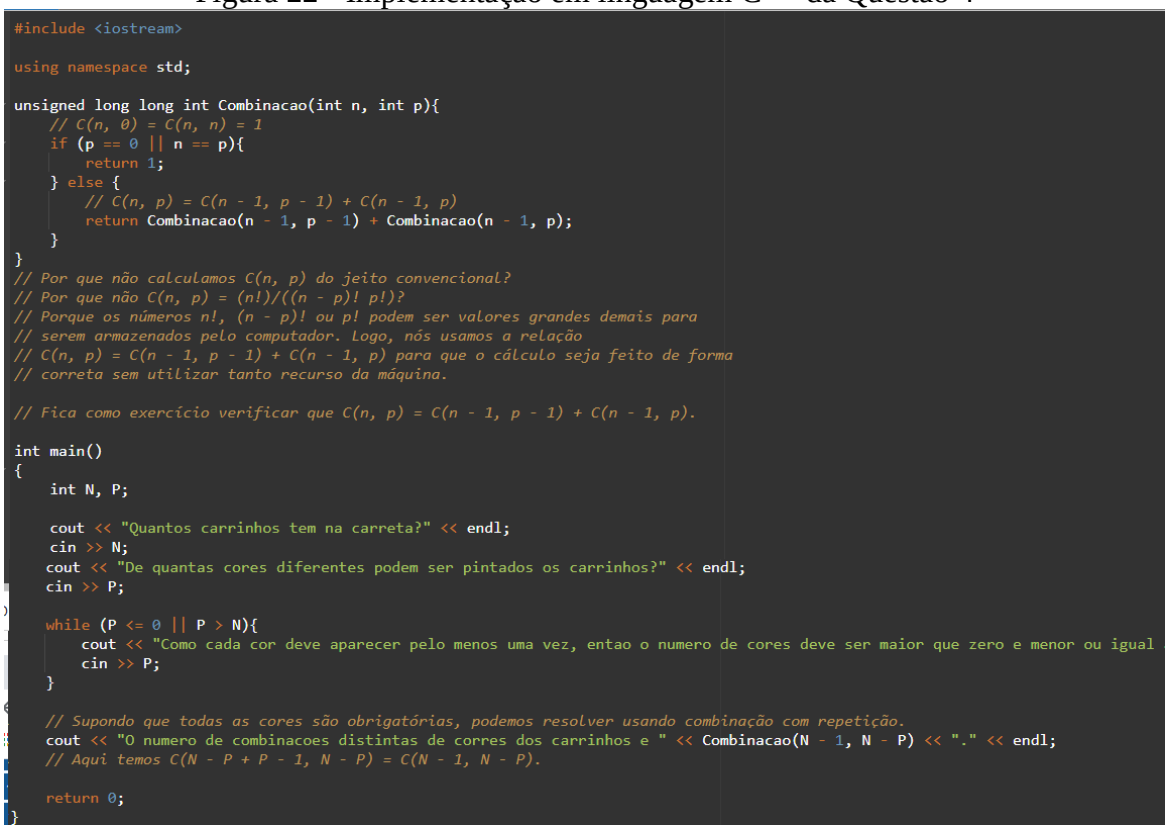
C – Abstração

```
Inicio ( )
    qtd_carros: natural
    qtd_cores: natural
    aux: natural
    resultado: natural
    Leia(qtd_carros, qtd_cores)
    carros_pintados = qtd_cores
    fatorial = 1
    aux = 1
    para 1 ≤ aux ≤ qtd_carros - qtd_cores -1
        fatorial = fatorial*aux
        aux = aux + 1
    fim para
    aux = 1
    fatorial2 = 1
    para 1 ≤ aux ≤qtd_cores
        fatorial2 = fatorial2*aux
        aux = aux + 1
    fim para
    aux = 1
    fatorial3 = 1
    para 1 ≤ aux ≤ qtd_carros - 1
        fatorial3 = fatorial3*aux
        aux = aux + 1
    fim para
    resultado =(fatorial)/(fatorial2 * fatorial3)
    imprimir ("existem", resultado, "possibilidades")
Fim (  )
```

Fonte: Os autores, 2021.

D – Implementação – Linguagem C ++

Figura 22 - Implementação em linguagem C++ da Questão 4



```
#include <iostream>

using namespace std;

unsigned long long int Combinacao(int n, int p){
    // C(n, 0) = C(n, n) = 1
    if (p == 0 || n == p){
        return 1;
    } else {
        // C(n, p) = C(n - 1, p - 1) + C(n - 1, p)
        return Combinacao(n - 1, p - 1) + Combinacao(n - 1, p);
    }
}
// Por que não calculamos C(n, p) do jeito convencional?
// Por que não C(n, p) = (n!)/((n - p)! p!)?
// Porque os números n!, (n - p)! ou p! podem ser valores grandes demais para
// serem armazenados pelo computador. Logo, nós usamos a relação
// C(n, p) = C(n - 1, p - 1) + C(n - 1, p) para que o cálculo seja feito de forma
// correta sem utilizar tanto recurso da máquina.

// Fica como exercício verificar que C(n, p) = C(n - 1, p - 1) + C(n - 1, p).

int main()
{
    int N, P;

    cout << "Quantos carrinhos tem na carreta?" << endl;
    cin >> N;
    cout << "De quantas cores diferentes podem ser pintados os carrinhos?" << endl;
    cin >> P;

    while (P <= 0 || P > N){
        cout << "Como cada cor deve aparecer pelo menos uma vez, entao o numero de cores deve ser maior que zero e menor ou igual
        cin >> P;
    }

    // Supondo que todas as cores são obrigatórias, podemos resolver usando combinação com repetição.
    cout << "O numero de combinacoes distintas de corres dos carrinhos e " << Combinacao(N - 1, N - P) << "." << endl;
    // Aqui temos C(N - P + P - 1, N - P) = C(N - 1, N - P).

    return 0;
}
```

Fonte: Os autores, 2021

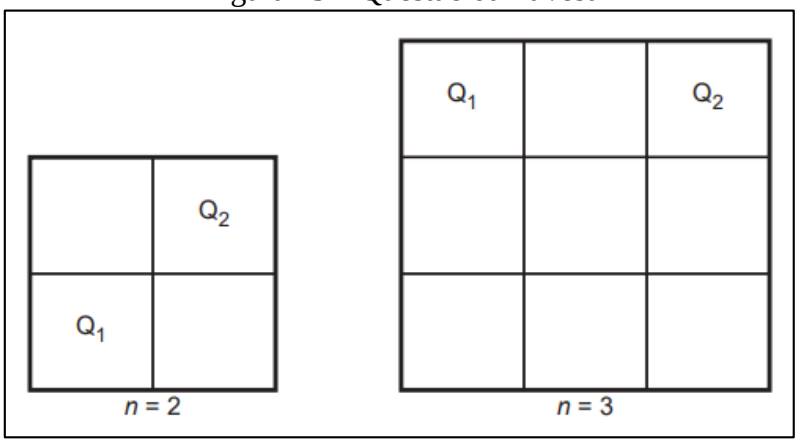
4.4 Considerações didáticas

Na resolução matemática, trabalhamos com a ideia de permutação com repetição. No fluxograma, no pensamento computacional com a etapa de abstração e na implementação vemos uma estrutura computacional que reproduz o comportamento da permutação com repetição, através de estrutura enquanto e na abstração usamos a estrutura para.

A questão 4 trata-se de um problema de contagem mas ao desenvolver a solução deste utilizamos a estratégia da classificação, pois precisamos ter pelo menos um carro de cada cor de modo a utilizar pelo menos uma vez todas as cores disponíveis. Ao generalizarmos a solução dessa questão nos deparamos com uma fórmula de permutação com repetição já ao pensarmos com o uso do pensamento computacional podemos observar que a cada cálculo de permutação ou combinação é utilizada a estrutura de repetição *para* ou *enquanto* a fim de simular o cálculo matemático. Tal problema nos faz refletir se é possível ter solução para o problema dado k cores e carros e as estratégias chamamos de existência.

Questão 5 - (FUVEST 2017) Um quadriculado é formado por $n \times n$ quadrados iguais, conforme ilustrado para $n = 2$ e $n = 3$. Cada um desses quadrados será pintado de azul ou de branco. Dizemos que dois quadrados Q_1 e Q_2 do quadriculado estão conectados se ambos estiverem pintados de azul e se for possível, por meio de movimentos horizontais e verticais entre quadrados adjacentes, sair de Q_1 e chegar a Q_2 passando apenas por quadrados pintados de azul.

Figura 23 – Questão da Fuvest



Fonte: Fuvest, 2017.

a) Se n = 2 , de quantas maneiras distintas será possível pintar o quadriculado de modo que o quadrado Q_1 do canto inferior esquerdo esteja conectado ao quadrado Q_2 do canto superior direito?

b) Suponha que n = 3 e que o quadrado central esteja pintado de branco. De quantas maneiras distintas será possível pintar o restante do quadriculado de modo que o quadrado Q_1 do canto superior esquerdo esteja conectado ao quadrado Q2 do canto superior direito?

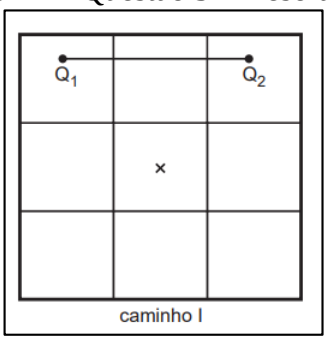
c) Suponha que n = 3. De quantas maneiras distintas será possível pintar o quadriculado de modo que o quadrado Q_1 do canto superior esquerdo esteja conectado ao quadrado Q_2 do canto superior direito?

5.1 Resolução Matemática

a) Para o caso da malha 2x2, devemos pintar Q_1 e Q_2 de azul. Cada um dos outros quadriculados pode ser ou não pintados. Mas para Q_1 esteja conectado com Q_2 é necessário pelo menos que um deles esteja pintado de azul. Devemos descartar o caso em que ambos estão pintados de branco, pois essa não é uma solução para o problema apresentado. Logo existem $2^2 - 1 = 3$ maneiras de conectar Q_1 até Q_2.

b) Para o caso da malha 3x3 em que o quadrado central esteja pintado de branco, vamos dividir o problema em dois casos. No primeiro caso em que vamos realizar um caminho direto de Q_1 até Q_2 conforme representado na figura a seguir:

Figura 24 – Questão 5 - Resolução A

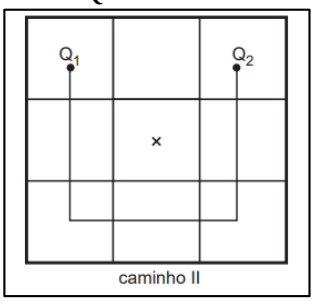


Fonte: Fuvest, 2017.

Existem 9 quadriculados. Se pensarmos que vamos realizar o caminho I, sobram 6 quadriculados, mas ainda temos que retirar o quadriculado central. Então podemos pintar as outras 5 casas de azul ou branco e assim existem 2^5 maneiras de pintar esses quadriculados.

Para o segundo caso, vamos considerar o caminho II representado na figura a seguir:

Figura 25 – Questão 5 - Resolução B



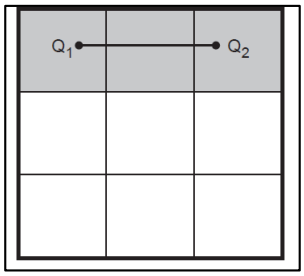
Fonte: Fuvest, 2017.

Se pensarmos que o caminho a ser realizado é diferente do caminho I e retirarmos o quadriculado central, temos apenas um caminho para chegar de Q_1 até Q_2.

Assim, no total temos $2^5 + 1$ caminhos para conectar Q_1 até Q_2.

c) Se pensarmos que vamos conectar Q_1 até Q_2 diretamente sobram 6 quadriculados para serem preenchidos como quisermos. Existem $2^6 = 64$ possibilidades.

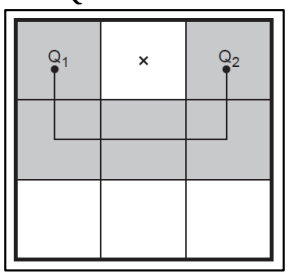
Figura 26 - Questão 5 - Resolução C



Fonte: Fuvest, 2017.

O segundo caso é pensarmos que a casa superior é pintada de branco, então as duas casas vizinhas devem ser pintadas de azul, podemos pintar as três casas restantes como quisermos tendo $2^3 = 8$ possibilidades

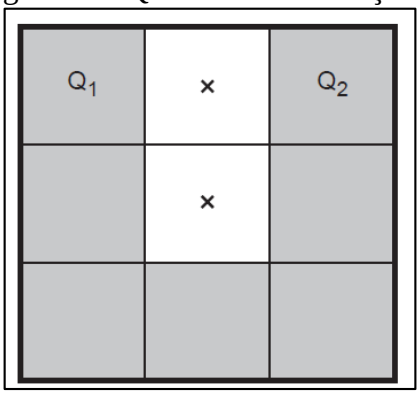
Figura 27 - Questão 5 - Resolução D



Fonte: Fuvest, 2017.

E o último caso em que pintamos a casa superior e a central de branco. Nesse caso temos apenas uma possibilidade.

Figura 28 - Questão 5 - Resolução E



Fonte: Fuvest, 2017.

Assim, existem 64 + 8 + 1 = 73 possibilidades.

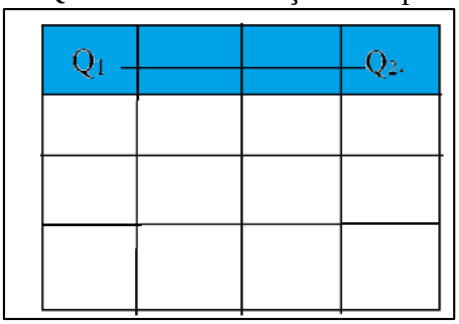
5.2 Resolução Pensamento Computacional

5A - Decomposição

Vamos observar o problema para um quadriculado 4x4.

O primeiro caso seria um caminho reto pela primeira linha conectando Q_1 à Q_2.

Figura 29 - Questão 5 - Resolução Computacional A

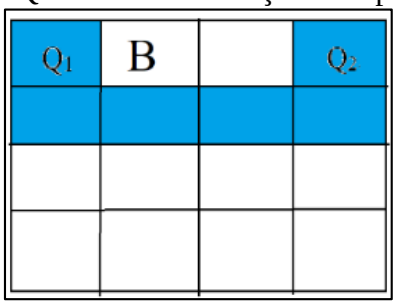


Fonte: Os autores, 2021.

Nesse caso sobram 12 quadriculados e 2^{12} possibilidades. Essa possibilidade que retiramos é quando todos os quadriculados são brancos.

O segundo caso seria em que os dois quadriculados inferiores a Q_1 e Q_2 fossem pintados de azul.

Figura 30 - Questão 5 - Resolução Computacional B

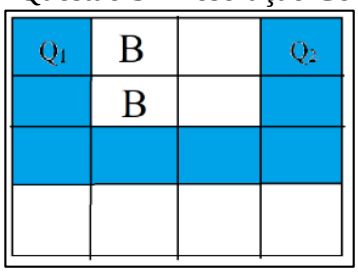


Fonte: Os autores, 2021.

Precisamos fixar um quadriculado em branco para não cairmos no primeiro caso. Sobram nesse caso 2^9 possibilidades de pintarmos os outros casos.

O terceiro caso seria o seguinte caminho:

Figura 31 - Questão 5 - Resolução Computacional C

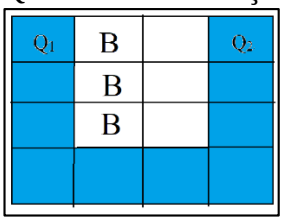


Fonte: Os autores, 2021

Precisamos fixar 2 quadriculados brancos para não cairmos no primeiro nem no segundo casos já contados. Assim sobram 6 quadriculados e 2^6 possibilidades de colorir esse quadriculado

E por último temos

Figura 32 - Questão 5 - Resolução Computacional D



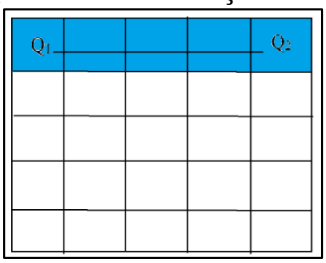
Fonte: Os autores, 2021.

Temos 3 quadriculados restantes e 2^3 possibilidades de colori-los.

Então para uma malha 4x4 podemos resolvê-la de $2^{12} + 2^9 + 2^6 + 2^3$ possibilidades.

Para n = 5, teremos uma malha com 25 quadriculados.

Figura 33 - Questão 5 - Resolução Computacional E

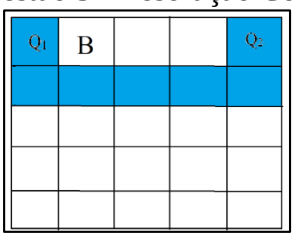


Fonte: Os autores, 2021.

Vemos que sobram 20 quadriculados e, portanto, 2^{20} possibilidades de pintá-los.

O segundo caso temos o seguinte esquema:

Figura 34 - Questão 5 - Resolução Computacional F

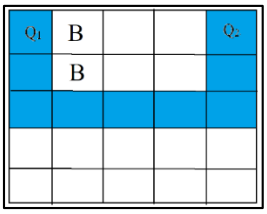


Fonte: Os autores, 2021.

Temos que garantir um quadriculado em branco na primeira linha para não recairmos no primeiro caso. Sobram então 18 quadriculados e 2^{17} possibilidades de pintura destes.

O terceiro caso é o seguinte:

Figura 35 - Questão 5 - Resolução Computacional G

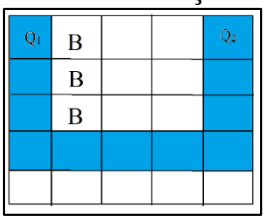


Fonte: Os autores, 2021.

Temos que pintar de branco um quadriculado na primeira e segunda linha para não cairmos nos casos anteriores. Assim sobram 14 quadriculados para serem pintados de 2^{14} maneiras.

O quarto caso é:

Figura 36 - Questão 5 - Resolução Computacional H

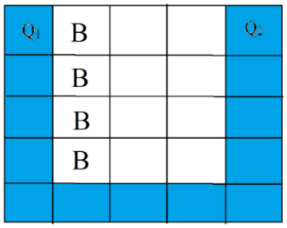


Fonte: Os autores, 2021.

Novamente, precisamos pintar um quadriculado em cada uma das três linhas anteriores para garantir que não contamos novamente os casos já analisados. Dessa maneira sobram 11 quadriculados para serem pintados de 2^{11} maneiras distintas

O quinto caso é o seguinte:

Figura 37 - Questão 5 - Resolução Computacional I



Fonte: Os autores, 2021

Novamente, vamos colocar um quadriculado branco nas quatro linhas anteriores para garantir que não recairemos nos casos já contados antes. Assim, sobram 8 quadriculados para serem pintados de 2^8 maneiras distintas.

No total temos $2^{20} + 2^{17} + 2^{14} + 2^{11} + 2^8$ possibilidades.

5B - Padrões

Observamos que dividimos o problema em casos como na letra b. E que para cada um desses casos existe uma relação entre a quantidade de possibilidades e quantas vezes precisamos repetir esse processo.

Percebemos que para n = 3, temos três casos e em cada um deles o expoente decai em razão 3. O primeiro número de possibilidades é de $2^{9-3} = 2^6$ e assim o expoente decai em três unidades em cada um dos outros processos.

Para n = 4, existem 4 subcasos. O primeiro deles tem $2^{16-4} = 2^{12}$ possibilidades e assim os próximos casos decaem o expoente em 4 unidades

Para n = 5 o mesmo acontece, temos que precisamos de 5 casos, em que o primeiro tem $2^{25-5} = 2^{20}$ casos

5C - Abstração

Diante das observações que fizemos podemos dizer que se temos uma malha quadriculada n x n, com n^2 quadriculados.

Então teremos que dividir em n casos. No primeiro caso teremos $2^{n^2 - n}$ possibilidades. No segundo caso temos $2^{n^2 - n - 3}$ maneiras, no terceiro temos $2^{n^2 - n - 6}$ e assim vamos calculando até chegar no nº caso que seria $2^{n^2 - n - 3(n-1)} = 2^{n^2 - 4n + 3}$

E o total de possibilidades seria a soma desses casos $2^{n^2 - n} + ... + 2^{n^2 - 4n + 3}$

5D - Algoritmo

```
Inicio ( )
i, n, soma: natural
Leia (n)
i = n² - n
Para 1 ≤ i ≤ n faça
  Soma = soma + 2^i
  i = i – 3
fim para
Imprimir (“Existem”, soma, “possibilidades de pintar a malha”).
Fim.
```

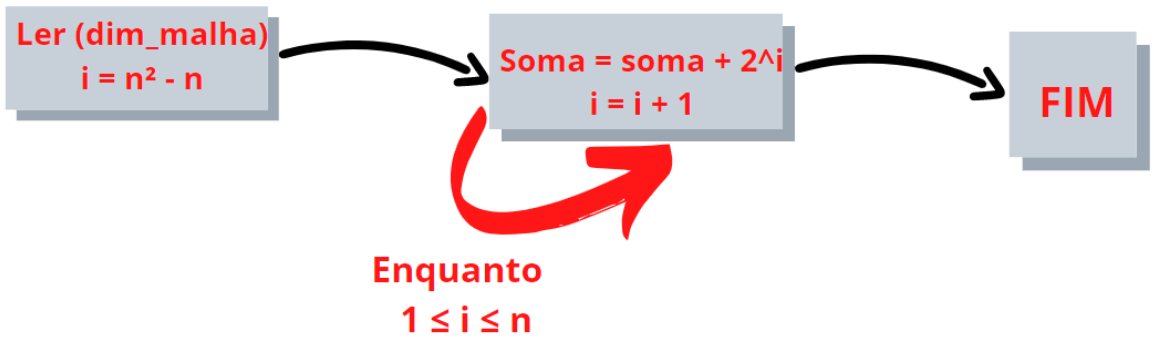
5.3 Comparando Soluções

Tabela 7 - Comparando Soluções – Questão 5

A – Resolução Matemática

a) Existem $2^2 - 1 = 3$ maneiras de conectar Q_1 até Q_2.
b) Temos $2^5 + 1$ caminhos para conectar Q_1 até Q_2.
c) Existem $64 + 8 + 1 = 73$ possibilidades.

B – Fluxograma



C – Abstração

```
Inicio ( )
    i, n, soma: natural
    Leia (n)
    i = n² - n
    Para 1 ≤ i ≤ n faça
        Soma = soma + 2^i
        i = i - 3
    fim para
    Imprimir ("Existem", soma, "possibilidades de pintar a malha").
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 38 - Implementação em linguagem C++ da Questão 5

```cpp
#include <iostream>

using namespace std;

int main()
{
    int N;
    unsigned long long int soma, expoente;

    cout << "Qual o tamanho da malha?" << endl;
    cin >> N;
    expoente = N * (N - 1);

    soma = 0;
    for (int i = 0; i < N; ++i){
        soma += 1 << expoente;
        // Maneira rápida de conseguirmos 2^expoente pela linguagem C
        expoente -= 3;
    }

    cout << "Existem " << soma << " possibilidades de pintar a malha." << endl;

    return 0;
}
```

Fonte: Os autores, 2021

5.4 Considerações didáticas

A questão 5 pode ser classificada como contagem. Por outro lado, também podemos utilizar a estratégia da enumeração para casos menores como resolução do problema para um melhor entendimento do que é pedido, bem como para ressaltar que alguns problemas de combinatória são facilitados através da estratégia de numeração. Essa estratégia de resolução auxilia no entendimento e na generalização da matemática para chegarmos à etapa de abstração. Além disso, ela também pode ser vista como uma questão que pode ser transformada de existência de modo que nos perguntamos se existe um caminho que obedeça às regras impostas pelo problema. Dentro da solução desta questão ao evidenciarmos quais eram as restrições para solução do problema estamos utilizando a estratégia de classificação dos casos possíveis.

Na resolução matemática, analisamos cada caso separadamente até encontrarmos um padrão e percebermos como era calculado para malhas maiores. Na etapa de abstração do

pensamento computacional, notamos que através de generalização dos estudos de casos conseguimos encontrar uma forma geral na qual se dá de maneira recursiva. Essa estrutura representamos através de uma estrutura computacional de repetiação que pode ser representada pela estrutura enquanto, trabalhada no fluxograma, quanto pela estrutura para, usada na implementação e na abstração. Já na etapa de implementação também usamos tal estrutura ao invés da ideia de recursividade.

Questão 6 - (Unicamp – 2002) Em Matemática, um número natural a é chamado palíndromo se seus algarismos, escritos em ordem inversa, produzem o mesmo número. Por exemplo, 8, 22 e 373 são palíndromos. Pergunta-se:

a) Quantos números naturais palíndromos existem entre 1 e 9.999?

6.1 Resolução Matemática:

Vamos dividir em casos, O primeiro caso são os palíndromos que são formados por 1 algarismo. Nesse caso temos 9 palíndromos. O segundo caso são os palíndromos formados por dois algarismos. Temos 9 também, pois devemos retirar o caso do algarismo 0.

Para os palíndromos com três algarismos, também devemos retirar o caso do algarismo 0. Nesse caso temos que pensar nos algarismos que tem por exemplo o formato 212, 222, 232, ou seja, a posição da unidade e centena é o mesmo algarismo e no meio varia.

Temos então números no formato X_X em que x é qualquer algarismo de 1 até 9 e no meio pode ser qualquer algarismo de 0 até 9. Assim temos 9,10.1 = 90 tipos de palíndromos com três algarismos. E finalmente, temos os casos com 9 palíndromos que são por exemplo do formato 2332, 4554, etc. Ou seja, a primeira e última posição devem ser igual (posição das unidades e unidade de milhar) e as posições da casa da dezena e centena devem ser iguais. Em outras palavras, é do tipo XYYX. Temos então que retirar da posição X o algarismo 0 senão o número deixa de ter quatro dígitos. Então temos 9.10.1.1 = 90 possibilidades.

No total temos 9 + 9 + 90 + 90 = 198 palíndromos de 1 até 9.999

6.2 Resolução Pensamento Computacional

6A - Decomposição

Dividimos o problema em alguns casos. O primeiro é contar os palíndromos com 1 dígito. O segundo são os palíndromos com dois, três ou quatro dígitos.

6B - Padrões

Quando dividimos em casos devemos lembrar que a partir dos números com 2 algarismos, devemos desconsiderar o zero como na posição de maior valor relativo. Então na posição de maior valor relativo sempre teremos 9 possibilidades, caso não haja nenhuma restrição no problema. Devemos lembrar também que devemos pensar no número da seguinte maneira: contamos quantos dígitos ele possui. Se for uma quantidade par então teremos metade

desses dígitos do lado esquerdo e a outra metade do lado direito terá que ser de acordo com os dígitos escolhidos para as posições no lado esquerdo. Ou vice-versa.

Se temos uma quantidade ímpar de dígitos, devemos pensar na mediana das posições para servir de referencial. A partir dela a quantidade de algarismos à esquerda é a mesma à direita. E os dígitos escolhidos à esquerda devem ser os mesmos escolhidos à direita ou vice-versa. O digito que se encontra na mediana tem 10 opções de ser escolhido.

6C - Abstração

Pensemos nos casos com 1 dígito. Podemos escolher qualquer algarismo de 1 até 9. Logo temos 9 possibilidades. Para o caso com 2 dígitos, temos 9 escolhas para a primeira posição e apenas uma para a segunda, pois depende do primeiro escolhido. No formato XX, temos 9.1 = 9 palíndromos. Para o caso de 3 dígitos, temos o formato XYX, em que X temos 9 escolhas e em Y temos 10 escolhas, isso resulta em 9.10.1 = 90 palíndromos

Para o caso de 4 dígitos temos o formato XYYX, em que temos 9 escolhas para a posição X e 10 escolhas para a posição Y resultando em 9.10.1.1 = 90 palíndromos.

Para o caso com 5 dígitos, temos o formato XYZYX, logo temos 9.10.10.1.1 = 900 palíndromos. Para o caso de 6 algarismos, temos o formato XYZZYX, que nos resulta em 9.10.10.1.1.1 = 900 palíndromos.

Se temos uma quantidade par de palíndromos, teremos n posições em que n é do formato 2k. Logo temos k posições que podemos escolher qualquer algarismo, exceto na primeira posição que deve ser 9.10.10.....10.1.1...1 = 9.10^{k-1}. Se tivermos uma quantidade ímpar de posições então n é do formato 2k + 1. Assim teremos 9.10.10....10....1.1.1...1 = 9.10^{k+1}

6D - Algoritmo

```
Inicio ( )
Leia (qtd_de_digitos)
Conta possibilidades ( )
        Se resto(i/2) = 0 então
                Possibilidades = possibilidades + 9.10^((i/2) – 1)
        Senão Possibilidades = possibilidades + 9.10^((i/2) + 1)
Fim conta possibilidades( )
Para 1 ≤ i ≤ qtd_de_digitos faça
        Conta possibilidades (i)
        i = i +1
fim para
imprimir(“existem”, possibilidades, “palíndromos”).
Fim
```

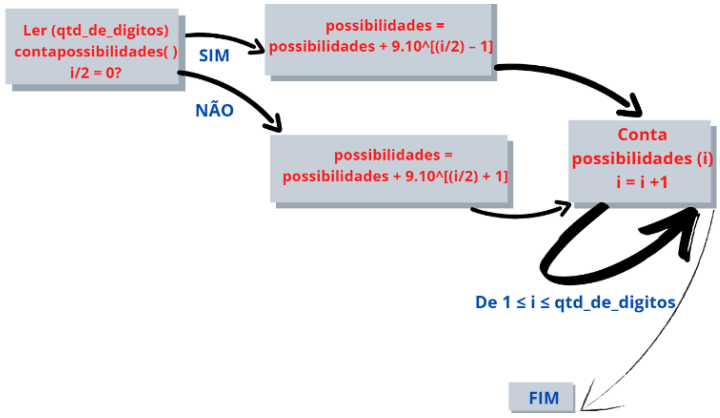
6.3 Comparando Soluções

Tabela 8 - Comparando Soluções – Questão 6

A – Resolução Matemática

De 1 até 9 – são 9 palíndromos
De 10 até 99 são 9 palíndromos
De 100 até 999 são 90 palíndromos
De 1000 até 9999 são 90 palíndromos
Logo há 9 + 9 + 90 + 90 = 198 palíndromos de 1 até 9.999

B – Fluxograma



C – Abstração

```
Inicio ( )
    Leia (qtd_de_digitos)
        Conta possibilidades ( )
        Se resto(i/2) = 0 então
        Possibilidades = possibilidades + 9.10^[(i/2) – 1]
        Senão
        Possibilidades = possibilidades + 9.10^[(i/2) + 1]
    Fim conta possibilidades( )
    Para 1 ≤ i ≤ qtd_de_digitos faça
        Conta possibilidades (i)
        i = i +1
    fim para
    imprimir("existem", possibilidades, "palíndromos").
Fim
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 39 - Implementação em linguagem C++ da Questão 6

```cpp
#include <iostream>

using namespace std;

bool ehPalindromo(int numero){
    string palindromo = to_string(numero);

    int comeco = 0;
    int final = palindromo.length() - 1;

    while (comeco < final){
        if (palindromo[comeco] != palindromo[final]){
            return false;
        }
        comeco += 1;
        final -= 1;
    }

    return true;
}

int main()
{
    int comeco, final, contador;

    cout << "De onde?" << endl;
    cin >> comeco;

   cout << "Ate onde?" << endl;
   cin >> final;

   while (final < comeco){
       cout << "O final nao pode estar antes do começo. Escolha outro numero." << endl;
       cin >> final;
   }

   contador = 0;
   for(int i = comeco; i <= final; ++i){
       if (ehPalindromo(i)){
           ++contador;
       }
   }
   cout << "Ha " << contador << " palindromos entre " << comeco << " e " << final << "." << endl;

   return 0;
}
```

Fonte: Os autores, 2021

6.4 Considerações didáticas

A questão 6 trabalha com a ideia de palíndromo e é classificada como de contagem, mas ela tem caráter enumerativo quando o aluno utiliza a enumeração ou a listagem das possíveis configurações de palíndromo para entendimento e resolução de quantidades pequenas, assim como ela tem caráter de existência, se perguntássemos ao contrário: dado uma certa quantidade de palíndromos, esta é possível dentro de um intervalo numérico de números inteiros positivos? É viável ou não?

Há ainda o caráter de classificação uma vez que separamos a resolução e abstração desse problema através de intervalos numéricos de 1 a 9, de 10 e 99, 100 a 999 e assim sucessivamente.

A resolução matemática segue uma solução em divisão de casos . Já o fluxograma e a abstração seguem uma estrutura de repetição e uma função com condicional. Já na implementação vemos o uso de uma nova estrutura que até então não havia aparecido. É usado a estrutura lógica de verdadeiro ou falso para o cálculo do palíndromo em cada caso separadamente conforme vimos nas representações anteriores. Vemos que a implementação difere um pouco da abstração e facilita em certos casos o cálculo matemático. Como por exemplo, o tempo de execução, a busca por um contraexemplo para conjecturas com cálculos mais complexos e para simulação de sistemas biológicos.

Questão 7 - (Unesp – 2003) O conselho administrativo de um sindicato é constituído por doze pessoas, das quais uma é o presidente deste conselho. A diretoria do sindicato tem quatro cargos a serem preenchidos por membros do conselho, sendo que o presidente da diretoria e do conselho não devem ser a mesma pessoa. De quantas maneiras diferentes esta diretoria poderá ser formada?

a) 40
b) 7920
c) 10890
d) 11!
e) 12!

7.1 Resolução Matemática:

Podemos resolver o problema através do princípio fundamental da contagem. O cargo de diretoria do sindicato não pode ser preenchido pelo presidente do conselho. Então sobram 11 pessoas para se candidatarem a ser o presidente da diretoria. Para o segundo cargo de diretoria, sobram 11 pessoas, pois o presidente do conselho pode ser diretor. No terceiro cargo de diretoria sobram 10 pessoas e por último 9 pessoas. Assim, temos no total 10.890 maneiras distintas de preencher esses quatro cargos.

Porém, podemos resolver essa questão também por arranjo. O cargo de presidente da diretoria há 11 opções. Mas para os outros três cargos de chefia, se mudarmos a ordem na qual essas pessoas são escolhidas, sua diretoria muda. Logo a ordem de escolha importa e por isso se trata de uma questão de arranjo. No caso temos $A_{11}^{3} = \frac{11!}{(11-3)!} = \frac{11.10.9.8!}{8!} = 990$. E finalmente temos 11.990 = 10.890. Letra c

7.2 Resolução Pensamento Computacional

7A - Decomposição

O problema tem algumas restrições e vamos começar por estas. O presidente do conselho não pode se candidatar para a presidência da diretoria. Como são 12 pessoas, então sobram 11 pessoas para esse cargo. O presidente do conselho pode se candidatar para os outros cargos de diretoria. Então temos 11 pessoas para a segunda opção de cargo de diretoria.

7B - Padrões

A ordem importa nessa questão, pois quem está como presidente da diretoria não pode estar como presidente do conselho. A pessoa que está como presidente do conselho pode se candidatar para qualquer cargo de diretoria que sobrou, três posições, então pode ter dois cargos ao mesmo tempo. Se mudarmos a ordem de escolha dos três cargos de diretoria restantes fora o presidente da diretoria, nossa solução muda, uma vez que cada cargo diretoria tem uma atribuição diferente.

Por isso esse problema pode ser resolvido tanto pelo princípio fundamental da contagem como por arranjo.

7C - Abstração

Seja n a quantidade de candidatos aos cargos de chefia. E k o número de cargos de diretoria. O presidente do conselho não pode ser presidente da diretoria. Assim existem n – 1 candidatos ao cargo de presidente da diretoria. Já para o segundo cargo de diretoria também temos n -1, pois o presidente do conselho pode tentar esse cargo. Para o terceiro cargo de diretoria temos n – 2 candidatos. Para o quarto cargo temos n – 3. Esse padrão se repete até o k-ésimo cargo de diretoria. No k-ésimo cargo, temos n – k – 1 pessoas para se candidatar.

7D - Algoritmo

```
Inicio ( )
n,k: naturais
Leia (n, k)
Produtorio = (n – 1)
i = n - 1
Para (n – 1 ≤ i ≤ n – k – 1)  faça
        Produtorio = produtorio* i
        Imprimir(produtorio)
Fim para
            Imprimir (“Existem”, produtorio, “possibilidades”)
Fim.
```

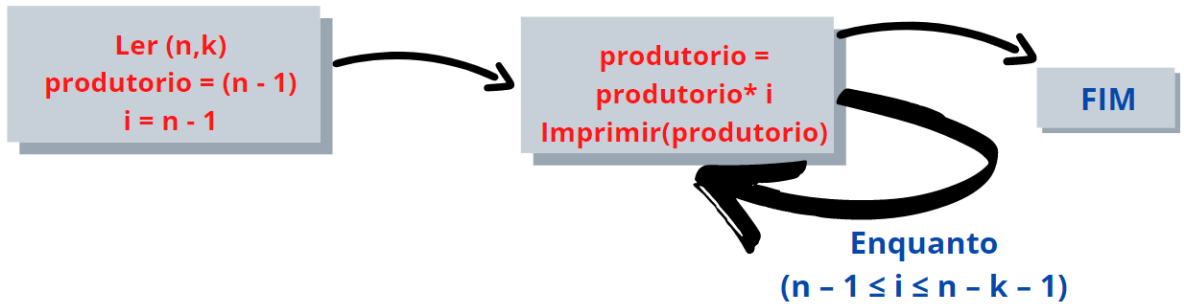
7.3 Comparando Soluções

Tabela 9 - Comparando Soluções – Questão 7

A – Resolução Matemática

O cargo de diretoria do sindicato não pode ser preenchido pelo presidente do conselho. Então sobram 11 pessoas para se candidatarem a presidência da diretoria. Para o segundo cargo de diretoria, sobram 11 pessoas, pois o presidente do conselho pode ser diretor. No terceiro cargo de diretoria sobram 10 pessoas e por último 9 pessoas. Assim, temos no total 10.890 maneiras distintas de preencher esses quatro cargos.

B – Fluxograma



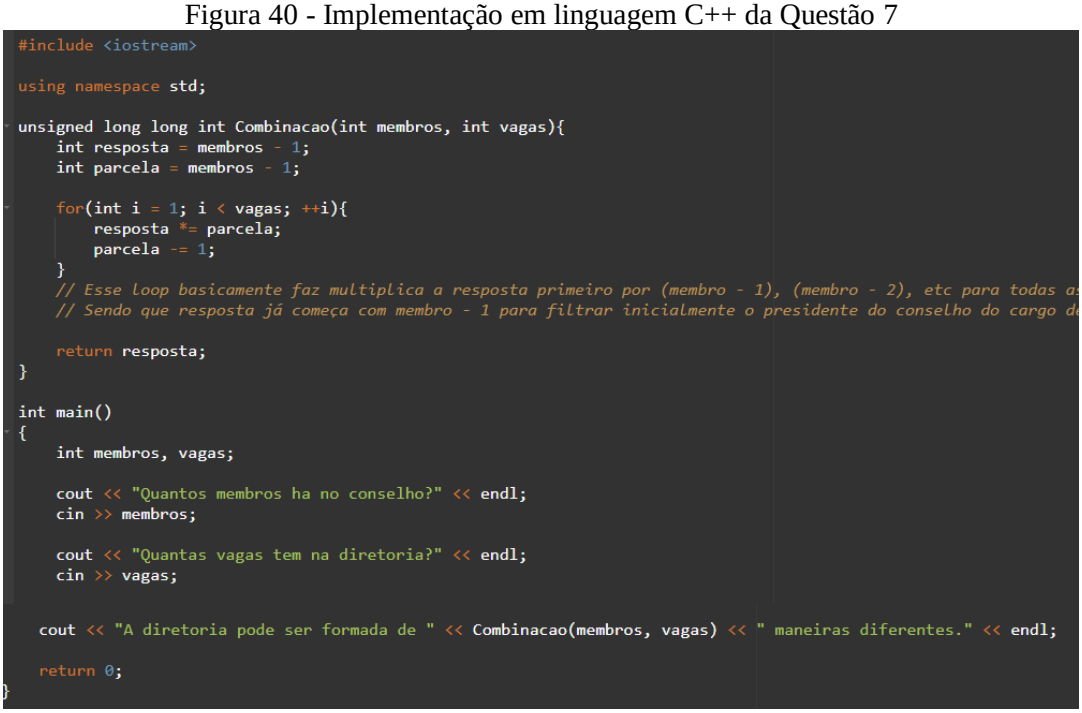
C – Abstração

```
Inicio ( )
    n,k: naturais
    Leia (n, k)
    Produtorio = (n - 1)
    i = n - 1
    Para (n - 1 ≤ i ≤ n - k - 1)  faça
        Produtorio = produtorio* i
        Imprimir(produtorio)
    Fim para
    Imprimir ("Existem", produtorio, "possibilidades")
Fim.
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 40 - Implementação em linguagem C++ da Questão 7



```
#include <iostream>

using namespace std;

unsigned long long int Combinacao(int membros, int vagas){
    int resposta = membros - 1;
    int parcela = membros - 1;

    for(int i = 1; i < vagas; ++i){
        resposta *= parcela;
        parcela -= 1;
    }
    // Esse loop basicamente faz multiplica a resposta primeiro por (membro - 1), (membro - 2), etc para todas as
    // Sendo que resposta já começa com membro - 1 para filtrar inicialmente o presidente do conselho do cargo de

    return resposta;
}

int main()
{
    int membros, vagas;

    cout << "Quantos membros ha no conselho?" << endl;
    cin >> membros;

    cout << "Quantas vagas tem na diretoria?" << endl;
    cin >> vagas;


   cout << "A diretoria pode ser formada de " << Combinacao(membros, vagas) << " maneiras diferentes." << endl;

   return 0;
}
```

Fonte: Os autores, 2021

7.4 Considerações didáticas

A questão 7 trata de um problema de contagem, porém que pode ser resolvido através de enumeração, ainda que de maneira não eficiente para valores mais altos. Tem características de combinatória de existência, visto que para a solução ser viável é necessário que exista a quantidade de pessoas mínimas necessárias para o cálculo do problema e pode ser visto como com características de classificação, uma vez que existem condições para que o problema possa ser resolvido: o presidente da diretoria e do conselho não podem ser a mesma pessoa. Isso faz com que o problema de contagem tenha uma restrição, uma regra para sua solução existir. Na implementação dessa questão, podemos perceber que a etapa de abstração da solução matemática segue o mesmo raciocínio que a abstração no pensamento computacional para a construção do algoritmo.

O problema pode ser resolvido pelo princípio fundamental da contagem ou por arranjo. O cálculo do arranjo pode ser representado pela estrutura de repetição *para*, tanto na abstração, quanto no fluxograma. Já na implementação em linguagem C++ é usada uma função para o cálculo das possibilidades e a estrutura de repetição *para* no cálculo total. Apesar de parecidas

as abordagens possuem diferenças sutis tais como não ser necessário escrever mais de uma vez a variável soma. quando realizamos soma a variável mais uma vez.

Questão 8 - De quantos modos se pode dispor 5 moças e 5 rapazes em torno de uma mesa circular de modo que não fiquem juntos nem 2 rapazes, nem 2 moças?

8.1 Resolução Matemática:

Vamos fixar os homens e permutar as mulheres de 5! = 120 maneiras. Já os homens permutam circularmente de PC(5) = 4! = 24 maneiras. Logo existem 120.24 = 2880 maneiras.

8.2 Resolução Pensamento Computacional

8A – Decomposição

Devemos separar homens e mulheres em dois grupos distintos e analisá-los da seguinte maneira:

Os homens estão dispostos em uma mesa circular, então se trata de uma permutação circular. Como são 5 homens, há 4! maneiras.

Já, para permutar as mulheres é possível imaginar como se estivessem em fila e com isso tratamos como permutação.

8B - Padrões

Quando nos deparamos com problemas em que precisamos organizar n objetos distintos em n lugares que estão igualmente espaçados sob um círculo considerado iguais as sequências de objetos que possam coincidir por rotação temos um problema de permutação circular. No caso do problema proposto, queremos permutar 5 homens em 5 cadeiras de formato circular e então resolvemos por permutação com repetição. E ao fixarmos homens, tratamos as mulheres como se estivessem numa fila e fossemos permutá-las entre as cadeiras. Nesse caso se trata de uma permutação sem repetição.

8C – Abstração

Vamos imaginar que temos n homens e m mulheres. Então trataremos as mulheres com permutação simples. Logo existem m! maneiras de dispor todas as moças de modo que uma não fique ao lado da outra.

Por outro lado, trataremos os n homens como permutação circular. Nesse caso existem (n – 1) maneiras de permutá-los numa mesa circular.

No total teremos (n – 1)!m! maneiras.

8D – Algoritmo

```
Inicio ( )
Leia(n,m)
Produtorio,i = 1
Para (1 ≤ i ≤ n -1) faça
        Produtorio = produtorio * i
        Imprimir (produtorio)
Fim para
Para (1 ≤ j ≤ m) faça
        Produtorio = produtorio * j
        Imprimir (produtorio)
Fim para
Fim
```

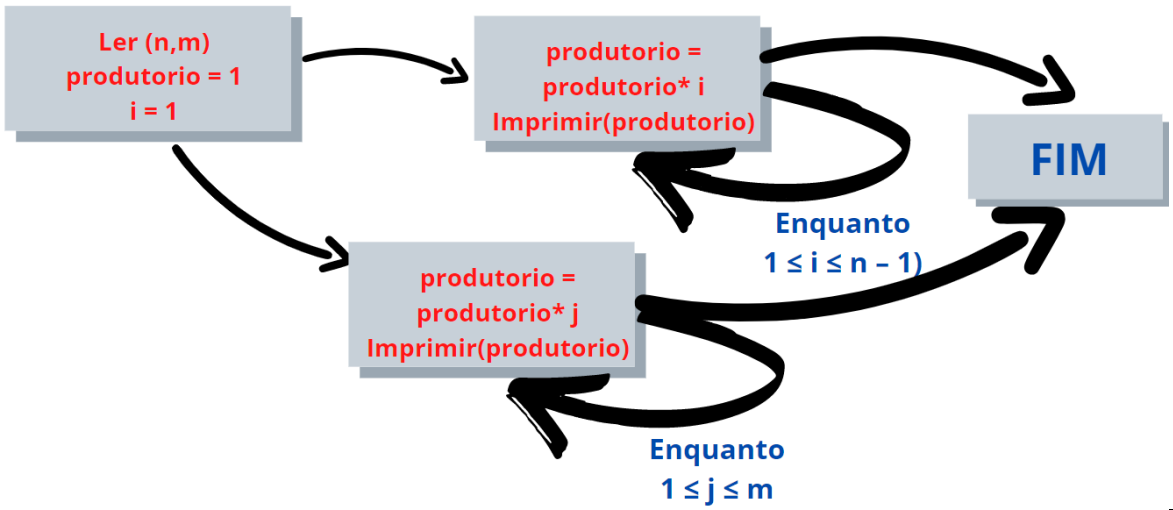
8.3 Comparando Soluções

Tabela 10 - Comparando Soluções – Questão 8

A – Resolução Matemática

Vamos fixar os homens e permutar as mulheres de 5! = 120 maneiras. Já os homens permutam circularmente de PC(5) = 4! = 24 maneiras. Logo existem 120.24 = 2880 maneiras.

B – Fluxograma



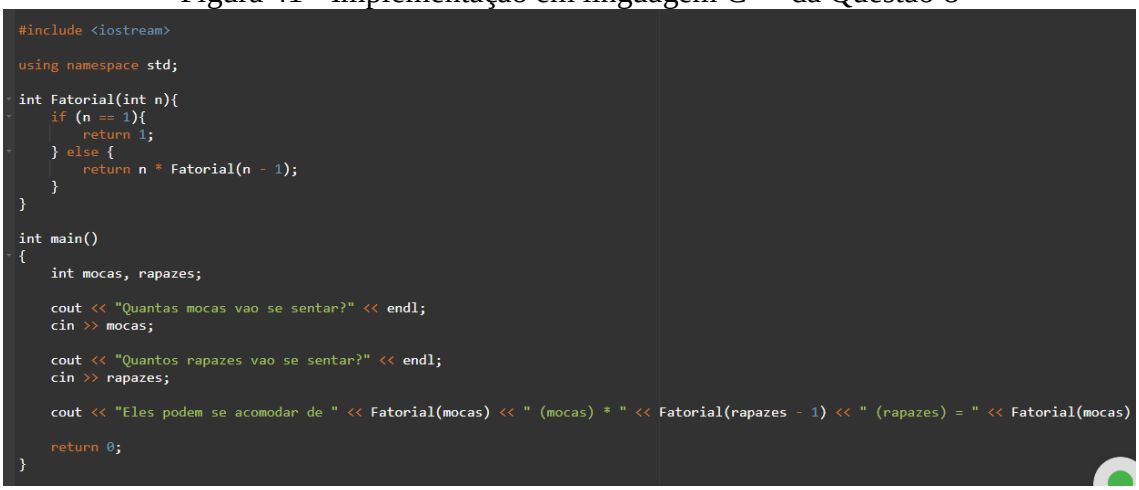
C – Abstração

```
Inicio ( )
    Leia(n,m)
    Produtorio,i = 1
    Para (1 ≤ i ≤ n -1) faça
        Produtorio = produtorio * i
        Imprimir (produtorio)
    Fim para
    Para (1 ≤ j ≤ m) faça
        Produtorio = produtorio * j
        Imprimir (produtorio)
    Fim para
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 41 - Implementação em linguagem C++ da Questão 8



```
#include <iostream>

using namespace std;

int Fatorial(int n){
    if (n == 1){
        return 1;
    } else {
        return n * Fatorial(n - 1);
    }
}

int main()
{
    int mocas, rapazes;

    cout << "Quantas mocas vao se sentar?" << endl;
    cin >> mocas;

    cout << "Quantos rapazes vao se sentar?" << endl;
    cin >> rapazes;

    cout << "Eles podem se acomodar de " << Fatorial(mocas) << " (mocas) * " << Fatorial(rapazes - 1) << " (rapazes) = " << Fatorial(mocas)

    return 0;
}
```

Fonte: Os autores, 2021

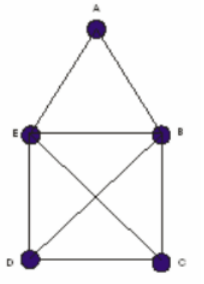
8.4 Considerações didáticas

Enquanto na resolução matemática usamos ideia de permutação, no fluxograma e na abstração utilizamos uma estrutura equivalente a essa que é a de repetição *para*. Essa estrutura realiza comandos de repetição de ações. No caso desse problema, repete o produto entre número no cálculo do fatorial. Já na implementação computacional essa repetição foi resolvida através de uma recursão, o que diminui a complexidade do programa, bem como o tempo de execução de seu resultado.

A questão 12 trata de um problema de contagem que envolve permutação circular e para resolvê-lo precisamos obedecer a algumas regras na hora de dispor homens e mulheres sentados em torno da mesa circular. Por isso, esse problema tem característica ou natureza de classificação e pode ser resolvido através do método da exaustão, ou também conhecido como enumeração de todos os casos. Tal resolução pode não ser eficiente quando manualmente realizada, mas dependendo da quantidade de pessoas envolvidas a enumeração pode ser calculada computacionalmente e auxiliar no raciocínio combinatório do aluno, assim como no entendimento da questão, pois ele pode observar as possíveis configurações de resposta. Por fim, essa questão pode ser considerada de existência: para que haja solução é necessário ter condições mínimas iniciais para que possamos realizar o cálculo ou ainda podemos refletir o contrário: se dado uma quantidade m de pessoas nas quais n são mulheres e y são homens, existe solução de modo a sentar homens e mulheres alternadamente? A implementação nesse caso do algoritmo foi análoga à abstração matemática apresentada na questão.

Questão 9 - É possível desenhar a figura a seguir sem retirar o lápis do papel e sem repetir nenhuma linha?

Figura 42 – Questão 9



Fonte: Os autores, 2021

9.1 Resolução Matemática:

Podemos iniciar do vértice A, B, C, D ou E. Temos 8 arestas no total e assim 8! maneiras de escolher esse caminho iniciando em alguma dessas arestas. Vamos enumerar alguns exemplos:

DE → EA → AB → BC → CD → DB → BE → EC essa é uma possível configuração da existência de um caminho sem retirar o lápis do papel. Trata-se de um caminho Euleriano. Sabemos que esse problema tem solução porque temos uma quantidade par de vértices de grau ímpar. No caso temos dois vértices de grau 3, ou seja, os vértices D e C.

9.2 Resolução Pensamento Computacional

9A – Decomposição

Primeiro podemos representar as conexões entre os vértices através de uma matriz de adjacência, em que atribuiremos 0 quando não existir arestas e 1 quando houver aresta entre os vértices em questão.

Podemos percorrer na matriz entre os vértices i e j que possuírem $a_{ij} = 1$ os relacionando.

Figura 43 – Resolução computacional

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	1	1
C	0	1	0	1	1
D	0	1	1	0	1
E	1	1	1	1	0

Fonte: Os autores, 2021.

9B – Padrões

Nenhum vértice tem caminho para si mesmo, o que é chamado de laço na Teoria de Grafos. A matriz de adjacência é simétrica, ou seja, AB tem o mesmo valor que BA na matriz.

9C – Abstração

Para uma matriz de adjacência, podemos partir de qualquer vértice. Vamos começar do A. Vamos ver onde A tem ligação com outro vértice. Ele tem ligação com B e. Vamos para o B. A partir de B, podemos ir para C, D ou E. Decidimos ir para o C. A partir do C podemos ir para o D. E do vértice D podemos ir para o vértice E. Dessa maneira fizemos o caminho AB → BC → CD → DE → EA. Mas esse caminho não abrange todas as arestas existentes no grafo.

Vamos partir de um grafo de grau 3. Para ver isso, basta somarmos em cada linha da matriz para ver qual tem grau ímpar. No caso temos a linha do vértice D e C.

Se partirmos de C, vamos para o vértice B (CB). De B partimos para A (BA). De A vamos para E (AE). De E vamos para B (EB) . De B só podemos ir para D (BD). De D vamos para C (DC). De C só podemos ir para E (CE). E finalmente de E vamos para D (ED). Temos então o seguinte caminho CB → BA → AE → EB → BD → DC → CE → ED e assim passamos por todos as arestas sem repeti-las.

9D – Algoritmo

Para realizar o caminho anterior, utilizamos o algoritmo de Fleury. Nele partimos de um vértice e partir desse vamos apagando as arestas por onde passamos. Como estamos pensando na implementação desse problema, representamos essas arestas através de uma matriz de adjacência a seguir:

Figura 44 – Resolução computacional – Etapa 1

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	1	1
C	0	1	0	1	1
D	0	1	1	0	1
E	1	1	1	1	0

Fonte: Os autores, 2021.

A partir dela vemos o grau de cada vértice. Para sabermos isso, basta somarmos o valor de cada linha. Veremos que os vértices C e D possuem grau três que é ímpar. Vamos iniciar do vértice C. A partir de C decidimos ir para o vértice B. Com isso zeramos na matriz esses espaços que destacamos em vermelho a seguir. Ao zerarmos estamos dizendo que já passamos por aquela aresta e não podemos repeti-la.

Figura 45 – Resolução computacional – Etapa 2

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	0	1	1
C	0	0	0	1	1
D	0	1	1	0	1
E	1	1	1	1	0

Fonte: Os autores, 2021.

Em B vamos para o vértice A. Assim apagamos a aresta AB na matriz:

Figura 46 – Resolução computacional – Etapa 3

	A	B	C	D	E
A	0	0	0	0	1
B	0	0	0	1	1
C	0	0	0	1	1
D	0	1	1	0	1
E	1	1	1	1	0

Fonte: Os autores, 2021

.

Em A partimos para o vértice E zeramos a aresta AE:

Figura 47 – Resolução computacional – Etapa 4

	A	B	C	D	E
A	0	0	0	0	0
B	0	0	0	1	1
C	0	0	0	1	1
D	0	1	1	0	1
E	0	1	1	1	0

Fonte: Os autores, 2021.

Em E partimos para o vértice B e apagamos a aresta EB:

Figura 48 – Resolução computacional – Etapa 5

	A	B	C	D	E
A	0	0	0	0	0
B	0	0	0	1	0
C	0	0	0	1	1
D	0	1	1	0	1
E	0	0	1	1	0

Fonte: Os autores, 2021.

Em B partimos para o vértice D e apagamos a aresta BD:

Figura 49 – Resolução computacional – Etapa 6

	A	B	C	D	E
A	0	0	0	0	0
B	0	0	0	0	0
C	0	0	0	1	1
D	0	0	1	0	1
E	0	0	1	1	0

Fonte: Os autores, 2021.

Em D vamos para o vértice C e apagamos a aresta DC:

Figura 50 – Resolução computacional – Etapa 7

	A	B	C	D	E
A	0	0	0	0	0
B	0	0	0	0	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	1	1	0

Fonte: Os autores, 2021.

Em C só podemos ir para o vértice E apagamos a aresta CE:

Figura 51 – Resolução computacional – Etapa 8

	A	B	C	D	E
A	0	0	0	0	0
B	0	0	0	0	0
C	0	0	0	0	0
D	0	0	0	0	1
E	0	0	0	1	0

Fonte: Os autores, 2021.

Em E só podemos ir para D e apagamos a aresta DE:

Figura 52 – Resolução computacional – Etapa 9

	A	B	C	D	E
A	0	0	0	0	0
B	0	0	0	0	0
C	0	0	0	0	0
D	0	0	0	0	0
E	0	0	0	0	0

Fonte: Os autores, 2021.

Assim, passamos por todas as arestas sem repetir nenhuma e chegamos na matriz nula que significa que esgotamos todos os caminhos possíveis do grafo e finalizamos o algoritmo de Fleury.

```
Inicio ( )
        Leia(grafo)
        aloca(grafo)
        enquanto nao percorrer todo(grafo)
                se (vértice a_i tem ligação com outro vértice e não passou por ele) então
                        vá para o vértice a_j e guarda(caminho)
                senão
                        (vértice a_i tem ligação com outro vértice e não passou por ele)
                então
                        vá para o vértice a_k e guarda(caminho)
                senão
                        i = i + 1
        fim enquanto
        imprime(caminho)
Fim ( )
```

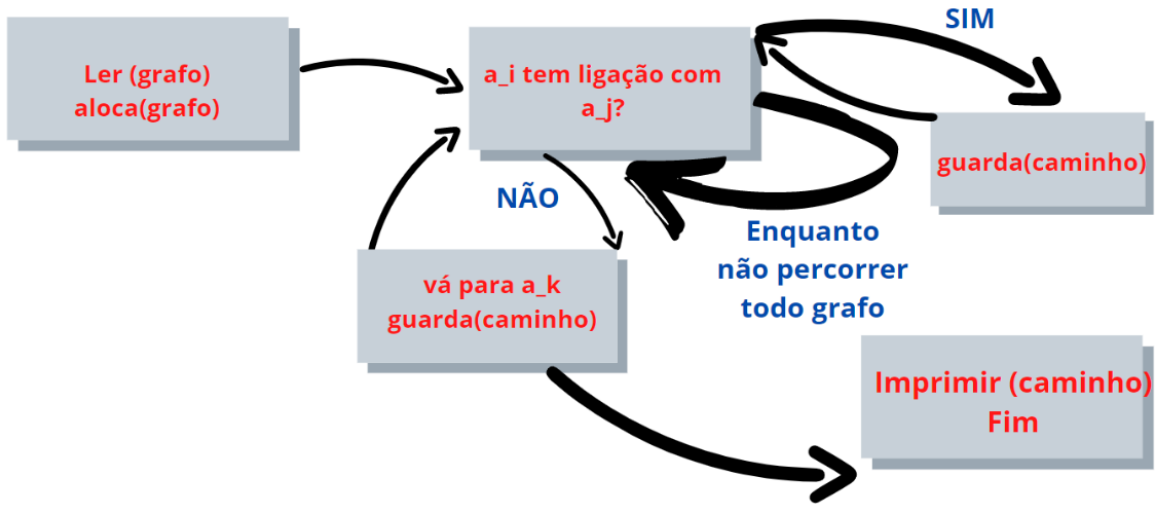
9.3 Comparando Soluções

Tabela 11 - Comparando Soluções – Questão 9

A – Resolução Matemática

Podemos iniciar do vértice A, B, C, D ou E. Temos 8 arestas no total e assim 8! maneiras de escolher esse caminho iniciando em alguma dessas arestas.
Vamos enumerar alguns exemplos:
DE → EA → AB → BC → CD → DB → BE → EC essa é uma possível configuração da existência de um caminho sem retirar o lápis do papel. Trata-se de um caminho euleriano. Sabemos que esse problema tem solução porque temos uma quantidade par de vértices de grau ímpar. No caso temos dois vértices de grau 3, ou seja, os vértices D e C

B – Fluxograma



C – Abstração

```
Inicio ( )
    Leia(grafo)
    aloca(grafo)
    enquanto nao percorrer todo(grafo)
        se (vertice a_i tem ligacao com outro vertice e nao passou por ele) entao
            vá para o vertice a_j e guarda(caminho)
        senao
            (vertice a_i tem ligacao com outro vertice e nao passou por ele) entao
            va para o vertice a_k e guarda(caminho)
        senao
        i = i + 1
    fim enquanto
    imprime(caminho)
Fim ( )
```

Fonte: Os autores, 2021.

D – Implementação – Linguagem C ++

Figura 53 - Implementação em linguagem C++ da Questão 9

```
#include <iostream>
#include <vector>

using namespace std;

vector<vector<int> > caminhos;

bool Eureliano(vector<vector<int> > desenho){
    int contador = 0;
    for(int i = 1; i < desenho.size(); ++i){
        if (desenho[i].size() % 2 == 1){
            contador += 1;
            if (contador > 2){
                return false;
            }
        }
    }
    if (contador == 1){
        return false;
    }
    return true;
}
// Só é considerado um grafo eureliano se tiver zero ou dois vértices de grau ímpar.

void Solucao(vector<vector<int> > desenho, vector<int> caminho, int u, int v, int m){
    caminho.push_back(v);
    // Toda vez que olhamos um vértice devemos colocá-lo no caminho em questão.

    desenho[u].erase(remove(desenho[u].begin(), desenho[u].end(), v), desenho[u].end());
    desenho[v].erase(remove(desenho[v].begin(), desenho[v].end(), u), desenho[v].end());
    // E retiramos a aresta do nosso grafo temporariamente para não repeti-la.

    if (caminho.size() == (m + 1)){
        for (int i = 1; i < desenho.size(); ++i){
            if (desenho[i].size() > 0){
                return;
            }
        }
        // Verificamos se todas as arestas foram cobertas pelo caminho.
        caminhos.push_back(caminho);
    }else{
        for(int i = 0; i < desenho[v].size(); ++i){
            int vizinho = desenho[v][i];

            // Vamos para o vizinho verificar o caminho através dele.
            Solucao(desenho, caminho, v, vizinho, m);
        }
    }
}
// Basicamente, fazemos uma busca em profundidade para encontrarmos todos os caminhos válidos, que passam por todas as arestas sem repe

int main()
{
    vector<vector<int> > desenho;
    int n, m, u, v;

    cout << "Quantos vertices ha no desenho?" << endl;
    cin >> n;
    for(int i = 0; i <= n; ++i){
        desenho.push_back(vector<int>());
    }

    cout << "Quantas arestas ha no desenho?" << endl;
    cin >> m;

    cout << "Descreva as " << m << " arestas do seu desenho. Exemplo: 1 3 significa uma aresta entre os vertices 1 e 3." << endl;
    for(int i = 0; i < m; ++i){
        cin >> u >> v;
        desenho[u].push_back(v);
        desenho[v].push_back(u);
    }

    if (Eureliano(desenho)){
        for(int i = 1; i < desenho.size(); ++i){
            vector<vector<int> > outro_desenho(desenho);
            outro_desenho[0].push_back(i);
            outro_desenho[i].push_back(0);
            // Criamos um grafo temporário apenas para nos auxiliar na nossa busca.
            // E criamos uma aresta auxiliar prum vértice imaginário só para podermos começar a busca de um vértice específico.
    if (Eureliano(desenho)){
        for(int i = 1; i < desenho.size(); ++i){
            vector<vector<int> > outro_desenho(desenho);
            outro_desenho[0].push_back(i);
            outro_desenho[i].push_back(0);
            // Criamos um grafo temporário apenas para nos auxiliar na nossa busca.
            // E criamos uma aresta auxiliar prum vértice imaginário só para podermos começar a busca de um vértice específico.
            Solucao(desenho, vector<int>(), 0, i, m);
        }
        if (caminhos.size() > 0){
            cout << "Estas sao as maneiras possiveis de fazer o desenho sem retirar o lapis do papel nem repetir nenhuma linha:" << endl;
            for(int i = 0; i < caminhos.size(); ++i){
                cout << caminhos[i][0];
                for(int j = 1; j < caminhos[i].size(); ++j){
                    cout << " -> " << caminhos[i][j];
                }
                cout << endl;
            }
        } else {
            cout << "Nao e possivel fazer esse desenho sem tirar o lapis da folha e sem passar por cima de uma linha." << endl;
        }
    } else {
        cout << "Nao e possivel fazer esse desenho sem tirar o lapis da folha e sem passar por cima de uma linha." << endl;
    }

    return 0;
}
```

Fonte: Os autores, 2021.

9.4 Considerações didáticas.

A questão 13 é classificada como de natureza combinatória de existência, uma vez que o comando: existe um caminho Euleriano? Esse tipo de questão de combinatória de existência observamos que ela possui característica de enumeração, porque para encontrarmos um caminho que passe apenas uma vez por cada vértice, necessitamos testar possíveis caminhos existentes entre os vértices e eliminar aqueles que não satisfazem à regra. Uma maneira de realizar essa questão é enumerar todos os caminhos existentes.

Na solução pelo pensamento computacional, ao pensarmos na implementação desse problema observaremos que a característica de enumerar todos os possíveis caminhos pode ser otimizada computacionalmente pelo comando *se*, uma estrutura de comando condicional, reduzimos o tempo de busca da solução ótima. Por termos regras de quais caminhos podemos enumerar, essa questão também possui natureza de classificação, pois a solução existe apenas se seguirmos as condições expressas na questão, o qual é passar em cada vértice apenas uma vez sem repetir nenhum deles e voltar ao vértice de origem. Ao pensarmos na abstração dessa questão, nos deparamos com uma possível solução com o auxílio do uso de tabelas, na qual marcamos as conexões existentes com o número 1 entre os vértices e 0 quando não existentes e através desta tabela, seguindo um raciocínio algorítmico, ou seja, de procedimentos, conseguimos encontrar um caminho Euleriano

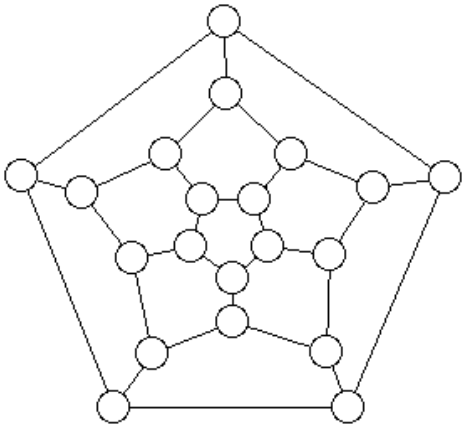
Já na implementação algorítmica dessa questão observamos a noção de uma estrutura de dados usada na área da Ciência da Computação e que é comum a matemática: Teoria dos Grafos. Tal teoria e estrutura computacional são eficientes em diversas situações e problemas tais como: de caminho ou percursos eulerianos, combinatória de existência em caminhos, e em combinatória de otimização. Ao simularmos na tabela um possível caminho de encontrar tal solução, através de uma estratégia lógica, tal ação nos auxilia no entendimento do que a máquina realiza; como é feita a transcrição da linguagem matemática para linguagem computacional; como podemos representar equivalentemente problemas da mesma natureza em outras palavras toda vez que nos deparamos com uma questão de caminho na qual precisamos saber se existe um percurso Euleriano basta pensarmos numa Matriz de adjacência e repetirmos os processos algorítmicos apresentados na solução da questão.

Na solução matemática, encontramos uma estratégia de percorrer o grafo todo sem repetir vértices. Posteriormente, na etapa de abstração buscamos uma outra estratégia de solução com o auxílio de uma matriz e percorremos a matriz até passarmos por todos os vértices sem repetir nenhum deles. Já na etapa de abstração buscamos representar essa ideia de uma

maneira mais geral. No fluxograma a ideia seguida foi a mesma usada na abstração. Já na implementação foi utilizado o auxílio de estrutura de vetores para armazenar os caminhos percorridos, remover os que não obedeciam às regras e de estruturas de lógica. Podemos perceber pela implementação que esse tipo de problema não é de simples solução e quanto maior for a matriz a ser percorrida, mais tempo é necessário para encontrar a solução. A implementação foi capaz de generalizar a ponto de que o aluno diga qual é o formato do grafo. Então ele pode ser usado para outros grafos.

Questão 10 - Um passeio escolar levará 20 alunos do Colégio Pedro II em Cabo Frio para conhecer a Praia do Forte e o Bairro da Passagem com toda sua história da cidade. O ônibus vai buscar cada aluno em sua casa. Precisa economizar gasolina de modo a passar apenas uma vez em cada casa. O esquema das ruas e das casas está na figura a seguir:

Figura 54 – Questão 10



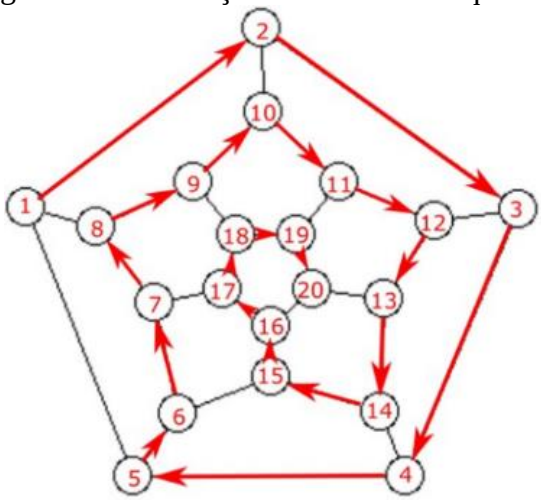
Fonte: Os autores, 2021.

É possível que o motorista consiga realizar tal feito?

10.1 Resolução Matemática:

Mostramos a seguir uma possível solução para o problema.

Figura 55 – Resolução matemática da questão 10



Fonte: Os autores, 2021

10.2 Resolução Pensamento Computacional

10A – Decomposição

Precisamos encontrar um caminho que passe por todos os vértices uma única vez. Esse tipo de caminho é conhecido como Hamiltoniano.

10B - Padrões

Ao passar por uma aresta, podemos apagar esse vértice da matriz para não o repetir na matriz de incidência.

10C - Abstração

Para um grafo com n vértices, queremos encontrar um caminho que passe por todos os vértices sem repetir nenhum. Tal caminho terá n – 1 arestas utilizadas.

10D - Algoritmo

Podemos utilizar o algoritmo do caixeiro viajante para descobrir esse caminho. Para isso, montamos uma matriz de adjacência em que 0 significa que não existe ligação entre os vértices e 1 é a ligação através de uma aresta.

Figura 56 – Matriz de Adjacência da questão 10

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
3	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
5	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0
8	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0
10	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0
12	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1
14	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
15	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1
17	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0
18	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0
19	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1
20	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0

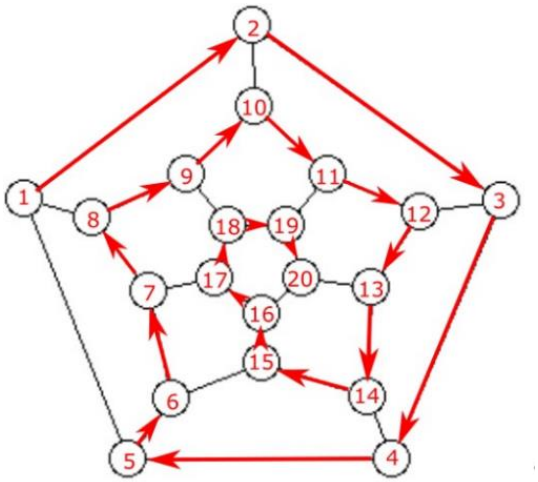
Iniciamos o caminho no vértice rotulado como 1. A partir dele vamos realizando um caminho e a cada passo que damos vamos zerando na matriz acima as ligações já utilizadas.

Como já usamos o vértice 1 então não voltaremos nele, pois a ideia é não repetir vértices. Vamos para o vértice 2 e seguimos em diante. Ou seja, para cada passo que damos não voltamos na linha (ou coluna) visitada na matriz. A matriz é simétrica em relação a diagonal principal, que é formada por zeros. Isso significa que podemos percorrer a matriz tanto em linha, quanto em coluna e teremos o mesmo resultado.

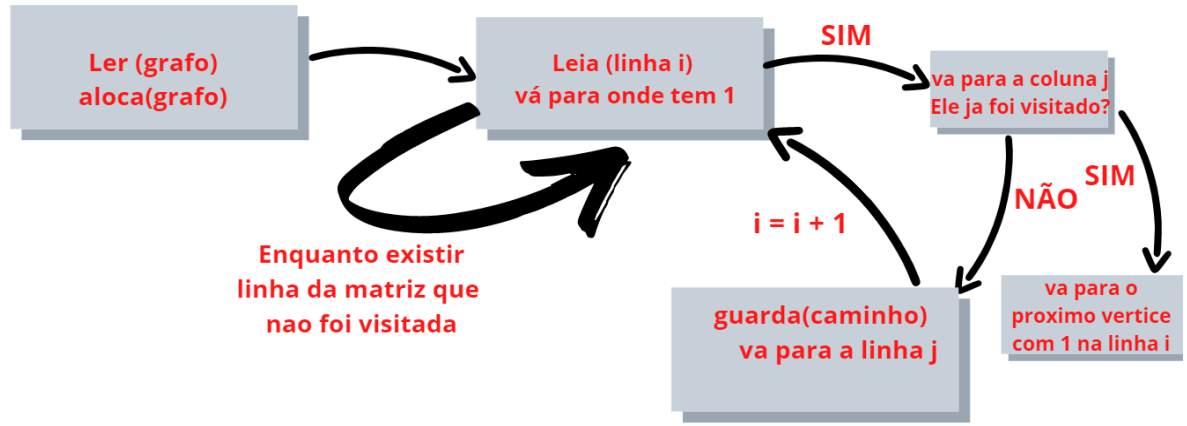
10.3 Comparando Soluções

Tabela 12 - Comparando Soluções – Questão 10

A – Resolução Matemática



B – Fluxograma



C – Abstração

```
Inicio ( )
    Leia(grafo)
    Aloca(matriz_grafo)
    Enquanto (existir linha da matriz nao visitada faça)
        leia linha i
        onde tem 1? (significa que existe ligacao entre esse vertice da linha e a coluna correspondente)
        va para a coluna j
        esse vertice ja foi visitado?
            se sim entao va para o proximo vertice
        senao guarda(caminho) e va para a linha j
        fim se
        i = i + 1
    fim enquanto
fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 57 - Implementação em linguagem C++ da Questão 10

```cpp
#include <iostream>
#include <vector>

using namespace std;

vector<vector<int> > mapa;
int n, m;
vector<vector<int> > caminhos;

void Hamiltoniano(vector<int> caminho, int u, vector<bool> visitados){
    caminho.push_back(u);
    if (caminho.size() == n){
        // Se nosso caminho tem todos os vértices, temos um caminho hamiltoniano.
        caminhos.push_back(caminho);
    } else {
        visitados[u] = true;
        for(int i = 0; i < mapa[u].size(); ++i){
            int vizinho = mapa[u][i];

            if (!visitados[vizinho]){
                // Se o vizinho não for visitado até então, podemos visitá-lo.
                Hamiltoniano(caminho, vizinho, visitados);
            }
        }
    }
}

int main()
{
    vector<bool> visitados;
    int u, v;

    cout << "Quantos vertices ha no mapa?" << endl;
    cin >> n;
    for(int i = 0; i <= n; ++i){
        mapa.push_back(vector<int>());
        visitados.push_back(false);
    }

    cout << "Quantas arestas ha no desenho?" << endl;
    cin >> m;

    cout << "Descreva as " << m << " arestas do seu desenho. Exemplo: 1 3 significa uma aresta entre os vertices 1 e 3." << endl;
    for(int i = 0; i < m; ++i){
        cin >> u >> v;
        mapa[u].push_back(v);
        mapa[v].push_back(u);
    }

    for(int i = 1; i < mapa.size(); ++i){
        vector<bool> visitados;
        visitados.assign(n + 1, false);
        Hamiltoniano(vector<int>(), i, visitados);
    }

    if (caminhos.size() > 0){
        cout << "Estas sao as maneiras possiveis de buscar todos os alunos passando em cada aluno uma unica vez:" << endl;
        for(int i = 0; i < caminhos.size(); ++i){
            cout << caminhos[i][0];
            for(int j = 0; j < caminhos[i].size(); ++j){
                cout << " -> " << caminhos[i][j];
            }
            cout << endl;
        }
    } else {
        cout << "Nao e possivel buscar todos os alunos passando em cada aluno uma unica vez." << endl;
    }

    return 0;
}
```

Fonte: Os autores, 2021

10.4 Considerações didáticas

Observamos que a solução matemática a estratégia de iniciar no vértice mais externo e posteriormente percorrer os outros vértices de maneira a não repetir nenhum deles e no fluxograma, bem como na abstração reparamos que os passos não ficariam tão claros devida a dificuldade de abstrair a implementação de maneira detalhada. Já na implementação em si vemos quantos passos e procedimentos são necessários para buscar uma solução hamiltoniana.

A questão 10 possui características semelhantes com a anterior, porém o intuito desta é encontrar um caminho hamiltoniano, percurso que passa por todas as arestas sem repetir nenhuma. Tem característica de enumeração por podermos resolver através da listagem dos caminhos existentes no grafo e tem caráter de classificação, pois devemos seguir uma regra para que ela tenha solução.

Se pensarmos além do que é pedido, essa questão poderia ter potencial de se tornar uma pergunta de otimização, se a imagem em questão tivesse nas arestas peso variado, o que tornaria a imagem um grafo valorado e, nesse caso, poderíamos perguntar qual o caminho a seguir de modo que passássemos por todas as arestas andando o mínimo possível. Na solução matemática podemos perceber que a técnica para resolver a questão do caminho hamiltoniano não é a mais eficiente e tal característica se expande para a implementação computacional, pois a maneira que é encontrada a solução dessa questão não é a mais eficiente e de menor complexidade possível. Isso porque é considerada de complexidade NP difícil, ou seja, não se conhece até então um algoritmo eficiente e de rápida resolução que busque a resposta dessa pergunta. Essa característica evidencia que, nem tudo o que é calculado computacionalmente, terá resultado imediato e ser eficaz.

Por outro lado, muitos problemas têm grande potencial de ter o tempo minimizado e a eficiência da solução ser ótima. Vale lembrarmos de muitos problemas matemáticos que antes não puderam ser provados a sua não existência e foram resolvidos através de auxílio de implementações computacionais, seja por processo de numeração ou de existência e conseguiu mostrar um exemplo no qual determinada conjectura falhava.

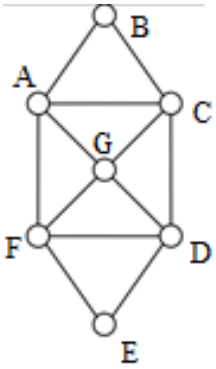
Na própria solução da questão 14 ao observarmos a matriz de adjacência do grafo trabalhado na questão, vemos o tamanho dessa matriz, bem como podemos imaginar o espaço na memória ocupado para armazenar tais dados, o cálculo necessário para a solução desse problema e comentarmos que a simulação desses caminhos é feita através de uma estrutura de

repetição que pode ser um *enquanto* ou *para. A* exceção à regra a ser obedecido pode ser simulada através condicional isso mostra também que fórmulas combinatoriais matemáticas e implementações computacionais de uma de um mesmo problema possuem diferenças ou estruturas equivalentes entre si para calcular soluções. Isso ressalta a diferença da abstração matemática e abstração para a implementação algorítmica ou para a construção do algoritmo no pensamento computacional que é levantada por Dantas (2019).

Nessa questão vale comentar que estrutura tais como de lógica e ponteiro são utilizadas na implementação, ou seja algumas características matemáticas, tais como a tabela-verdade ou análise de proposições para verificar se uma sentença é verdadeira ou falsa através da lógica matemática, na implementação computacional é simulada através de uma rotina lógica ou então através de uma estrutura condicional, que verifica se a afirmação é verdadeira ou falsa. O armazenamento da matriz na memória computacional é realizado através de um ponteiro e que tem papel de armazenar o valor naquele instante que está guardado na variável envolvida no cálculo. Tal atributo de armazenar o valor da variável em um local da memória do computador temporariamente é uma característica da implementação computacional e que não se tem no papel na solução matemática, mas que podemos associar a memória humana quando estamos realizando cálculos e guardamos em nossa memória o valor de determinado produto e utilizamos ele novamente em outra etapa do cálculo, por exemplo, ou ainda a matriz que guarda informações em seus elementos.

Questão 11 - Daniela, Luana, Viviane, Vanessa, Ivail, Marco Antônio e Renato encomendaram salgados de comemoração aos 200 anos do Colégio Pedro II. O entregador passará pela casa deles entregando a encomenda de modo a não repetir as ruas e tampouco passar novamente por cada uma das casas. O esquema das casas e ruas se encontram a seguir:

Figura 58 - Questão 11



Fonte: Os autores, 2021

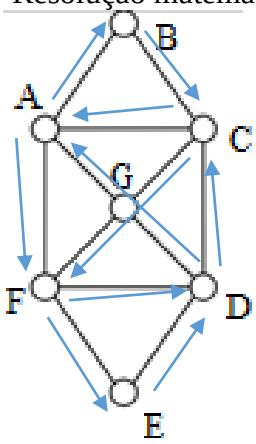
É possível entregar os salgados em todas as casas sem repeti-las e passando por todas as ruas?

11.1 Resolução Matemática:

Passando apenas uma vez por cada casa é possível por exemplo da seguinte maneira:

A → B → C → D → E → G seria um caminho possível. Esse caminho chamamos de caminho hamiltoniano. Vamos tentar encontrar uma Resolução que passe por todas as ruas sem repetir nenhuma.

Figura 59 – Resolução matemática questão 11



Fonte: Os autores, 2021

AB → BC → CA →AF → FE → ED →DC →CF →FD →DG → GA

11.2 Resolução Pensamento Computacional

11A - Decomposição

Este problema trata de três casos: é possível realizar um caminho que passa por todas as casas sem repeti-las? Se sim, temos um caminho hamiltoniano. O outro caso é se é possível passar por todas as ruas sem repetir nenhuma. Se for possível então teremos um caminho Euleriano.

E por último, é possível passar por todos os vértices e arestas uma única vez sem repetir nenhum?

11B – Padrões

Todos os vértices têm grau par. Trata-se de um grafo ciclo. Trata-se de um grafo planar. Os graus dos vértices são 4 ou 2.

11C – Abstração

Para encontrarmos uma Resolução para o grafo em questão ou para qualquer outro, computacionalmente precisamos implementar a matriz de incidência. E precisamos lembrar também que para caminhos hamiltonianos ou eulerianos não temos algoritmos eficazes e tais problemas são de alta complexidade e demoram a ser solucionados ainda que computacionalmente. Quanto maior a dimensão do grafo, mais tempo levará e o crescimento é extraordinário como vimos em Muniz (2007).

Figura 60 – Tempo de processamento com o tempo disposto

Função tempo/ complexidade	Quantidade de dados: N				
	10	20	30	40	50
N	0,00001s	0,00002s	0,00003s	0,00004s	0,00005s
N^2	0,0001s	0,0004s	0,0009s	0,0016s	0,0036s
N^3	0,001s	0,008s	0,027s	0,064s	0,125s
2^N	0,001s	1,0s	17,19 min	12,7 dias	35,7 anos
3^N	0,059s	58 min	6,5 anos	3.855 séculos	200.000,000 séculos

Fonte: Muniz, 2007.

11D – Algoritmo

A matriz de incidência é dada por:

Figura 61 – Matriz de incidência – etapa 1

	A	B	C	D	E	F	G
A	0	1	1	0	0	1	1
B	1	0	1	0	0	0	0
C	1	1	0	1	0	0	1
D	0	0	1	0	1	1	1
E	0	0	0	1	0	1	0
F	1	0	0	1	1	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021

Vamos encontrar um caminho que passe por todos os vértices sem repetir nenhum. Vamos começar no vértice A e vamos para o vértice B:

Figura 62 – Matriz de incidência – etapa 2

	A	B	C	D	E	F	G
A	0	0	1	0	0	1	1
B	0	0	1	0	0	0	0
C	1	1	0	1	0	0	1
D	0	0	1	0	1	1	1
E	0	0	0	1	0	1	0
F	1	0	0	1	1	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021

Do vértice B vamos para o C:

Figura 63 – Matriz de incidência – etapa 3

	A	B	C	D	E	F	G

A	0	0	1	0	0	1	1
B	0	0	0	0	0	0	0
C	1	0	0	1	0	0	1
D	0	0	1	0	1	1	1
E	0	0	0	1	0	1	0
F	1	0	0	1	1	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021.

Do vértice C vamos para o vértice D:

Figura 64 – Matriz de incidência – etapa 4

	A	B	C	D	E	F	G
A	0	0	1	0	0	1	1
B	0	0	0	0	0	0	0
C	1	0	0	0	0	0	1
D	0	0	0	0	1	1	1
E	0	0	0	1	0	1	0
F	1	0	0	1	1	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021.

Do vértice D vamos para o vértice E:

Figura 65 – Matriz de incidência – etapa 5

	A	B	C	D	E	F	G
A	0	0	1	0	0	1	1
B	0	0	0	0	0	0	0
C	1	0	0	0	0	0	1
D	0	0	0	0	0	1	1
E	0	0	0	0	0	1	0
F	1	0	0	1	1	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021.

Do vértice E vamos para o F:

Figura 66 – Matriz de incidência – etapa 6

	A	B	C	D	E	F	G
A	0	0	1	0	0	1	1
B	0	0	0	0	0	0	0
C	1	0	0	0	0	0	1
D	0	0	0	0	0	1	1
E	0	0	0	0	0	0	0
F	1	0	0	1	0	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021

Do vértice F vamos para o vértice G:

Figura 67 – Matriz de incidência – etapa 6

	A	B	C	D	E	F	G
A	0	0	1	0	0	1	1
B	0	0	0	0	0	0	0
C	1	0	0	0	0	0	1
D	0	0	0	0	0	1	1
E	0	0	0	0	0	0	0
F	1	0	0	1	0	0	0
G	1	0	1	1	0	0	0

Fonte: Os autores, 2021

Vale ressaltarmos aqui que nesse tipo de algoritmo, esgotamos todos os vértices, atribuímos zero aos caminhos por onde passamos, mas não esgotamos todas as arestas. Outra característica dessa matriz é que onde está em vermelho significa que trocamos o valor e essa matriz é simétrica e com diagonal principal zerada. Essa matriz inicial também será usada para encontrarmos um caminho Euleriano. Essa Resolução apresentada até aqui é um caminho hamiltoniano.

Agora veremos uma solução para encontrar um caminho Euleriano usando o algoritmo de Fleury. Vamos iniciar com a matriz de incidência:

Figura 68 – Algoritmo – Questão 11 – etapa 1

	A	B	C	D	E	F	G
A	0	1	1	0	0	1	1
B	1	0	1	0	0	0	0
C	1	1	0	1	0	0	1
D	0	0	1	0	1	1	1
E	0	0	0	1	0	1	0
F	1	0	0	1	1	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021

Vamos iniciar de um vértice diferente. Vamos começar pelo vértice D e vamos para o C:

Figura 69 Algoritmo – Questão 11 – etapa 2

	A	B	C	D	E	F	G
A	0	1	1	0	0	1	1
B	1	0	1	0	0	0	0
C	1	1	0	0	0	0	1
D	0	0	0	0	1	1	1
E	0	0	0	1	0	1	0
F	1	0	0	1	1	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021

Vamos de C pra B:

Figura 70 – Algoritmo – Questão 11 – etapa 3

	A	B	C	D	E	F	G
A	0	1	1	0	0	1	1
B	1	0	0	0	0	0	0
C	1	0	0	0	0	0	1
D	0	0	0	0	1	1	1
E	0	0	0	1	0	1	0
F	1	0	0	1	1	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021

Vamos de B para A:

Figura 71 – Algoritmo – Questão 11 – etapa 4

	A	B	C	D	E	F	G
A	0	0	1	0	0	1	1
B	0	0	0	0	0	0	0
C	1	0	0	0	0	0	1
D	0	0	0	0	1	1	1
E	0	0	0	1	0	1	0
F	1	0	0	1	1	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021

Vamos de A para F:

Figura 72 – Algoritmo – Questão 11 – etapa 5

	A	B	C	D	E	F	G
A	0	0	1	0	0	0	1
B	0	0	0	0	0	0	0
C	1	0	0	0	0	0	1
D	0	0	0	0	1	1	1
E	0	0	0	1	0	1	0
F	0	0	0	1	1	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021

Vamos de F para D:

Figura 73 – Algoritmo – Questão 11– etapa 6

	A	B	C	D	E	F	G
A	0	0	1	0	0	0	1
B	0	0	0	0	0	0	0
C	1	0	0	0	0	0	1
D	0	0	0	0	1	0	1
E	0	0	0	1	0	1	0
F	0	0	0	0	1	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021

Vamos de D para E:

Figura 74– Algoritmo – Questão 11 – etapa 7

	A	B	C	D	E	F	G
A	0	0	1	0	0	0	1
B	0	0	0	0	0	0	0
C	1	0	0	0	0	0	1
D	0	0	0	0	0	0	1
E	0	0	0	0	0	1	0
F	0	0	0	0	1	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021

Vamos de E para F:

Figura 75 – Algoritmo – Questão 11 – etapa 8

	A	B	C	D	E	F	G
A	0	0	1	0	0	0	1
B	0	0	0	0	0	0	0
C	1	0	0	0	0	0	1
D	0	0	0	0	0	0	1
E	0	0	0	0	0	0	0
F	0	0	0	0	0	0	1
G	1	0	1	1	0	1	0

Fonte: Os autores, 2021

Vamos de F para G:

Figura 76 - Algoritmo – Questão 11 – etapa 9

	A	B	C	D	E	F	G
A	0	0	1	0	0	0	1
B	0	0	0	0	0	0	0
C	1	0	0	0	0	0	1
D	0	0	0	0	0	0	1
E	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0
G	1	0	1	1	0	0	0

Fonte: Os autores, 2021

Vamos de G para A:

Figura 77 – Algoritmo – Questão 11 – etapa 10

	A	B	C	D	E	F	G
A	0	0	1	0	0	0	0
B	0	0	0	0	0	0	0
C	1	0	0	0	0	0	1
D	0	0	0	0	0	0	1
E	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0
G	0	0	1	1	0	0	0

Fonte: Os autores, 2021

Vamos de A para C:

Figura 78– Algoritmo – Questão 11 – etapa 11

	A	B	C	D	E	F	G
A	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0
C	0	0	0	0	0	0	1
D	0	0	0	0	0	0	1
E	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0
G	0	0	1	1	0	0	0

Fonte: Os autores, 2021

Vamos de C para G:

Figura 79 – Algoritmo – Questão 11 – etapa 12

	A	B	C	D	E	F	G
A	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0
D	0	0	0	0	0	0	1
E	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0
G	0	0	0	1	0	0	0

Fonte: Os autores, 2021

Vamos de G para D:

Figura 80 – Algoritmo – Questão 11 – etapa 13

	A	B	C	D	E	F	G
A	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0

Fonte: Os autores, 2021

Então o caminho que fizemos foi: DC → CB → BA →AF → FD → DE → EF → FG → GA → AC → CG → GD. Esse é outro exemplo de caminho Euleriano, pois esgotamos todas

as arestas do grafo através do algoritmo de Fleury. Nesse tipo de algoritmo fica nítido que chegamos a uma matriz nula, diferente da matriz da Resolução do caminho hamiltoniano.

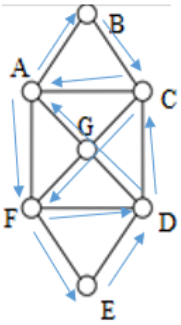
Sobre a pergunta se conseguimos fazer no grafo em questão um caminho que não repita nem aresta nem vértice, mas que passe por todos eles uma única vez, a resposta é que nesse grafo não é possível. Basta observarmos que para resolver o problema de caminho Euleriano acabamos passando mais de uma vez por alguns vértices como o vértice A.

11.3 Comparando Soluções

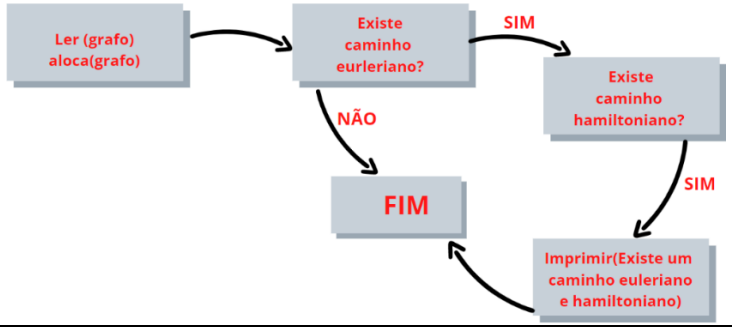
Tabela 13 - Comparando Soluções – Questão 11

A – Resolução Matemática

Passando apenas uma vez por cada casa é possível por exemplo da seguinte maneira: A → B → C → D → E → G seria um caminho possível. Esse caminho chamamos de caminho hamiltoniano. Vamos tentar encontrar uma solução que passe por todas as ruas sem repetir nenhuma.



B – Fluxograma



C – Abstração

```
Inicio ( )
    Leia(grafo)
    Aloca(matriz_grafo)
    Enquanto (existir linha da matriz nao visitada faça)
        leia linha i
        onde tem 1? (significa que existe ligacao entre esse vertice da linha e a coluna correspondente)
        va para a coluna j
        esse vertice ja foi visitado?
            se sim entao va para o proximo vertice
        senao guarda(caminho) e va para a linha j
        fim se
        i = i + 1
        aux = verdadeiro
    fim enquanto
    Enquanto (existir linha da matriz nao visitada faça)
        leia linha i
        onde tem 1? (significa que existe uma aresta entre o vertice dessa linha e a coluna correspondente)
        va para a coluna j
        guarda(caminho) e coloca 0 nessa posicao
        va para a linha j
        i = i + 1
        aux2 = verdadeiro
    fim enquanto
    Se aux e aux2 forem Verdadeiros entao
        Imprimir(É possivel fazer um caminho euleriano e hamiltoniano ao mesmo tempo)
    senao
        Imprimir (nao é possivel fazer um caminho euleriano e hamiltoniano ao mesmo tempo)
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 81 - Implementação em linguagem C++ da Questão 11

```cpp
#include <algorithm>
#include <iostream>
#include <vector>

using namespace std;

vector<vector<int> > caminhos;

void Solucao(vector<vector<int> > mapa, vector<int> caminho, int u, int v, vector<bool> visitados, int n){
    caminho.push_back(v);
    mapa[u].erase(remove(mapa[u].begin(), mapa[u].end(), v), mapa[u].end());
    mapa[v].erase(remove(mapa[v].begin(), mapa[v].end(), u), mapa[v].end());
    visitados[v] = true;

    if (caminho.size() == n){
        for(int i = 1; i <= n; ++i){
            if (mapa[i].size() > 0){
                return;
            }
        }
        caminhos.push_back(caminho);
    } else {
        for(int i = 0; i < mapa[v].size(); ++i){
            int vizinho = mapa[v][i];

            if (!visitados[vizinho]){

                Solucao(mapa, caminho, v, vizinho, visitados, n);
            }
        }
    }
}

int main()
{
    vector<vector<int> > mapa;
    vector<bool> visitados;
    int n, m, u, v;

    cout << "Quantos vertices ha no mapa?" << endl;
    cin >> n;
    for(int i = 0; i <= n; ++i){
        mapa.push_back(vector<int>());
        visitados.push_back(false);
    }

    cout << "Quantas arestas ha no desenho?" << endl;
    cin >> m;

    cout << "Descreva as " << m << " arestas do seu desenho. Exemplo: 1 3 significa uma aresta entre os vertices 1 e 3." << endl;
    for(int i = 0; i < m; ++i){
        cin >> u >> v;
        mapa[u].push_back(v);
        mapa[v].push_back(u);
    }

   if (caminhos.size() > 0){
       cout << "Estao sao as maneiras possiveis do entregador passar pela casa de cada um uma unica vez sem repetir ruas:" << endl;
       for(int i = 0; i < caminhos.size(); ++i){
           cout << caminhos[i][0];
           for(int j = 0; j < caminhos[i].size(); ++j){
               cout << " -> " << caminhos[i][j];
           }
           cout << endl;
       }
   } else {
       cout << "Nao e possivel passar pela casa de todos uma unica vez sem repetir ruas." << endl;
   }

   return 0;
}
```

Fonte: Os autores, 2021

11.4 Considerações didáticas

Na solução matemática mostramos uma solução para um caminho Euleriano e outro para um caminho hamiltoniano, mas para os dois ao mesmo tempo não é possível. Essa ideia foi trabalhada no fluxograma de maneira mais abstrata, uma vez que já analisamos nas questões anteriores como encontrar um caminho Euleriano ou hamiltoniano diante de grafo. O diferencial dessa questão é que perguntamos se é possível fazer os dois ao mesmo tempo. Nesse caso não é possível. Observamos que a implementação é também mais complexa e trabalha com estrutura auxiliares, inclusive com a ideia de grafos, de busca com o auxílio de vetores. Fica nítido a importância da organização na linguagem de programação para que fique mais claro o entendimento do que o programa está realizando.

A questão 11 também se trata de um problema combinatorial de existência. Nela são feitas duas perguntas e com isso a sua solução pode ser particionada em três perguntas. A primeira é se existe um caminho Euleriano que passe por todos os vértices sem repetir nenhum deles, outra pergunta é se existe um caminho hamiltoniano e finalmente a pergunta que junta as duas: é possível passar por todos os vértices e arestas sem repetir nenhum dos dois?

Analogamente como as anteriores, tem caráter enumerativo, além de ter a classificação mais reforçada e evidente pois trabalhamos com 3 condicionais. Tem o potencial de se tornar uma questão de otimização se colocarmos valores nas arestas, ou ainda se pensarmos de maneira genérica, que ao buscarmos um caminho hamiltoniano que passe por todas as arestas sem repetir nenhuma queremos economizar ou evitar repetição desnecessária, que geraria um custo maior dependendo do contexto em que esse problema estivesse inserido.

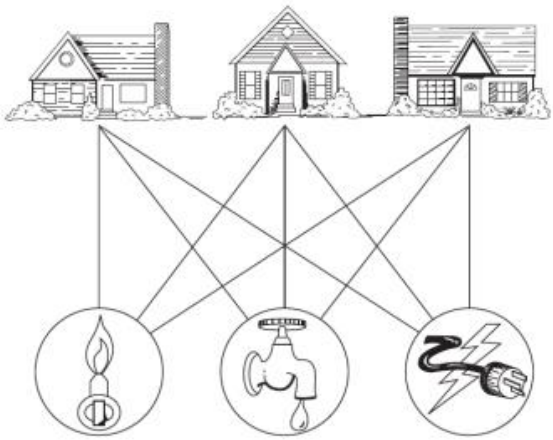
Essa questão também tem uma característica de lógica, pois as perguntas podem ser verdadeiras separadamente, ou seja, dado, um grafo ele pode ser Euleriano e não hamiltoniano, e, portanto, ele não será Euleriano e hamiltoniano mesmo tempo; ou ainda ele pode ser não Euleriano e hamiltoniano o que gera um grafo que não é Euleriano e hamiltoniano ao mesmo tempo e; ainda; ele pode ser um grafo não Euleriano e tampouco hamiltoniano e consequentemente não ser um grafo Euleriano e hamiltoniano. Essa análise lógica pode ser realizada através de uma estrutura lógica dentro da implementação computacional.

O tempo gasto e a complexidade desse tipo de problema no grafo implementado não aparecem muito por se tratar de um grafo simples, mas se pensarmos em Sistemas de redes ou estradas no nível macro, o tempo de execução e eficiência desse problema não são ótimos.

O uso planilha, com o auxílio de procedimentos algorítmicos, podem simular o que acontece computacionalmente e é uma alternativa para escolas que não possuem internet ou computadores trabalhar com seus alunos de forma desplugada a simulação da implementação algoritimica, desenvolver o pensamento computacional e também evidenciar a velocidade que levamos para resolver essa questão manualmente, ou a velocidade que ganhamos se resolvermos a questão via uma editor de planilhas como Excel e como se torna ainda mais veloz implementarmos tal problema computacionalmente. Ou seja, questões de enumeração ou de existência em que seja necessário uma extensa e numeração dos casos, a eficiência computacional é muito maior do que a manual ainda que não seja ótima.

Questão 12 - Sérgio, Eduardo e Claudia moram em casas adjacentes. Eles querem pedir a instalação de água, luz e gás em suas residências de modo que os fios não se encontrem e ocorra um curto-circuito. Observe:

Figura 82 – Questão 12



Fonte: https://homepages.dcc.ufmg.br/~loureiro/md/md_9Grafos_MaterialExtra

É possível realizar essa representação?

12. 1 Resolução Matemática:

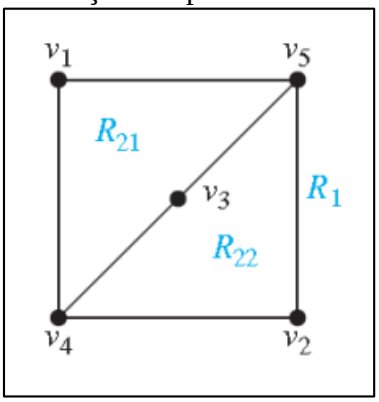
O problema é sobre um grafo em que as casas e as instalações são os vértices e as ligações são as arestas. Cada vértice tem grau três e eles podem ser representados através de dois conjuntos disjuntos e com isso se trata de um grafo bipartido. Este grafo é conhecido como $K_{3,3}$, pois para cada conjunto de vértices, todo tem grau três. Tal grafo não é planar. Ou seja, não temos como fazer uma representação do grafo $K_{3,3}$ de modo que as arestas não tenham intersecção entre si. Para termos certeza de que ele não é planar, vamos nos basear no Teorema de Euler. Este teorema nos garante que a quantidade de regiões existentes num grafo desenhado, se ele for planar, obedece a seguinte fórmula:

$$R = e - v + 2$$

Além disso, uma das regiões é ilimitada.

Nesta fórmula, r são as regiões, e são as arestas e v são os vértices. Através dessa relação, teríamos que $R = 9 - 6 + 2 \rightarrow R = 5$ regiões. Mas conseguimos mostrar uma representação que possui mais de 5 regiões, veja:

Figura 83 – Resolução via pensamento computacional



Fonte: Os autores, 2021

Ou seja, existem 3 regiões, sendo uma delas ilimitada. Logo o grafo $K_{3,3}$ não é planar.

12.2 Resolução Pensamento Computacional

12A – Decomposição

Devemos ver em quantas regiões podemos dividir o grafo que é candidato a ser planar.

Contamos os vértices e as arestas.

Tentamos usar o Teorema de Euler para saber o grafo é planar ou não.

12B – Padrões

Todo grafo bipartido K_1, K_2 são planares.

Se o grafo for K_4 ou K_3 precisamos usar o Teorema de Euler para garantir que seja planar.

O mesmo acontece para K_5.

12C - Abstração

Se um grafo possui um subgrafo do formato $K_{3,3}$ ou K_5 então ele não é planar.

12D - Algoritmo

Não há um algoritmo prático para vermos se um grafo é planar. Uma maneira seria utilizar o algoritmo de redução:

1 – Vamos determinar os componentes do grafo $G = G_1, G_2, ..., G_k$. Devemos testar cada componente G_i do grafo.

2 – Removemos todos os loops.

3 – Eliminamos as arestas paralelas, deixando apenas uma aresta entre cada par de vértices.

4 – Eliminamos os vértices de grau dois através da fusão de duas arestas. (Arestas em série não afetam a planaridade).

5 – Repetimos os passos 3 e 4 enquanto for possível.

De uma maneira geral, após aplicar o procedimento 1 a cada uma das componentes Gi, o grafo reduzido, Hi é:

a) uma aresta; ou

b) um grafo completo com 4 vértices; ou

c) um grafo simples com $n \geq 5$ e $m \geq 7$.

Se todos os grafos reduzidos Hi satisfizerem os itens a) ou b), o grafo G é planar. Caso contrário é necessário verificar se $m \leq 3n - 6$.

Se o grafo reduzido não satisfaz esta inequação então o grafo G é não planar. Se a inequação for satisfeita, é necessário fazer testes adicionais.

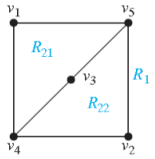
Observação: Usando o algoritmo e o Teorema anterior podemos identificar claramente a planaridade de um grafo para casos em que o grafo tem menos que 5 vértices e menos que 7 arestas.

12.2 Comparando Soluções

Tabela 14 - Comparando Soluções – Questão 12

A – Resolução Matemática

Nesta fórmula, r são as regiões, e são as arestas e v são os vértices. Através dessa relação, teríamos que R = 9 – 6 + 2 → R = 5 regiões. Mas conseguimos mostrar uma representação que possui mais de 5 regiões, veja:



Ou seja, existem 3 regiões, sendo uma delas ilimitada. Logo o grafo $K_{3,3}$ não é planar.

B – Fluxograma

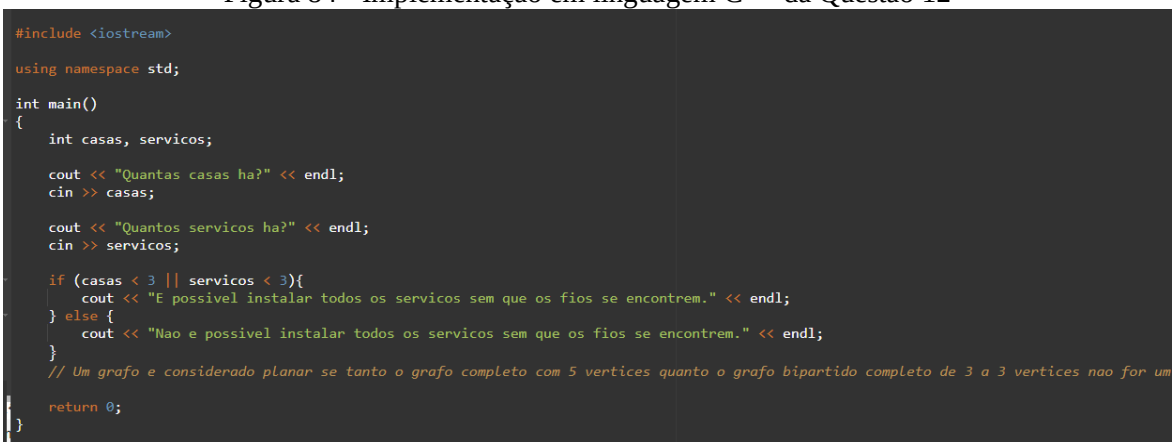
C – Abstração

```
Inicio ( )
    1. Determinar o grafo de componentes G = G1, G2, ..., Gk
    2. Testar cada componente Gi do grafo.
    3. Remover todos os loops
    4. Eliminar as arestas paralelas, deixando apenas uma aresta entre cada par de vértices..
    5. Eliminar os vértices de grau dois através da fusão de duas arestas. (Arestas em série não afetam a planaridade).
    6. Repetimos os passos 3 e 4 enquanto for possível.
    Quando reduzirmos o grafo, chegamos em um grafo Hi
    Se Hi for uma aresta ou um grafo completo com 4 vértices entao ele é um grafo planar
        senao verificar
            se m ≤ 3 n - 6 entao fazer testes adicionais e pode ser planar
            senao nao é planar
        fim se
    fim se
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 84 - Implementação em linguagem C++ da Questão 12



```cpp
#include <iostream>

using namespace std;

int main()
{
    int casas, servicos;

    cout << "Quantas casas ha?" << endl;
    cin >> casas;

    cout << "Quantos servicos ha?" << endl;
    cin >> servicos;

    if (casas < 3 || servicos < 3){
        cout << "E possivel instalar todos os servicos sem que os fios se encontrem." << endl;
    } else {
        cout << "Nao e possivel instalar todos os servicos sem que os fios se encontrem." << endl;
    }
    // Um grafo e considerado planar se tanto o grafo completo com 5 vertices quanto o grafo bipartido completo de 3 a 3 vertices nao for um

    return 0;
}
```

Fonte: Os autores, 2021

12.4 Considerações didáticas

A questão 12 é classificada como uma questão de existência, em particular trabalha com a planaridade de grafos regulares cúbicos, ou seja, grafos cujos vértices todos têm grau 3 e se é possível a planaridade desses, em outras palavras se é possível escrever sem tirar o lápis do papel a realizar um caminho que interligue todos os vértices através de três arestas que não se sobreponham. Sabemos baseados na teoria dos grafos que a esse problema não tem solução, mas para chegarmos a essa conclusão a solução matemática e computacional não é usual e nem eficiente ou ótima. Podemos notar pelo desenvolvimento da solução matemática que diversas propriedades e teorias são utilizadas para mostrar que o problema não tem solução e que na implementação computacional do algoritmo não é tão simples e claro de se entender, bem como de ser implementado.

Trata-se, assim, de um problema que não tem eficiência computacional para ser resolvido rapidamente. Isso nos remete aos problemas de modelagem matemática nos quais são utilizados os sistemas matemáticos para imitar fenômenos naturais biológicos, mas nos quais muitas variáveis são descartadas para que a simulação computacional consiga imitar o comportamento esperado e mostra que nem sempre a modelagem matemática e simulação computacional são capazes, até onde se sabe, de resolver sistemas mais complexos.

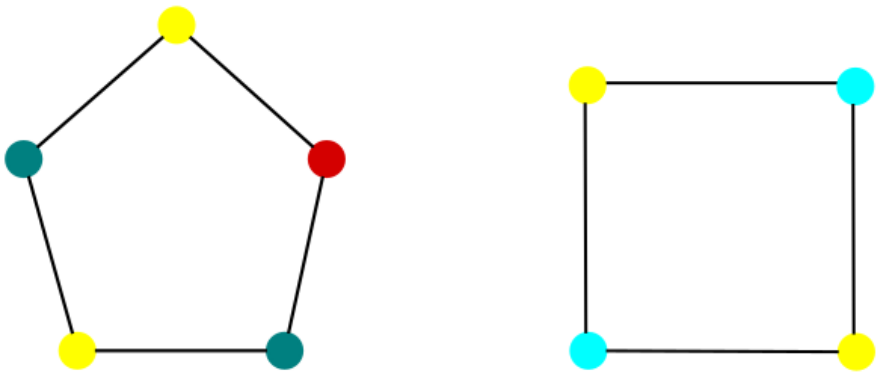
Notamos que usamos uma relação matemática para verificar se é planar ou não. Já no fluxograma, usamos a ideia de um algoritmo mais abstrato de como saber se o grafo é planar ou não. E ele é trabalhado na etapa de abstração. E por último, na implementação é usado o resultado que já é sabido que o grafo $K_{3,3}$ não é planar.

Questão 13 - É possível colorir os grafos C_4 e C_5 usando duas cores de modo que os vértices adjacentes não tenham a mesma cor? Justifique sua resposta.

13.1 Solução Matemática:

Vamos iniciar pensando no ciclo par. Se pintarmos alternadamente cada vértice de uma cor veremos que precisamos apenas de duas cores conforme sugerido no enunciado. Já para o grafo de ciclo ímpar, duas cores não são suficientes, pois quando chegamos no ultimo vértice, ele se depara com o primeiro vértice. Por isso são necessárias três cores conforme esquema abaixo:

Figura 85 – Questão 13



Fonte: Os autores, 2021

13.2 Resolução Pensamento Computacional

13A - Decomposição

Vamos dividir em dois casos: quando temos uma quantidade par ou ímpar de vértices.

13B - Padrões

Quando temos quatro vértices, precisamos de duas cores. Basta irmos alternando as cores de acordo com a vez do preenchimento, mas quando temos uma quantidade ímpar, no caso cinco vértices, precisamos de três cores, pois o último vértice tem ligação com o primeiro e se neste colocarmos a mesma cor vamos desobedecer a regra de vértices adjacentes terem a mesma cor.

13C - Abstração

Quando temos uma quantidade par de vértices, conseguimos colorir com duas cores e o problema tem solução. Basta colorirmos alternadamente cada vértice. Mas quando temos uma quantidade ímpar, o problema não tem solução com duas cores e precisamos de três cores para que o primeiro e o ultimo vértice não sofram choque de cores.

13D - Algoritmo

Inicio( )
Leia (n)
Se n é par, então faça
    Para $1 \leq i \leq n$
        Se resto(i/2) = 0 então
        A_i = cor 1
        Senão
        A_i = cor 2
        Fim se
        i = i + 1
    Fim para
Fim

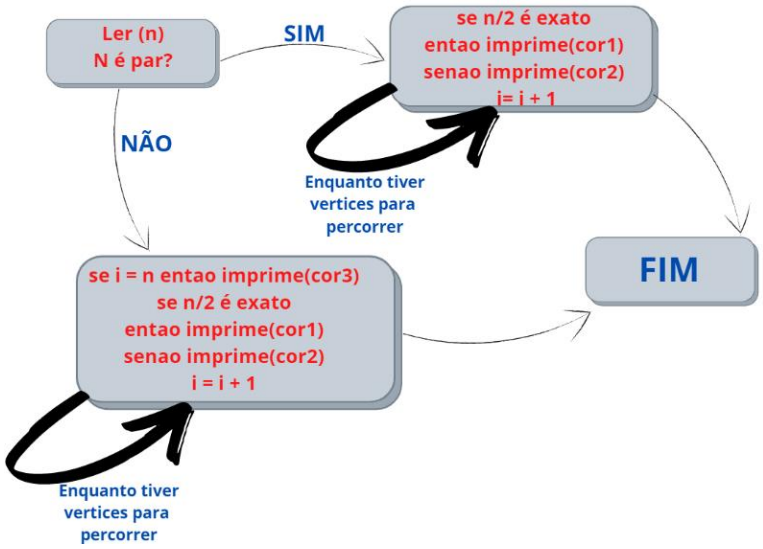
13.3 Comparando Soluções

Tabela 15 - Comparando Soluções – Questão 13

A – Resolução Matemática

Vamos iniciar pensando no ciclo par. Se pintarmos alternadamente cada vértice de uma cor veremos que precisamos apenas de duas cores conforme sugerido no enunciado. Já para o grafo de ciclo ímpar, duas cores não são suficientes, pois quando chegamos no ultimo vértice, ele se depara com o primeiro vértice. Por isso são necessárias três cores.

B – Fluxograma



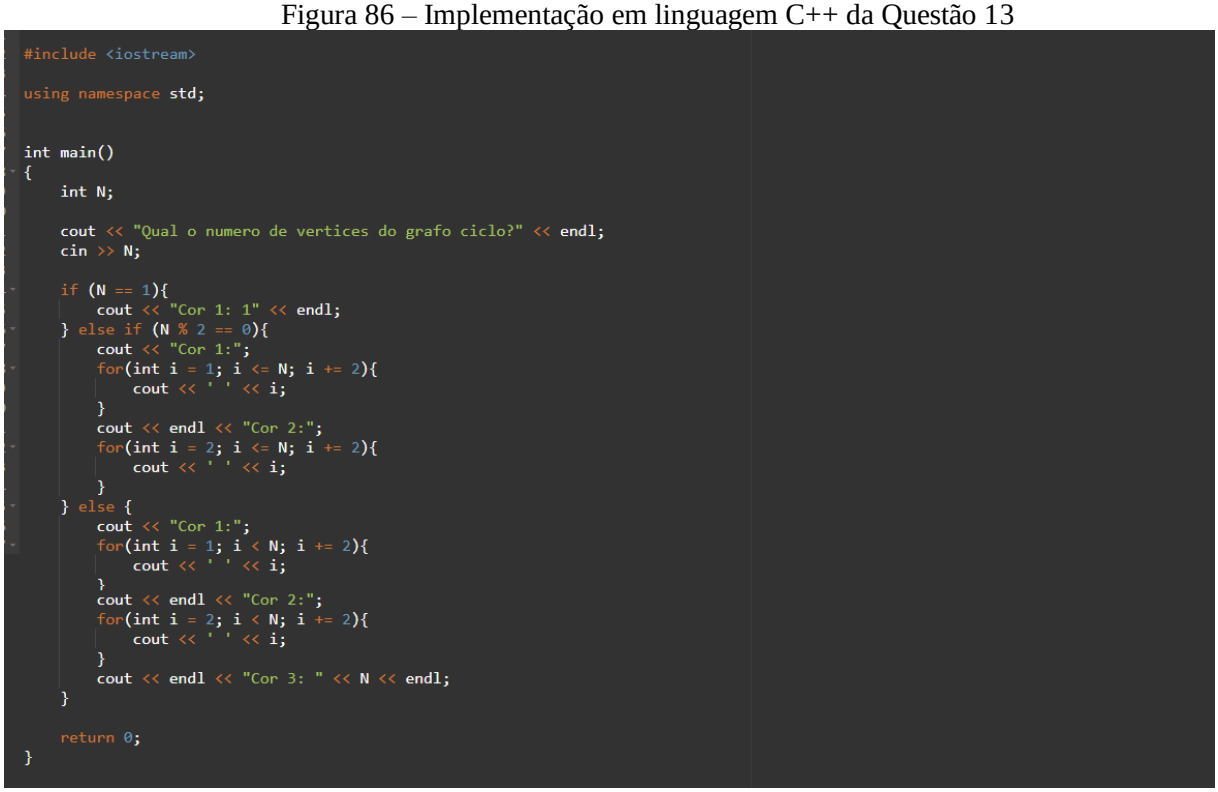
C – Abstração

```
Inicio( )
    Leia (n)
    Se n é par, então faça
        Para 1 ≤ i ≤ n
            Se resto(i/2) = 0 então
                Ai = cor 1
            Senão
                Ai = cor 2
            Fim se
            i = i + 1
        Fim para
    Senao
        Para 1 ≤ i ≤ n
            Se (i = n) então imprimir(An = cor 3)& Fim Para
            Se resto(i/2) = 0 então
                Ai = cor 1
            Senão
                Ai = cor 2
            Fim se
            i = i + 1
        Fim para
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 86 – Implementação em linguagem C++ da Questão 13



```cpp
#include <iostream>

using namespace std;


int main()
{
    int N;

    cout << "Qual o numero de vertices do grafo ciclo?" << endl;
    cin >> N;

    if (N == 1){
        cout << "Cor 1: 1" << endl;
    } else if (N % 2 == 0){
        cout << "Cor 1:";
        for(int i = 1; i <= N; i += 2){
            cout << ' ' << i;
        }
        cout << endl << "Cor 2:";
        for(int i = 2; i <= N; i += 2){
            cout << ' ' << i;
        }
    } else {
        cout << "Cor 1:";
        for(int i = 1; i < N; i += 2){
            cout << ' ' << i;
        }
        cout << endl << "Cor 2:";
        for(int i = 2; i < N; i += 2){
            cout << ' ' << i;
        }
        cout << endl << "Cor 3: " << N << endl;
    }

    return 0;
}
```

Fonte: Os autores, 2021

13.4 Considerações didáticas

A questão 13 trabalha com a noção de coloração de grafos e possui diversas aplicações em teoria de grafos ou em áreas interdisciplinares com a matemática tais como podemos ver em Muniz (2007); Lovaz et al (2006); Malta (2008); Borges Muniz (2018). Para resolver o problema matematicamente utilizamos uma estratégia de classificação na qual colorimos os vértices rotulados pares com uma cor e os vértices rotulados ímpares uma segunda cor. Ao usarmos essa estratégia, podemos perceber que essa questão se trata então de um problema de existência, pois o comando da pergunta é se é possível, ou seja se existe uma coloração com duas cores de um grafo de ciclo par e ímpar, mas também possui características de otimização uma vez que ao colorir com apenas duas cores minimizamos a quantidade de cores ou de rótulos necessários para representar situações e essa é uma das aplicações da coloração de grafos.

Ao solucionarmos a questão também vemos a natureza de classificação, uma vez que dividimos em 2 casos: o qual temos um ciclo par e o que temos um ciclo ímpar. Essa estratégia de saber a quantidade de cores pode ser vista como de contagem, pois podemos usar duas cores,

ou três no caso de ciclo ímpar, bem como podemos pensar em cada vértice pintar de uma cor diferente, apesar de que essa questão não seria eficiente, mas podemos trazer essa reflexão à tona e ao colorir cada vértice de uma cor distinta das cores usadas anteriormente esse tipo de solução não é ótimo e tampouco eficaz. Vale também ressaltar que essa questão tem natureza enumerativa pois, dependendo do nível escolar com que estamos trabalhando podemos mostrar a solução aos alunos através da enumeração das possíveis configurações de coloração de cada um dos círculos, seja ela no papel, seja através da enumeração exposta na tela do computador calculada pela implementação algorítmica. A Abstração matemática e computacional nessa questão são bem semelhantes pois a conclusão que chegamos aqui em ambos os casos Se temos uma quantidade par de vértices utilizamos duas cores é possível resolver a questão e se temos uma quantidade ímpar a questão tem solução. Em ambos os casos essa conclusão é utilizada na etapa de abstração.

A ideia da resolução matemática é feita por ciclos pares com duas cores e ímpares com três cores. Já no fluxograma a ideia trabalhada é a mesma e na implementação também. Na implementação vemos o uso da estrutura de condicional.

Questão 14 - Numa turma de 40 alunos, existe pelo menos duas pessoas que fazem aniversário no mesmo mês. Podemos afirmar isso com segurança? Justifique.

14.1 Resolução Matemática:

Pelo princípio das gavetas de Dirichlet se colocássemos esses 40 alunos em 39 gavetas então pelo menos uma delas teria dois alunos. No caso como estamos falando de meses do ano e esses são 12 no total, se colocarmos cada aluno representando um mês do ano, haverá pelo menoses duas pessoas que fazem aniversário no mesmo mês.

14.2 Resolução Pensamento Computacional

14A - Decomposição

Vamos colocar cada um dos alunos aleatoriamente em um mês. Assim teremos analisado 12 alunos, já que o ano tem 12 meses. Mas ainda sobraram 18 pessoas que não analisamos. Se viermos colocando cada uma delas novamente em um mês, teremos 2 pessoas em cada mês e ainda sobram 6 alunos sem análise. Esses podem ser colocados em 6 meses restantes. Temos segurança de que existe duas pessoas que fazem aniversário no mesmo mês

14B - Padrões

Sempre colocamos uma pessoa em cada um dos meses e como existem mais pessoas que o número de posições para colocá-las, isso nos garante a existência de pelo menos duas pessoas em um mesmo mês.

14C - Abstração

Para n alunos de uma turma, dispomos esses n alunos ao longo dos 12 meses de modo que pelo menos cada um fique em um mês representando o aniversariante daquele mês. Nesse caso n é um número maior ou igual a 12.

14D - Algoritmo

Inicio ( )
Leia (n)
A_1 = janeiro
A_2 = fevereiro
A_3 = março
A_4 = abril
....
A_{12} = dezembro

Para ($1 \leq i \leq n$) faça
Coloque i no mês a_i
i = i + 1
Fim para
Fim

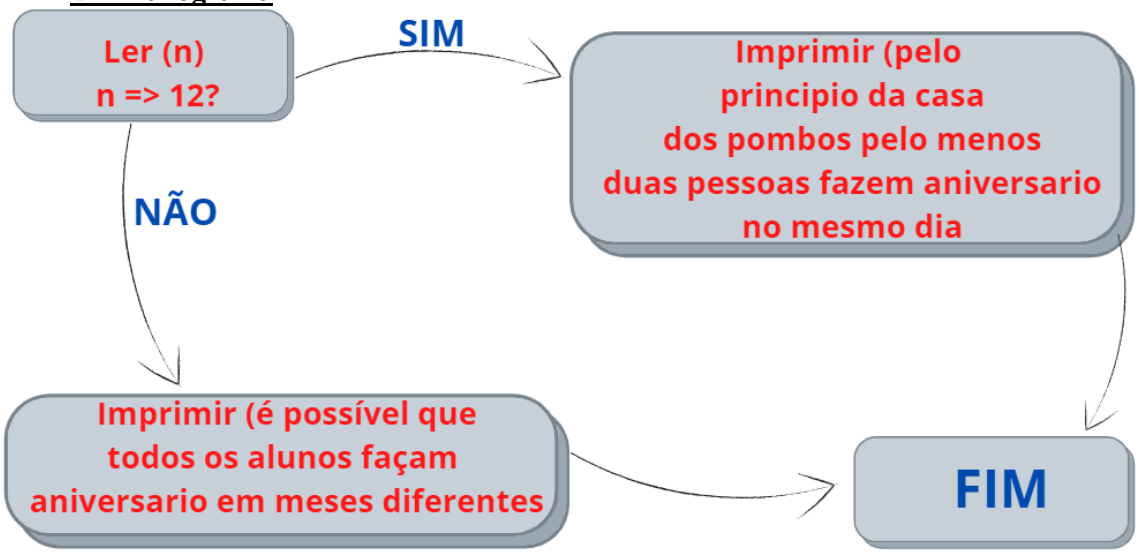
14.3 Comparando Soluções

Tabela 16 - Comparando Soluções – Questão 14

A – Resolução Matemática

Pelo princípio das gavetas de Dirichlet se colocássemos esses 40 alunos em 39 gavetas então pelo menos uma delas teria dois alunos. No caso como estamos falando de meses do ano e esses são 12 no total, se colocarmos cada aluno representando um mês do ano, haverá pelo menos duas pessoas que fazem aniversário no mesmo mês.

B – Fluxograma



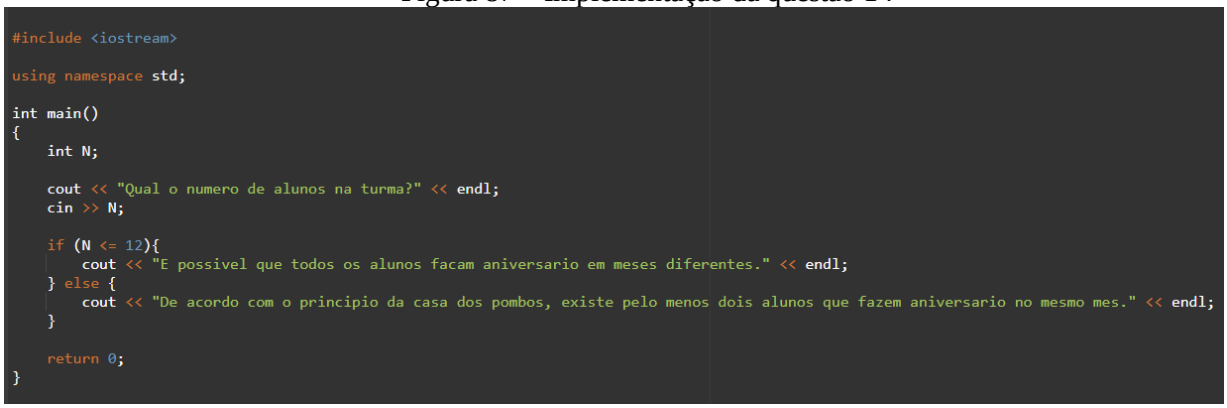
C – Abstração

```
Inicio (  )
    Leia (n)
    A1 = janeiro
    A2 = fevereiro
    A3 = março
    A4 = abril
    ...
    A12 = dezembro
        Para (1 ≤ i ≤ n) faça
            Coloque i no mês ai
            i = i + 1
        Fim para
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 87 – Implementação da questão 14



```
#include <iostream>

using namespace std;

int main()
{
    int N;

    cout << "Qual o numero de alunos na turma?" << endl;
    cin >> N;

    if (N <= 12){
        cout << "E possivel que todos os alunos facam aniversario em meses diferentes." << endl;
    } else {
        cout << "De acordo com o principio da casa dos pombos, existe pelo menos dois alunos que fazem aniversario no mesmo mes." << endl;
    }

    return 0;
}
```

Fonte: Os autores, 2021

14.4 Considerações didáticas

A questão 14 trabalha com o princípio da casa dos pombos, ou conhecido também como princípio das gavetas, e é muito usado na computação e na matemática discreta. Esse problema tem natureza de combinatória de existência. Mas ele também pode ser visto como um problema com caráter enumerativo, pois ao buscarmos a solução desse problema pensamos na quantidade total de elementos que temos e a quantidade total de locais para armazenar esses elementos, de modo que em cada local tenham a mesma quantidade mínima de elementos armazenados. Podemos fazer uma analogia com alocação de memória na computação paralela, na qual dividimos uma tarefa em sub-rotinas. Esse problema tem aplicação computacional, como por exemplo em situações de alocação de objetos igualmente distribuídos dentre uma quantidade de espaços para armazenamento, e matemática apesar de parecer muito abstrato, ele tem grande característica recursiva. Tal questão também pode ser vista como de natureza de otimização, porque ao buscarmos armazenar uma quantidade de objetos num espaço limitado, com uma quantidade de gavetas, nosso raciocínio é guardar o máximo possível de elementos em cada uma das gavetas de modo a não faltar gaveta e armazenar número máximo possível de elementos e isso recai na noção de computação paralela mencionada anteriormente. Finalmente, essa questão também possui caráter de contagem pois ao perguntarmos se existe solução acabamos contando quantos elementos cabem em cada posição.

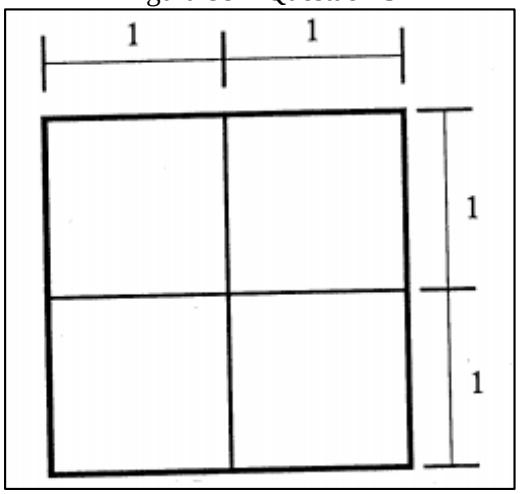
A noção da solução matemática, por abstração, o fluxograma e a implementação foram baseados no mesmo Princípio da Casa dos Pombos. Vemos a estrutura de condicional na implementação e que não é necessária na solução matemática.

Questão 15 - Escolhemos ao acaso cinco pontos sobre a superfície de um quadrado de lado 2. Existe pelo menos um dos segmentos que eles determinam que tem comprimento menor ou igual a √2?

15.1 Resolução Matemática:

Vamos dividir o quadrado de lado 2 em quatro quadrados de lado 1. Dos 5 pontos, há pelo menos dois que pertencerão a um mesmo quadrado de lado 1. E a distância entre tais pontos será de no máximo igual a diagonal do quadrado que é √2. Esse argumento permite concluir que existe um segmento que obedece a regra estipulada no problema.

Figura 88 – Questão 15



Fonte: Os autores, 2021

15.2 Resolução Pensamento Computacional

15A – Decomposição

Para o quadrado de comprimento de lado 2 pensamos em dividi-lo em quadrados menores de lado 1. Como a diagonal é menor do que o valor dos lados isso garante que ela será igual a √2.

15B - Padrão

Podemos repetir o processo quantas vezes for necessário para obtermos medidas do tamanho que quisermos

15C - Abstração

Seja um quadrado de lado n. Temos n + 1 pontos sobre a superfície desse quadrado. Queremos encontrar um segmento de medida igual ou menor do que √n. Vamos dividir o quadrado de segmento n em quadrados menores de medida $\frac{n}{2}$.

Para o quadrado de segmento $\frac{n}{2}$, temos que sua diagonal vale:

$d^2 = (\frac{n}{2}.)^2 + (\frac{n}{2}.)^2$

$d^2 = \frac{2n^2}{4} = \frac{n^2}{2}.$

$d = \frac{\sqrt{n}}{2}$. Logo $d < \sqrt{n}$.

15D - Algoritmo

Inicio ( )

Leia (n)

Segmento = n/2

Diagonal = $\frac{\sqrt{n}}{2}$

Imprimir (“O quadrado de medida”, $\frac{n}{2}$, “tem diagonal medindo”, $\frac{\sqrt{n}}{2}$, “e que é menor do que”, √n.)

Fim

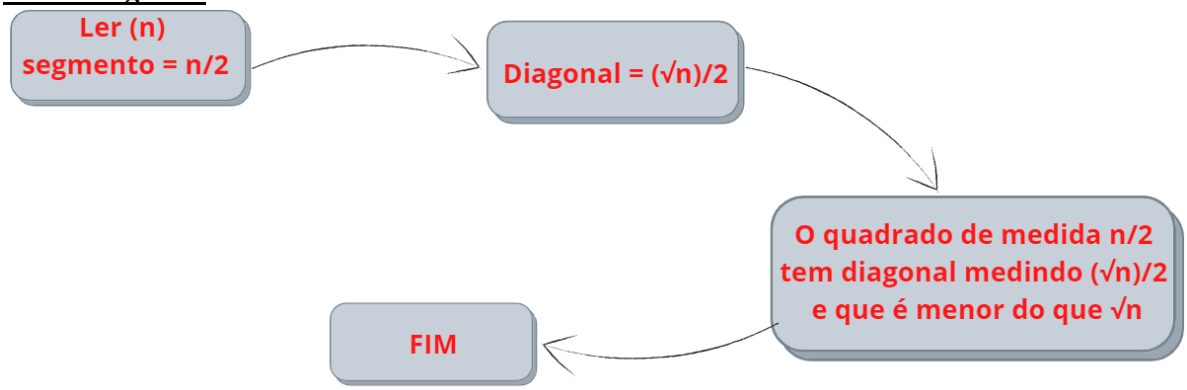
15.3 Comparando Soluções

Tabela 17 - Comparando Soluções – Questão 15

A – Resolução Matemática

Vamos dividir o quadrado de lado 2 em quatro quadrados de lado 1. Dos 5 pontos, há pelo menos dois que pertencerão a um mesmo quadrado de lado 1. E a distância entre tais pontos será de no máximo igual a diagonal do quadrado que é √2. Isso conclui que existe um segmento que obedece a regra estipulada no problema.

B – Fluxograma



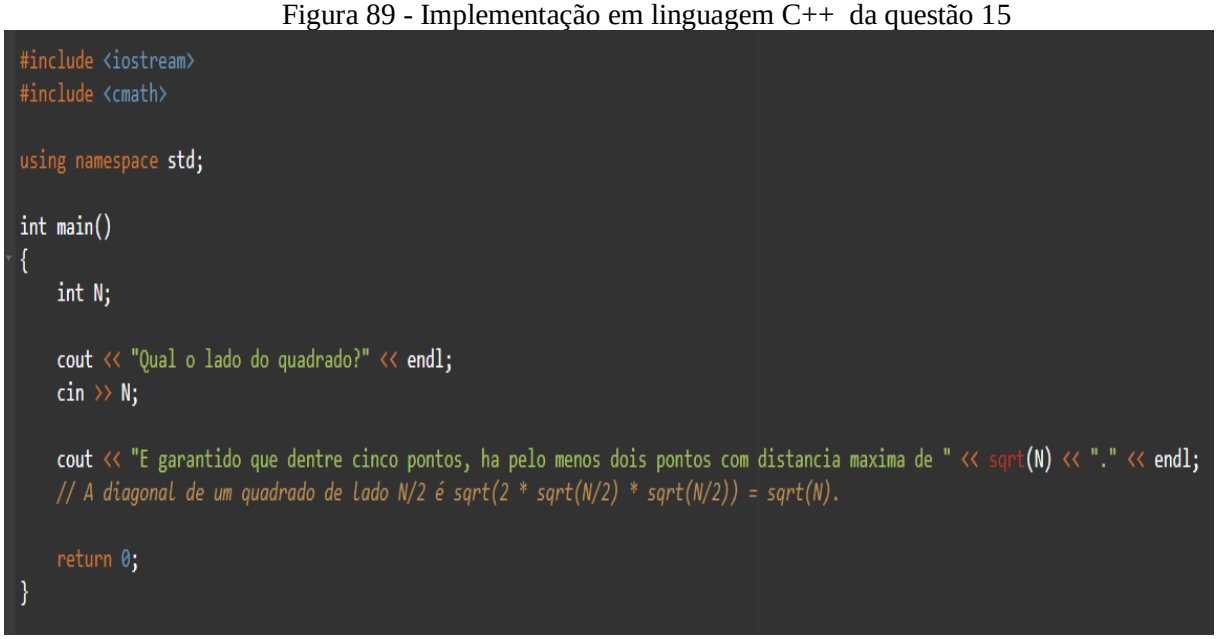
C – Abstração

```
Inicio ( )
    Leia (n)
    Segmento = n/2
    Diagonal = (√n)/2
    Imprimir ("O quadrado de medida", n/2, "tem diagonal medindo", (√n)/2, "e que é menor do que", √n.)
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 89 - Implementação em linguagem C++ da questão 15



```
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    int N;

    cout << "Qual o lado do quadrado?" << endl;
    cin >> N;

    cout << "E garantido que dentre cinco pontos, ha pelo menos dois pontos com distancia maxima de " << sqrt(N) << "." << endl;
    // A diagonal de um quadrado de lado N/2 é sqrt(2 * sqrt(N/2) * sqrt(N/2)) = sqrt(N).

    return 0;
}
```

Fonte: Os autores, 2021

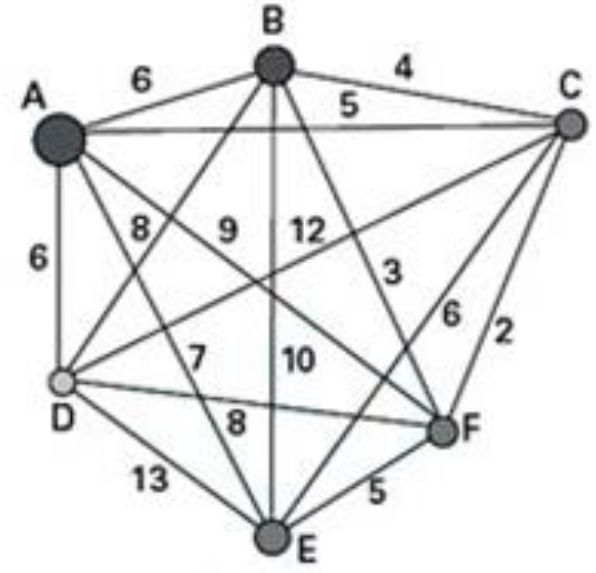
15.4 Considerações didáticas

Na solução matemática vemos a ideia da solução e como generalizá-la para um caso de termos um quadrado de lado n. Essa ideia utilizamos no fluxograma e na abstração e implementação. Podemos perceber que nessas três etapas já foi diretamente utilizado o valor de que a diagonal é do tipo Diagonal = $\frac{\sqrt{n}}{2}$ e isso se dá devido aos cálculos matemáticos para encontrarmos o valor da diagonal de um quadrado qualquer. Nesse problema tanto a implementação quanto a parte matemática são muito semelhantes.

Questão 16 - (ENEM – 2010- Adaptado) João mora na cidade A e precisa visitar cinco clientes, localizados em cidades diferentes da sua. Cada trajeto possível pode ser representado por uma sequência de 7 letras. Por exemplo, o trajeto ABCDEFA, informa que ele sairá da cidade A, visitando as cidades B, C, D, E F nesta ordem, voltando para a cidade A. Além disso, o número indicado entre as letras informa o custo do deslocamento entre as cidades. A figura mostra o custo de deslocamento entre cada uma das cidades.

Como João quer economizar, ele precisa determinar qual o trajeto de menor custo para visitar os cinco clientes. Somente parte das sequências, pois os trajetos ABCDEFA e AFEDCBA têm o mesmo custo. Ele gasta 1min30s para examinar uma sequência e descartar sua simétrica, conforme apresentado.

Figura 90 – Questão 16



Fonte: ENEM (2010).

O tempo mínimo necessário para João verificar todas as sequências possíveis no problema é de

a) 60 min

b) 90 min

c) 120 min

d) 180 min

e) 360 min

Qual o caminho que ele deve seguir em que economizará ao máximo?

16.1 Solução Matemática

As possibilidades de João visitar as cidades é de $\frac{5!}{2}$ = 60 possibilidades, desconsiderando as possibilidades simétricas. Como ele gasta 1 min e 30 s para analisar cada caminho, então existem 60.90 = 5400 s = 90 min. Letra b.

Tabela 18 – Tabela de Resolução matemática – Questão 16

	A	B	C	D	E	F
A	0	6	5	6	7	9
B	6	0	4	8	12	3
C	5	4	0	12	6	2
D	6	8	12	0	13	8
E	7	12	6	13	0	5
F	9	3	2	8	5	0

Fonte: Os autores, 2021.

Podemos tentar encontrar o menor caminho através da enumeração das possibilidades. Mas como vimos anteriormente, enumerarmos 60 possibilidades se tornaria algo cansativo e ineficaz. Então, vamos fazer alguns para ilustrar e posteriormente ver uma solução com o uso de um algoritmo de otimização.

ABCFEDA: 6 + 4 + 2 + 5 + 13 + 6 = 36

ACFEDBA: 5 + 2 + 5 + 13 + 8 + 6 = 39

AFCBDEA: 9 + 2 + 4 + 8 + 13 + 7 = 43

AEDBCFA: 7 + 13 + 8 + 4 + 2 + 9 = 43

ADCBFEA: 6 + 12 + 4 + 12 + 8 + 7 = 49

16.2 Resolução Pensamento Computacional

16 A - Decomposição

Precisamos esquematizar as possibilidades através de um processo de enumeração, por exemplo. Após escrevermos alguns exemplos, veremos que é muito longo e esse método de exaustão tem grande possibilidade de errarmos. Vamos colocar em uma matriz com o valor das distâncias entre as cidades

16B - Padrões

Um caminho ABCDEFA é o mesmo que AFEDCBA e é chamado simétrico dele. Precisamos encontrar o menor caminho, mais econômico. Vamos usar a ideia do caixeiro viajante.

16C – Abstração

Tabela 19 – Tabela de Resolução computacional – Questão 16

	A	B	C	D	E	F
A	0	6	5	6	7	9
B	6	0	4	8	12	3
C	5	4	0	12	6	2
D	6	8	12	0	13	8
E	7	12	6	13	0	5
F	9	3	2	8	5	0

Fonte: Os autores, 2021

Vamos partir da cidade A. Dentre as conexões que A realiza, a mais próxima é com a cidade C. Então iremos para essa cidade e andamos 5 u.c. Na cidade C a conexão mais próxima é a cidade F e vamos para essa e andando 2 u.c.(unidades de comprimento). Na cidade F a mais próxima, sem ser a C pois já viemos de lá, é a cidade B e andamos 3 u.c. Até o momento fizemos o caminho ACFB. Na cidade B a mais próxima é F, mas já passamos por ela. A segunda mais próxima seria C, mas também já passamos por ela. A terceira mais próxima é a cidade A, mas também já passamos por ela. Então partimos para a cidade D e andamos 8 u.c. ACFBD

Na cidade D vamos a cidade A como a mais próxima, mas já passamos por ela. Então vamos para as cidades B e F com a mesma distância, que também já passamos. Posteriormente, temos a cidade C, que também já passamos. Então nos resta ir para a cidade E andar 13 u.c. Logo andamos ACFBDE.

Finalmente, retornamos para A e andamos mais 7 u.c. No total fizemos o caminho ACFBDEA e andamos no total: $5 + 2 + 3 + 4 + 13 + 7 = 34$ u.c

16D - Algoritmo

```
Inicio ( )
Leia(grafo)
n:= recebe a quantidade de vértices do grafo
A_n = matriz de adjacência do grafo
Para (1 ≤ i ≤ n ) faça
        Não volta no vértice que já foi visitado
        Na linha i, vá para a posição de menor valor

        Soma = soma + posição de menor valor
        Caminho = caminho + vértice para onde foi
Fim para
Imprimir (soma, caminho)
Fim
```

16.3 Comparando Soluções

Tabela 20 - Comparando Soluções – Questão 16

A – Resolução Matemática

As possibilidades de João visitar as cidades é de $\frac{5!}{2}$ = 60 possibilidades, desconsiderando as possibilidades simétricas. Como ele gasta 1 min e 30 seg para analisar cada caminho, então existem 60.90 = 5400 seg = 90 min. Letra b.

B – Fluxograma

C – Abstração

```
Inicio ( )
    Leia(grafo)
    n:= recebe a quantidade de vértices do grafo
    An = matriz de adjacência do grafo
    Para (1 ≤ i ≤ n ) faça
        Não volta no vértice que já foi visitado
        Na linha i, vá para a posição de menor valor
        Soma = soma + posição de menor valor
        Caminho = caminho + vértice para onde foi
    Fim para
    Imprimir (soma, caminho)
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 91 – Implementação em linguagem C++ da questão 16

```cpp
#include <iostream>
#include <climits>
#include <vector>

using namespace std;

int mapa[100][100];
int n, m;
vector<vector<int> > caminhos;
int min_resposta = -1;

void Solucao(vector<int> caminho, vector<bool> visitados, bool first, int i, int soma){
    caminho.push_back(i);
    if (caminho.size() == n){
        if (mapa[caminho[n - 1]][1] != -1){
            caminho.push_back(1);
            soma += mapa[caminho[n - 1]][1];

            if (min_resposta == -1 || soma <= min_resposta){
                if (soma < min_resposta){
                    caminhos.clear();
                }
                caminhos.push_back(caminho);
                min_resposta = soma;
            }
        }
    } else {
        visitados[i] = true;

        for(int j = 1; j <= n; ++j){
            if (mapa[i][j] != -1 && !visitados[j]){
                Solucao(caminho, visitados, first, j, soma + mapa[i][j]);
            }
        }
    }
}

int main()
{
    vector<bool> visitados;
    int x, y, p;
    cout << "Quantas cidades Joao precisa visitar?" << endl;
    cin >> n;
    n += 1;
    cout << "n = " << n << endl;
    for(int i = 0; i <= n; ++i){
        visitados.push_back(false);
        for(int j = 0; j <= n; ++j){
            mapa[i][j] = -1;
        }
    }

    cout << "Quantas ruas existem entre as cidades?" << endl;
    cin >> m;

    cout << "Descreva cada uma das ruas com seu respectivo peso." << endl;
for(int i = 0; i < m; ++i){
    cin >> x >> y >> p;
    mapa[x][y] = p;
    mapa[y][x] = p;
}

Solucao(vector<int>(), visitados, true, 1, 0);

if (caminhos.size() > 0){
    cout << "Os caminhos de peso minimo (" << min_resposta << ") que passam por todas as cidades e voltam a cidade de Joao sao:" << endl
    for(int i = 0; i < caminhos.size(); ++i){
        cout << caminhos[i][0];
        for(int j = 1; j < caminhos[i].size(); ++j){
            cout << " -> " << caminhos[i][j];
        }
        cout << endl;
    }
} else {
    cout << "Nao ha um caminho minimo que passa por todas as cidades e volta a cidade de Joao." << endl;
}

return 0;
```

Fonte: Os autores, 2021

16.4 Considerações didáticas

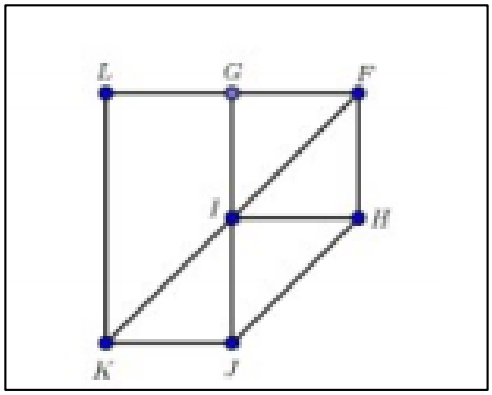
A questão 16 pode ser classificada como de otimização por buscarmos o menor tempo possível, mas também possui características de enumeração já que o aluno pode buscar a resposta de maneira intuitiva, sem saber teoria dos grafos, simplesmente listando os possíveis caminhos a percorrer e o tempo gasto em cada um deles. E possui natureza de contagem visto que, como queremos o menor tempo possível, é necessário que contemos o tempo usado em cada um dos caminhos. Interessante nessa questão é lembrar que a pergunta é de minimização, porém o comando fala em economia máxima que requer uma maior atenção do aluno na leitura para não interpretar incorretamente.

Na solução matemática apresentada vemos um processo de contagem para resolver essa situação, porém é apresentado também uma representação do gráfico através de uma tabela na qual colocamos o valor de cada uma das arestas e através de um processo enumerativo buscamos a solução ótima. Esse tipo de questão é interessante para ressaltarmos que a teoria de grafos aliada aos processos enumerativos e a problemas de otimização são grandes recursos que temos para que o aluno possa aprender ou desenvolver a capacidade de resolver problemas sem que tenha conhecimento prévio da teoria de grafos ou que ele seja obrigado a lembrar de fórmulas combinatoriais. Isto também acentua a intercessão desse tipo de questão com a implementação computacional de tal grafo através de uma matriz adjacência e de técnicas de algoritmo para buscar a solução ótima. Podemos usar a ideia do algoritmo do vizinho mais próximo ou do caixeiro viajante para resolver essa questão. Sua implementação mostra como a estrutura do pensamento computacional construído é diferente do algoritmo idealizado através da abstração matemática, pois na implementação é utilizado outras estruturas computacionais que não existem na linguagem matemática, tais como ponteiros ou vetores para armazenar dados estruturas condicionais e de lógica.

Na solução matemática apresentamos a solução dada diante do grafo apresentado no problema. Já no fluxograma, na abstração e implementação o problema foi generalizado para um grafo qualquer que seja introduzido no início do programa. Notamos que o pensamento computacional auxilia na abstração e na generalização da solução, mas ao implementarmos as estruturas e organização do programa se encontram muito mais detalhados em relação ao fluxograma apresentado. Isso porque são necessários vários pequenos detalhes, tais como a estrutura lógica do programa e estrutura para armazenamento de resultados e do grafo em si.

Questão 17 - (ENEM – 2011) Um técnico em refrigeração precisa revisar todos os pontos de saída de ar de um escritório com várias salas. Na imagem apresentada, cada ponto indicado por uma letra é a saída do ar e os segmentos são as tubulações.

Figura 92 – Questão 17



Fonte: Os autores, 2021

Iniciando a revisão pelo ponto K e terminando em F, sem passar mais de uma vez por cada ponto, o caminho será passando pelos pontos

a) K, I e F

b) K, J, I, G, L e F

c) K, L, G, I, J, H e F

d) K, J, H, I, G, L e F

e) K, L, G, I, H, J e F

17.1 Resolução Matemática:

Vamos representar o grafo acima através de uma matriz de adjacência:

Tabela 21 – Tabela de Resolução matemática – Questão 17

	F	G	H	I	J	K	L
F	0	1	1	1	0	0	0
G	1	0	0	1	0	0	1
H	1	0	0	1	1	0	0
I	1	1	1	0	1	1	0
J	0	0	1	1	0	1	0
K	0	0	0	1	1	0	1
L	0	1	0	0	0	1	0

Fonte: Os autores, 2021.

Vamos de K e terminar em F. Vamos tentar enumerar:

K – I – G – L e a solução aqui para

K – J – H – I – G – L e a solução para aqui

K – L – G – I – H – J e a solução para aqui

K – L – G – I – J – H – F

17.2 Resolução Pensamento Computacional

17A - Decomposição

Vamos enumerar alguns caminhos que o computador vai percorrer. Ele deve partir do vértice de rótulo K e chegar ao vértice de rótulo F sem repetir nenhum vértice e respeitando as ligações entre os vértices.

Vamos representar o grafo através de uma matriz de adjacência em que 1 significa ligação entre os vértices e 0 é quando não há ligação entre os vértices analisados.

Iniciamos de K e vamos para o primeiro vértice que aparecer atribuído 1 na linha do vértice K. E a partir dele vamos montando o caminho.

K – I – G – L não é solução porque não tem mais como seguir em frente sem repetir vértice.

K – J – H – I – G – L não é solução porque não tem mais como seguir em frente sem repetir vértice.

K – L – G – I – H – J não é solução porque não tem mais como seguir em frente sem repetir vértice

K – L – G – I – J – H – F é a solução, pois não repete vértice e obedecer às regras do enunciado.

17B - Padrão

Montar uma matriz 8 x 8 simétrica rotulada com F,G, H, I, J, K, L. Nessa matriz, vamos iniciar o caminho na linha 7 em que está o rótulo K (ou na coluna 7) e vamos para o primeiro vértice em que esteja com 1. Vamos seguindo em frente enquanto for possível seguir em frente sem repetir vértices de modo que não repitamos vértices e terminemos em F. O caminho percorrido é K – L – G – I – J – H – F.

17C - Abstração

Seja um grafo com n vértices rotulados.

Vamos criar uma matriz (n + 1) x (n + 1) com os rótulos dos vértices.

Atribuímos 0 aos vértices que não tem conexão e 1 para os vértices que tem conexão.

Iniciamos no vértice que for estabelecido para iniciar, por exemplo a_i.

Vamos para o primeiro vértice que tem número 1 na linha do vértice a_i.

A partir de a_i vamos para o próximo que aparecer com 1 na linha desse vértice.

Vamos seguindo esse procedimento até chegarmos ao vértice a_j

17D - Algoritmo

Inicio ( )
    Leia (grafo, n)
    Construa uma matriz quadrada (n + 1)
    Coloque 1 se a_i está associado com a_j
    Coloque 0 se a_i está associado com a_j
    Saia do vértice inicio
    Vá para o vértice da coluna a_j que está com 1.
    Repita esse processo até percorrer todos os vértices e não repetir o vértice anterior e chegar só vértice saída
    Imprimir (caminho)
Fim ( )

17.3 Comparando Soluções

Tabela 22 - Comparando Soluções – Questão 17

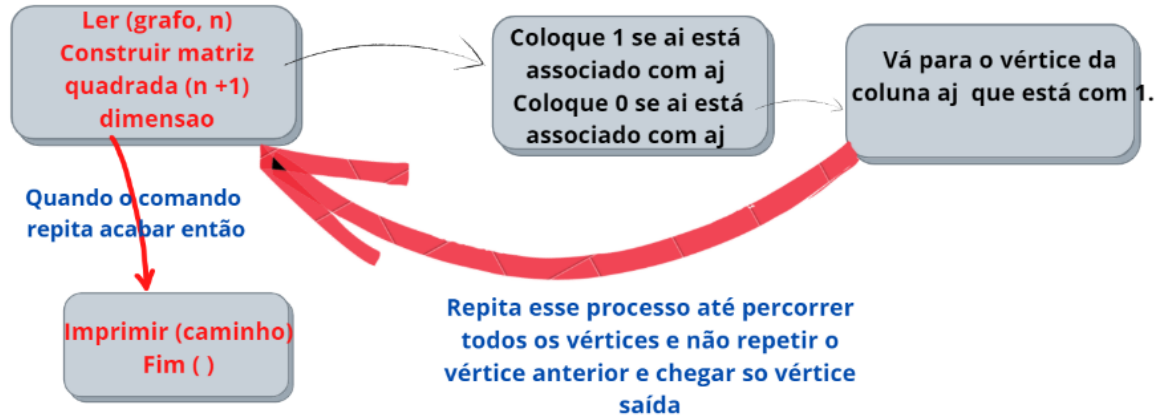
A – Resolução Matemática

Vamos representar o grafo acima através de uma matriz de adjacência:

	F	G	H	I	J	K	L
F	0	1	1	1	0	0	0
G	1	0	0	1	0	0	1
H	1	0	0	1	1	0	0
I	1	1	1	0	1	1	0
J	0	0	1	1	0	1	0
K	0	0	0	1	1	0	1
L	0	1	0	0	0	1	0

Vamos de K e terminar em F. Vamos tentar enumerar:
K – I – G – L e a solução aqui para
K – J – H – I – G – L e a solução para aqui
K – L – G – I – H – J e a solução para aqui
K – L – G – I – J – H – F

B – Fluxograma



C – Abstração

```
Inicio ( )
    Leia (grafo, n)
    Construa uma matriz quadrada (n + 1)
    Coloque 1 se ai¬ está associado com aj
    Coloque 0 se ai¬ está associado com aj
    Saia do vértice inicio
    Vá para o vértice da coluna aj  que está com 1.
    Repita esse processo até percorrer todos os vértices e não repetir o vértice anterior e chegar so vértice saída
    Imprimir (caminho)
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 91 – Implementação em linguagem C++ da questão 17

```
#include <iostream>
#include <vector>

using namespace std;

vector<vector<int> > mapa;
vector<vector<int> > caminhos;
int entrada, saida;
int n, m;

void Hamiltoniano(vector<int> caminho, int u, vector<bool> visit
    caminho.push_back(u);
    if (caminho.size() == n){
        if (caminho[n - 1] == saida){
            caminhos.push_back(caminho);
        }
    } else {
        visitados[u] = true;
        for(int i = 0; i < mapa[u].size(); ++i){
            int vizinho = mapa[u][i];

            if (!visitados[vizinho]){
                Hamiltoniano(caminho, vizinho, visitados);
            }
        }
    }
}


int main()
{
   vector<bool> visitados;
   int u, v;

   cout << "Quantos vertices ha no mapa?" << endl;
   cin >> n;
   for(int i = 0; i <= n; ++i){
       mapa.push_back(vector<int>());
       visitados.push_back(false);
   }

   cout << "Quantas arestas ha no desenho?" << endl;
   cin >> m;

   cout << "Descreva as " << m << " arestas do seu desenho. Exemplo: 1 3 significa uma aresta entre os vertices 1 e 3." << endl
   for(int i = 0; i < m; ++i){
       cin >> u >> v;
       mapa[u].push_back(v);
       mapa[v].push_back(u);
   }

   cout << "Qual o vertice de entrada?" << endl;
   cin >> entrada;

   cout << "Qual o vertice de saida?" << endl;
   cout << "Estao sao as maneiras possiveis de passar por todos os pontos comecando em " << entrada << " e terminando em " << saida << ":"
   Hamiltoniano(vector<int>(), entrada, visitados);

   if (caminhos.size() > 0){
       for(int i = 0; i < caminhos.size(); ++i){
           cout << caminhos[i][0];
           for(int j = 1; j < caminhos[i].size(); ++j){
               cout << " -> " << caminhos[i][j];
           }
           cout << endl;
       }
   } else {
       cout << "Nao ha como passar por todos os pontos uma unica vez comecando em " << entrada << " e terminando em " << saida << "." << er
   }

   return 0;
```

Fonte: Os autores, 2021

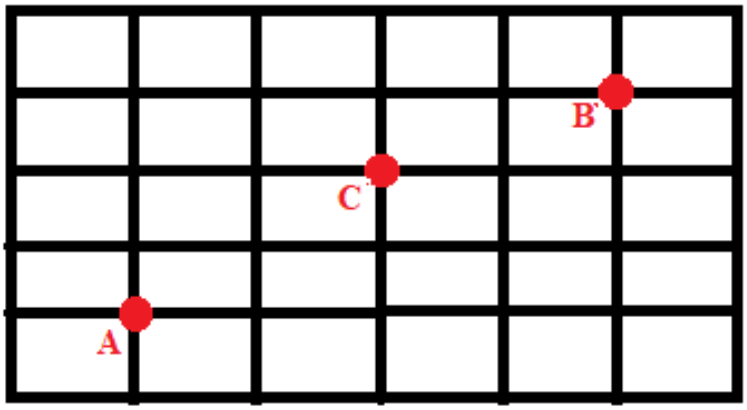
17.4 Considerações didáticas

A questão 17 apesar de trabalhar com grafo e aparecer ser de otimização, se trata de uma questão de existência pois se pergunta se é possível fazer um caminho saindo de do vértice K e chegando no vértice F, sem repetir em nenhum vértice nesse caminho. Ou seja. é uma questão que trabalha com a ideia de caminho euleriano dentro de um subconjunto de vértices do grafo, ou podemos interpretar como um subconjunto de vértices do grafo com uma condicional de partida e chegada diferente das questões anteriores, nas quais o aluno poderia iniciar e terminar o caminho em quaisquer um dos vértices. Ela tem caráter enumerativo, pois o aluno pode através da numeração encontrar a resposta tem como de otimização pois ao não repetirmos nenhum vértice estamos economizando algum recurso nem que seja o tempo e tem natureza de classificação, pois existe a condição de começar no vértice k e terminar no vértice F. A solução matemática apresentada é feita através de um processo enumerativo baseado na matriz de adjacência, o que torna essa busca através da listagem mais eficiente. Já em pensamento computacional essa matriz também foi usada e a ideia do vértice mais próximo para percorrer possíveis caminhos e eliminar aqueles que não obedecem a classificação estipulada na questão.

Vemos através do fluxograma e da abstração que nem sempre é fácil detalhar todos os passos que a implementação do problema usará. No caso a implementação vemos estruturas tais como a lógica, a de vetores, a de apontador, condicionais, repetição etc.

Questão 18 - (Enem – 2021) Três amigos, André, Bernardo e Carlos, moram em um condomínio fechado de uma cidade. O quadriculado representa a localização das ruas paralelas e perpendiculares, delimitando quadras de mesmo tamanho nesse condomínio, em que nos pontos A, B e C estão localizadas as casas de André, Bernardo e Carlos, respectivamente.

Figura 92 - Questão 18



Fonte: ENEM (2021)

André deseja deslocar-se da sua casa até a casa de Bernardo, sem passar pela casa de Carlos, seguindo ao logo das ruas do condomínio, fazendo sempre deslocamentos para a direita (→) ou para cima (↑) , segundo o esquema da figura. O número de diferentes caminhos que André poderá utilizar para realizar o deslocamento nas condições propostas é:

a) 4
b) 14
c) 17
d) 35
e) 48

18.1 Resolução Matemática:

Destacamos essa solução porque ela é similar a questão 18 e é do Enem desse ano. Temos o mesmo modelo de questão e resolução, por isso vamos direto à solução. Mas existe uma exceção, o problema quer que não passe pela casa do Carlos. Então vamos ver o total de caminhos possíveis e subtraímos a quantidade que passa pela casa do Carlos.

$$P_7^{4,3} = \frac{4!\,3!}{7!} = \frac{7.6.5.4!}{3!\,4!} = \frac{7.6.5}{3.2.1} = 35$$

$$P_4^{2,2} x P_3^{2,1} = \frac{4!}{2!\,2!} . \frac{3!}{2!\,1!} = \frac{12}{2} . \frac{3}{1} = 18$$

Então existe 35 – 18 = 17 caminhos, opção c.

18.2 Resolução Pensamento Computacional

18A - Decomposição

Análogo à questão 24, temos que dividir nesse caso em três etapas. A primeira é contarmos quantos caminhos podemos fazer de A até C no total, fazendo apenas movimentos para a direita ou para cima. A segunda é contarmos quantos desses caminhos passam pela casa do Carlos. E por último, vamos subtrair o total de caminhos que existem que passam por C. Na primeira e segunda etapa usamos a permutação com repetição e posteriormente fazemos a diferença entre elas.

18B - Padrão

Vemos que existe uma exceção e sempre precisamos atacar o problema por aquilo que não queremos acontecer. Ou seja, perguntar o que não queremos que aconteça: que não passe pela casa do Carlos.

Depois disso, calculamos a permutação com repetição de A até B.

18C - Abstração

Contamos quantas vezes podemos ir para a direita e para cima. Calculamos a permutação com repetição referente a essa etapa.

Contamos quantas vezes podemos ir para a direita e para cima passando pela casa de Carlos (C) e calculamos a permutação nesse caso.

Calculamos a diferença entre o total de permutações e a exceção que não queremos que aconteça.

18D - Algoritmo

```
Inicio ( )
        Imprima (Qual o tamanho da malha do seu caminho)
        Leia (cima, direita)
        Imprima ("Quantos pontos haverão no trecho?")
Leia (paradas)
```

Produto = 1
Função calcula_trecho( )
Permutacao = $\frac{cima!direita!}{(cima+direita)!}$
Fim função calcula_trecho ( )
Enquanto (paradas <> 0) faça
Imprimir("Quais são as coordenadas do ponto" A_{parada} ?)
Leia ($ponto_{parada}$)
Calcula_trecho (calcula permutação com repetição do trecho A_i até $ponto_{parada}$)
Produto = calcula trecho * produto
Parada = parada ++
Fim enquanto ( )
Função exceção ( )
exceção = Calcula_trecho(calcula a permutação com repetição que passa pela casa C)
Fim exceção
Imprimir (produto - exceção)
Fim

18.3 Comparando Soluções

Tabela 23 - Comparando Soluções – Questão 18

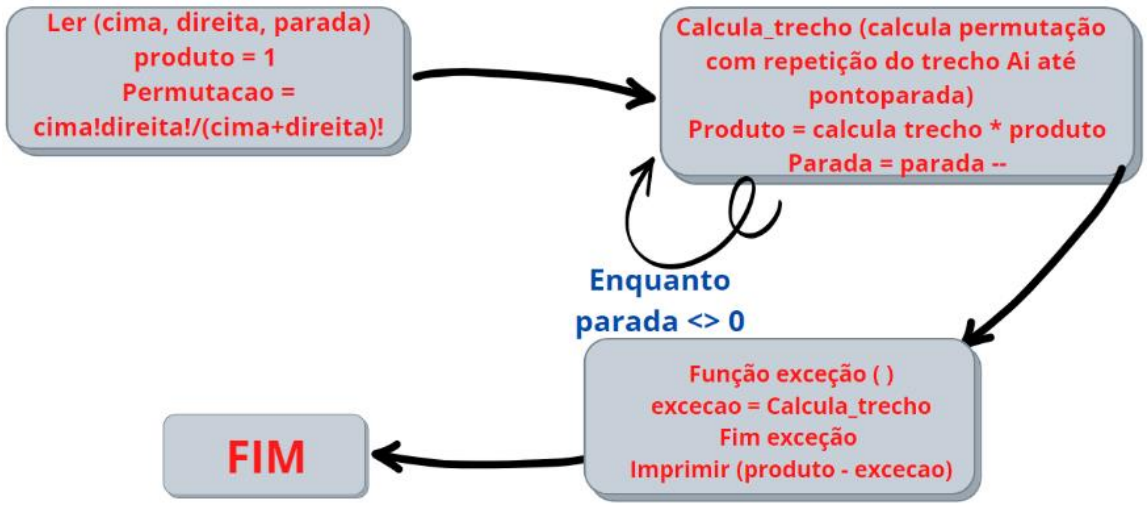
A – Resolução Matemática

$$P_7^{4,3} = \frac{4!\,3!}{7!} = \frac{7.6.5.4!}{3!\,4!} = \frac{7.6.5}{3.2.1} = 35$$

$$P_4^{2,2} x P_3^{2,1} = \frac{4!}{2!\,2!} . \frac{3!}{2!\,1!} = \frac{12}{2} . \frac{3}{1} = 18$$

Então existe 35 – 18 = 17 caminhos

B – Fluxograma



C – Abstração

```
Inicio ( )
    Imprima (Qual o tamanho da malha do seu caminho)
    Leia (cima, direita)
    Imprima ("Quantos pontos haverá no trecho?")
    Leia (paradas)
    Produto = 1
    Função calcula_trecho( )
        Permutacao = cima!direita!/(cima+direita)!
    Fim função calcula_trecho ( )
    Enquanto (paradas <> 0) faça
        Imprimir("Quais são as coordenadas do ponto" Aparada ?)
        Leia (pontoparada)
        Calcula_trecho (calcula permutação com repetição do trecho Ai até pontoparada)
        Produto = calcula trecho * produto
        Parada = parada --
    Fim enquanto ( )
    Função exceção ( )
        excecao = Calcula_trecho(calcula a permutação com repetição que passa pela casa C)
    Fim exceção
    Imprimir (produto - excecao)
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 93 – Implementação em linguagem C++ da questão 18

```cpp
#include <iostream>

using namespace std;

int Fatorial(int n){
    if (n == 1){
        return 1;
    } else {
        return n * Fatorial(n - 1);
    }
}

int PermutacaoComRepeticao(int cima, int direita){
    return Fatorial(cima + direita)/(Fatorial(cima) * Fatorial(direita));
}
// A resposta é o número de permutações com repetição envolvendo as vezes que João precisa ir pra cima e as que ele precisa ir para a direi


void EscreveCaminhos(string caminho, int cima, int direita){
    if (cima == 0 && direita == 0){
        cout << caminho << endl;
    } else if (cima > 0 && direita > 0) {
        EscreveCaminhos(caminho + 'C', cima - 1, direita);
        EscreveCaminhos(caminho + 'D', cima, direita - 1);
    } else if (direita == 0) {
        EscreveCaminhos(caminho + 'C', cima - 1, direita);
    } else if (cima == 0) {
        EscreveCaminhos(caminho + 'D', cima, direita - 1);
    }
}

int main()
{
    int Ax, Ay, Bx, By, Cx, Cy;
    int resposta1, resposta2, resposta3;

    cout << "Escreva as coordenadas da casa do Andre no formato x y. Exemplo: 1 1 indica que a casa de Andre esta em (1, 1)." << endl;
    cin >> Ax >> Ay;

    cout << "Escreva as coordenadas da casa da Bernardo no formato x y." << endl;
    cin >> Bx >> By;
    while (Bx < Ax || By < Ay){
        cout << "A casa do Bernardo precisa estar para cima e para direita da casa do Andre. Por favor, digite novas coordenadas." << endl;
        cin >> Bx >> By;
    }
    resposta1 = PermutacaoComRepeticao(By - Ay, Bx - Ax);

    cout << "Escreva as coordenadas da casa de Carlos no formato x y." << endl;
    cin >> Cx >> Cy;
    while ((Cx < Ax || Cy < Ay) || (Cx > Bx || Cy > By)){
        cout << "A casa do Carlos precisa estar entre as casas do Andre e do Bernardo. Por favor, digite novas coordenadas." << endl;
        cin >> Bx >> By;
    }
    resposta2 = PermutacaoComRepeticao(Cy - Ay, Cx - Ax);
    resposta3 = PermutacaoComRepeticao(By - Cy, Bx - Cx);

    cout << "O numero de caminhos que existem de Andre ate a casa de Bernardo passando pela casa de Carlos e ";
    if (Cx <= Bx && Cy <= By){
        cout << resposta1 - resposta2 * resposta3 << "." << endl;
    } else {
        cout << resposta1 << "." << endl;
    }

    cout << "Esses sao os possiveis caminhos da casa do Andre ate a casa do Bernardo:" << endl;
    EscreveCaminhos("", By - Ay, Bx - Ax);

    cout << "Esses sao os possiveis caminhos da casa do Andre ate a casa do Carlos:" << endl;
    EscreveCaminhos("", Cy - Ay, Cx - Ax);

    cout << "Esses sao os possiveis caminhos da casa do Carlos ate a casa do Bernardo:" << endl;
    EscreveCaminhos("", By - Cy, Bx - Cx);

    return 0;
}
```

Fonte: Os autores, 2021

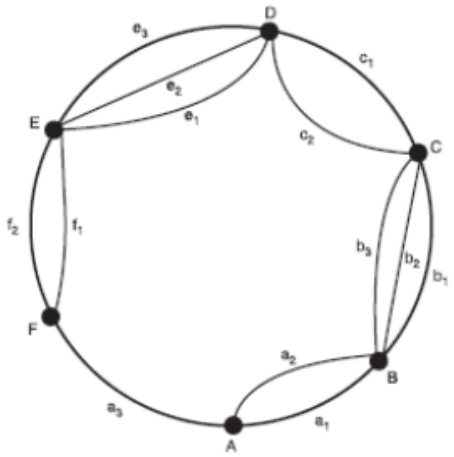
18.4 Considerações didáticas

Esse problema é muito parecido com o anterior, difere apenas no fato de que existe uma exceção de não passar em determinado ponto. Tal situação modificou o fluxograma, a solução matemática e a abstração no ponto que tivemos que criar uma função de exceção daquilo que não queremos que aconteça. Já na implementação percebemos que a estrutura aumentou consideravelmente em relação a questão anterior.

A questão 18 é muito parecida com a anterior bem como na resolução, mas cabe colocá-la na dissertação pois foi uma questão do Enem de 2021 e somente difere da questão da FUVEST de 93 no fato de que é exigido que do ponto de partida até a chegada a pessoa passe pelo ponto c no meio do caminho, mas os movimentos são os mesmos: para cima ou para direita. Essa questão também se trata de um problema de combinatória de contagem, mas envolve uma condição que é a passar pelo ponto c no meio do caminho com isso acabamos calculando duas permutações com repetição e pelo princípio multiplicativo obtemos as possibilidades de não passar pelo ponto desejado e o total de caminhos possíveis. Assim, através do princípio da inclusão e exclusão fazemos a diferença entre o total de caminhos possíveis e os caminhos não viáveis.

Questão 19 - (UFRN) A figura abaixo representa um mapa das estradas que interligam as comunidades A, B, C, D, E F.

Figura 94 – Questão 19



Fonte: UFRN

Assinale a opção que indica quantos percursos diferentes existem para se chegar à comunidade D (partindo-se de A), sem que se passe mais de uma vez numa mesma comunidade, em cada percurso.

a) 72

b) 12

c) 18

d) 36

19.1 Resolução Matemática:

Podemos partir de A e seguir pela direita ou pela esquerda. Se formos pela direita, passaremos pelos pontos B e C. Existem 2 caminhos de A para B, 3 caminhos de B para C e 2 caminhos de C até D. Logo existem por esse percurso, 2.3.2 = 12 caminhos.

Mas também podemos ir pela esquerda. Nesse caso passaremos A, F, E D. De A para F existe 1 caminho, de F para E existem 2 caminhos, de E para D existem 3 maneiras. Logo podemos ir de 1.2.3 = 6 maneiras distintas.

Então, podemos ir pela esquerda ou pela direita. Então temos no total 12 + 6 = 18 maneiras distintas sem repetir pontos. Letra c.

19.2 Resolução Pensamento Computacional

19A - Decomposição

Observamos que se trata de um percurso circular com início e fim determinados. Isso divide o percurso em dois casos. De um lado podemos ir pela direita ou pela esquerda. E em cada um deles aplicamos o princípio fundamental da contagem

19B - Padrão

Separamos em dois casos e contamos entre cada vértice quantos caminhos existem. Fazemos o produto entre os possíveis caminhos e depois somamos os dois casos.

19C - Abstração

Seja um caminho circular com m pontos n caminhos. Vamos sair de A_1 e vamos chegar em A_m sem repetir nenhuma cidade. Então podemos pela esquerda ou direita de A_1. Se formos pela direita, pelo princípio fundamental da contagem teremos o produto da quantidade de caminhos entre cada vértice até chegarmos em A_m.

Por outro lado, se formos por pela esquerda, de maneira análoga faremos o produto entra a quantidade de caminhos entre cada vértice até atingirmos A_m.

Pelo princípio aditivo, podemos ir pela direita ou pela esquerda e assim adicionamos ambos resultados.

19D - Algoritmo

```
Inicio ( )
Imprimir ("Quantas cidades têm no problema?")
Ler (qtd_cidades)
Imprimir ("Quantas cidades têm entre", A_origem "e a cidade", A_destino "pela esquerda?")
Ler (qtd_cidades_esquerda)
qtd_cidades_direita = qtd_cidades – qtd_cidades_esquerda
Enquanto (qtd_cidades_esquerda <> 0 ) faça
        Imprimir ("Quantos caminhos existem de", A_qtd_cidades_esquerda, "até", A_qtd_cidades_esquerda -1?)
        Ler (num)
        produto_esquerda = num * produto_esquerda
        qtd_cidades_esquerda = qtd_cidades_esquerda – 1
fim enquanto ( )
Enquanto (qtd_cidades_direita <> 0 ) faça
        Imprimir ("Quantos caminhos existem de", A_qtd_cidades_direita, "até", A_qtd_cidades_direita -1?)
        Ler (num)
        produto_direita = num * produto_direita
        qtd_cidades_direita = qtd_cidades_direita – 1
fim enquanto
resultado = produto_cidades_direita + produto_cidades_esquerda
imprimir ("existem", resultado, caminhos de", A_origem, "até", A_destino) fim
```

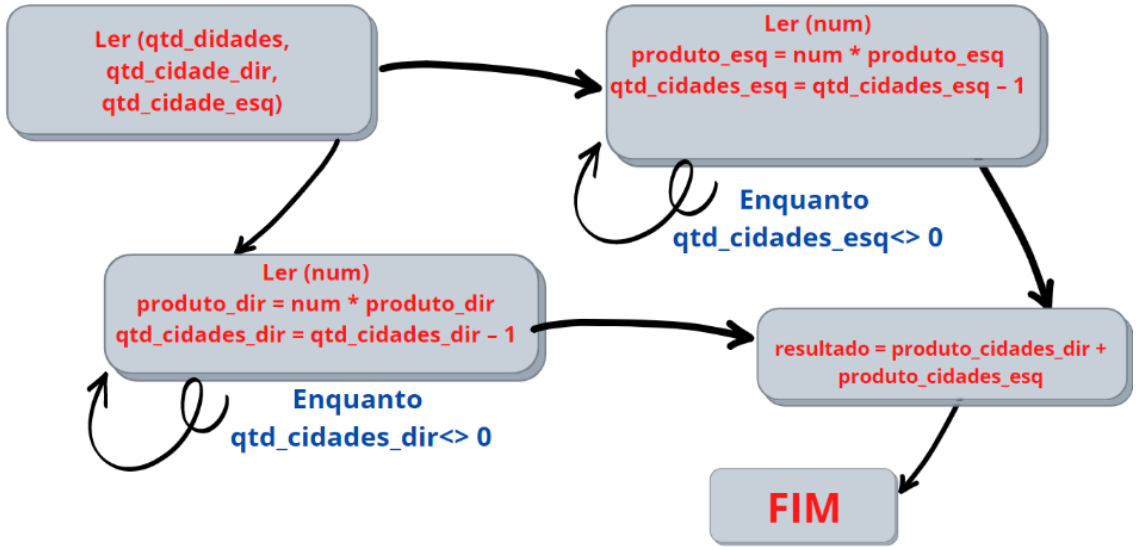
19.3 Comparando Soluções

Tabela 24 - Comparando Soluções – Questão 19

A – Resolução Matemática

Podemos partir de A e seguir pela direita ou pela esquerda. Se formos pela direita, passaremos pelos pontos B e C . Logo existem por esse percurso, 2.3.2 = 12 caminhos. Mas também podemos ir pela esquerda. Nesse caso passaremos A, F, E D. Logo podemos ir de 1.2.3 = 6 maneiras distintas. Então temos no total 12 + 6 = 18 maneiras distintas sem repetir pontos.

B – Fluxograma



C – Abstração

```
Inicio ( )
    Imprimir ("Quantas cidades têm no problema?")
    Ler (qtd_cidades)
    Imprimir ("Quantas cidades têm entre",  Aorigem "e a cidade", Adestino "pela esquerda?")
    Ler (qtd_cidades_esquerda)
    qtd_cidades_direita = qtd_cidades – qtd_cidades_esquerda
    Enquanto (qtd_cidades_esquerda <> 0 ) faça
        Imprimir ("Quantos caminhos existem de", Aqtd_cidades_esquerda, "até", Aqtd_cidades_esquerda -1?)
        Ler (num)
        produto_esquerda = num * produto_esquerda
        qtd_cidades_esquerda = qtd_cidades_esquerda – 1
    fim enquanto ( )
    Enquanto (qtd_cidades_direita <> 0 ) faça
        Imprimir ("Quantos caminhos existem de", Aqtd_cidades_direita, "até", Aqtd_cidades_direita -1?)
        Ler (num)
        produto_direita = num * produto_direita
        qtd_cidades_direita = qtd_cidades_direita – 1
    fim enquanto
    resultado = produto_cidades_direita + produto_cidades_esquerda
    imprimir ("existem", resultado, caminhos de", Aorigem, "até", Adestino)
fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 95 – Implementação em linguagem C++ da questão 19

```cpp
#include <algorithm>
#include <iostream>
#include <vector>

using namespace std;

int main()
{
    vector<int> ligacoes, produtos;
    int n, x, entrada, saida, resposta1, resposta2;

    cout << "Quantas comunidades ha no mapa?" << endl;
    cin >> n;

    for(int i = 1; i < n; ++i){
        cout << "Quantas ligacoes tem entre as cidades " << i << " e " << i + 1 << "?" << endl;
        cin >> x;
        ligacoes.push_back(x);
    }
    cout << "Quantas ligacoes tem entre as cidades " << n << " e 1?" << endl;
    cin >> x;
    ligacoes.push_back(x);

    cout << "De qual comunidade sair?" << endl;
    cin >> entrada;
    entrada -= 1;

    cout << "Pra qual comunidade chegar?" << endl;
    cin >> saida;
    while (entrada == saida){
        cout << "A entrada e a saida nao podem ser no mesmo lugar. Escolha outro destino." << endl;
        cin >> saida;
    }
    saida -= 1;

    resposta1 = 1;
    x = entrada;
    while (x != saida){
        resposta1 *= ligacoes[x];
        produtos.push_back(ligacoes[x]);
        x = (x + 1) % n;
    }
    cout << "Indo da cidade " << entrada << " ate a cidade " << saida << " em sentido anti-horario, temos ";
    cout << produtos[0];
    for(int i = 1; i < produtos.size(); ++i){
        cout << " * " << produtos[i];
    }
    cout << " = " << resposta1 << " possibilidades." << endl;

    resposta2 = 1;
    x = entrada;
    produtos.clear();
    while (x != saida){
        x = (x + n - 1) % n;
        resposta2 *= ligacoes[x];
        produtos.push_back(ligacoes[x]);
    }
    cout << "Indo da cidade " << entrada << " ate a cidade " << saida << " em sentido anti-horario, temos ";
    cout << produtos[0];
    for(int i = 1; i < produtos.size(); ++i){
        cout << " * " << produtos[i];
    }
    cout << " = " << resposta2 << " possibilidades." << endl;

    entrada += 1;
    saida += 1;
    cout << "Logo, temos " << resposta1 << " + " << resposta2 << " = " << resposta1 + resposta2 << " caminhos diferentes da cidade " << entr

    return 0;
}
```

Fonte: Os autores, 2021

19.4 Considerações didáticas

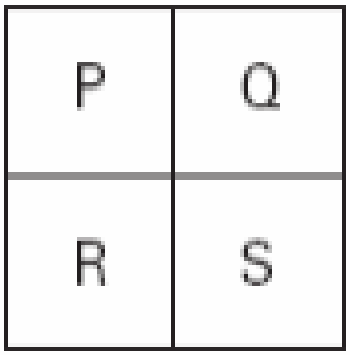
Na questão 19 temos um tipo de questão de combinatória de contagem, mas ela tem natureza enumerativa, uma vez que podemos listar todos os caminhos possíveis para sair do ponto a até o ponto b. Ela também possui natureza de existência, uma vez que para contarmos o total de caminhos devemos saber se existe um caminho possível de escrever. Ela tem natureza de otimização, pois podemos nos perguntar se existe um caminho menor do que o outro para sair de A e chegar até o ponto D e ela tem caráter de classificação, uma vez que podemos separar o trajeto indo pela esquerda ou pela direita.

A solução matemática apresentada no trabalho soluciona a questão através do princípio multiplicativo, porém poderíamos explorar tal questão através de uma solução enumerativa ou ainda pensarmos em implementar a solução desta questão computacionalmente de modo que o algoritmo exibisse todos os trajetos possíveis. A abstração da questão se deu através de pensarmos na quantidade de cidades existentes em cada caminho, já na implementação computacional levamos em consideração a quantidade de ligações entre uma cidade e outra que não foi citado no pensamento matemático. Isso mostra que quando implementamos problemas matemáticos, outros detalhes surgem talvez que na prática ou com auxílio de imagem não seja necessário implementar computacionalmente.

As quatro representações convergem para a mesma ideia. A solução matemática resolve um caso particular, a abstração e fluxograma representam uma generalização do problema e a implementação em linguagem C++ mostra as diferenças entre a etapa da abstração e a implementação. Na implementação vemos o uso de uma biblioteca, ou seja um acervo de comandos específicos, de vetores e utiliza essa ferramenta de auxílio para armazenar os dados do grafo em questão. E há também o uso de um vetor com um ponteiro. Isso significa que onde o ponteiro aponta o valor que ali se encontra fica alocado e não é perdido ou trocado ao longo da programação.

Questão 20 - (Vunesp – 2003) Dispomos de 4 cores distintas e temos que colorir o mapa mostrado na figura com os países P, Q, R e S, de modo que países cuja fronteira é uma linha não podem ser coloridos com a mesma cor.

Figura 96 – Questão 20



Fonte: Vunesp (2003)

Responda, justificando sua resposta, de quantas maneiras é possível colorir o mapa, se:

a) os países P e S forem coloridos com cores distintas?

20.1a Resolução Matemática:

Para colorirmos P, temos 4 cores. Ao escolher a cor 1 para pintar P, precisamos escolher outra cor para colorir S. Nesse caso sobram 3 cores para pintá-la.

Mas para colorir Q ou R, podemos começar por onde quisermos. Para o primeiro vértice escolhido dentre os restantes, teremos 2 cores e para o último 1 cor. Assim, podemos pintar de 4.3.2.1 = 4! = 24 maneiras essa bandeira

20.2a Resolução Pensamento Computacional

<u>20A - Decomposição</u>

Devemos analisar o vértice P e a quantidade de cores disponíveis. Escolher uma cor e colocar o valor de possibilidades de escolha como sendo a quantidade de cores disponíveis para pintar o vértice rotulado P. Para pintar o vértice temos a quantidade de cores menos uma unidade que já usamos em P. Para os outros restantes temos 2 e 2 e, portanto, 4.3.2.2= 48 maneiras de pintar seguindo essas restrições.

20B - Padrão

A dimensão da matriz é a mesma quantidade de cores que temos disponível: quatro.

Usamos o princípio fundamental da contagem de modo que $a_{11} <> a_{22}$, existem 2.4! Maneiras de pintar essa malha quadriculada

20C - Abstração

Dada uma malha quadriculada n x n e com n cores para colorir todos esses n espaços de modo que quadriculados adjacentes não podem ter a mesma cor. Então temos n! maneiras de pintar essa malha de modo que os vértices a_{11} e a_{nn} não tenham a mesma cor.

20D - Algoritmo

```
Inicio ( )
        Imprimir ("Quantos quadriculados têm a sua malha?")
        Ler (qtd_quadriculados)
        Para (0 < i < qtd_quadriculados) faça
            i = cor i
            qtd_quadriculados = qtd_quadriculados – 1
        fim para
        imprimir (Existem qtd_quadriculados! a ser usado nessa pintura)
fim
```

b) os países P e S forem coloridos com a mesma cor?

20.1b Resolução Matemática:

Temos quatro escolhas para pintar P e apenas uma para S, uma vez que ele precisa da mesma cor que P. Sobram então 3 cores para o R ou Q. Quem for escolhido por último terá 2 opções. Isso totaliza 4.3.3.1= 36 opções.

20.2b Resolução Pensamento Computacional

20A - Decomposição

Primeiro pintamos os quadrados a_{11} e a_{22}. No primeiro temos quatro cores para escolher e no caso do a_{22} teremos uma opção que é a cor usada no a_{11}. E para os quadrados restantes sobram 3 e 3 cores para colorir e no total resulta em 4.3.3.1 = 36 opções de pintura

20B - Padrão

Vamos observar padrões. Para uma malha 3 x 3:

Tabela 25 -Resolução Matemática – Questão 20A

9	8	8
8	7	7
8	7	1

Fonte: Os autores, 2021

Para uma malha 4 x 4:

Tabela 26 -Resolução Matemática – Questão 20B

16	15	15	15
15	14	14	14
15	14	14	13
15	14	13	1

Fonte: Os autores, 2021

Para uma malha 5 x 5:

Tabela 27 -Resolução Matemática – Questão 20C

25	24	24	24	24
24	23	23	23	23
24	23	23	23	23
24	23	23	23	22
24	23	23	22	1

Fonte: Os autores, 2021

Percebemos alguns padrões nas malhas analisadas. Primeiro que a quantidade de cores que podemos pintar o a_{11} é a mesma que a quantidade de quadrados existentes na malha quadrada. Sempre teremos uma opção de pintura para o último quadrado da malha, já que esse depende da cor escolhida na quadrado a_{11}. Quando pintamos a primeira linha e coluna, com exceção do a_{11} temos a quantidade de quadrados menos uma unidade de escolhas para pintar esses quadrados. Para colorir os quadrados acima do último quadrado do canto direito inferior e o quadrado à sua esquerda temos o total de quadrados da malha menos 3 unidades como opções de escolha. E para o restante dos quadrados temos a quantidade de malhas menos duas unidades para pintar cada um dos quadrados.

20C - Abstração

Se tivermos uma malha n x n, existem n^2 quadrados para colorirmos. O quadrado a_{11} tem n^2 opções de pintarmos. O quadrado a_{nn} tem uma opção, que é a mesma cor que pintamos o a_{11}. Temos na primeira linha e coluna $2n - 2$ quadrados para pintarmos. Esses podem ser coloridos de $n^2 - 1$ maneiras. Então no total temos $(2n - 2).(n^2 - 1)$. Para os quadrados ao lado do quadrado a_{nn}, ou seja, os quadrados $a_{n(n-1)}$ e $a_{(n-1)n}$ temos $(n^2 - 3)$ opções de cores para cada

um. Então são 2.(n^2 - 3). Sobram para pintar ao longo da malha n^2 - 4 – (2n – 2) = n^2 - 2n – quadrados para pintarmos usando (n^2 - 2) cores. Isso nos dá uma opção de colorir de (n^2 - 2n – 2).(n^2 - 2) maneiras esses quadrados.

Agora vamos fazer o produto desses casos juntos: n^2.1.(2n–2).(n^2-1).2.(n^2-3).(n^2-2n-2).(n^2 - 2) maneiras de colorir essa malha quadrada.

20D - Algoritmo

Inicio ( )
Imprimir (“Qual a dimensão da malha quadrada?”)
Leia (n)
Imprimir (“existe” n^2 “maneiras de pintar” a_{11})
Imprimir (“existe” 1 “maneira de pintar”, a_{nn})
Imprimir(“existe” , 2.(n^2 - 3), “maneiras de colorir”, $a_{n(n-1)}$ e $a_{(n-1)n}$)
Imprimir (“existe”, (n^2 - 2n – 2).(n^2 - 2), “maneiras de colorir os quadrados restantes”)
Imprimir (“existe”, n^2.1.(2n – 2).(n^2 - 1).2.(n^2 - 3). (n^2 - 2n – 2).(n^2 - 2), “maneiras de colorir essa malha quadrada”)
Fim

20.3 Comparando Soluções

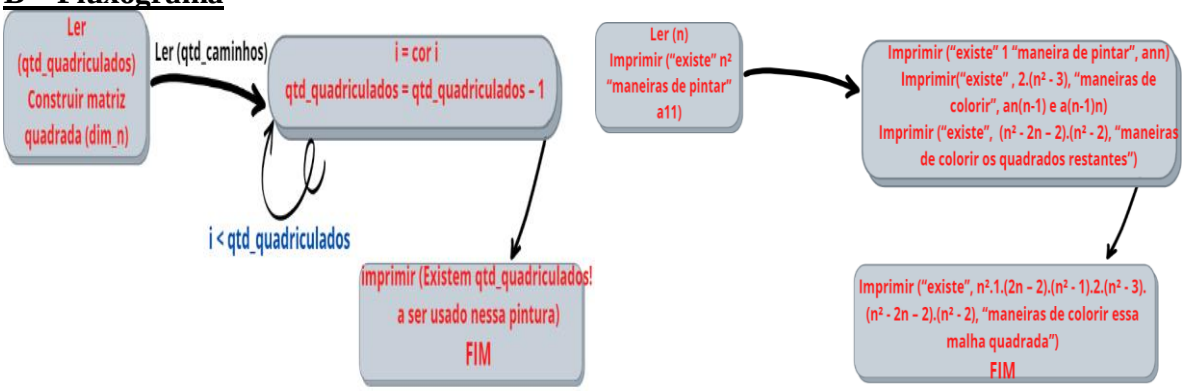
Tabela 28 - Comparando Soluções – Questão 20

A – Resolução Matemática

Para colorirmos P, temos 4 cores. Ao escolher a cor 1 para pintar P, precisamos escolher outra cor para colorir S. Nesse caso sobram 3 cores para pintá-la.

Mas para colorir Q ou R, podemos começar por onde quisermos. Para o primeiro vértice escolhido dentre os restantes, teremos 2 cores e para o último 1 cor. Assim, podemos pintar de 4.3.2.1 = 4! = 24 maneiras essa bandeira. Já no segundo caso, temos quatro escolhas para pintar P e apenas uma para S, uma vez que ele precisa da mesma cor que P. Sobram então 3 cores para o R ou Q. Quem for escolhido por último terá 2 opções. Isso totaliza 4.3.3.1= 36 opções.

B – Fluxograma



C – Abstração

```
Inicio ( )
        Imprimir ("Quantos quadriculados têm a sua malha?")
        Ler (qtd_quadriculados)
        Para (0 < i < qtd_quadriculados) faça
            i = cor i
            qtd_quadriculados = qtd_quadriculados - 1
        fim para
    imprimir (Existem qtd_quadriculados! a ser usado nessa pintura)
fim ( )
```

```
Inicio ( )
    Imprimir ("Qual a dimensão da malha quadrada?")
    Leia (n)
    Imprimir ("existe" n² "maneiras de pintar" a11)
    Imprimir ("existe" 1 "maneira de pintar", ann)
    Imprimir("existe" , 2.(n² - 3), "maneiras de colorir", an(n-1) e a(n-1)n)
    Imprimir ("existe",  (n² - 2n - 2).(n² - 2), "maneiras de colorir os quadrados restantes")
    Imprimir ("existe", n².1.(2n - 2).(n² - 1).2.(n² - 3). (n² - 2n - 2).(n² - 2), "maneiras de colorir essa malha quadrada")
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 97 – Implementação em linguagem C++ da questão 20

```cpp
/*

Questão 28

Dispomos de 4 cores distintas e temos que colorir o mapa mostrado na figura com os países P, Q, R e S,
de modo que países cuja fronteira é uma linha não podem ser coloridos com a mesma cor.

Responda, justificando sua resposta, de quantas maneiras é possível colorir o mapa, se:

a) os países P e S forem coloridos com cores distintas?

b) os países P e S forem coloridos com a mesma cor?

Input recomendado:
3
*/

#include <iostream>
#include <vector>

using namespace std;

int main()
{
    vector<int> produto;
    int N, cores;

    cout << "Qual e o tamanho da malha? (digite n tal que a malha seja n x n)" << endl;
    cin >> N;

    cores = N * N;

    // Opções de cores para (1, 1)
    produto.push_back(cores);
    // Opções de cores para linha 1 inteira e coluna 1 inteira
    for(int i = 0; i < 2 * (N - 1); ++i){
        produto.push_back(cores - 1);
    }
    // Opções pro resto dos elementos
    while (produto.size() < cores - 3){
        produto.push_back(cores - 2);
    }
  // Opções de cores para os dois elementos (N - 1, N) e (N, N - 1)
  // Um adendo é que só pode ser adicionada essa restrição caso N seja maior que 2
  if (N > 2){
      produto.push_back(cores - 3);
      produto.push_back(cores - 3);
  }
  // Opções de cores para (N, N)
  produto.push_back(cores - 1);

  cout << "Ha " << produto[0];
  for(int i = 1; i < produto.size(); ++i){
      cout << " * " << produto[i];
  }
  cout << " possibilidades de cores para esses paises se os paises em (1, 1) e (" << N << ", " << N << ") tiverem cores diferentes." << endl;

  // Alterando as opções de cores para (N, N)
  produto[cores - 1] = 1;
  cout << "Ha " << produto[0];
  for(int i = 1; i < produto.size(); ++i){
      cout << " * " << produto[i];
  }
  cout << " possibilidades de cores para esses paises se os paises em (1, 1) e (" << N << ", " << N << ") tiverem cores iguais." << endl;

  return 0;
```

Fonte: Os autores, 2021

20.4 Considerações didáticas

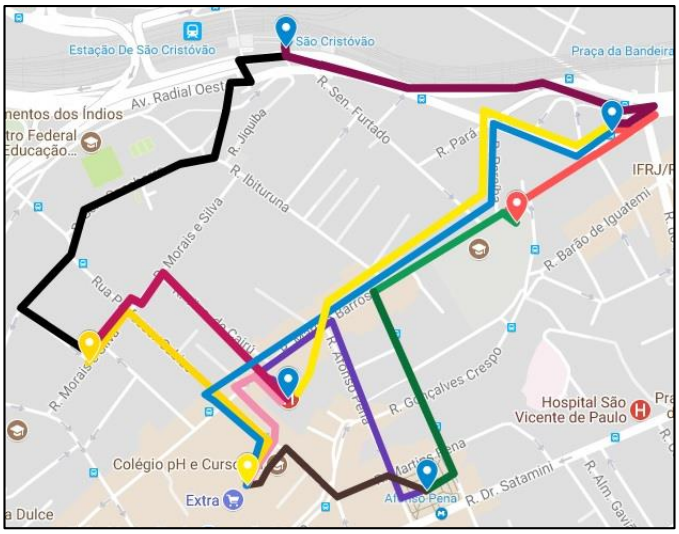
A questão 20 trabalha com a ideia de coloração e pode ser classificada como sendo de contagem pois ainda que trabalha com coloração e coloração nos remete à ideia de utilização coloração mínima, nesse caso o comando da pergunta é quantos tem e não se trata de uma questão de contagem mas a sua natureza pode ser de otimização também, se pensarmos em usar a menor quantidade de cores possível para colorir tal mapa, de modo que regiões adjacentes não tenham a mesma cor. A questão também tem caráter enumerativo, pois como se trata de uma situação simples podemos enumerar, se necessário as possíveis combinações de coloração. Tem natureza de existência, pois podemos nos perguntar se tal configuração é possível e ainda de classificação, pois temos regras a seguir para chegar na Resolução esperada.

A questão se desdobra em duas perguntas. Na segunda pergunta utilizamos uma matriz para representar os valores esperados e buscarmos um padrão para termos uma abstração da questão e posteriormente após encontrarmos padrão implementá-lo computacionalmente.

Na solução matemática, utilizamos o princípio multiplicativo e resolvemos os casos separadamente. No fluxograma e na abstração buscamos generalizar o padrão observado na solução matemática também analisando de forma disjunta. Mas já na implementação utilizamos uma única implementação que já apresenta os dois resultados juntos. É utilizado uma estrutura de repetição enquanto e ponteiros para localizar e armazenar os dados.

Questão 21 - (Borges&Muniz) Lucia é uma menina muito estudiosa. Por tirar excelentes notas, seu pai deu-lhe de presente, no último Dia das Crianças, uma bicicleta nova. Pedalando pelas ruas do Bairro, ela percebeu que, dependendo do lugar de partida e de chegada, poderia chegar mais rápido se escolhesse bem o trajeto por onde passaria. Com seu relógio, Lucia foi medindo quanto tempo, em minutos, gastava para se deslocar entre determinados locais e construiu o desenho abaixo.

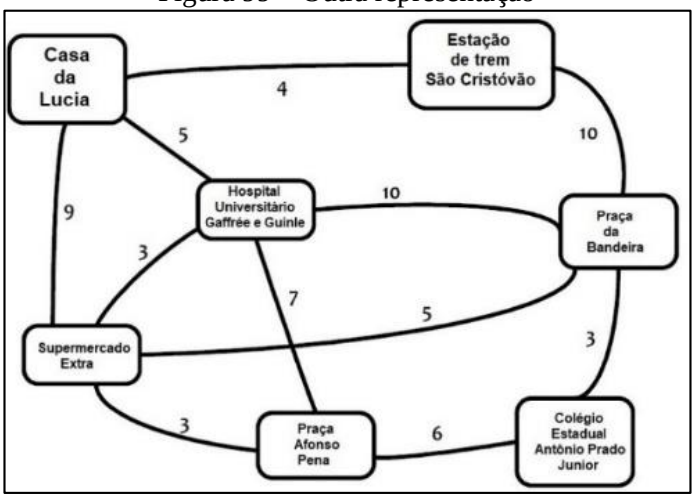
Figura 98 – Questão 21



Fonte: Borges, Muniz (2018)

Posteriormente, representou esse mapa através de um esquema mais simples:

Figura 99 – Outra representação



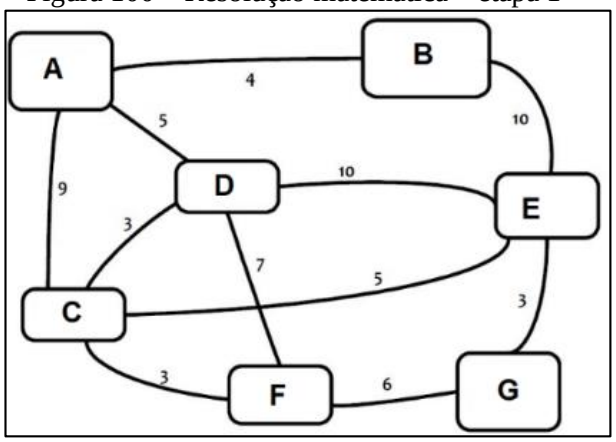
Fonte: Os autores, 2021

Qual o caminho mais rápido para Lucia sair da casa dela e chegar até o Colégio Estadual Antônio Prado Junior?

21.1 Resolução Matemática:

Vamos representar os locais como sendo pontos e enumerar possíveis caminhos:

Figura 100 – Resolução matemática – etapa 1



Fonte: Borges, Muniz (2018)

ABEG – 4 + 10 + 3 = 17

ABECDFG – 4 + 10 + 5 + 3 + 7 + 6 = 35

ABECFG – 4 + 10 + 5 + 3 + 6 = 28

ABEDFG – 4 + 10 + 10 + 7 + 6 = 37

ADEG – 5 + 10 + 3 = 18

ADECFG – 5 + 10 + 5 + 3 + 6 = 29

ADCFG - 5 + 3 + 3 + 6 = 17

ADCEG - 5 + 3 + 5 + 3 = 16

ACFG - 9 + 3 + 6 = 18

ACDEG - 9 + 3 + 10 + 3 = 25

ACEDFG - 9 + 5 + 10 + 7 + 6 = 37

ACDFG - 9 + 3 + 7 + 6 = 25

Aparentemente, realizar esse exercício pelo método da exaustão enumerando todos os caminhos possíveis a ser percorridos não é uma solução eficiente. Por isso, vamos usar uma estratégia mais eficiente.

Vamos sair de A e queremos chegar em G. Em A podemos ir para B, D ou C. Vamos marcar em cada um desses vértice o quanto andamos. Se formos para B, ali temos 4 min, no vértice D temos 5 min e no vértice C temos 9 min. Agora vamos analisar o vértice D, pois

podemos chegar nele através de A ou através de ACD. Qual deles tem o menor tempo? Ora, se formos por AD levamos 5 min, enquanto, por ACD levamos 12 min, então mantemos o percurso AD com 5 min. Agora vamos analisar C, podemos chegar pelo caminho AC e andamos 9 min ou pelo caminho ADC e andamos 8 min, então optamos por esse caminho e anotamos em C que levamos 8 min para chegar até ali. Para chegar em E podemos ir por ABE e andamos 14 min, ou vamos o ADE e levamos 15 min, ou vamos por ADCE e andamos 13 min e vamos optar por esse que é o menor caminho. Não precisamos testar o caminho ACE, pois já analisamos no passo anterior que o mais rápido de A até C já era o caminho ADC. Para chegarmos em F, podemos ir por ADCF e levamos 11 min ou ADF que levaria 12 min. Então o mais rápido é ADCF. Para chegarmos em G podemos ir por ADCEG e levaremos 16 min e por ADCFG levaremos 17 min. Então a maneira mais rápida de chegar na escola é fazendo o trajeto ADCEG em 16 min. Ao utilizarmos essa estratégia temos segurança de que esse é o menor caminho, pois esgotamos em cada vértice todas as possibilidades até ali de chegarmos e escolhemos sempre a menor. Isso é conhecido como algoritmo e é um recurso pertencente ao pensamento computacional.

21.2 Resolução Pensamento Computacional

21A - Decomposição

Queremos sair de A e chegar em G andando o mínimo possível. Vamos em cada vértice ver qual o menor caminho para chegar até ele e repetir esse procedimento até chegarmos em G.

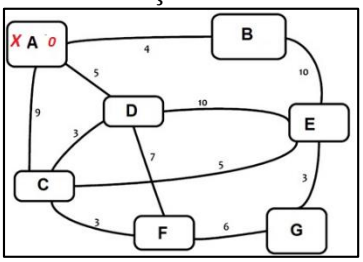
21B – Padrões

Esse algoritmo é guloso e a cada passo esgota toda as possibilidades anteriores. É um problema de otimização discreta que busca o menor caminho possível.

21C - Abstração

1- Marque com um X o vértice A. Coloque o valor 0 nele.
2-

Figura 101 – Resolução matemática – etapa 2



Fonte: Borges Muniz (2018)

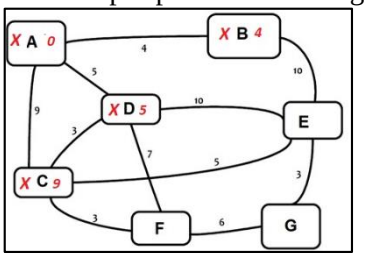
3- Quais são os vértices vizinhos diretos de A? Coloque o valor do tempo de A até cada um deles no respectivo vértice.

4- Escolha um dos vértices vizinhos ao A e marque um X nele.

5- Quais são os vértices vizinhos ao que você está e que não estão marcados com X?

6- Para cada vértice vizinho, coloque o valor do vértice onde você está somado ao caminho até ele.

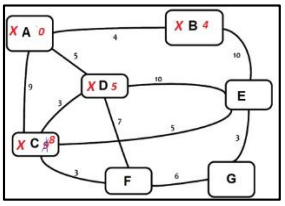
Figura 102 – Etapas posteriores do algoritmo



Fonte: Borges, Muniz (2018)

7- Se o vértice vizinho já tiver um valor atribuído a ele, veja qual o menor resultado e deixe este no vértice

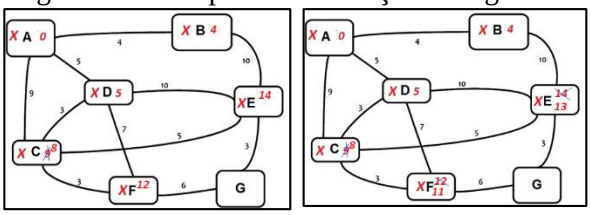
Figura 103 – Etapas do algoritmo



Fonte: Borges, Muniz (2018)

8- Repita os passos 4,5 e 6 até marcar todos os vértices.

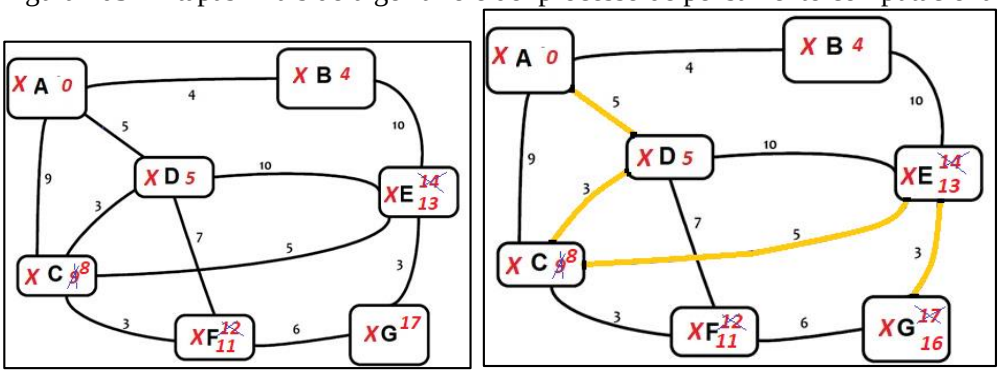
Figura 104 – Etapas de finalização do algoritmo



Fonte: Borges, Muniz (2018)

9- Qual foi o menor tempo encontrado? Marque o caminho que resulta nesse tempo.

Figura 105 – Etapas finais do algoritmo e do processo de pensamento computacional



Fonte: Os autores, 2021

21D - Algoritmo

Inicio ( )

Imprimir (Vamos partir de A)

B = calcula_distancia (A,B)

C = calcula_distancia (A,C)

D = calcula_distancia (A,D)

D = min(distancia(A,D), distancia(ACD))

Repetir calcula_distancia ( ) até chegar em G

Imprimir (“A menor distância é”, x)

Fim

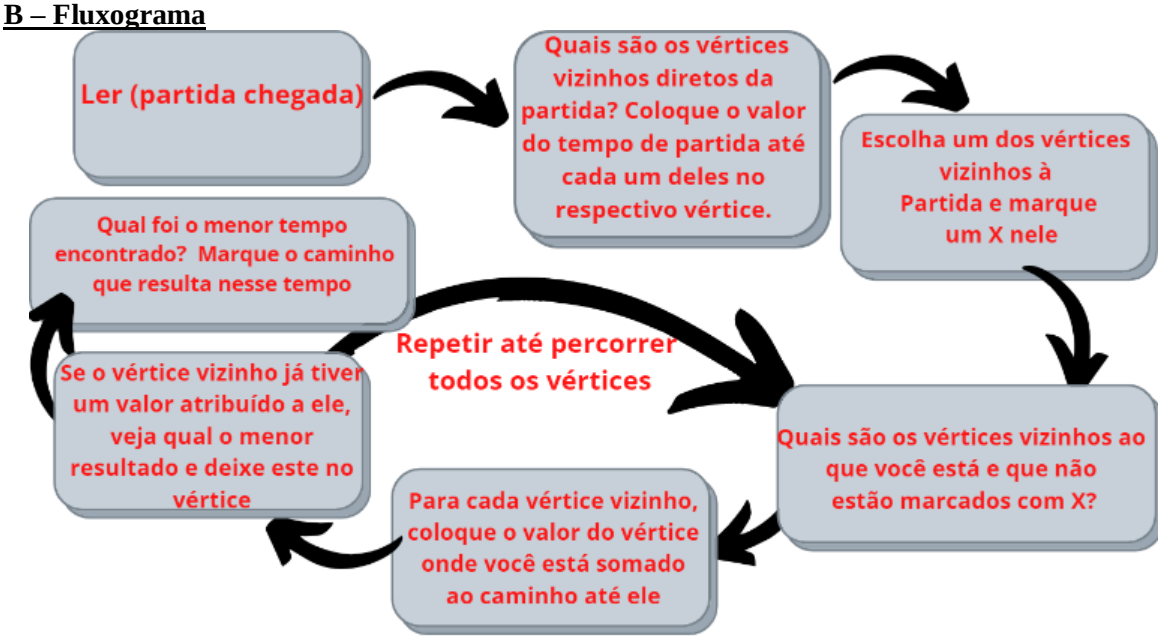
21.3 Comparando Soluções

Tabela 29 - Comparando Soluções – Questão 21

A – Resolução Matemática

Pelo método da exaustão, o menor caminho é 16.
ABEG – 4 + 10 + 3 = 17
ABECDFG – 4 + 10 + 5 + 3 + 7 + 6 = 35
ABECFG – 4 + 10 + 5 + 3 + 6 = 28
ABEDFG – 4 + 10 + 10 + 7 + 6 = 37
ADEG – 5 + 10 + 3 = 18
ADECFG – 5 + 10 + 5 + 3 + 6 = 29
ADCFG - 5 + 3 + 3 + 6 = 17
ADCEG - 5 + 3 + 5 + 3 = 16
ACFG - 9 + 3 + 6 = 18
ACDEG - 9 + 3 + 10 + 3 = 25
ACEDFG - 9 + 5 + 10 + 7 + 6 = 37
ACDFG - 9 + 3 + 7 + 6 = 25

B – Fluxograma



C – Abstração

```
Inicio ( )
        Imprimir (Vamos partir de A)
        funcao calcula_distanca ( )
        B = calcula_distancia (A,B)
        C = calcula_distancia (A,C)
        D = calcula_distancia (A,D)
        D = min(distancia(A,D), distancia(ACD))
        Repetir calcula_distancia ( ) até chegar em G
        Imprimir ("A menor distancia é",  x)
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 106 - Implementação em linguagem C++ da questão 21

```cpp
#include <iostream>
#include <climits>
#include <queue>

using namespace std;

vector<int> distancias;
vector<bool> visitados;
int grafo[100][100];
int n, m;

int VerticeMinimo(vector<int> distancias, vector<bool> visitados){
    int min_distancia = INT_MAX;
    int min_posicao;

    for(int i = 1; i <= n; ++i){
        if (!visitados[i] && distancias[i] <= min_distancia){
            min_distancia = distancias[i];
            min_posicao = i;
        }
    }

    cout << "O menor vertice escolhido desta vez foi o vertice " << min_posicao << " que tem distancia " << min_distancia << " do vertice

    return min_posicao;
}


int Dijsktra(int origem, int destino){
    distancias[origem] = 0;

    for(int i = 0; i < n; ++i){
        int v = VerticeMinimo(distancias, visitados);
        // Para cada etapa do algoritmo, se visita sempre o vertice com menor
        // distância que ainda não foi visitado, fazemos isso n vezes para certificar
        // que todos os vértices serão visitados alguma hora

        visitados[v] = true;

        for(int j = 1; j <= n; ++j){
            int peso = grafo[v][j];

            if (peso != -1){
                // Só atualiza a resposta se o vértice a ser atualizado não foi visitado ainda,
                // se a distância do vértice atual não for infinita e se a distância a ser atualizada
                // é menor do que a cadastrada para o vizinho
                if (!visitados[j] && distancias[v] != INT_MAX && distancias[v] + peso < distancias[j]){
                    cout << "A distancia do vertice " << j << " para o vertice 1 era " << distancias[j];
                    distancias[j] = distancias[v] + peso;
                    cout << ", mas agora ela foi atualizada para " << distancias[v] << " + " << peso << " = " << distancias[j] << "." << e
                }
            }
        }
    }

    return distancias[destino];
}
// Esse é o algoritmo de Dijsktra, que visita todos os vértices atualizando a
// distância entre ele e seus vizinhos a um vértice específico (vértice de origem).
// O algoritmo calcula a distância mínima do vértice de origem a todos os demais,
// mas nesse caso estamos interessados em só um vértice específico, o de destino.

int main()
{
    int x, y, p;

    cout << "Quantos lugares Lucia pode visitar?" << endl;
    cin >> n;
    n += 1;
    for(int i = 1; i <= n; ++i){
        for(int j = 1; j <= n; ++j){
            grafo[i][j] = -1;
        }
    }

    for(int i = 0; i <= n; ++i){
        distancias.push_back(INT_MAX);
        visitados.push_back(false);
    }


    cout << "Quantas ruas tem na cidade entre esses lugares?" << endl;
    cin >> m;

    cout << "Descreva as ruas da cidade. Exemplo: 1 2 5 conecta os lugares a casa da Lucia com o lugar 2 com distancia 5 (a casa da Lucia e
    for(int i = 0; i < m; ++i){
        cin >> x >> y >> p;
        grafo[x][y] = p;
        grafo[y][x] = p;
    }

    cout << "Qual lugar Lucia quer visitar?" << endl;
    cin >> x;
    while (x == 1){
        cout << "O lugar 1 ja e a casa de Lucia. Escolha outro lugar." << endl;
        cin >> x;
    }

    int resposta = Dijsktra(1, x);

    cout << "A distancia minima da casa da Lucia ate o lugar desejado e " << resposta << "." << endl;

    return 0;
}
```

Fonte: Os autores, 2021

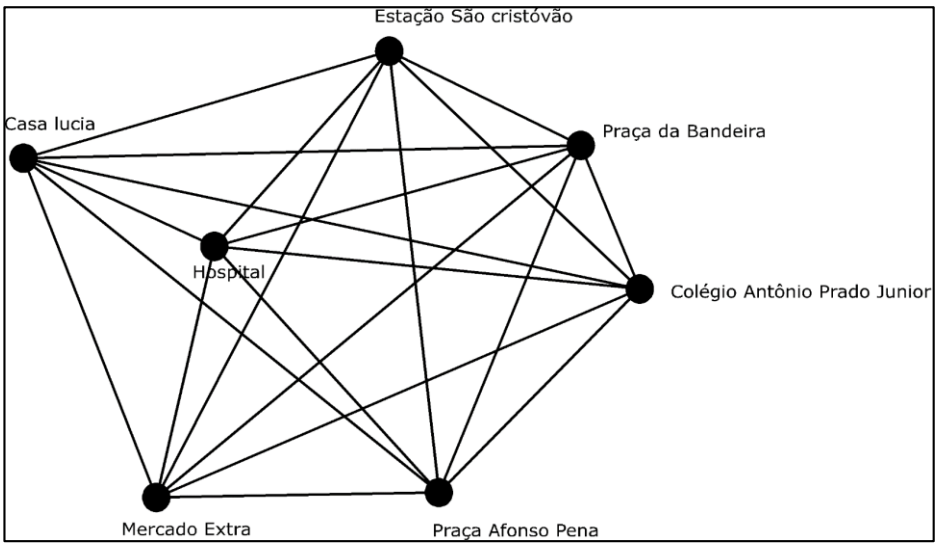
21.4 Considerações didáticas

A Resolução matemática se deu por enumeração e posteriormente através da estratégia do uso do algoritmo de Dijkstra. Já no fluxograma ficou claro os passos realizados nesse algoritmo. Na abstração destacamos uma implementação menos detalhada e na implementação da linguagem C++ em si trabalhos com a função auxiliando a busca pelo menor caminho. Objetivos – trabalhar a relação entre grafos, problemas de otimização discreta e o ganho que temos ao resolvê-lo através do pensamento computacional.

Na questão 21 observamos que existe os problemas de otimização com a natureza enumerativa, de existência e classificação. A questão é resolvida inicialmente através de um processo enumerativo no qual buscamos o menor caminho listando aqueles possíveis. Porém temos o grande potencial dessa questão de ser resolvido através de um procedimento de algoritmo que no caso seria o Djisktra e sua implementação se deu através de um algoritmo justamente o que foi indicado para a solução do problema em sala de aula.

Questão 22 - (Borges & Muniz, 2018) Lucia é estudante do Colégio Estadual Antônio Prado Junior e pensou em alguns pontos próximos à escola que fazem parte de sua rotina: estação de metrô de São Cristóvão, Praça da Bandeira, Praça Afonso Pena, Hospital Gaffreé Guinle, sua casa na Rua Moraes Silva, mercado Extra e o colégio. Ela observou que poderia fazer diversos caminhos que passassem por todos esses lugares. A figura abaixo representa os possíveis caminhos que ela poderia fazer:

Figura 107 – Questão 22



Fonte: Borges, Muniz (2018)

Lucia também pesquisou no Google Maps qual era a distância a pé, em quilômetros, entre cada um desses lugares:

Tabela 30 – Distâncias entre os locais – Questão 22

	Prado Junior	São Cristóvão	Extra	Casa da Lucia	Praça Afonso Pena	Hospital	Praça da Bandeira
Prado Junior	0	0,8	0,9	1,1	0,85	0,65	0,6
São Cristóvão	0,8	0	1,4	0,9	1,5	1,2	1,3
Extra	0,9	1,4	0	0,55	0,45	0,35	0,55
Casa da Lucia	1,1	0,9	0,55	0	0,8	0,4	1,6
Praça Afonso Pena	0,85	1,5	0,45	0,8	0	0,6	1,3
Hospital	0,65	1,2	0,35	0,4	0,6	0	1,6
Praça da Bandeira	0,6	1,3	0,55	1,6	1,3	1,6	0

Fonte: Borges, Muniz (2018)

Qual o menor caminho que ela pode fazer da escola passando por todos os lugares e sem repetir nenhum e retornar à escola?

22.1 Resolução Matemática:

Essa questão é um grande exemplo de como o pensamento computacional pode auxiliar-nos em problemas de combinatória. Se tentarmos fazer a questão por enumeração, teremos 6! = 720 caminhos distintos para percorrer e pelo método da exaustão, ou seja, escrever todos eles muito provavelmente errarão. Então vamos resolvê-la através do algoritmo do caixeiro viajante.

22.2 Resolução Pensamento Computacional

22A - Decomposição

Partiremos da escola sempre iremos para o local mais próximo sem repetir por onde já passamos. Se o lugar mais próximo for algum pelo qual já passamos, então pegaremos o segundo mais próximo e se necessário repetimos esse processo até que sigamos em frente.

A cada passo dado, vamos somando os caminhos pelos quais já passamos.

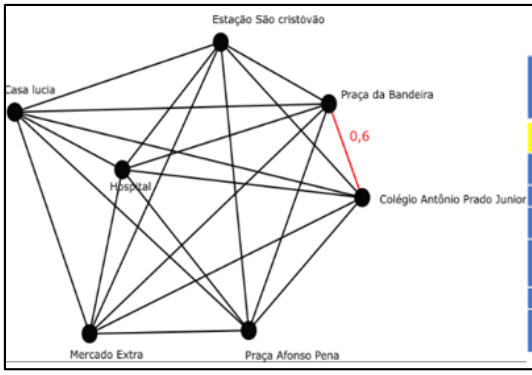
22B - Padrões

A matriz é simétrica, então podemos ver a distância entre os locais, tanto pela linha, quanto pela coluna. É um problema de otimização discreta, no qual sempre vamos para o mais próximo.

22C – Abstração

O local mais próximo da escola é a Praça da Bandeira:

Figura 106 – Questão 22 – etapa 1

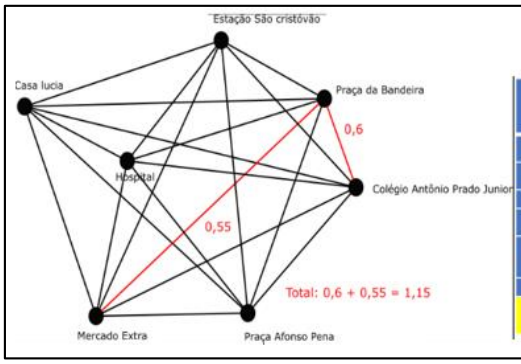


	Prado Junior	São Cristóvão	Extra	Casa da Lucia	Praça Afonso Pena	Hospital	Praça da Bandeira
Prado Junior	0	0,8	0,9	1,1	0,85	0,65	0,6
São Cristóvão	0,8	0	1,4	0,9	1,5	1,2	1,3
Extra	0,9	1,4	0	0,55	0,45	0,35	0,55
Casa da Lucia	1,1	0,9	0,55	0	0,8	0,4	1,6
Praça Afonso Pena	0,85	1,5	0,45	0,8	0	0,6	1,3
Hospital	0,65	1,2	0,35	0,4	0,6	0	1,6
Praça da Bandeira	0,6	1,3	0,55	1,6	1,3	1,6	0

Fonte: Borges, Muniz (2018)

A partir da Praça da Bandeira, o local mais próximo é o Mercado Extra:

Figura 107 – Questão 22 – etapa 2

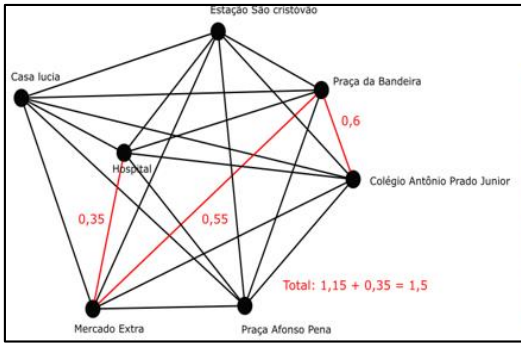


	Prado Junior	São Cristóvão	Extra	Casa da Lucia	Praça Afonso Pena	Hospital	Praça da Bandeira
Prado Junior	0	0,8	0,9	1,1	0,85	0,65	0,6
São Cristóvão	0,8	0	1,4	0,9	1,5	1,2	1,3
Extra	0,9	1,4	0	0,55	0,45	0,35	0,55
Casa da Lucia	1,1	0,9	0,55	0	0,8	0,4	1,6
Praça Afonso Pena	0,85	1,5	0,45	0,8	0	0,6	1,3
Hospital	0,65	1,2	0,35	0,4	0,6	0	1,6
Praça da Bandeira	0,6	1,3	0,55	1,6	1,3	1,6	0

Fonte: Borges, Muniz (2018)

A partir do Mercado Extra o local mais próximo ainda não visitado é o Hospital:

Figura 108 – Questão 22 – etapa 3

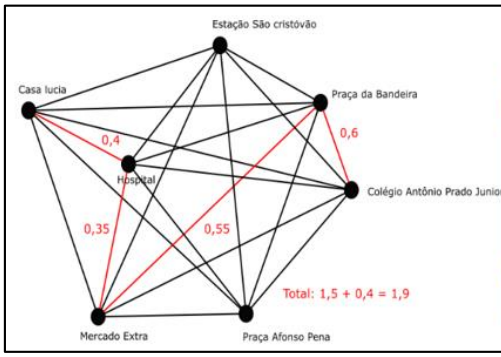


	Prado Junior	São Cristóvão	Extra	Casa da Lucia	Praça Afonso Pena	Hospital	Praça da Bandeira
Prado Junior	0	0,8	0,9	1,1	0,85	0,65	0,6
São Cristóvão	0,8	0	1,4	0,9	1,5	1,2	1,3
Extra	0,9	1,4	0	0,55	0,45	0,35	0,55
Casa da Lucia	1,1	0,9	0,55	0	0,8	0,4	1,6
Praça Afonso Pena	0,85	1,5	0,45	0,8	0	0,6	1,3
Hospital	0,65	1,2	0,35	0,4	0,6	0	1,6
Praça da Bandeira	0,6	1,3	0,55	1,6	1,3	1,6	0

Fonte: Borges, Muniz (2018)

A partir do Hospital o local mais próximo que ainda não passamos é a Casa da Lucia:

Figura 109 – Questão 22 – etapa 4

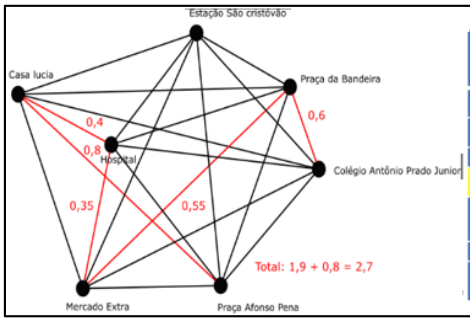


	Prado Junior	São Cristóvão	Extra	Casa da Lucia	Praça Afonso Pena	Hospital	Praça da Bandeira
Prado Junior	0	0,8	0,9	1,1	0,85	0,65	0,6
São Cristóvão	0,8	0	1,4	0,9	1,5	1,2	1,3
Extra	0,9	1,4	0	0,55	0,45	0,35	0,55
Casa da Lucia	1,1	0,9	0,55	0	0,8	0,4	1,6
Praça Afonso Pena	0,85	1,5	0,45	0,8	0	0,6	1,3
Hospital	0,65	1,2	0,35	0,4	0,6	0	1,6
Praça da Bandeira	0,6	1,3	0,55	1,6	1,3	1,6	0

Fonte: Borges, Muniz (2018)

Da Casa da Lucia o local mais próximo é a Praça Afonso Pena:

Figura 110 – Questão 22 – etapa 5

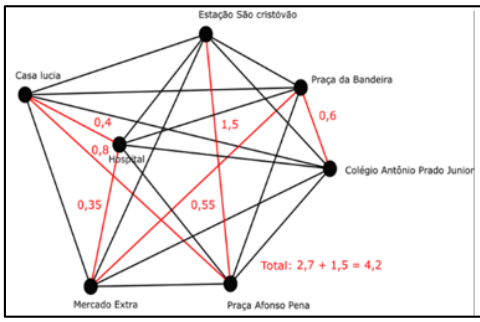


	Prado Junior	São Cristóvão	Extra	Casa da Lucia	Praça Afonso Pena	Hospital	Praça da Bandeira
Prado Junior	0	0,8	0,9	1,1	0,85	0,65	0,6
São Cristóvão	0,8	0	1,4	0,9	1,5	1,2	1,3
Extra	0,9	1,4	0	0,55	0,45	0,35	0,55
Casa da Lucia	1,1	0,9	0,55	0	0,8	0,4	1,6
Praça Afonso Pena	0,85	1,5	0,45	0,8	0	0,6	1,3
Hospital	0,65	1,2	0,35	0,4	0,6	0	1,6
Praça da Bandeira	0,6	1,3	0,55	1,6	1,3	1,6	0

Fonte: Borges, Muniz (2018)

Da Praça Afonso Pena o local mais próximo para irmos é a Estação de São Cristóvão:

Figura 111 – Questão 22 – etapa 6

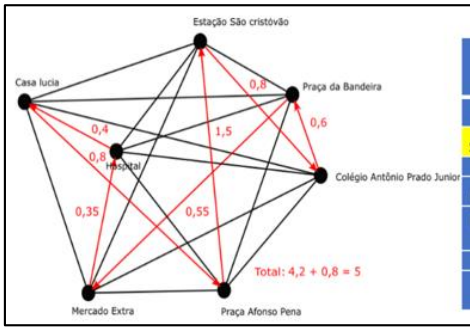


	Prado Junior	São Cristóvão	Extra	Casa da Lucia	Praça Afonso Pena	Hospital	Praça da Bandeira
Prado Junior	0	0,8	0,9	1,1	0,85	0,65	0,6
São Cristóvão	0,8	0	1,4	0,9	1,5	1,2	1,3
Extra	0,9	1,4	0	0,55	0,45	0,35	0,55
Casa da Lucia	1,1	0,9	0,55	0	0,8	0,4	1,6
Praça Afonso Pena	0,85	1,5	0,45	0,8	0	0,6	1,3
Hospital	0,65	1,2	0,35	0,4	0,6	0	1,6
Praça da Bandeira	0,6	1,3	0,55	1,6	1,3	1,6	0

Fonte: Borges, Muniz (2018)

Da Estação de São Cristóvão vamos para escola novamente e andamos no total 5km:

Figura 112 – Questão 22 – etapa 7



	Prado Junior	São Cristóvão	Extra	Casa da Lucia	Praça Afonso Pena	Hospital	Praça da Bandeira
Prado Junior	0	0,8	0,9	1,1	0,85	0,65	0,6
São Cristóvão	0,8	0	1,4	0,9	1,5	1,2	1,3
Extra	0,9	1,4	0	0,55	0,45	0,35	0,55
Casa da Lucia	1,1	0,9	0,55	0	0,8	0,4	1,6
Praça Afonso Pena	0,85	1,5	0,45	0,8	0	0,6	1,3
Hospital	0,65	1,2	0,35	0,4	0,6	0	1,6
Praça da Bandeira	0,6	1,3	0,55	1,6	1,3	1,6	0

Fonte: Borges, Muniz (2018)

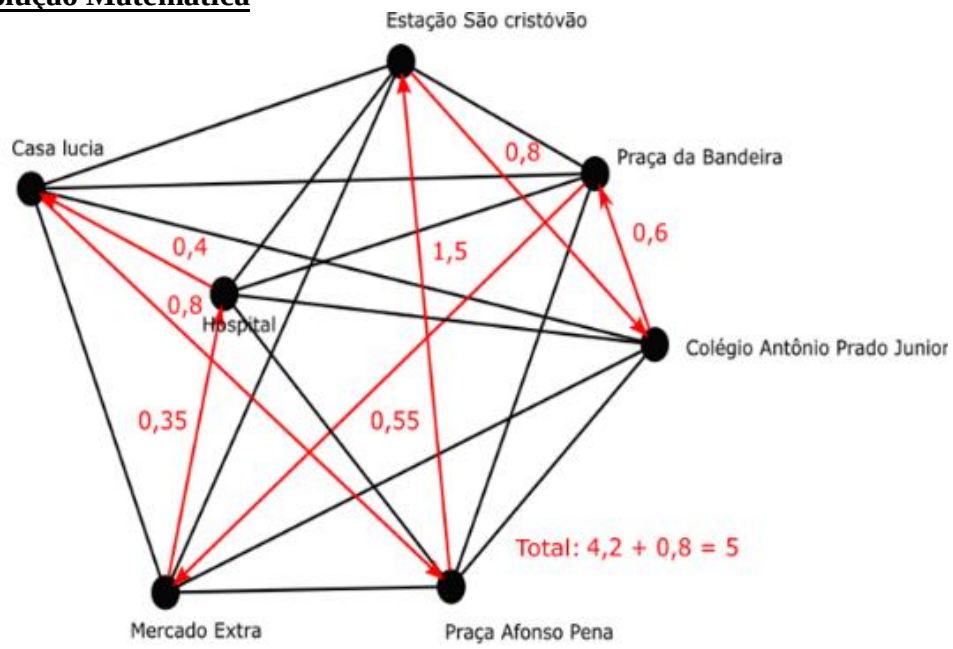
22D - Algoritmo

Inicio ( )
Leia (matriz A_{8x8})
Escolher a menor_distancia do colégio aos outros lugares
Soma = Soma + menor_distacia( )
Repetir o processo menor_distancia( ) e soma = soma + menor_distancia( ) até passar por todos os vértices e voltar ao de origem
Imprimir (“O menor caminho a percorrer é”, soma)
Fim

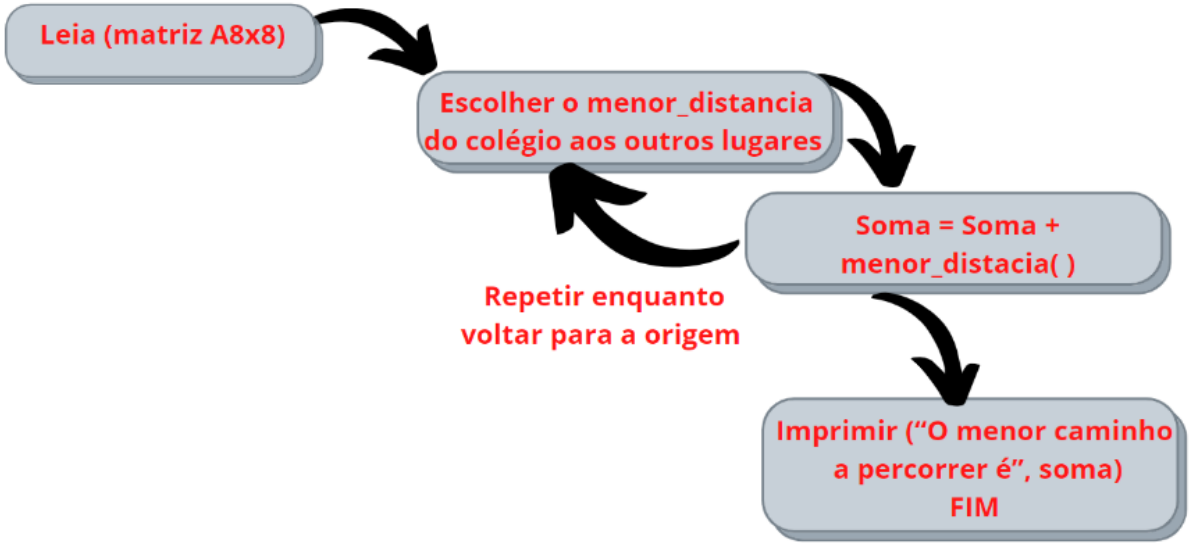
22.3 Comparando Soluções

Tabela 31 – Distâncias entre os locais – Questão 22

A – Resolução Matemática



B – Fluxograma



C – Abstração

```
Inicio ( )
        Leia (matriz A8x8)
        Escolher o menor_distancia do colégio aos outros lugares
        Soma = Soma + menor_distacia( )
        Repetir o processo menor_distancia( ) e soma = soma + menor_distancia( ) até passar por todos os vértices e voltar ao de origem
        Imprimir ("O menor caminho a percorrer é", soma)
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 113 - Implementação em linguagem C++ da questão 22

```
#include <iostream>
#include <vector>

using namespace std;

double mapa[100][100];
vector<vector<int> > caminhos;
double min_resposta = -1;
int n, m;

void Solucao(vector<int> caminho, vector<int> visitados, double peso_total, int origem, int destino){
    caminho.push_back(origem);

    if (caminho.size() == n){
        double peso = mapa[caminho[n - 1]][destino];
        if (peso != -1){
            caminho.push_back(destino);
            peso_total += peso;
            if (min_resposta == -1 || peso_total <= min_resposta){
                if (peso_total < min_resposta){
                    caminhos.clear();
                }
                min_resposta = peso_total;
                caminhos.push_back(caminho);
            }
        }
    } else {
        visitados[origem] = true;
        for(int i = 1; i <= n; ++i){
            double peso = mapa[origem][i];

            if (peso != -1 && !visitados[i]){
                Solucao(caminho, visitados, peso_total + peso, i, destino);
            }
        }
    }
}

int main()
{
    vector<int> visitados;
    int x, y;
    double p;

    cout << "Quantos vertices tem no grafo?" << endl;
    cin >> n;
    for(int i = 0; i <= n; ++i){
        for(int j = 0; j <= n; ++j){
            mapa[i][j] = -1;
        }
        visitados.push_back(false);
    }

    cout << "Quantas arestas tem no grafo?" << endl;
    cin >> m;

    cout << "Descreva as arestas do grafo com seus respectivos pesos." << endl;
    for(int i = 0; i < m; ++i){
        cin >> x >> y >> p;
        mapa[x][y] = p;
        mapa[y][x] = p;
    }

    cout << "Qual e o vertice de entrada/saida?" << endl;
    cin >> x;

    Solucao(vector<int>(), visitados, 0, x, x);

    cout << "Os menores caminhos (de tamanho " << min_resposta << ") sao:" << endl;
    for(int i = 0; i < caminhos.size(); ++i){
        cout << caminhos[i][0];
        for(int j = 1; j < caminhos[i].size(); ++j){
            cout << " -> " << caminhos[i][j];
        }
        cout << endl;
    }

    return 0;
}
```

Fonte: Os autores, 2021

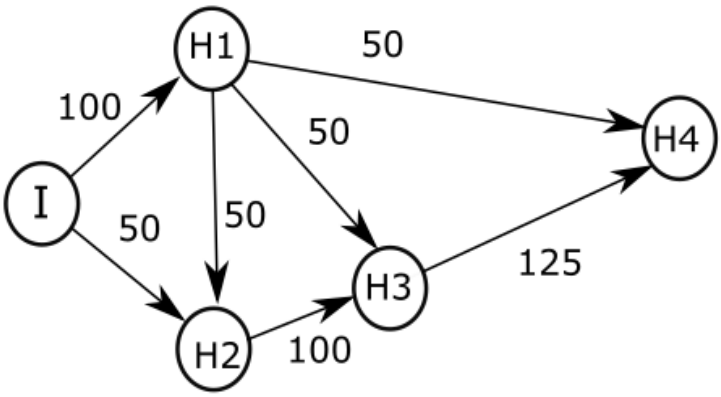
22.4 Considerações didáticas

A questão 22 também aborda um problema de otimização no qual é necessário passar por todos os locais, ou seja os vértices, e voltar ao local de origem e a técnica apresentada nesse problema é a ideia do caixeiro-viajante. Caso não utilizemos o algoritmo teríamos de trabalhar com um processo numerativo mas que nesse caso não seria eficiente, uma vez que estamos lidando com um grafo de grau 6 em todos os seus vértices o que aumenta muito a quantidade de configurações distintas que podemos trilhar nesse percurso Vale ressaltar que trabalhamos com uma tabela que pode ser implementada computacionalmente através de uma matriz de adjacência e tal matriz é simétrica, a fim de facilitar os cálculos. Mas se formos simular tempo real em aplicativo de busca veremos que as rotas e os valores das rotas variam uma vez que o aplicativo muda a cada instante a rota e o tempo gasto de acordo com os outros fatores que influenciam tal situação. Realizamos a simulação através da análise da Imagem e da tabela de como seria a busca pelo menor caminho ou o maior caminho através do algoritmo do caixeiro-viajante e é interessante essa análise, uma vez que essa trajetória é realizada pelo computador, de maneira muito mais rápida e eficiente desde que o programa tenha sido implementado corretamente. Isso é um ponto interessante levantar: de nada adianta ter uma estrutura computacional se a lógica do algoritmo ou se alguma característica do algoritmo não estiver correta, pois não encontraremos a solução ótima. A complexidade desse algoritmo é np-completo, ou seja, ele é um problema que não é rápido de se resolver tampouco no processo enumerativo como no processo computacional.

A solução matemática utilizou o algoritmo do caixeiro viajante como estratégia de solução. Já no fluxograma colocamos a representação do algoritmo do caixeiro viajante. E na abstração também seguimos esses passos. Já na implementação utilizamos estruturas de ponteiro, vetores e grafos, que nos auxiliam na solução, mas que não são aplicáveis na solução matemática em si.

Questão 23 - (Borges & Muniz) Há um incêndio na Rua Conde de Bonfim. Devido à crise, o carro do corpo de bombeiros está sem seu abastecimento normal de água. Felizmente há um hidrômetro no local que se interliga com outros pelo bairro da Tijuca. Cada hidrômetro é interligado através de canos com diferentes capacidades de fluxo de água por segundo. O esquema abaixo representa essa rede de hidrômetros interligados. Um bombeiro está no marco I onde a água é fornecida para esta rede e precisa saber a quantidade máxima de água que ele pode fornecer para que o incêndio seja apagado o mais rápido possível.

Figura 114 – Questão 23



Fonte: Borges, Muniz (2018)

23.1 Resolução Matemática:

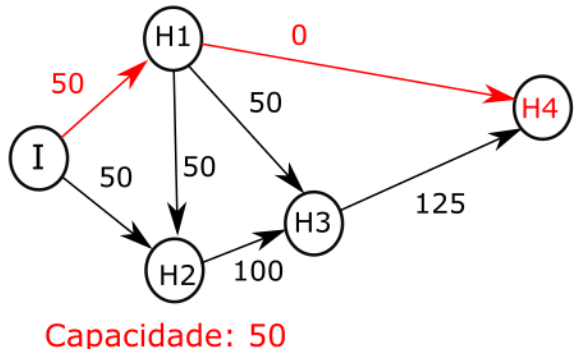
O problema é de otimização e visa pegar o máximo de capacidade possível até o hidrante H_4. O esquema representado se trata de um grafo direcionado com peso nas arestas. Vamos resolver esse problema através de enumeração e usando a ideia do Algoritmo de Ford-Fulkerson.

23.2 Resolução Pensamento Computacional

23A - Decomposição:

Vamos fazer o caminho I → H_1 → H_4 = máximo(100, 50) = 50. Ou seja, a capacidade máxima que passa por esse sistema é e 50 unidades de volume. Assim, ainda sobra 50 u.v. pelo I → H_1 e esgota o escoamento em H_1→ H_4 conforme ilustra o esquema a seguir:

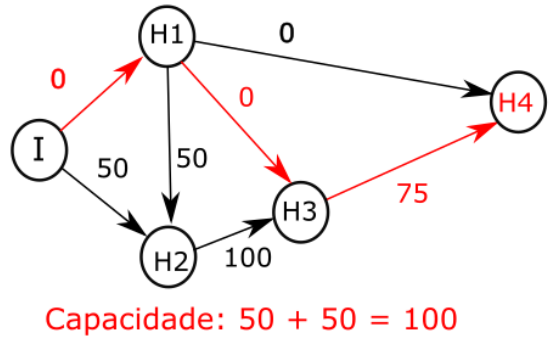
Figura 115 – Resolução A



Fonte: Os autores, 2021

Agora, faremos o trajeto I → H_1 → H_3 → H_4 = máximo(50, 50 , 125) = 50. Ou seja, conseguimos mandar mais 50 u.v. por esse fluxo de escoamento:

Figura 116 – Resolução B

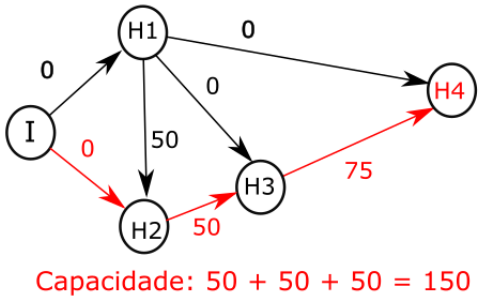


Fonte: Os autores, 2021

O caminho I → H_1 → H_2 → H_3 → H_4 não pode ser usado, pois toda a capacidade do cano I → H_1 foi usada. Assim o cano H_1 → H_2 ficou sem utilidade.

Mas ainda podemos fazer o trajeto I → H_2 → H_3 → H_4 = máximo (50, 100, 75) = 50

Figura 117 – Resolução C



Fonte: Os autores, 2021

Logo, conseguimos fazer chegar 150 unidades de volume de água até o hidrante H_4.

23B - Padrões:

O esquema se trata de um grafo direcionado com peso nas arestas.

Sempre buscamos o melhor caminho e encontramos o valor máximo entre as arestas possível de passa pelo percurso observado.

Tabela 32 – Resolução Computacional – Questão 23A

	I	H1	H2	H3	H4
I	0	100	50	0	0
H1	0	0	50	50	50
H2	0	0	0	100	0
H3	0	0	0	0	125
H4	0	0	0	0	0

Fonte: Os autores, 2021.

23C - Abstração

Vamos fazer o processo através da matriz que realizamos nas imagens para entendermos como o computador funciona na resolução desse problema.

Tabela 33 – Resolução Computacional – Questão 23B

	I	H1	H2	H3	H4
I	0	100	50	0	0
H1	0	0	50	50	50
H2	0	0	0	100	0
H3	0	0	0	0	125
H4	0	0	0	0	0

Fonte: Os autores, 2021

Devemos iniciar em I. O primeiro vértice que aparece é o H_1. Vamos para ele e dele vamos para o H_2. Vamos continuar esse percurso. De H_2 decidimos ir para H_3. E de H_3 vamos para H_4. Nesse trajeto I → H_1 → H2 → H_3 → H_4 vamos pegar o min(100, 50, 100, 125) = 50. Então conseguimos enviar 50 unidades de volume e nossa nova matriz será:

Tabela 34 – Resolução Computacional – Questão 23C

	I	H1	H2	H3	H4
I	0	50	50	0	0
H1	0	0	0	50	50
H2	0	0	0	50	0
H3	0	0	0	0	75
H4	0	0	0	0	0

Fonte: Os autores, 2021

A partir de I podemos ir H_1 novamente e de H_1 para H_4. A capacidade é min (50, 50) = 50. Então conseguimos passar 50 unidades de volume pelo percurso I → H_1 → H_4. Vamos retirar isso da matriz, uma vez que já usamos esse recurso. E até aqui conseguimos enviar 100 u.v. de água para o hidrante H_4.

Tabela 35 – Resolução Computacional – Questão 23D

	I	H1	H2	H3	H4
I	0	0	50	0	0
H1	0	0	0	50	0
H2	0	0	0	50	0
H3	0	0	0	0	75
H4	0	0	0	0	0

Fonte: Os autores, 2021

Agora a partir de I podemos ir para H_2, a partir desse ir para H_3, de H_3 para H_4 e escolher a melhor opção no min(50, 50, 75) = 50 pelo caminho I → H_2 → H_3 → H_4. Logo conseguimos passar a capacidade de 100 + 50 = 150 unidades de volume. E assim atualizando a matriz temos o seguinte esquema:

Tabela 36 – Resolução Computacional – Questão 23E

	I	H1	H2	H3	H4
I	0	0	0	0	0
H1	0	0	0	50	0
H2	0	0	0	0	0
H3	0	0	0	0	25
H4	0	0	0	0	0

Fonte: Os autores, 2021

A partir daí já não podemos continuar o caminho, pois esgotamos todas as saídas possíveis I para qualquer hidrante. Mas podemos reparar que ainda sobrou espaço nos canos H_3 e H_4.

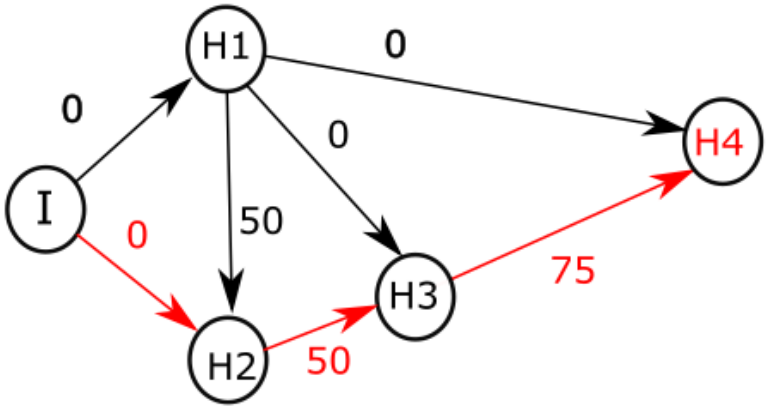
23D – Algoritmo

inicio ( )
leia (grafo, partida, chegada)
analisa os caminhos possíveis de partida -> chegada
//escolhe o menor valor das arestas que estão no caminho:
1 - No caminho escolher min(v1, v2, ..., vn) = aux
2 - soma = aux + soma
3 - v1 = v1 - aux,
4 - v2 = v2 - aux
...
5 - vn = vn - aux
repete o processo dos passos 1 a 5 enquanto houver caminho possível a ser percorrido
imprimir (soma)
fim ( )

23.3 Comparando Soluções

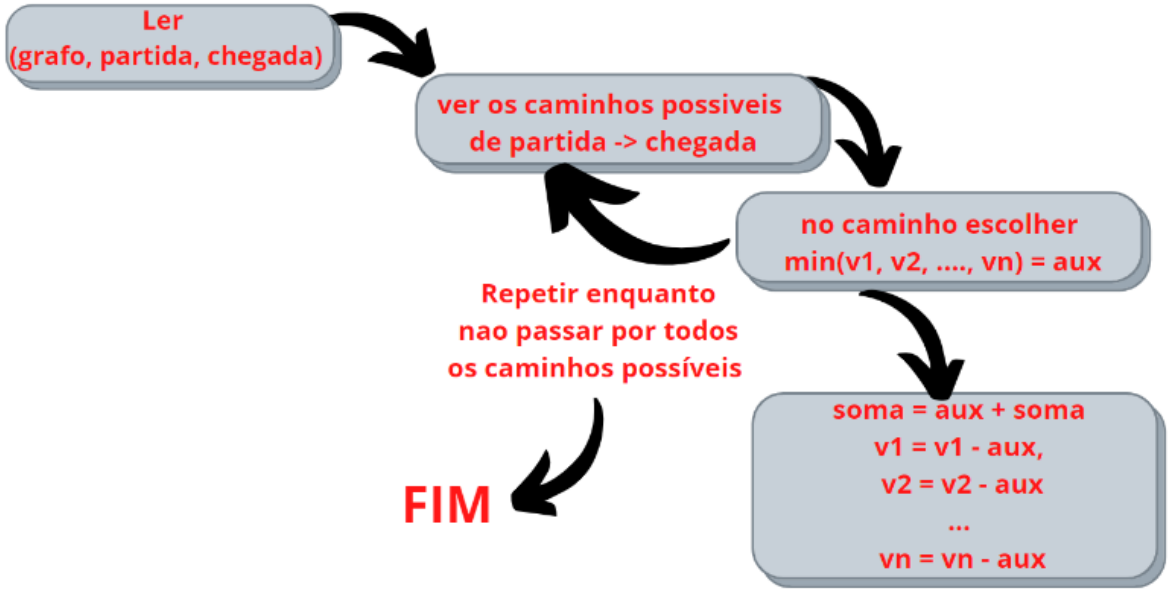
Tabela 37 - Comparando Soluções – Questão 23

A – Resolução Matemática



Capacidade: 50 + 50 + 50 = 150

B – Fluxograma



C – Abstração

```
inicio ( )
    leia (grafo, partida, chegada)
    analisa os caminhos possiveis de partida -> chegada
    //escolhe o menor valor das arestas que estao no caminho:
    1 - no caminho escolher min(v1, v2, ...., vn) = aux
    2 - soma = aux + soma
    3 - v1 = v1 - aux,
    4 - v2 = v2 - aux
    ...
    5 - vn = vn - aux
    repete o processo dos passos 1 a 5 enquanto houver caminho possivel a ser percorrido
    imprimir (soma)
fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 118 – mplementação em linguagem C++ da questão 23

```cpp
#include <iostream>
#include <climits>
#include <cmath>
#include <queue>

using namespace std;

vector<int> pai;
int mapa[100][100];
int n, m;

bool BuscaEmLargura(int fonte, int sumidouro){
    vector<bool> visitado;
    queue<int> fila;

    visitado.assign(n + 1, false);

    fila.push(fonte);
    visitado[fonte] = true;

    while (!fila.empty()){
        int u = fila.front();
        fila.pop();

        for(int i = 0; i <= n; ++i){
            if (!visitado[i] && mapa[u][i] > 0){
                fila.push(i);
                visitado[i] = true;
                pai[i] = u;
            }
        }
    }

    return visitado[sumidouro];
}
// A Busca em Largura em grafos busca cada vértice colocando em uma fila (representa como queue aqui).
// Para cada vértice da busca, ele coloca seus vizinhos numa fila, o que dá um aspecto de busca em níveis característico da busca

int FordFulkerson(int fonte, int sumidouro){
    int fluxo_maximo = 0;

    pai.assign(n + 1, -1);

    while (BuscaEmLargura(fonte, sumidouro)){
    // Enquanto for possível chegar ao sumidouro pela fonte
        int fluxo = INT_MAX;

        int s = sumidouro;
        while (s != fonte){
            fluxo = min(fluxo, mapa[pai[s]][s]);
            s = pai[s];
        }
        // Ver o fluxo pelo mínimo que se pode passar pelo caminho do sumidouro à fonte.

    return fluxo_maximo;
}
// O algoritmo de Ford-Fulkerson conta com dois vértices especiais: a fonte e o sumidouro.
// A fonte precisa ser um vértice que só possui vértices de saída e o sumidouro outro que só possui vértices de entrada.
// A partir daí, enquanto se puder ir da fonte ao sumidouro, testa-se fluxos e atualiza o grafo de acordo.

int main()
{
    int x, y, p;

    cout << "Quantos hidrantes tem?" << endl;
    cin >> n;

    for(int i = 0; i <= n; ++i){
        for(int j = 0; j <= n; ++j){
            mapa[i][j] = 0;
        }
    }

    cout << "Quantas ligacoes no total (entre hidrantes e entre um hidrante e o bombeiro)?" << endl;
    cin >> m;

    cout << "Descreva as ligacoes entre hidrantes e entre hidrantes e o bombeiro. Exemplo: 1 6 5 significa que o hidrante 1 pode p
    for(int i = 0; i < m; ++i){
        cin >> x >> y >> p;
        mapa[x][y] = p;
    }


    cout << "Qual dos vertices eh o hidrante final?" << endl;
    cin >> x;

    cout << "O maximo que pode se passar de agua do bombeiro ate o hidrante " << x << " e " << FordFulkerson(0, x) << "." << endl;

    return 0;
}
```

Fonte: Os autores, 2021

23.4 Considerações didáticas

A questão 23 trabalha com um tipo de grafo que chamamos de grafo direcionado, ou seja, o caminho percorrido só pode ser em uma direção que é apontada na seta. Esse é a primeira questão que aparece nesse trabalho com esse tipo de estrutura de grafo e ela visa captar o fluxo máximo possível de água na rede analisada. Ela é otimização, mas também pode ser analisada por sua natureza enumerativa, de existência e de classificação. É enumerativa no sentido de listarmos os caminhos possíveis partindo do início e chegando no hidrante final. É de existência se nos questionarmos se existe um caminho e é de classificação no sentido de escolhemos caminhos e posteriormente retirarmos o valor já usado da capacidade restante.

A noção de enumeração pode ser trabalhada através do auxílio de um grafo, ou seja, através de uma matriz de adjacência na qual o valor de cada elemento representa o peso ou a capacidade de fluxo entre quaisquer dois vértices desse sistema. O algoritmo implementado nesse tipo de resolução de problema é o do Bellman-Ford. E ele possui aplicação tanto prática matemática quanto computacional.

Ao olharmos para implementação de tal problema percebemos que estruturas tais como de repetição são utilizadas e funções sua complexidade é alta mas pode ser que muitas vezes determinados algoritmos que não sejam do conhecimento de quem está implementando tenham complexidade menor do que aquela apresentada pelo programador por exemplo, a noção recursiva pode ser implementada através de uma função recursiva que se chama dentro de si, ou pode ser representada através de uma estrutura de repetição *for* e tem uma complexidade maior do que a função recursiva.

Na solução matemática com o auxílio de algoritmo, vemos que temos a certeza de esgotarmos todas as possibilidades, enquanto se fizéssemos apenas pelo método de enumeração, tal resultado poderia estar incorreto, por não termos a certeza de esgotar todas as possibilidades de caminhos. Já no fluxograma e na abstração vemos o passo a passo adotado na solução matemática que tem o auxílio do pensamento computacional. Já na implementação o aluno tem oportunidade de aprender por exemplo como entrar com os dados de um grafo que está em formato de imagem e traduzi-lo para uma linguagem de matriz na qual o computador consegue ler. Ou seja, ele pode ver a transformação de uma imagem em linguagem de alto nível na implementação em C ++. E vemos o uso da estrutura fila na implementação e que na solução matemática não é usada.

Questão 24 - (Mack-2008) Na figura, o quadrado ABCD é formado por 9 quadrados congruentes. O total de triângulos distintos, que podem ser construídos, a partir dos 16 pontos,

Figura 119 – Questão 24

Fonte: Mack, 2008

a) 516

b) 520

c) 526

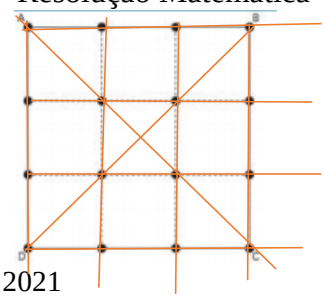
d) 532

e) 546

24.1 Resolução Matemática:

Para formarmos triângulos precisamos de 3 pontos escolhidos no total de 16. Logo isso pode acontecer de $C_{16,3}$, mas se os vértices escolhidos forem colineares tanto nos segmentos horizontais, ou verticais ou na diagonal, então eles não formarão triângulos. Temos que retirar esses casos em que de 4 vértices nos segmentos são escolhidos 3 vértices. Se observarmos temos 10 segmentos de reta que são os segmentos verticais formados por 4 vértices, os segmentos horizontais e as diagonais desse quadrado. Destacamos esses dez segmentos a seguir:

Figura 120 – Resolução Matemática – Etapa A

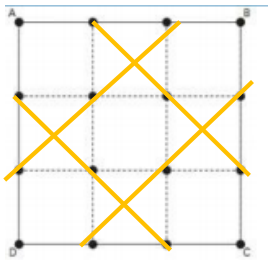


Fonte: Os autores, 2021

Isso pode acontecer de 10. $C_{4,3}$ maneiras distintas.

Mas ainda temos outro conjunto de segmentos com 3 pontos colineares:

Figura 121 – Resolução Matemática – Etapa B



Fonte: Os autores, 2021

Ou seja, mais quatro casos de colinearidade para retirarmos. Também podemos ver como sendo 4 casos de $C_{3,3}$.

Logo existem $C_{16,3}$ - 10. $C_{4,3}$ - 4. $C_{3,3}$ = 560 – 40 – 4 = 516 opção a.

24.2 Resolução Pensamento Computacional

24A - Decomposição

Sempre iremos escolher 3 dentre o total de pontos dado na malha quadriculada. Devemos retirar os casos em que temos pontos colineares em segmentos de reta

24B - Padrão

Sempre teremos que dividir o problema em casos menores. O primeiro é escolher 3 pontos dentre o total de pontos. Depois ver os casos em que podemos ter segmentos com pontos colineares e fazer esse cálculo e por último realizar a diferença entre esses valores.

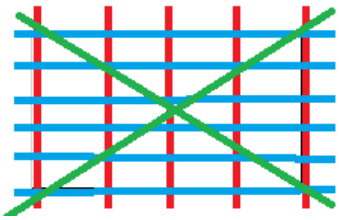
24C - Abstração

Vamos ver como seria a Resolução de uma malha com 25 pontos:

Vamos escolher 3 dentre 25. Mas temos as exceções:

A primeira são os segmentos de retas com cinco pontos em que temos pontos colineares. Neles temos os segmentos verticais, horizontais e diagonais:

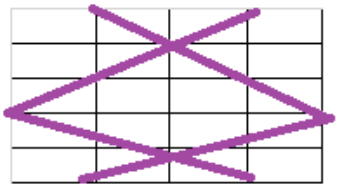
Figura 122 – Resolução Matemática – Etapa C



Fonte: Os autores, 2021

No total temos 12 segmentos na qual escolheremos 3 no total de 5 pontos então teremos 12. $C_{5,3}$. Temos também os casos em que temos segmentos com 4 pontos e escolhemos 3 pontos colineares:

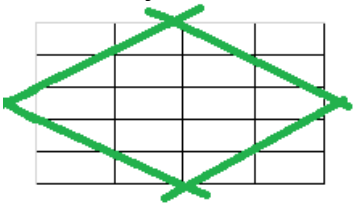
Figura 123 – Resolução Matemática – Etapa D



Fonte: Os autores, 2021

Então temos 4 segmentos de reta de modo que 4. $C_{4,3}$. E finalmente temos os casos em que temos segmentos formados por 3 pontos e vamos escolher os 3 colineares:

Figura 124 – Resolução Matemática – Etapa E



Fonte: Os autores, 2021

Então temos 4 segmentos de reta de modo que 4. $C_{3,3}$

Então podemos montar triângulos da seguinte maneira:

$C_{25,3} - 12.C_{5,3}$ - 4. $C_{4,3}$ −4. $C_{3,3}$ = 50600 – 12.10 – 16 – 4 = 50460 maneiras

Podemos perceber alguns padrões.

Se tivermos n² pontos. Podemos combinar de $C_{n^2,3}$ maneiras diferentes.

Depois vamos retirando as exceções. A primeira seriam os segmentos verticais, horizontais e as duas diagonais. No total teremos 2n + 2 segmentos e assim teremos (2n + 2) $C_{n,3}$

maneiras distintas. Depois vamos subtraindo cada uma das combinações: $4.C_{(n-1),3} - 4.\ C_{(n-2),3}$ - .... - $4.\ C_{4,3} - 4.C_{3,3}$

24D - Algoritmo

Inicio ( )
Imprimir ("Qual a dimensão da malha de pontos? (escolha dentre 4,9,16, 25, 36, 49, 64, 81,100, 121)")
Leia (n)
Resultado = $C_{n^2,3}$ – calcula(exceção)
Imprimir ("Existem", resultado, "maneiras de formarmos triângulos")
Fim

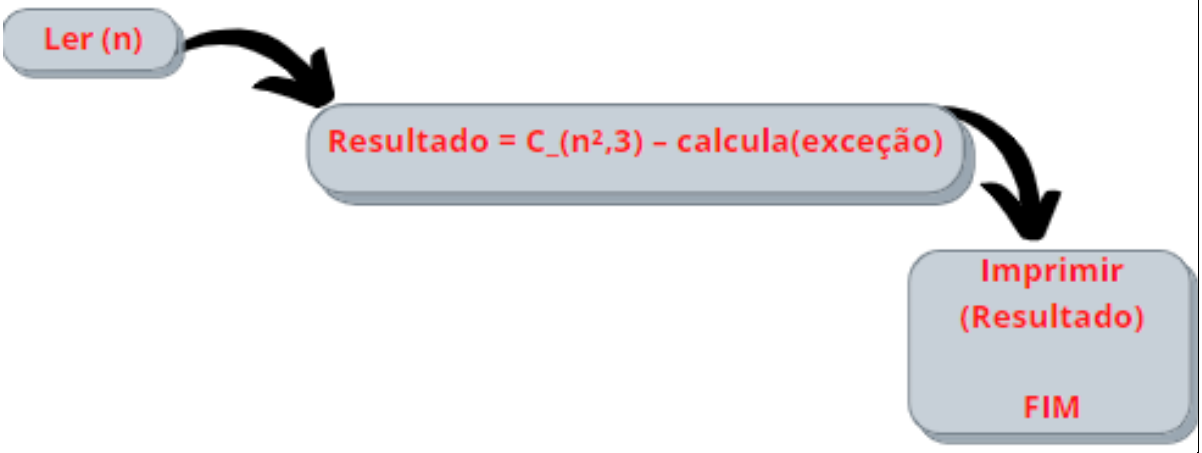
24.3 Comparando Soluções

Tabela 38 - Comparando Soluções – Questão 24

A – Resolução Matemática

Existem $C_{16,3}$ - 10. $C_{4,3}$ - 4. $C_{3,3}$ = 560 – 40 – 4 = 516

B – Fluxograma



C – Abstração

```
Inicio ( )
        Imprimir ("Qual a dimensão da malha de pontos? (escolha dentre 4,9,16, 25, 36, 49, 64, 81,100, 121)")
        Leia (n)
        Resultado = C_(n²,3) - calcula(exceção)
        Imprimir ("Existem", resultado, "maneiras de formarmos triângulos")
Fim ( )
```

Fonte: Os autores, 2021

D – Implementação – Linguagem C ++

Figura 125 - Implementação em linguagem C++ da questão 24

```cpp
#include <iostream>

unsigned long long int Combinacao(int n, int p){
    if (p == 0 || n == p){
        return 1;
    } else {
        return Combinacao(n - 1, p - 1) + Combinacao(n - 1, p);
    }
}

using namespace std;

int main()
{
    int n, pontos;

    cout << "Qual e o tamanho da malha? (escrever n tal que tenha n x n pontos no plano)" << endl;
    cin >> n;

    if (n == 1){
        cout << "Pode-se construir 0 triangulos distintos em uma malha com 1 ponto." << endl;
        return 0;
    } else if (n == 2){
        cout << "Pode-se construir 4 triangulos distintos em uma malha com 4 pontos." << endl;
        return 0;
    }

    pontos = n * n;

    cout << "Pode-se construir";

    unsigned long long int resposta = Combinacao(pontos, 3);
    // C(n * n, 3) dá o número de combinações possíveis de 3 pontos envolvendo todos os pontos da malha.
    cout << " C(" << pontos << ", " << 3 << ")";

    resposta -= (2 * n + 2) * Combinacao(n, 3);
    // n * C(n, 3) + n * C(n, 3) + 2 * C(n, 3) são as combinações de pontos nas linhas, colunas e diagonais.
    cout << " - " << 2 * n + 2 << " * C(" << n << ", " << 3 << ")";

    for(int i = 3; i < n; ++i){
        resposta -= 4 * Combinacao(i, 3);
        cout << " - " << 4 << " * C(" << i << ", " << 3 << ")";
    }
    // Isso se refere a cada "meia-diagonal" existente na malha, que também não pode ser escolhida.

    cout << " = " << resposta << " triangulos distintos em uma malha com " << pontos << " pontos." << endl;

    return 0;
}
```

Fonte: Os autores, 2021

24.4 Considerações didáticas

Nesse problema 24 também podemos observar a interseção entre a geometria, combinatória e otimização. A questão é classificada como de contagem pois queremos saber a quantidade de triângulos que podem ser formados dentro da malha quadriculada, mas ela possui um potencial caráter de otimização, pois queremos saber a quantidade máxima de triângulos existentes dentro da malha quadriculada. Por outro lado, também é uma questão de existência, pois para contarmos é necessário que exista e pode ser vista também como de classificação, pois para que haja solução devem ser escolhidos três pontos não colineares a fim de formarmos um triângulo. A solução matemática apresentada se dá através do uso de uma técnica de contagem que é a combinação, porém a busca pelo padrão ou por uma abstração nesta questão não é usual então buscamos através da decomposição do problema em situações menores buscar um padrão que é apresentado na dissertação e dessa maneira conseguimos encontrar uma configuração que tentamos levar para o pensamento computacional. Esses padrões foram implementados através do uso de estruturas de condição e fórmulas recursivas a fim de trazer uma resposta correta.

Esta questão tem o potencial do pensamento combinatório e computacional na intersecção de ser resolvida através da decomposição em casos menores e essa característica ser em comum com as duas áreas de conhecimento, bem como pela beleza de trabalhar a geometria junto da combinatória e do pensamento computacional.

5.2 – Conexões e articulações adicionais com a BNCC

Diante das análises das questões de acordo com as classificações dos tipos de questões de combinatória e as habilidades da BNCC que podem ser relacionadas com combinatória e pensamento computacional obtemos a seguinte tabela:

Tabela 39 – Classificação dos Problemas Combinatoriais e habilidades da BNCC

QUESTÃO	COMBINATÓRIAS					ENSINO FUNDAMENTAL																ENSINO MÉDIO					
						EF01MA				EF04MA		EF05MA		EF 06 MA					EF07MA			EF08MA	EM13MAT					
	ENU	COT	EXI	OTI	CLA	2	11	12	20	6	28	9	25	3	4	23	33	34	5	6	7	11	106	310	315	507	508	405
1																												
2																												
3																												
4																												
5																												
6																												
7																												
8																												
9																												
10																												
11																												
12																												
13																												
14																												
15																												
16																												
17																												
18																												
19																												
20																												
21																												
22																												
23																												
24																												

Fonte: Os autores, 2021

Para a construção da tabela, levamos em consideração o comando de cada problema resolvido nessa dissertação. Se a pergunta fosse sobre quantidade, classificamos como um problema de contagem e colocamos a cor azul escuro. Se o comando da pergunta fosse de existência, então categorizamos como problema combinatório de existência e colorimos com a cor vermelha. Se o comando era pela solução máxima, ou mínima, classificamos como problema de otimização e colocamos a cor verde escuro. Caso a pergunta fosse para exibir todas as possíveis soluções então classificamos como problema de enumeração e colorimos com a cor laranja.

Além disso, observamos ao longo desse trabalho a categorização típica de problemas de combinatória, mas que ao serem solucionados via pensamento computacional nos fizeram perceber que dentro de um problema combinatório podemos observar outras características concomitantemente.

Assim, um problema combinatório pode ter uma classificação devido ao comando da pergunta, mas também ter características ou propriedades combinatoriais que não são do mesmo tipo, tais como problemas de contagem que se conectam com características de otimização.

Para representar essas intersecções e múltiplas tipagens combinatoriais, utilizamos cores mais claras para evidenciar quais os outros tipos de natureza combinatória existem em cada uma das questões classificadas.

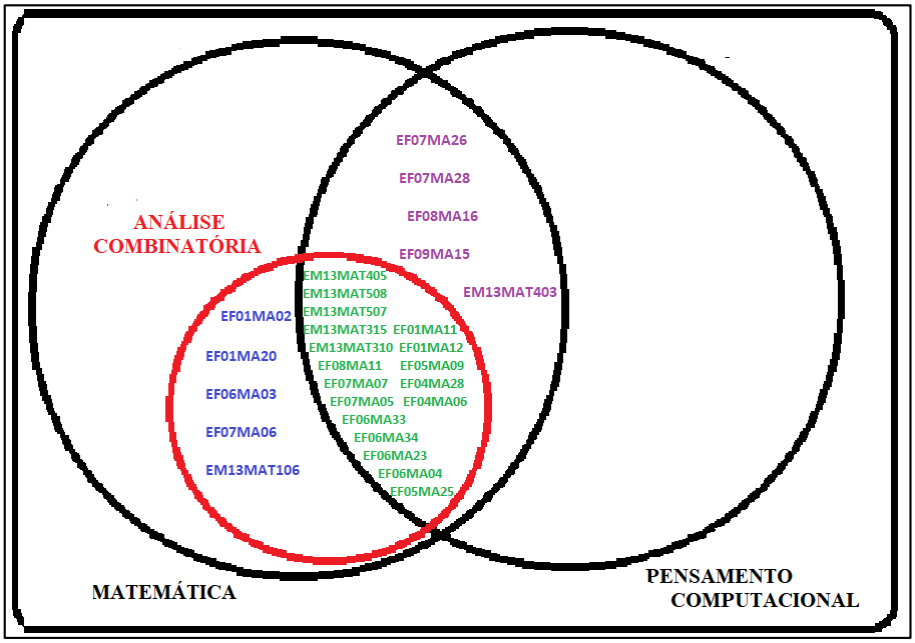
Assim, por exemplo, se olharmos para o problema 26 na tabela, veremos que ele é classificado como um problema de contagem. Isso significa que o comando do problema envolve a palavra *quantos.* Mas no mesmo problema, notamos que as colunas de existência, otimização, enumeração e classificação estão pintadas em azul claro.

Isto significa que no problema 26 notamos características combinatoriais desses quatro tipos. E na mesma linha cada coluna que esta pintada de azul escuro significa as habilidades da BNCC presentes nesse problema e que estão diretamente ligadas com a área de combinatória ou pensamento computacional.

As observações sobre as classificações dos problemas combinatoriais já foi realizada no capítulo 3. A partir daqui vamos evidenciar algumas características observadas em relação as habilidades da BNCC presentes na tabela. Estas habilidades foram escolhidas, pois elas têm características combinatoriais ou de pensamento computacional.

Diante disso, separamos cada uma dessas habilidades e as categorizamos como sendo características de análise combinatória ou como características de pensamento computacional. Representamos essa propriedade através da imagem a seguir, na qual trabalhamos com dois conjuntos: o conjunto da matemática e o conjunto do pensamento computacional. Observe:

Figura 126 – Relação entre BNCC, análise combinatória e pensamento computacional



Fonte: Os autores, 2021

Este conjunto nos evidencia que, a matemática presente na BNCC, em particular a área de combinatória, tem como habilidades que podem ser classificadas como sendo de combinatória sem característica de pensamento computacional as que estão escritas em azul, como por exemplo a habilidade EF01MA02.

Por outro lado, observamos habilidades da BNCC que podem ser classificadas como tendo propriedades matemáticas e de pensamento computacional, mas que não envolvem propriedades combinatoriais. Estas habilidades estão em roxo, como por exemplo a habilidade EF09MA15.

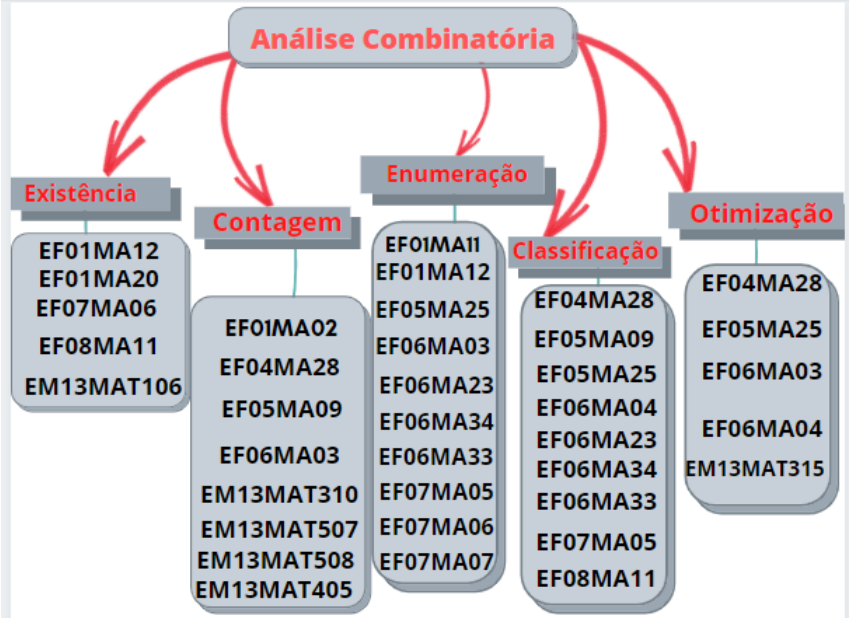
Finalmente, colocamos em verde as habilidades presentes na BNCC que podem ser classificadas como tendo propriedades combinatoriais e de pensamento computacional ao mesmo tempo. E essas são a grande maioria ao olharmos para o conjunto. Isso é mais uma afirmação da importância de atrelarmos a solução de problemas de análise combinatória com o pensamento computacional na educação básica e como elas estão conectas diante das

habilidades da BNCC. Por exemplo, as habilidades EM13MAT405 e EF05MA25 são classificadas como habilidades computacional e combinatorial.

É notável a grande quantidade de habilidades presentes na BNCC no ensino fundamental que tem propriedades das duas áreas. Isso reforça, junto as observações levantadas dos problemas dessa dissertação, a possibilidade e necessidade de trabalharmos mais questões combinatoriais envolvendo pensamento computacional no ensino fundamental em todos os anos.

Além dessa análise, relacionamos as habilidades da BNCC presentes nesse trabalho com as classificações de problemas de análise combinatória e chegamos ao seguinte resultado:

Figura 127 – Relações entre Classificação combinatorial e habilidades da BNCC



Fonte: Os autores, 2021

Por exemplo, a habilidade EF01MA12, a qual aborda a descrição da localização de pessoas ou objetos no espaço segundo um dado ponto de referência, pode ser classificada como sendo de existência, pois assim como a questão do ENEM de caminho entre dois pontos tem essa habilidade, se a solução exibida tem essa característica, significa que se trata de um problema que pode ser categorizado como de existência. Ao seguirmos esse mesmo raciocínio, essa habilidade pode ser relacionada com problemas de enumeração, uma vez que ao encontrarmos um caminho que envolve tal habilidade e este existe podemos nos perguntar como é a configuração dessa solução. E nesse caso, estamos enumerando a solução.

Seguindo esse mesmo raciocínio, a habilidade EF04MA24, que se refere a, *realizar pesquisa envolvendo variáveis categóricas e numéricas e organizar dados coletados por meio de tabelas e gráficos de colunas simples ou agrupadas, com e sem uso de tecnologias digitais,* pode ser relacionada a problemas de combinatória do tipo: otimização, classificação e contagem.

Relacionamos com a natureza de otimização, pois diversos problemas de otimização foram resolvidos através do auxílio de tabelas, seja na solução matemática ou por pensamento computacional, tais como as questões 4, 13, 14, 15, 23, 28, 30 e 31. Ela também pode ser relacionada com questões de classificação, pois ao resolvermos esse tipo de problema, podemos nos deparar com a divisão do problema em casos e termos que decidir qual a melhor estratégia a seguirmos. E pode ser classificada como de contagem, pois ao nos perguntarmos quantos caminhos existem entre um vértice e outro, por exemplo, esta informação pode ser obtida através de uma tabela de adjacência do grafo em questão no qual contamos cada um dos diferentes caminhos que podemos encontrar entre dois vértices. Essa característica pode ser observada na questão 4. Se construíssemos a tabela de adjacência desse grafo e perguntássemos quantos caminhos distintos existem entre os vértices a_6 e a_4, essa relação então poderia ser observada nas questões 13, 14 e 15.

Além disso, em nosso trabalho Borges (2018), analisamos livros didáticos adotados naquele ano no ensino básico e notamos vários com questões de combinatória envolvendo grafos e outras considerações foram realizadas. Diante da BNCC e dos livros de projetos, analisamos alguns que possam ter nos projetos as habilidades presentes nesse trabalho ou a intersecção de combinatória e pensamento computacional.

O livro Nani (2020) no primeiro projeto e etapa 1 trabalha com temas interdisciplinares e notamos a presença de fractais e reconhecimento de padrões. Esse projeto tem características presentes no pensamento computacional e que corretamente adaptadas podem ser construídas junto à combinatória.

O livro Leonardo (2020) apresenta na página 131 um projeto que envolve localização no qual podemos explorar questões de combinatória envolvendo localização e locomoção no estilo das questões 29 e 30 do nosso trabalho, por exemplo. Nesse mesmo estilo observamos Trevizan et al (2020) no projeto 3.

Em Silva (2020), temos o projeto, página 76, no qual trabalha com criptografia e habilidades presentes no nosso trabalho, tais como EM13MAT405. Esse tema aliado a

habilidade pode ser adaptado para questões de combinatória solucionados via pensamento computacional

No livro de Pereira et al (2020) encontramos três projetos com potenciais de trabalhar a temática de nossa dissertação: o projeto 1 com as habilidades EM13MAT508, o projeto 5, que tem presente teoria de grafos e o projeto 6, o qual trabalha com as habilidades EM13MAT405, EM13MAT315 e EM13MAT507.

6 CONSIDERAÇÕES E ALINHAMENTOS FINAIS

Este trabalho apresentou uma investigação de diferentes tipos de problemas de combinatória (contagem, existência, enumeração, classificação e otimização), em nível do Ensino Médio, ora abordados pelas usuais técnicas de contagem ensinadas na Educação Básica, ora abordados por meio de processos que caracterizam um tipo de pensamento computacional, mostrando conexões, potencialidades e limitações das duas abordagens, inclusive para o ensino de combinatória na Educação Básica.

Vimos nitidamente a dificuldade em separar o pensamento matemático do pensamento computacional. A definição de pensamento computacional depende de quem usa. Ou seja, um programador tem uma visão, alguém da área de TI terá outra visão, assim como para um aluno do ensino médio ou professor da educação básica. Tentamos separar os dois pensamentos ao longo da resolução dos problemas, mas não conseguimos deixar isso evidente onde começa um pensamento ou em que ponto termina o outro.

Mas com isso, mostramos que para programar um computador é necessário entender as partes de um problema, inclusive para resolver problemas de combinatória, enriquecemos a visão de resolução de problemas de combinatória através de uma visão computacional e ver que existem outras maneiras de resolver um problema de combinatória fora das resoluções usuais, tais como através de enumeração via implementação computacional.

Uma primeira contribuição do nosso estudo, e de certa forma, que foi sendo percebida na medida que investigávamos os problemas, da resolução a implementação dos algoritmos, é que a classificação dos problemas de combinatória deve ser considerada para ampliar a visão sobre diferentes problemas de combinatória, bem como, diferentes formas de resolver um mesmo problema. Ou seja, usamos as categorias para organizar e ressaltar características especiais dos problemas, sem ter a intenção de limitar possibilidades de estratégia, resolução e produção de significados, sendo, portanto, um convite à ampliação da visão sobre a natureza dos problemas de combinatória, e da dinâmica da produção de significados. E isso foi se confirmando ao longo da nossa investigação.

Mas não foi só isso. Nosso estudo aponta variadas conexões entre combinatória e pensamento computacional, gerando formas alternativas de selecionar, categorizar, abordar, resolver e pensar problemas de combinatória, tanto pela sua natureza como pelas diferentes estratégias de solução, tais como: novas possibilidades de problemas de contagem, tais como os de existência e otimização, a partir do uso de processos recursivos; o uso de algoritmos,

fluxogramas e programas criando oportunidades de novas percepções e produção de significados sobre as etapas de resolução de problemas de combinatória; a ampliação das estratégias enumerativas devido à natureza da estrutura algorítmica e do aumento e velocidade do poder computacional; possibilidades de novas leituras sobre a dinâmica dos limites e das intersecções dos tipos de problemas de combinatória; a ampliação das estratégias enumerativas devido à natureza da estrutura algorítmica e do aumento e velocidade do poder computacional; o dinâmica do papel das ténicas de contagem em processos computacionais, dentre outras.

Para sintetizar os principais resultados obtidos a partir do objetivo central que traçamos, e evidenciarmos pontos que nos parecem ser importantes, vamos nos basear nos objetivos específicos apresentados na introdução.

(i) Identificar e selecionar problemas de combinatória de contagem, existência e otimização que podem ser explorados via estratégias do pensamento computacional e pelo pensamento combinatório.

Os 24 problemas que selecionamos, ora conversam com a sala de aula, pois estão presentes em livros didáticos de matemática, ENEM ou outros exames de acesso aos cursos de graduação de diferentes universidades. Os problemas de combinatória de existência e otimização ampliam o leque, mostrando processos possíveis para a Educação Básica, que já foram pesquisados e abordados, conforme as pesquisas de Muniz (2007) e Borges (2018), dentre outras.

Nesse aspecto de ampliar possiblidades, nosso trabalho se alinha com a perspectiva de Dantas (2019), ao possibilitar a produção de significados além daqueles que os materiais didáticos habituais possibilitam.

Visto isso, ao analisarmos os resultados e reflexões ao longo do nosso trabalho, percebemos que dentre os trinta e cinco problemas de combinatória, a maioria é classificada como contagem. Ou seja, dezessete são classificados como de contagem, nove são classificados como de existência e oito são classificados como de otimização. Dentre as questões de contagem, todas poderiam ser resolvidas, para um conjunto pequeno é claro, através da enumeração das possíveis configurações existentes no problema. Ou seja, a resolução de problemas via enumeração pode ser uma estratégia para resolver problemas de contagem, desde que o número de soluções não seja suficientemente grande a ponto de o poder computacional disponível não dar conta de exibir todas elas.

Resolver um problema por meio de um algoritmo e implementar em uma linguagem de programação também foi um critério. Aqui chamamos atenção do leitor para a estrutura dos

programas e os processos de resolução: há implementações computacionais que seguem a linha da solução matemática e outros que usam outros caminhos que não estão presentes na solução matemática. A experimentação, muitas vezes via enumeração, e muito presente nos problemas de combinatória de existência, privilegiam aspectos numéricos que não costumam ser privilegiados em sala de aula, quer pela questão do tempo, quer pela limitação computacional. Testar muitas soluções geralmente não é viável nas atividades de sala de aula.

Nesse sentido, nossa escolha de problemas, e a investigação destes mostram possibilidades de trazer a experimentação para o centro da atividade de produção de conhecimento matemático, conforme se pode ver em Muniz (2007) e Dantas (2019).

Esse é um resultado que consideramos importante do nosso trabalho. Passemos assim, para os dois outros objetivos específicos.

(ii) Resolver os problemas selecionados usando as duas estratégias, analisando-as e comparando-as nas perspectivas combinatorial, computacional e educacional

(iii) Apresentar considerações didáticas, incluindo conexões entre pensamento computacional, combinatória e habilidades da BNCC.

No estudo, ampliamos nossa visão de classificação combinatorial e as possibilidades de novos questionamentos que podemos realizar sobre o problema, ampliando a visão da natureza de cada problema, além da classificação usual que já realizamos.

Nos problemas de combinatória de existência percebemos a presença de processos da categoria classificação, pois seguimos alguns critérios a fim de buscar a existência do que é perguntado, como por exemplo, quando buscamos um caminho Euleriano.

Também percebemos a presença do processo enumerativo, pois se existe uma resposta, é possível listá-la. Nas questões classificadas como de otimização identificamos processos de enumeração (ao listarmos soluções até encontramos a solução ótima, ou criarmos uma lista de configurações até encontrarmos a solução) e a de existência, pois se exibimos um resultado é porque esse existe. Existe também o caráter de contagem, uma vez que se queremos o máximo é necessário que contemos qual é o valor máximo.

Isso se alinha com a perspectiva de Dantas, que levanta a importância de tirar do cenário principal os processos puramente algébricos e pôr em cena a interação numérica, a visualização geométrica dinâmica e a produção de automações. (DANTAS, 2019)

Observamos intersecções entre o pensamento computacional e a estratégia de solução matemática nas etapas de abstração, na qual a abstração matemática não é a mesma da computacional, mas em alguns casos possuem aspectos em comum. Destacamos a questão 3

que foi resolvida recursivamente tanto pela solução matemática, quanto pela implementação. O uso da Teoria de Grafos é um exemplo disso. Na matemática é um ente matemático para solução e representação de problemas, enquanto no pensamento computacional é uma estrutura algorítmica usada como ferramenta para a representação e resolução de problemas. Temos a presença dessa estrutura nas questões 4, 13, 14, 15, 16, 17, 20. 29, 30, e 31.

Problemas resolvidos por estratégia matemática ao serem resolvidos por estratégia do pensamento computacional ou quando implementados demandam outros questionamentos, tais como: de que maneira representar uma imagem na implementação computacional ou como representar a localização de um ponto, como na questão do ENEM 2021 em que na implementação utilizamos coordenadas.

Na primeira questão, do aperto de mão, cuja solução matemática exibiu a quantidade de apertos possíveis, a implementação busca resolver se dado uma quantidade n de apertos de mão é possível acontecer tal configuração obedecendo às regras do enunciado.

Essa mudança entre a solução matemática a computacional evidencia que um problema de contagem ao ser generalizado, de acordo com as habilidades EF07MA06 e EF07MA07, pode se transformar num problema de existência e a busca em saber se ele tem solução ou não está de acordo com EF01MA20.

Há termos em matemática que tem significado diferente de computação, como por exemplo a palavra variável (CIEB) ou algoritmo (DANTAS, 2019) e essa diferença pode ser notada na questão dois. Nela, a etapa de abstração na solução matemática nos remete a solução de uma equação de segundo grau, enquanto na implementação do pensamento computacional essa resolução da equação não se dá através do cálculo da fórmula, mas pelo uso de estruturas condicionais e do prévio conhecimento matemático do cálculo do discriminante e suas propriedades.

A estrutura do condicional reduz o número de possibilidades a ser analisado. O processo de contagem em questões de enumeração ou otimização, tal como as questões 4, 13, 14, 15, 16, 17 e 20, o condicional limita quais caminhos serão analisados e assim o programa ganha tempo e eficiência na resolução do problema.

Ao criarmos uma solução com estratégias de pensamento computacional, percebemos que certas estruturas matemáticas ou problemas matemáticos acabam perdendo parte de caráter matemático na hora da implementação e ganham novas características típicas do pensamento computacional, tais como a estrutura de repetição. Ao pensarmos nesse tipo de solução, que envolve combinação ou permutação, por exemplo, pelo pensamento computacional, veremos que sua estrutura pode ser representada computacionalmente através de uma estrutura de

repetição (enquanto, para) ou de recursividade. Mas, ao resolvermos um problema de contagem por recursão na implementação computacional, veremos que o tempo de execução e sua complexidade caem. Por exemplo, no problema de permutação caótica, no problema três a solução matemática podemos calcular através da fórmula de desordenamento ou por recursão e no problema seis e sete, cujas soluções foram implementadas recursivamente.

O pensamento computacional é uma ferramenta para solucionar problemas matemáticos de otimização, que apenas pelo processo manual de exaustão nem sempre é possível exibir todas as configurações de solução. Uma estratégia de solução matemática via pensamento computacional não necessariamente necessita de profundo conhecimento matemático, ou seja, ao saber resolver problemas e a lógica computacional a pessoa pode resolver usando estruturas computacionais ao invés de pensar em soluções matemáticas.

Por outro lado, o conhecimento matemático pode ser muito importante na etapa de implementação computacional, como por exemplo nas questões 23 e 24 que ao serem implementadas foi necessário a noção de plano de cartesiano para saber localizar o ponto de partida e chegada e está noção também caracteriza uma das habilidades da BNCC (EF01MA11) e (EF01MA12).

A implementação computacional pode auxiliar no melhor entendimento do conteúdo de combinatória através por exemplo da enumeração dos resultados tais como de caminhos, rotulações etc. Por exemplo, destacamos as questões 13, 14 e 15. Todo problema quando implementado é considerado ótimo se ele é resolvível em pouco tempo e se é eficiente, mas nem sempre isso acontece como no caso da sequência de Fibonacci. Se tentarmos resolver a sequência de maneira recursiva no computador para o valor de n muito alto veremos essa característica, comum de problemas NP-completos.

Identificamos também que todo problema de contagem também pode ser visto como se fosse um problema de existência, na medida que cada solução só pode ser contada, se ela existe.

Apenas uma das questões apresentou classificação enumerativa. Então nos parece que os problemas de combinatória tem em si mais as definições de enumeração e classificação como naturezas ou propriedades de problemas combinatórias do que como uma classificação. A classificação das questões combinatórias se perde muitas vezes dentro da própria natureza da questão, uma questão pode ser classificada como sendo de contagem e sempre ter o caráter enumerativo, por exemplo.

Outra questão é que ao analisarmos os problemas através das habilidades da BNCC podemos observar que existem muito mais habilidades relacionadas a combinatória ou pensamento computacional no ensino fundamental do que no ensino médio: existem seis

habilidades no Ensino Fundamental I, onze habilidades no Ensino Fundamental II e seis habilidades no Ensino Médio. Isso nos evidencia que a combinatória é um objeto de conhecimento da área da matemática inerente da formação do discente.

A questão 34 é de contagem, mas ela possui um potencial caráter de otimização, pois queremos saber a quantidade máxima de triângulos existentes dentro da malha quadriculada. Isso reforça a conexão entre problemas de contagem e otimização. E nos mostra que podemos trabalhar questões de contagem e otimização no ensino fundamental, pois estão diretamente conectados.

Na resolução de problemas os alunos tendem a considerar que soluções não inteiras são incorretas e que todo problema tem solução quando proposto em sala de aula. A questão 2 generalizada computacionalmente nos dá um ganho no sentido de mostrar ao aluno que em certas situações o valor inserido não tem solução ótima, o que reforça o ganho que o pensamento computacional aliado, quando possível, à implementação trazem na reflexão dessa questão e ajuda a quebrar essa falsa crença de que todo problema tem solução e que esta é sempre um número inteiro e positivo.

Através da análise da questão 3, podemos perceber que, ainda que uma questão seja de contagem, ela pode ser analisada pelo professor com potencial para criar questões de existência, bem como utilizar estratégias de enumeração e classificação para a solução do problema.

Dentre as possíveis representações de solução de problemas combinatórios, podemos usar a ideia de abstração de uma questão matemática, a representação através de um fluxograma, conforme presente na BNCC em EF06MA34, ou através de algoritmos, presente em EF06MA04.

Na implementação vemos que é usado algumas estruturas que não estão presentes em nenhuma das estratégias de solução matemática. Estruturas de lógica ou vetorial. Outra estrutura é a de função e rotina para que o programa fique mais organizado e rotinas de repetição e condicionais. Isso pode ser observado no problema 4.

Alguns problemas de contagem podem ter regras para o seu cálculo. A isto chamamos combinatória de classificação, como é o caso da questão 5. E ao pensarmos na implementação de problema de maneira abstrata e geral, nos perguntamos se é possível resolver o problema de acordo com as condições impostas inicialmente. Nesta etapa da implementação problema ganha classificação de existência.

A partir de um problema de combinatória, ao resolvermos via pensamento computacional, novas perguntas ou reflexões surgem e trazem consigo outras estratégias combinatoriais de resolução de problemas.

Problemas matemáticos que são necessários serem divididos em casos para o cálculo, na implementação computacional podem ser simulados através de uma estrutura lógica, tal como é resolvida no problema 10.

Problemas combinatoriais de classificação podem ser resolvidos através do método da exaustão, ou também conhecido como enumeração de todos os casos, como a questão 12 e dado um problema de contagem ele está diretamente conectado a ideia de existência. Isso porque se pensarmos em generalizar a questão e implementá-la veremos que podemos analise se existe ou não solução para o problema dado o resultado total da contagem, tal como no P12 ou P1.

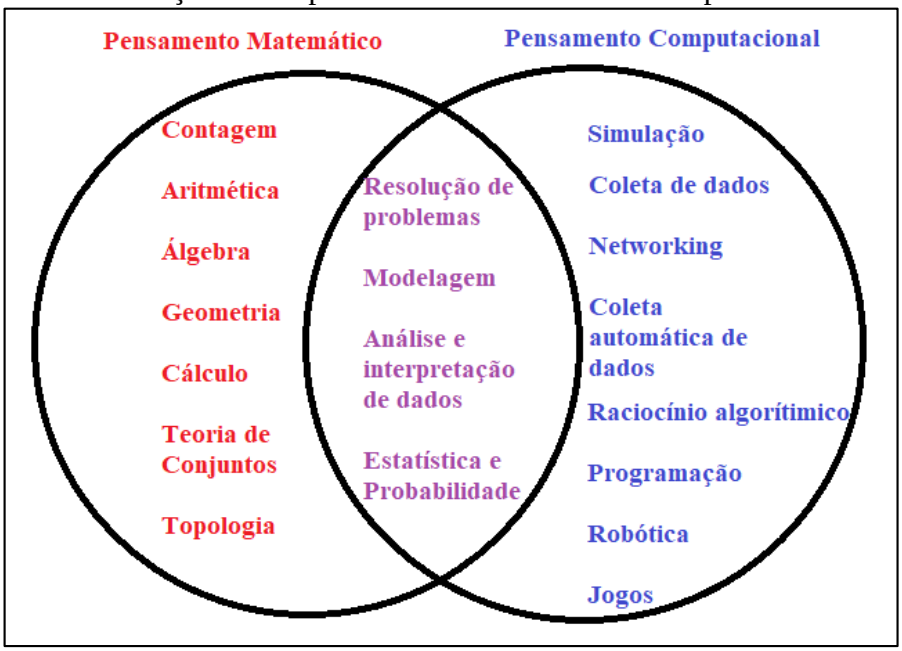
Soluções para problemas combinatoriais usando estratégias matemáticas ou computacionais podem usar tabelas ou matrizes como auxiliar na busca da solução ótima. O uso de tabelas e matrizes são objeto comum entre soluções matemáticas ou computacionais, como podemos observar nas questões 4, 13, 14, 15 e 23. E o uso de tabelas está de acordo com a habilidade EF06MA33 presente na BNCC.

A questão 6 quando foi solucionada via estratégia matemática foi dividida em casos. Essa divisão da solução combinatória em etapas caracteriza a definição de questões de classificação. Mas essas quando foram implementadas computacionalmente, ganharam a característica de contagem, pois utilizamos o padrão recursivo encontrado em cada uma delas para usar no cálculo algorítmico. Observamos que questões de classificação ao serem analisadas via pensamento computacional podem ganhar propriedades de contagem.

Finalmente, a noção de existência combinatorial está diretamente ligada a problemas de contagem e otimização, pois quando estas duas possuem solução, isso significa que tal solução pode ser exibida e isso garante a sua existência. Isto aliado com a habilidade EF01MA20 reafirmam a possibilidade de trabalharmos questões de contagem, otimização e enumeração no ensino fundamental, uma vez que elas estão conectadas entre si.

A seguir apresentamos um esquema traduzido de Kotsopoulos et al (2019) sobre as áreas em comum do pensamento combinatório e computacional.

Figura 128 – Relações entre pensamento matemático e computacional: uma reflexão



Fonte: traduzido de Kotsopoulos et al (2019).

Mesmo considerando todas as conexões, ideias e visões sobre a relação dentre combinatória e pensamento computacional, por meio da resolução de problemas, nossas reflexões e aprendizados ao longo da investigação apresentada nos trouxeram mais indagações do que respostas, gerando uma cadeia de oportunidades para trabalhos futuros, os quais podem partir de algumas perguntas, que deixamos para o leitor.

Até onde vai o pensamento combinatório e inicia o computacional e vice-versa? O pensamento computacional seria uma ferramenta para o pensamento combinatório na busca de solução de problemas? Como separar ambas as áreas de maneira disjunta? Contagem realmente é um pensamento combinatório apenas? Ao coletarmos dados por exemplo a quantidade de pessoas com menos de 30 anos numa cidade, utilizamos nessa etapa o processo de contagem, que até então foi considerado pensamento combinatório. Nesse caso então a contagem seria uma ferramenta para o pensamento computacional encontrar os resultados desejados? Ao criarmos uma modelagem matemática para entendermos processos naturais tais como o funcionamento de uma célula endócrina (Borges et al 2020), utilizamos da simulação para encontrarmos o intervalo numérico ideal em que temos o comportamento esperado.

Nessa etapa podemos ter a simulação como ferramenta do pensamento computacional para resolver essa modelagem? Analogamente nessa modelagem temos implementada na linguagem de computação equações de sistemas matemáticos oriundos do cálculo. Então o cálculo seria uma ferramenta do pensamento combinatório para resolver esse problema? Então

os dois pensamentos podem ser vistos como ferramentas para a solução de problemas em comum às duas áreas?

Será que quanto temos uma questão que envolve pensamento computacional esta pode ser vista como tendo recursos do pensamento combinatórios e computacional concomitantes considerados como ferramenta para solução de problemas? Qual é a diferença entre pensamento combinatório, pensamento computacional, pensamento combinatório como ferramenta para solução de problemas, ou pensamento computacional como ferramenta para a solução de problemas? Quando e como definir qual pensamento sobrepõe o outro na busca de solução para problemas do nosso cotidiano? Como podemos classificar os problemas como sendo de matemática ou computacional? O pensamento computacional pode sobrepor o matemático? A solução de problemas de combinatória pode ser sempre resolvida via pensamento computacional? Sendo assim, vale a pena descartar a solução matemática diante de problemas potencialmente programáveis?

Quais são as vantagens e desvantagens de solucionarmos problemas utilizando o pensamento combinatório ou o computacional? O que cada um desses pensamentos possui que o outro não tem?

Essas e outras questões ficam em aberto para refletirmos e em futuros trabalhos buscarmos trazer uma análise desses questionamentos e outros que porventura apareçam.

Esperamos que esse trabalho contribua para que professores e alunos produzam novos significados, aprendam e ampliem a capacidade de resolver problemas, e que isso se transforme em ações mais eficientes, de alguma forma, para uma sociedade mais justa, menos desigual, onde a complexidade dos problemas não seja tratada de forma simplória e mesquinha, e sim com o cuidado e estratégias compatíveis com uma visão de sociedade mais altruísta, eficiente para todos e com soluções mais humanas.

REFERÊNCIAS

AMÉRICO, Gilmar Virgolino. **Resolução de problemas sobre Combinatória para as Olimpíadas Brasileira de Matemática das Escolas Públicas OBMEP**. 2013. 30 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) - Instituto de Ciências Exatas e Naturais - Universidade Federal do Pará, Pará, 2013.

AMIEL, T.; REEVES, T. C. Design-Based Research and educational technology: rethinking technology and the research agenda. **Educational Technology & Society**, Athabasca, v. 11, n. 4, p. 29-40, Oct. 2008.

ANDRADE, D; CARVALHO, T; SILVEIRA, J.; CAVALHEIRO, S.; FOSS, L.; FLEISCHMAN, A.M.; AGUIAR, M.; REISER, R. Proposta de atividades para o desenvolvimento do pensamento computacional no ensino fundamental. In CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 2., 2013, Campinas. **Anais [...]** Porto Alegre: SBC, 2013. p.169-178. Disponível em: <http://br-ie.org/pub/index.php/wie/article/view/2645>. Acesso em: 10 nov. 2020.

BAIRRAL, Marcelo Almeida. As TIC e a licenciatura em matemática: em defesa de um currículo focado em processos. **Jornal Internacional de Estudos em Educação Matemática**, São Paulo, v. 6, n. 1, p.1-20, 2013. Disponível em: <https://revista.pgsskroton.com/index.php/jieem/article/view/97/0>. Acesso em 10 dez. 2020.

BARAB, S.; SQUIRE, K. Design-based research: putting a stake in the ground. **The Journal of the Learning Sciences**, New York, v. 13, n. 1, p. 1-14, 2004. Disponível em: < https://www.tandfonline.com/doi/abs/10.1207/s15327809jls1301_1>. Acesso em: 20 out. 2020.

BARBOSA, J. C.; OLIVEIRA, A. M. P. Por que a pesquisa de desenvolvimento na Educação Matemática?. **Perspectivas da Educação Matemática**, Mato Grosso do Sul, v. 8, n. 18, p. 526-546-19, 2015. Disponível em: <https://periodicos.ufms.br/index.php/pedmat/article/view/1462/969>. Acesso em: 30 out. 2020.

BARCELOS, T. S.; SILVEIRA, I. F. pensamento computacional e educação matemática: relações para o ensino de computação na educação básica. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO, 20., 2012, Curitiba. **Anais [...]** Curitiba: UFPR, 2012. p.219-228. Disponível em: <https://sol.sbc.org.br/index.php/wei/article/view/10976>. Acesso em: 10 nov. 2020.

BARCELOS, T. et al. Relações entre o Pensamento Computacional e a Matemática: uma Revisão Sistemática da Literatura. In: WORKSHOPS DO IV CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (WCBIE 2015), 2015, Brasília, DF. **Anais [...].** Brasília, DF: Editora da SBC, 2015. p. 1369-1378. Disponível em: <https://br-ie.org/pub/index.php/wcbie/article/view/6311/4420>. Acesso em: 13 mar. 2021.

BARICHELLO, L. **Pensamento Computacional**. Livro Aberto de Matemática. Rio de Janeiro: editora do Instituto Nacional de Matemática Pura e Aplicada, 2021. Disponível em <https://umlivroaberto.org/producao/pensamento-computacional/>. Acesso em: 10 ago. 2020.

BARICHELO, L. P. **Computação desplugada**. Campinas: Editora da UNICAMP. 2020. Disponível em:<www.desplugada.ime.unicamp.br>. Acesso em: 20 ago. 2020.

BASTOS, T. A. **Modelagem na educação matemática para o desenvolvimento de conceitos de combinatória em uma escola particular no Vale do Rio Doce em Minas Gerais.** 2020. 444 f. Dissertação (Mestrado em Educação Matemática) - Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto, Ouro Preto, 2020.

BATANERO, M. del C.; GODINO, J. D.; NAVARRO-PELAYO, V. **Razonamiento combinatorio en alumnos de secundaria**. Virgínia: Educación Matemática, 1997.

BATTY, M. A New Theory of Space Syntax, UCL Centre for Advanced Spatial Analysis. **Working Papers Series**, Londres, v. 211, n. 1, p. 1-34, mar. 2004. Disponível em:<http://discovery.ucl.ac.uk/211/1/paper75.pdf>. Acesso em: 21 jan. 2018.

BELLMAN, R. On a routing problem. **Quarterly of applied mathematics,** United States of America, v. 16, n. 1, p. 87-90, 1958. Disponível em: http://www.ams.org/journals/qam/1958-16-01/S0033-569X-1958-0102435-2/S0033-569X-1958-0102435-2.pdf. Acessado em: 15 abr. 2017.

BEZERRA, J. R. A. **Uma ferramenta didática para ajudar na fixação dos conceitos introdutórios de combinatória**. 2013. 48f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) - Universidade Federal do Rio Grande do Norte, Rio Grande do Norte, 2013. Disponível em: < https://repositorio.ufrn.br/jspui/bitstream/123456789/18656/1/JoseRAB_DISSERT.pdf>. Acesso em: 10 jan. 2019.

BIGGS, N. L. The roots of combinatorics. **História Mathematica**, Inglaterra, v. 6, n. 2, p. 109-136, 1979. Disponível em:<https://www.sciencedirect.com/science/article/pii/0315086079900740>. Acesso em: 15 jun. 2020.

BLIKSTEIN, P. **Pre-college computer science education: a survey of the field.** Mountain View: Google LLC, 2018. Disponível em: < https://research.google/pubs/pub49719/>. Acesso em: 20 out. 2020.

BOCHARDT, M.; ROGGI, I. Ciencias de la computación en los sistemas educativos de América Latina. **Ciudad Autónoma de Buenos Aires: Instituto Internacional de Planeamiento de la Educación IIPE-Unesco**. Buenos Aires, v. 79, n. 1, p. 30-33, 2019. Disponível em:<https://bit.ly/2tTAwK7>. Acesso em: 10 jan. 2021.

BORBA, R. E. S. R; ROCHA; C.A.; AZEVEDO; J. Estudos em Raciocínio Combinatório: investigações e práticas de ensino na Educação Básica. **Bolema**, Rio Claro (SP), v. 29, n. 53, p. 1348-1368, 2015. Disponível em:< http://www.scielo.br/pdf/bolema/v29n53/1980-4415-bolema-29-53-1348.pdf>. Acesso em: 02 jan. 2018.

BORGES, V. H.; MOURA, C. A. de; CORTEZ, C. M. **Modelagem da dinâmica de armazenamento hormonal em uma célula endócrina**. 2018. 56 f. Dissertação (Mestrado em

Ciências Computacionais) - Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro/ RJ, 2018.

BORGES, Vanessa Henriques e Muniz, Ivail. Otimização discreta com grafos no Ensino Médio. **Revista BoEM,** Joinville, v. 6, n.11, p. 206-221, 2018a. Disponível em: <https://www.revistas.udesc.br/index.php/boem/article/view/11921>. Acesso em: 20 nov. 2019.

BORGES, V. H. **Otimização Discreta com Grafos no Ensino Médio.** 2018. 126 f . Produto final (Programa de Residência Docente da Especialização em Educação Básica) - Colégio Pedro II, Rio de Janeiro, 2018b.

BORGES, V. H.; de Castro, M. C. S.; de Moura, C. A.; & Cortez, C. M. Model for the storage control in an endocrine gland based on Lyapunov function. In: AIP CONFERENCE PROCEEDINGS, 2019, Grécia. **Proceedings […]**. Grécia: Proceedings of the International Conference of Computational Methods in Sciences and Engineering 2019 (ICCMSE-2019), 2019. *v*ol. 2186, n. 1, p. 130006-1-130006-4.

BORGES, V. H.; MUNIZ JR, I.; DE MOURA, C. A.; SILVA, D.; CORTEZ, C. MARTINS; CASTRO, M. C. S. Computational Mathematical Model Based on Lyapunov Function for the Hormonal Storage Control. **International Journal for Innovation Education and Research**, Dhaka, Bangladesh, v. 8, n. 11, p. 375–391, 2020. DOI: 10.31686/ijier.vol8.iss11.2761. Disponível em: https://ijier.net/ijier/article/view/2761. Acesso em: 9 mar. 2021.

BORGES, V. H; MUNIZ JR, I. Soluções para problemas de otimização no ensino médio através da teoria de grafos. **Brazilian Journal of Development**, Paraná, v. 6, n. 8, p. 61999-62009, 2020a. Disponível em: <https://www.brazilianjournals.com/index.php/BRJD/article/view/15584>. Acesso em: 20 dez. 2020.

BRACKMANN C. P. **Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica**. 2017. Tese (Doutorado em Informática na Educação) – Centro Interdisciplinar de Novas Tecnologias na Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017. Disponível em: http://<hdl.handle.net/ 10183/172208>. Acesso em: 24 abr. 2019.

BRAGA, F. R. **Combinatória**: **técnicas** de **resoluções** de **exercícios**. 2017. 121f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) - Universidade Estadual de Maringá. Paraná. 2017.

BRASIL. **PCN+Ensino médio:** Orientações Educacionais Complementares aos Parâmetros Curriculares Nacionais: Ciências da Natureza, Matemática e suas Tecnologias. Brasília: MEC/Secretaria de Educação Básica, 2006. Disponível em: <http://portal.mec.gov.br /seb/ arquivos/pdf/CienciasNatureza.pdf>. Acesso em: 07 jul. 2019.

BRASIL. **Base Nacional Comum Curricular:** Ensino Médio. Brasília: MEC/Secretaria de Educação Básica, 2018.

INSTITUTO NACIONAL DE PESQUISAS EDUCACIONAIS – INEP (Brasil). **Matriz de Referência do Enem**: Matemática e suas Tecnologias – Ensino Médio. Brasília, 2010. Disponível em:<http://download.inep.gov.br/educacao_basica/encceja/matriz_competencia/Mat_Mat_Tec_EM.pdf>. Acesso em: 2 dez. 2017.

BRENNAN, K.; RESNICK, M. New frameworks for studying and assessing the development of computational thinking. In: ANNUAL MEETING OF THE AMERICAN EDUCATIONAL RESEARCH ASSOCIATION. **Proceedings […]**. Vancouver: AERA, 2012. p.25-46. Disponível em: https:// web.media.mit.edu/ ~ kbrennan/ files/ Brennan_Resnick_AERA2012_CT.pdf. Acesso em: 12 nov. 2019.

BRINDA, T.; PUHLMANN, H.; SCHULTE, C. Bridging ICT and CS: educational standards for computer science in lower secondary education. In: CONFERENCE ON INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION, 14TH., 2009, **Proceeedings […]**. New York: ACM, 2009. p. 288-292. Disponível em: http:// www.inf.fu-berlin.de/inst/ag-ddi/docs/bridgingICTandCS.pdf. Acesso em: 24 abr. 2019.

BUENOS AIRES a. Gobierno de la Ciudad de Buenos Aires. Ministerio de Educación. Dirección General de Planeamiento e Innovación Educativa. Gerencia Operativa de Currículum. **Diseño curricular nueva escuela secundaria de la Ciudad de Buenos Aires, ciclo básico**. Ciudad Autónoma de Buenos Aires: Ministerio de Educación del Gobierno de la Ciudad Autónoma de Buenos Aires. Dirección General de Planeamiento e Innovación Educativa, 2015. Disponível em:< http://www.buenosaires.gob.ar/areas/educacion/nes/pdf/DC_NES.pdf>. Acessado em: 18 dez. 2017.

BUENOS AIRES b. Gobierno de la Ciudad de Buenos Aires. Ministerio de Educación. Dirección General de Planeamiento e Innovación Educativa. Gerencia Operativa de Currículum. **Diseño curricular nueva escuela secundaria de la Ciudad de Buenos Aires: ciclo orientado del bachillerato, formación general**. Ciudad Autónoma de Buenos Aires: Ministerio de Educación del Gobierno de la Ciudad Autónoma de Buenos Aires. Dirección General de Planeamiento e Innovación Educativa, 2015. Disponível em:< http://www.buenosaires.gob.ar/areas/educacion/nes/pdf/2015/NES-Co-formacion-general_w.pdf>. Acesso em: 22 dez. 2017.

BÚRIGO, E. Z. et al. **A matemática na Escola:** novos conteúdos, novas abordagens - SEAD. Porto Alegre: Editora da UFRGS, 2012. Disponível em: <http://www.ufrgs.br/tri/sead/publicacoes/documentos/livro-matematica-escola>. Acesso em: 22 set. 2017.

CALEIRO, A. Um Exercício de Simulação de Ocupação dos Espaços Rurais e Urbanos. **Documento de Trabalho**, Évora, n. 2, p. 1-19, março 2012. Disponível em:< https://www.researchgate.net/profile/Antonio_Caleiro/publication/254413455_Um_Exercicio_de_Simulacao_de_Ocupacao_dos_Espacos_Rurais_e_Urbanos/links/578f991808ae81b44671d1ec/Um-Exercicio-de-Simulacao-de-Ocupacao-dos-Espacos-Rurais-e-Urbanos.pdf>. Acesso em: 14 abr. 2017.

CARNEIRO, J. F. **Levantamento e análise de aplicativos para dispositivos móveis, que**

possam ser utilizados no ensino de biologia, nos conteúdos anatomia e fisiologia humana. 2019. 25f. Trabalho de Conclusão de Curso (Especialização em Inovação e Tecnologias na Educação) – Universidade Tecnológica Federal do Paraná, Curitiba, 2019.

CARTIER, L. **Le graphe comme outil pour enseigner la preuve et la modélisation**. 2008. 338 f. Tesis (Doctoral) - Université Joseph-Fourier-Grenoble I, Université Joseph-Fourier, Grenoble I, 2008. Disponível em:< https://tel.archives-ouvertes.fr/file/index/docid/416598/filename/These_Cartier.pdf>. Acesso em: 10 dez. 2018.

CHOI, J.; AN, S.; LEE, Y. Computing education in Korea: current issues and endeavors. **ACM Transactions on Computing Education**, New York, v. 15, n. 2, artigo 18, p. 1– 22, 2015.

COLÉGIO PEDRO II. **Admissão de alunos do Ensino Médio do Colégio Pedro II**. Rio de Janeiro, 2009. Disponível em:<https://www.cp2.g12.br/concurso/alunos/ensino_medio/200809/regular/provas/Prova_Matematica_Diurno.pdf>. Acesso em: 2 jul. 2017.

COMPUTATIONAL THINKING TASK FORCE (CSTA). **Computational think flyer**. 2015. Disponível em: http:// csta.acm.org/ Curriculum/ sub/ CurrFiles/ CompThinkingFlyer.pdf. Acesso em: 25 abr. 2019.

CSTA. K-12 Computer Science Standards - Revised 2011 - **The CSTA Standards Task Force**. New York: Association for Computing Machinery, 2011.

DA SILVA, T. V. **O Pensamento computacional como ferramenta de resolução de problemas de matemática**. 2020. 133f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional). Universidade Federal de Campina Grande. Campina Grande. Paraíba. 2020.

DAMBROS, R. L. **Software interativo para o ensino de combinatória**. 2015. 87f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) - Fundação Universidade Federal de Mato Grosso do Sul (Campus de Campo Grande). Mato Grosso do Sul. Campo Grande. 2015

DE FARIAS, Fernando Ramos. **Uma sequência didática alternativa para o ensino de combinatória na educação básica**. 2013. 46 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional). Universidade Federal do Pará, Belém. 2013. Disponível em: <https://www.ppgme.propesp.ufpa.br/ARQUIVOS/dissertacoes/2013/Vers%C3%A3o%20Final%20Fernando.pdf>. Acesso em: 4 jun. 2020.

DEGGERONI, R. **Uma introdução à teoria dos grafos no ensino médio**. 2010. 56f. Trabalho de conclusão de curso (Graduação) - Universidade Federal do Rio Grande do Sul – Prof. Carlos Hoppen. Instituto de Matemática. Curso de Matemática: Licenciatura. Porto Alegre, 2010. Disponível em:< http://www.lume.ufrgs.br/bitstream/handle/10183/29152/000775831.pdf?sequence=1>. Acesso em: 20 out. 2017.

DANTAS, S. C. Pensamento combinatório, pensamento computacional: aproximações e distanciamentos. In: ENCONTRO PARANAENSE DE EDUCAÇÃO MATEMÁTICA – EPREM, 15, 2019, Londrina, **Anais** [...]. Londrina: EPREM, 2019. p.6 - p.10

DIJKSTRA, E.W., 1959. A note on two problems in connexion with graphs. **Numerische mathematik**, United States of America, v.1, n. 1, p. 269-271, 1959. Disponível em:< http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf >. Acesso em: 10 set. 2017.

DISESSA, A. A. **Changing minds: computers, learning, and literacy**. Cambridge: MIT, 2001. Disponível em: < https://mitpress.mit.edu/books/changing-minds>. Acesso em: 10 ago. 2020.

DISESSA, A. A. Computational Literacy and The Big Picture Concerning Computers in Mathematics Education. **Mathematical Thinking and Learning**, Filadelfia, v.20, n.1, p. 3-31, 2018. Disponível em: https://www.tandfonline.com/doi/full/10.1080/10986065.2020.1779012?src. Acesso em: 29 mai. 2020.

EDUCATION SCOTLAND. **Curriculum for excellence: technologies: experiences and outcomes**. Livingston: Education Scotland, 2009.

ESPINOZA, J.; ROA, R. La combinatória en libros de texto de matemática de educación secundaria en España. In: INVESTIGACIÓN EM EDUCACIÓN MATEMÁTICA, 18., 2014, Salamanca. **Anais [...].** Salamanca: Universidad de Salamanca, 2014. p. 277-286. Disponível em:<http://funes.uniandes.edu.co/6007/1/ Espinoza 2014 LacombinatoriaSEIEM.pdf>. Acesso em: 2 ago. 2017.

ESTEVES, A.; NOSCHANG, L.; RAABE, A.; FILHO, A.; Portugol Studio: Em direção a uma comunidade aberta para pesquisa sobre o aprendizado de programação. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 27, 2019, Belém. **Anais [...]** Porto Alegre: Sociedade Brasileira de Computação, july 2019. p. 513-522. DOI: https://doi.org/10.5753/wei.2019.6656.

FERNANDES, Fabio Martins de. **Conexões com a Matemática 2**. São Paulo: Editora Moderna, 2013.

FERREIRA, F. P. **Combinatória no Ensino Médio: uma abordagem sem o uso de fórmulas**. 2013. 95f. Dissertação (Mestrado em Matemática em Rede Nacional). Universidade Federal do Vale do São Francisco. Juazeiro, Bahia. 2013

FERREIRA, Gessé Pereira. **Viabilidade do Aprendizado de modelagem Discreta como atividade extracurricular.** 2009. 94 f. Dissertação (Mestrado) - Universidade do Grande Rio – Prof. José de Souza Herdy, Mestrado em Ensino das Ciências na Educação Básica, Duque de Caxias, 2009. Disponível em:< http://www.unigranrio.br/unidades_adm/pro_reitorias/propep/stricto_sensu/cursos/mestrado/ensino_ciencias/galleries/downloads/dissertacoes/dissertacao_gesse_pereira.pdf>. Acesso em: 20 out. 2017.

FLEURY, A.; NEVEUX, C. **Hamon: "le code informatique à l'école dès septembre"**. Le Journal du Dimanche, 20 jun. 2017. Disponível em: http:// www.lejdd.fr/Societe/Hamon-Le-code-informatiqu-a-l-ecole-des-septembre-675912. Acesso em: 24 abr. 2019.

FLEURY, Gérard et al. **Graphes au lycée**. France: Institut de Recherche sur l'Enseignement des Mathématiques. Complexe Scientifique des Cézeaux. Clermont-Ferrand, 2004. Disponível em:< http://numerisation.irem.univ-mrs.fr/CF/ICF04005/ICF04005.pdf>. Acesso em: 21 out. 2017.

FORD, Lester R.; FULKERSON, Delbert R. Maximal flow through a network. **Canadian Journal of Mathematics**, Canada, v. 8, n. 3, p. 399-404, 1956. Disponível em:< https://cms.math.ca/openaccess/cjm/v8/cjm1956v08.0399-0404.pdf>. Acesso em: 10 nov. 2017.

FRANÇA, R. S.; AMARAL, H. J. C. Proposta metodológica de ensino e avaliação para o desenvolvimento do pensamento computacional com o uso do Scratch. In: WORKSHOP DE INFORMÁTICA NA ESCOLA, 19.; CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 2., 2013, Campinas. **Anais [...]**. Campinas, Edidora da SBC, 2013. p.179-188.

FRANCE. Ministère de l'Éducation Nationale. Éduscol. **Lumière**: Fiche nº 2753. 2014. Disponível em :<http://eduscol.education.fr/bd/urtic/maths/index.php?commande=aper&id=423>. Acesso em: 21 jan 2018.

FRANCE. Ministère de l'Éducation Nationale. Éduscol. **Matrices avec Xcas**: Fiche nº 2462. 2013. Disponível em:<http://eduscol.education.fr/bd/urtic/maths/index.php?commande=aper&id=423>. Acesso em: 21 jan 2018.

FRANCE. Ministère de l'Éducation Nationale. Éduscol. **Notion de Graphe** : Une approche en Terminale ES: Fiche nº 423. 2002. Disponível em: < http://eduscol.education.fr/bd/urtic/maths/index.php?commande=aper&id=423>. Acessado em: 21 jan 2018.

FRANCE. Ministère de l'Éducation Nationale. Éduscol. **Quatre documents sur les grades em terminale ES**: Fiche nº 2450. 2013. Disponível em:<http://eduscol.education.fr/bd/urtic/maths/index.php?commande=aper&id=423>. Acesso em: 21 jan 2018.

FRANCISCO, Carlos Alberto. O Modelo dos Campos Semânticos como Instrumento de Leitura da Prática Profissional do Professor de Matemática. In: ENCONTRO BRASILEIRO DE ESTUDANTES DE PÓS-GRAUAÇÃO EM EDUCAÇÃO MATEMÁTICA, 12, 2008, São Paulo. **Anais[...]**. São Paulo: Universidade Estadual Paulista, 2008. p. 1-18.

FUSEL, André Thiago. **O ensino e a aprendizagem da combinatória dentro do contexto de telefonia**. 2013. 135 f. Dissertação (Mestrado em Ciências Exatas e da Terra) - Universidade Federal de São Carlos, São Carlos, 2013.

GONÇALVES, Rafaela Ramos Soares. **Uma abordagem alternativa para o ensino de combinatória no ensino médio: a utilização do princípio multiplicativo e da resolução de problemas como ferramenta didático-pedagógica**. 2014. 111f. Dissertação (Mestrado em Matemática) – Mestrado Profissional em Matemática em Rede Nacional. Instituto Nacional de

Matemática Pura e Aplicada. Rio de Janeiro. 2014.

GUALANDI, Jorge Henrique. **Investigações matemáticas com grafos para o ensino médio.** 2012. 117 f. Dissertação (Mestrado) - Pontifícia Universidade Católica de Minas Gerais – Profª Maria Clara Rezende Frota, Mestrado em Ensino da Matemática, Belo Horizonte, 2012. Disponível em:< http://www.biblioteca.pucminas.br/teses/EnCiMat_GualandiJH_1.pdf.pdf>. Disponível em: 20 jan. 2018.

GUEDES, Victor Emanuel Pinto. **Uma abordagem para o ensino da teoria dos grafos no Ensino Médio**. 2014. 40f. Dissertação (Mestrado) - Universidade Federal de Juiz de Fora – Mestrado Profissional em Matemática em Rede Nacional, Juiz de Fora, 2014. Disponível em:<https://repositorio.ufjf.br/jspui/bitstream/ufjf/770/1/victoremanuelpintoguedes.pdf>. Acesso em: 22 set. 2017.

IEZZI, Gelson et al. **Matemática, ciência e aplicações, Volume 2**. 7. ed. São Paulo: Editora Saraiva, 2013.

ISTE-CSTA, International Society for Technology in Education, Computer Science Teachers Association. **Computational thinking: leadership toolkit**. 2011. Disponível em: www.iste.org/ docs/ ct-documents/ ct-leadershipt-toolkit.pdf. Acesso em: 23 abr. 2019.

Jacinto, Diego Suzano Ferreira. **Utilizando o material concreto para o ensino de combinatória**. 2015. [77 f.]. Dissertação(Programa de Pós-Graduação em Matemática em Rede Nacional) - Universidade Federal Rural do Rio de Janeiro, [Seropédica-RJ] . 2015

JONES, S. P. **Computing at school: international comparisons**. Swindon: Computing at School, 2011 Disponível em: https:// community.computingatschool.org.uk/ files/ 6710/ original.pdf. Acesso em: 28 abr. 2019.

JULIO, Rejane Siqueira. Produzindo significado para Uma Leitura da Produção de Significados Matemáticos e Não-matemáticos para Dimensão. **Revista do Programa de Pós-Graduação em Educação Matemátia da Universidade Federal de Mato Grosso do Sul (UFMS)**, Mato Grosso do Sul, v. 9, n. 20, p.501-525, 2016. Disponível em:< http://seer.ufms.br/index.php/pedmat/article/download/2885/2252>. Acesso em: 24 fev. 2018.

KAPUR, J. Nevin. Combinatorial analysis and school mathematics. **Educational Studies in Mathematics**, London, v.3, n.1, p. 111-127, 1970.

KENNETH, R. Chelst; EDWARDS, Thomas G., **¿Avanzará esta fila alguna vez? Aplicaciones de la Investigación de Operaciones**. Chile: Editorial Universitaria, 2008.

KOTSOPOULOS, Donna; FLOYD, Lisa.; NELSON, Vivian; Makosz, Samantha. **Mathematical or Computational Thinking? An Early Years Perspective**. In: ROBINSON K., OSANA H., KOTSOPOULOS D. (eds) MATHEMATICAL LEARNING AND COGNITION IN EARLY CHILDHOOD**: Proceedings [...]**. Cham: Springer, 2019, p. 79 - 90. Disponível em: < https://lisaannefloyd.com/publications-and-media/>. Acesso em: 2 out. 2020.

LEIBNIZ, Gottfried Wilhelm. **Logical papers**, Estados Unidos: Oxford. 1966. Acesso em: < https://www.worldcat.org/title/logical-papers/oclc/351520>. Acesso em: 2 nov. 2020.

LOVÁSZ, L., PELIKÁN, J., VESZTERGOMBI, K.; **Matemática Discreta**, [tradução de Ruy de Queiroz], Rio de Janeiro: Editora da SBM, 2005. Disponível em: < https://docplayer.com.br/107275175-Matematica-discreta-elementar-e-alem-l-lovasz-j-pelikan-e-k-vesztergombi.html>. Acesso em: 20 out. 2020.

LUGO, M. T.; ITHURBURU, V. Políticas digitales en América Latina. Tecnologías para fortalecer la educación de calidad. **Revista Iberoamericana de Educación**, Buenos Aires, v. 79, n. 1, p. 11-31, 2019. Disponível em: https://rieoei.org/RIE/article/view/3398. Acesso em 20 dez. 2020.

MACEDO, José Alexandre; VITALI, Maycon Maia. **Algoritmo de Dijkstra:** Estudo e Implementação. 2011. Disponível em:<http://claudiaboeres. pbworks.com/f/apresentacao-JoseAlexandre-e-Maycon.pdf> . Acesso em: 31 de abr. 2017.

MACIEL, Wagner da Silva. **Conceitos de combinatória e suas aplicações por meio de situações problemas**. 2015. 66f. Dissertação (Mestrado em Matemática) – Mestrado Profissional em Matemática em Rede Nacional. Universidade Federal do Mato Grosso do Sul. Campo Grande. Mato Grosso do Sul. 2015.

MALTA, Glaucia Helena Sarmento. **Grafos no Ensino Médio – Uma Inserção Possível**. 2008. 154f. Dissertação (Mestrado em Ensino de Matemática) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2016. Disponível em:< http://www.lume.ufrgs.br/bitstream/handle/10183/14829/000668628.pdf?sequence=1 >. Acesso em: 3 mar. 2018.

MARTIN, George E. **Counting: The art of enumerative combinatorics**. XII ed. United States of America: Springer Science & Business Media, 2001.

MATTA, A. E. R.; SILVA, F. DE P. S. DA; BOAVENTURA, E. M. Design-Based Research ou pesquisa de desenvolvimento: metodologia para pesquisa metodologia para pesquisa de desenvolvimento: metodologia para pesquisa aplicada de inovação em educação do século XXI. **Revista da FAEEBA - Educação e Contemporaneidade**, Bahia v. 23, n. 42, p.11, 2014. Disponível em: <https://www.revistas.uneb.br/index.php/faeeba/article/view/1025/705>. Acesso em: 30 out. 2020

MESQUITA, Daniel da Rosa. **Resolução de Problemas relacionados à Teoria de Grafos no Ensino Fundamental**. 2015. 97 f. Dissertação (Mestrado em Matemática) – Universidade Federal do Rio Grande do Sul – Profª. Marilaine de Fraga Sant'Ana. Mestrado em Ensino da Matemática, Porto Alegre, 2015. Disponível em:< http://www.lume.ufrgs.br/bitstream/handle/10183/132240/000983445.pdf?sequence=1>. Acesso em: 10 ago. 2017.

MIOTTO, Eder. **A combinatória e seu ensino**. 2014. 89f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional). Universidade Tecnológica Federal do Paraná. Curitiba, Paraná. 2014.

MODESTE, Simon. **Enseigner l'algorithme pour quoi? Quelles nouvelles questions pour les mathématiques? Quels apports pour l'apprentissage de la preuve?**. 2012. 253f. Tesis (Doctoral) - Université de Grenoble – Prof. Gravier Sylvain, Doctoral, Spécialité en Mathematique – Informatique, France, 2012. Disponível em:< https://tel.archives-ouvertes.fr/tel-00783294/document>. Acessado em: 13 dez. 2018.

MULLER, Jonathan Gil**. Teoria dos Grafos para o Ensino Fundamental: Desafios Lúdicos**. 2015. 185 f. Dissertação (Mestrado em Matemática) – Universidade Regional de Blumenau – Profª. Tânia Baier, Mestrado Profissional em Ensino de Ciências Naturais e Matemática, Blumenau, 2015. Disponível em:<http://www.bc.furb.br/docs/DS/2015/360431_1_1.pdf >. Acesso em; 2 out. 2017.

MUNIZ, Ivail Junior. **Encontrando, Minimizando e Planejando Percursos: uma Introdução à Teoria dos Grafos no Ensino Médio**. 2007. 146f. Dissertação (Mestrado) - Centro Federal de Educação Tecn. Celso Suckow da Fonseca – Prof. Samuel Jurkiewickz, Mestrado Profissionalizante em Ensino de Ciências e Matemática, Rio de Janeiro, 2007. Disponível em:< http://dippg.cefet-rj.br/index.php?option=com_docman&task=doc_details&gid=1565&Itemid=167 >. Acesso em: 8 jun. 2017.

MYKKÄNEN, J.; LIUKAS, L. **Koodi 2016: ensiapua ohjelmoinnin opettamissen peruskoulussa**. Helsink: Lönnberg, 2014. Disponível em: https:// s3-eu-west-1. amazonaws.com/ koodi2016/ Koodi2016_LR.pdf. Acesso em: 24 abr. 2019.

NETZ, Reviel; ACERBI, Fabio; WILSON, Nigel. **Towards a reconstruction of Archimedes' Stomachion**, **SCIAMVS**, Kyoto, v. 5, p. 67-99, 2004. Disponível em: <https://www.matetam.com/sites/default/files/ostomachion.pdf>. Acesso em 10 jan. 2021.

NEVES, Maria Augusta Ferreira; BOLINHAS, Sandra; FARIA, Luísa. P**reparação para o exame final nacional – Matemática aplicada às Ciências Sociais – 11º ANO**. Portugal: Porto Editora, 2016.

NOGUEIRA, Daniel Klug. **Introdução à Teoria dos Grafos: Proposta para o Ensino Médio**. 2015. 114 f. Dissertação (Mestrado) – Universidade de Brasília – Prof. Adail de Castro Cavalheiro, Mestrado Profissional em Matemática em Rede Nacional, Brasília, 2015. Disponível em:<http://repositorio.unb.br/bitstream/10482/19363/1/2015_DanielKlugNogueira.pdf>. Acesso em; 17 jul. 2017.

NOSCHANG, Luiz Fernando; PELZ, Filipi; DE JESUS, Elieser; RAABE, André. Portugol Studio: Uma IDE para Iniciantes em Programação. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 22., 2014, Brasília. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2014. p. 1-10. ISSN 2595-6175.

OLGIN, Clarissa de Assis. **Critérios, Possibilidades e Desafios para o Desenvolvimento de Temáticas no Currículo de Matemática do Ensino Médio**. 2015. 266 f. Tese (Doutorado em Ensino de Ciências e Matemática) - Universidade Luterana Do Brasil, Canoas, 2015. Disponível em:< https://www.researchgate.net/profile/Clarissa_De_Assis_Olgin/publication/321109187_Criter

ios_possibilidades_e_desafios_para_o_desenvolvimento_de_tematicas_no_Curriculo_de_Matematica_do_Ensino_Medio/links/5a0dcafc45851541b707993b/Criterios-possibilidades-e-desafios-para-o-desenvolvimento-de-tematicas-no-Curriculo-de-Matematica-do-Ensino-Medio.pdf>. Acesso em: 17 ago. 2017.

Oliveira, CARLOS ALBERTO LOPES DOS SANTOS DE. **Combinatória: Raciocínio recursivo e processos de enumeração.** 2015. 104f. Dissertação de Mestrado Profissional em Matemática em Rede Nacional. Universidade Estadual do Norte Fluminense Darcy Ribeiro – UENF, Campos dos Goytacazes, Rio de Janeiro, 2015.

OLIVEIRA, Gleisiani de Fátima et al. **Ensino de combinatória: como classificar problemas**. 2017. 67f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional). Universidade Federal de Viçosa, Minas Gerais, Viçosa, 2017.

OLIVEIRA, M. L. S. et al. Ensino de lógica de programação no ensino fundamental utilizando o Scratch: um relato de experiência. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO, 22.; CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 34., 2014, Brasília, DF. **Anais [...]** Porto Alegre: SBC, 2014. p. 1525-1534. Disponível em: <https://sol.sbc.org.br/index.php/wei/article/view/10978>. Acesso em: 20 out. 2020.

PAPERT, S. **Teaching children thinking**, Estados Unidos, v.9, n.5, p. 245 – 255, 1972. Disponível em: <https://statics-submarino.b2w.io/sherlock/books/firstChapter/1448098561.pdf>. Acesso em: 2 dez. 2020.

PAPERT, S.; RESNICK, M. **Technological fluency, and the representation of knowledge: proposal to the National Science Foundation**. Cambridge: MIT Media Laboratory, 1995.

PATIÑO Avendaño, Bibiana; CHARRY, Oscar Guillermo. **La enseñanza de la teoria de grafos como estratégia para desarollar processos de matematización.** 2013. 165f. Dissertação (Mestrado em Docência e Investigação Universitária) – Universidad Sergio Arboleda, Bogotá, 2013. Disponível em:< http://repository.usergioarboleda.edu.co/bitstream/handle/11232/844/La%20ense%C3%B1anza%20de%20la%20teor%C3%ADa%20de%20grafos%20como%20estrategia.%20procesos%20de%20matematizaci%C3%B3n.pdf?sequence=2&isAllowed=y>. Acesso em: 26 set. 2017.

PELAYO, Virginia. Razonamiento combinatório en alunos de secundaria. **Revista educación matemática**, España, v.8, n.1, p. 26-39, 1996. Disponível em:< http://www.revista-educacion-matematica.org.mx/descargas/Vol8/1/05Navarro.pdf >. Acesso em: 12 nov. 2018.

PERLIS, A. J. The computer in the university. In: GREENBERGER, M. (ED.). COMPUTERS AND THE WORLD OF THE FUTURE. **Proceedings [...]** Cambridge: MIT Press, 1962. p. 180-219. Disponível em: < https://sites.tufts.edu/devtech/files/2015/10/Relkin-thesis.pdf>. Acesso em: 20 nov. 2020.

PESSOA, C; BORBA, R. Do Young Children Notice what Combinatorial Situations Require? In: 36TH CONFERENCE OF THE INTERNATIONAL GROUP FOR THE PSYCHOLOGY OF MATHEMATICS EDUCATION. **Proceedings [...]** Taiwan: PME. 2012. p. 261. Disponível em:

<https://www.researchgate.net/profile/Dina_Hassidov/publication/3209>. Acesso em: 02 jan 2018.

PETERS, Jehu; METZ, Don. Using Graph Theory to Understand First Nations Connections. **Mathematics Teacher**, Nova Iorque, v. 109, n. 4, p. 311-313, 2015. Disponível em:< https://eric.ed.gov/?id=EJ1082084>. Acesso em: 08 out. 2017.

PEZETA, Jefferson Ricart. **Resolução de problemas em contextos de ensino de Matemática: uma abordagem por meio da Teoria dos Grafos**. 2013. 153 f. Dissertação (Mestrado Profissional em Educação Matemática) - Pontifícia Universidade Católica De São Paulo, São Paulo, 2013. Disponível em:< https://sapientia.pucsp.br/bitstream/handle/10976/1/Jefferson%20Ricart%20Pezeta.pdf>. Acesso em: 17 nov. 2017.

PIRES, Delma Erks. **Uma proposta de interdisciplinaridade utilizando combinatória e o algoritmo de colônia de formigas no ensino médio**. 2019. 58 f. Dissertação (Mestrado em Matemática em Rede Nacional) - Universidade Federal de Goiás, Jataí, 2019. Disponível em: < https://repositorio.bc.ufg.br/tede/handle/tede/10389>. Acesso em: 3 out. 2020.

PITOMBEIRA, João Bosco. Princípio da casa dos pombos. **Revista do Professor de matemática**, v. 8, n. 4, p.20, 1986. São Paulo: Sociedade Brasileira de Matemática, 1986**.** Disponível em: < https://rpm.org.br/cdrpm/8/4.htm>. Acesso em: 20 out. 2020.

POLYA, G. **A arte de resolver problemas: Um novo aspecto do método matemático**. Tradução: Heitor Lisboa de Araújo. Interciência, Rio de Janeiro, 1995. Disponível em: < http://educonse.com.br/2011/cdroom/eixo%206/PDF/Microsoft%20Word%20-%20ARTE%20DE%20RESOLVER%20PROBLEMAS.pdf>. Acesso em: 5 out. 2020.

QUINN, Anne. Using apps to visualize graph theory. **Mathematics Teacher**, Nova Iorque, v. 108, n. 8, p. 626-631, 2015. Disponível em:< https://www.jstor.org/stable/10.5951/mathteacher.108.8.0626 >. Acesso em: 26 out. 2017.

RAABE, A.; ZORZO, A. F.; BLIKSTEIN, P. **Computação na Educação Básica: Fundamentos e Experiências**. Porto Alegre: Penso Editora, 2020.

RAABE, André Luís Alice. Pensamento Computacional na Educação: Para todos, por todos! **Revista Computação Brasil, SBC**, São Paulo, v. 10, n. 8, p. 54 - 63, 2017. Disponível em: < https://www.sbc.org.br/images/flippingbook/computacaobrasil/computa_34/pdf/cb_34_2017.pdf>. Acesso em: 8 out. 2020

RAABE, André; Zorzo, Avelino F.; Blikstein, Paulo. **Computação na Educação Básica: Fundamentos e Experiências (Tecnologia e Inovação na Educação Brasileira)** (p. 49, 115, 118). Penso Editora. Edição do Kindle. 2020.

RAMOS, F. O.; TEIXEIRA, L. S. Significação da aprendizagem através do pensamento computacional no ensino médio: uma experiência com Scratch. In: WORKSHOP DE INFORMÁTICA NA ESCOLA, 21.; CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 4., 2015, Maceió. **Anais [...]** Porto Alegre: SBC, 2015. P. 217-226.

RESNICK, M. **Learn to code, code to learn.** Portland. Edsurge. 2013. Disponível em: <https://www.edsurge.com/news/2013-05-08-learn-to-code-code-to-learn>. Acesso em: 12 nov. 2019.

RESNICK, M. Point of view: reviving Papert's dream. **Educational Technology**, Estados Unidos, v. 52, n. 4, p. 42-46, 2012. Disponível em: < https://web.media.mit.edu/~mres/papers/educational-technology-2012.pdf>. Acesso em: 9 nov. 2020.

RIBAS, Giovani Batista. **Aplicações de grafos no ensino médio**. 2019. 89 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) - Universidade Federal de Ouro Preto, Ouro Preto, 2019. Disponível em: <https://www.repositorio.ufop.br/bitstream/123456789/11917/1/DISSERTA%C3%87%C3%83O_Aplica%C3%A7%C3%B5esGrafosEnsino.pdf>. Acesso em: 2 nov. 2020.

RIBEIRO, Igor Schmidke. **COMBS - Um software de apoio ao ensino da combinatória**. 2013. Universidade Estadual de Santa Cruz. 43f. Dissertação (Mestrado Profissionalizante em Matemática)- Universidade Estadual de Santa Cruz, Ilhéus-BA, 2013. Disponível em: < http://www.biblioteca.uesc.br/biblioteca/bdtd/201160272D.pdf>. Acesso em 8 nov. 2020.

RIBEIRO, Pedro. **Distâncias Mínimas**. Porto: Universidade do Porto, Departamento de Ciência de Computadores, 2015. Disponível em:<http://www.dcc.fc.up.pt/~pribeiro/aulas/daa1415/slides/8_distancias_06122014.pdf>. Acesso em: 31 de março de 2018.

RIBEIRO, Pedro. **Fluxo Máximo**. Departamento de Ciência de Computadores. Universidade do Porto, 2015. Disponível em:<http://www.dcc.fc.up.pt/~pribeiro/aulas/daa1415/slides/9_fluxo_14122014.pdf>. Acessado em 01 de abril de 2018.

SA, Lauro Chagas e. **Construção e utilização de maquete eletrônica para ensino de grafos: aprendizagens discentes a partir de uma abordagem histórico-investigativa**. 2016. 150 f. Dissertação (Mestrado Profissional em Educação em Ciências e Matemática) – Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo, Espírito Santo, 2016. Disponível em:< https://www.15snhct.sbhc.org.br/resources/anais/12/1474041409_ARQUIVO_ArtigoSNHCT.pdf>. Acesso em: 02 dez. 2017.

SCOTLAND. Education. **Curriculum for excellence: technologies: experiences and outcomes**. Livingston: Education Scotland, 2009. Disponível em:<https://education.gov.scot/education-scotland/scottish-education-system/policy-for-scottish-education/policy-drivers/cfe-building-from-the-statement-appendix-incl-btc1-5/experiences-and-outcomes/>. Acesso em: 2 ago. 2017.

SEIBERT, Lucas Gabriel. **Uma proposta para o desenvolvimento da competência de "observar com sentido" na formação inicial de professores de matemática**. 2013. 101f. Dissertação (Mestrado em Ensino de Ciências e Matemática) – Universidade Luterana do Brasil, Canoas, 2013. Disponível em:< http://www.academia.edu/12870866/UMA_PROPOSTA_PARA_O_DESENVOLVIMENTO_DA_COMPET%C3%8ANCIA_DE_OBSERVAR_COM_SENTIDO_NA_FORMA%C3%8

7%C3%83O_INICIAL_DE_PROFESSORES_DE_MA_TEM%C3%81TICA >. Acesso em: 17 de ago. 2017.

SERRAZINA, Lurdes; OLIVEIRA, Isolina. O professor como investigador: Leitura crítica de investigações em educação matemática. In: XII SEMINÁRIO DE INVESTIGAÇÃO EM EDUCAÇÃO MATEMÁTICA, 12, 2014, Vila Real. **Actas[...]** Vila Real: Associação de Professores de Matemática. 2014. p. 29-55. Disponível em:<http://apm.pt/files/127552_gti2002_art_pp283-308_49c771bcc0338.pdf>. Acesso em: 07 jul. 2017.

SHUTE, V. J.; SUN, C.; ASBELL-CLARKE, J. Demystifying computational thinking. **Educational Research Review**, Estados Unidos, v. 22, n. 1, p. 142–158, 2017. Disponível em: < https://www.learntechlib.org/p/204418>. Acesso em: 3 nov. 2020.

SILVA B, Juliano Thadeo Alves da. **Pensamento computacional no ensino da Matemática: desafios e possibilidades**. 2020. 78f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) – Universidade Federal de Mato Grosso, Cuiabá, 2020.

SILVA, Alex Sandro Vaz. **Atividades escolares que envolvem a combinatória, a partir da expectativa do desenvolvimento da habilidade de contagem, segundo a BNCC**. 2020. 90 f. Dissertação (Mestrado em Matemática em Rede Nacional) – Universidade Federal de São Paulo, São Jose dos Campos, 2020. Disponível em: <https://sca.profmat-sbm.org.br/sca_v2/get_tcc3.php?id=170870231>. Acesso em: 06 jan. 2021.

SILVA, BRUNO CESAR SA DA. **Emparelhamento em grafos bipartidos no ensino médio**. 2016. 43 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) - Associação Instituto Nacional De Matemática Pura E Aplicada, Rio de Janeiro, 2016. Disponível em:< https://impa.br/wp-content/uploads/2016/12/TCCBrunoCesar.pdf>. Acesso em: 26 set. 2017.

SILVA, Carina Brunehilde Pinto da. **Combinatória: concentrando o ensino na resolução de problemas**. 2013. 52 f. Dissertação (Mestrado em Matemática em Rede Nacional) – Centro de Ciências, Universidade Federal do Ceará, Fortaleza, 2013. Disponível em: <http://www.repositorio.ufc.br/handle/riufc/5293>. Acesso em: 24 out. 2020.

SILVA, José Gabriel Gouveia. **Uma abordagem histórica e combinatória do Stomachion**. (2020). Graduação. (Monografia em Matemática) - Universidade Federal da Paraíba, Paraíba, 2020. Disponível em: < https://repositorio.ufpb.br/jspui/bitstream/123456789/17468/1/JGGS13052020.pdf>. Acesso em: 10 ago. 2019.

SILVA, Liliana Mota Cardoso Marques da. **A Teoria dos Grafos no Ensino**. 2009. 137f. Dissertação (Mestrado em Matemática/Educação) - Universidade Portucalense Infante D. Henrique, Portugal, 2009. Disponível em: <http://repositorio.uportu.pt:8080/bitstream/11328/529/2/TMMAT%20113.pdf>. Acesso em: 26 ago. 2017.

SILVA, V.; SOUZA, A.; MORAIS, D. Pensamento Computacional: um relato de práticas pedagógicas para o ensino de computação em escolas públicas. **Revista Tecnologias na Educação**, Minas Gerais, v. 16, n. 16, p. 1 - 22, 2016. Disponível em: < http://tecedu.pro.br/wp-

content/uploads/2016/09/Art5-Pensamento-computacional-Um-relato-de-pr%C3%A1ticas-pedag%C3%B3gicas....pdf>. Acesso em: 3 jun. 2020.

SILVEIRA, D. S.; LOIOLA, E. M.; FERREIRA, S. B. L. Uma metodologia de ensino de lógica aplicada em cursos de ciências humanas. **Revista de Administração Mackenzie**, Minas Gerais, v. 10, n. 2, p. 164-180, 2009. Disponível em: < https://www.scielo.br/scielo.php?script=sci_arttext&pid=S1678-69712009000200008>. Acesso em 4 jun 2020.

SMOLE, Katia Stocco. DINIZ. Maria Ignez. **Matemática ensino médio 2.** 8. ed. São Paulo: Editora Saraiva, 2013.

SOUZA, Joamir. GARCIA, Jacqueline da Silva Ribeiro. **# Contato matemática 2**. São Paulo: FTD, 2016.

SOUZA, Renato Ferreira de. **Resolução de problemas via teoria de grafos**. 2015. 48 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) – Universidade de São Paulo/São Carlos, São Paulo, 2015. Disponível em:<http://www.teses.usp.br/teses/disponiveis/55/55136/tde-06072015-103319/publico/Dissertacao_RenatoFdeSouza_Revisada.pdf>. Acesso em: 26 de dez. 2017.

STAGER, G.; MARTINEZ, S. Thirteen considerations for teaching coding to children. In: HUMBLE, S. (ORG.). CREATING THE CODING GENERATION IN PRIMARY SCHOOLS: A PRACTICAL GUIDE FOR CROSS-CURRICULAR TEACHING. **Proceedings […]** Abingdon: Routledge, 2017. Disponível em: < https://www.taylorfrancis.com/books/edit/10.4324/9781315545813/creating-coding-generation-primary-schools-steve-humble>. Acesso em: 30 out. 2020

TAVEIRA, Roberto Campos Lima et al. **Ensino da matemática por meio de problemas clássicos de otimização combinatória**. 2019. 65f. Dissertação (Programa de Mestrado Profissional em Matemática em Rede Nacional) - Universidade Federal do Triângulo Mineiro, Uberaba, Minas Gerais, 2019. Disponível em: <http://bdtd.uftm.edu.br/handle/tede/939>. Acesso em: 3 out. 2020.

TEIXEIRA, Bea Karla Flores Machado. **Teoria dos grafos a partir do ensino médio: uma abordagem no espectro do modelo dos campos semânticos**. 2015. 173f. Dissertação (Mestrado Profissional em Educação em Ciências e Matemática) – Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo, Vitória, 2015. Disponível em: <http://educimat.ifes.edu.br/images/stories/Publica%C3%A7%C3%B5es/Disserta%C3%A7%C3%B5es/2015_Bea_Karla_Flores_Machado_Teixeira.pdf>. Acesso em: 20 jul. 2017.

TROCADO, Nathália López. **Uma proposta didática para um curso básico de combinatória**. 2017. 133 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) – Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2017. Disponível em:<http://www.bdtd.uerj.br/tde_busca/arquivo.php?codArquivo=11575>. Acesso em: 3 jan. 2018.

VALENTE, J. A. A espiral da aprendizagem e as tecnologias da informação e comunicação: repensando conceitos. In: JOLY, M.C. (ED.) TECNOLOGIA NO ENSINO: IMPLICAÇÕES

PARA A APRENDIZAGEM. **Anais [...]** São Paulo: Casa do Psicólogo, 2002. p.15-37. Disponível em: <http://repositorio.unicamp.br/bitstream/REPOSIP/284458/1/Valente_JoseArmando_LD.pdf> .Acesso em: 3 out. 2020.

VALENTE, J. A. integração do pensamento computacional no currículo da educação básica: diferentes estratégias usadas e questões de formação de professores e avaliação do aluno. **E-Curriculum**, São Paulo, v. 14, n. 3, p. 864-897, 2016. Disponível em: < https://revistas.pucsp.br/index.php/curriculum/article/view/29051>. Acesso em: 3 nov. 2020.

VALENTE, J. A. **O professor no ambiente Logo: formação e atuação**. Campinas: Unicamp/ NIED, 1996

VAN de WALLE, J.A., KARP, K. and BAY-WILLIAMS, J.M. **Elementary and Middle School Mathematics: Teaching Developmentally**.Allyn & Bacon, 2010.

VEIRAS, Augusto Fornari. **O caso Count Van Diamond**. Santa Catarina: Universidade Federal de Santa Catarina, 2015. Disponível em: <http://www.inf.ufsc.br/grafos/ problema/assassino/index.html>. Acesso em: 2 de novembro de 2017.

VIEL, F.; RAABE, A. L. A.; ZEFERINO, C. A. Introdução a programação e à implementação de processadores por estudantes do ensino médio. In: WORKSHOP DE INFORMÁTICA NA ESCOLA, 20.; CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 3., 2014, Dourados. **Anais [...]** Porto Alegre: SBC, 2014. p. 248-257. Disponível em: <http://www.ufrgs.br/tri/sead/events/3o-congresso-brasileiro-de-informatica-na-educacao>. Acesso em: 3 nov. 2020.

WEINBERG, Aaron; WIESNER, Emilie; FUKAWA-CONNELLY, Tim. Mathematics lectures as narratives: Insights from network graph methodology. **Educational Studies in Mathematics**, Estados Unidos, v. 91, n. 2, p. 203-226, 2016. Disponível em:< https://link.springer.com/content/pdf/10.1007%2Fs10649-015-9663-6.pdf >. Acesso em: 2 jan. 2018.

WILSON, R.J. **Introduction to Graph Theory**. England, UK: Addison Wesley Longman Limited, 1996.

WING, J. Computational thinking. **Communications of ACM**, Estados Unidos, v.49, n. 3, p. 33-36, 2006. Disponível em: < https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf>. Acesso em: 8 jan. 2021.

WING, J. M. Computational thinking: what and why?. **The magazine of Carnegie Mellon University's School of Computer Science**, Estados Unidos, 2011. Disponível em: <https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why>. Acesso em: 3 jun. 2020.

APÊNDICE A – LISTA DOS LINKS COM AS 24 IMPLEMENTAÇÕES

Nº	Links
1	https://onlinegdb.com/NNOy1ncG5
2	https://onlinegdb.com/VmZu3Pre4
3	https://onlinegdb.com/xcN65VdFw
4	https://onlinegdb.com/2qY5ktoMa
5	https://onlinegdb.com/N8OJNxRQr
6	https://onlinegdb.com/HkCMlhZeO
7	https://onlinegdb.com/BJxQme2-eu
8	https://onlinegdb.com/BkFQxhWlO
9	https://onlinegdb.com/HJb4g2beu
10	https://onlinegdb.com/r1U4x2Wxd
11	https://onlinegdb.com/B1oNe2be_
12	https://onlinegdb.com/SokTxW2F-6
13	https://onlinegdb.com/Sk8SxhZxd
14	https://onlinegdb.com/Syirgnbld
15	https://onlinegdb.com/BylIenZlO
16	https://onlinegdb.com/2Z_N61Ws1
17	https://onlinegdb.com/lUR6dTJSCK
18	https://onlinegdb.com/PXswBNQin
19	https://onlinegdb.com/htho0CY9w
20	https://onlinegdb.com/INf-zZgcOC
21	https://onlinegdb.com/BkNRxBJbO
22	https://onlinegdb.com/SJkBZSJZu
23	https://onlinegdb.com/B1rw-HyWu
24	https://onlinegdb.com/S1HbepVld

APÊNDICE B – HABILIDADES DA BNCC – CONEXÕES ENTRE COMBINATÓRIA E PENSAMENTO COMPUTACIONAL

∞ (EF01MA02) Contar de maneira exata ou aproximada, utilizando diferentes estratégias como o pareamento e outros agrupamentos.

∞ (EF01MA11) Descrever a localização de pessoas e de objetos no espaço em relação à sua própria posição, utilizando termos como à direita, à esquerda, em frente, atrás.

∞ (EF01MA12) Descrever a localização de pessoas e de objetos no espaço segundo um dado ponto de referência, compreendendo que, para a utilização de termos que se referem à posição, como direita, esquerda, em cima, em baixo, é necessário explicitar-se o referencial.

∞ (EF01MA20) Classificar eventos envolvendo o acaso, tais como "acontecerá com certeza", "talvez aconteça" e "é impossível acontecer", em situações do cotidiano.

∞ (EF04MA06) Resolver e elaborar problemas envolvendo diferentes significados da multiplicação (adição de parcelas iguais, organização retangular e proporcionalidade), utilizando estratégias diversas, como cálculo por estimativa, cálculo mental e algoritmos.

∞ (EF04MA28) Realizar pesquisa envolvendo variáveis categóricas e numéricas e organizar dados coletados por meio de tabelas e gráficos de colunas simples ou agrupadas, com e sem uso de tecnologias digitais.

∞ (EF05MA09) Resolver e elaborar problemas simples de contagem envolvendo o princípio multiplicativo, como a determinação do número de agrupamentos possíveis ao se combinar cada elemento de uma coleção com todos os elementos de outra coleção, por meio de diagramas de árvore ou por tabela

∞ (EF05MA25) Realizar pesquisa envolvendo variáveis categóricas e numéricas, organizar dados coletados por meio de tabelas, gráficos de colunas, pictóricos e de linhas, com e sem uso de tecnologias digitais, e apresentar texto escrito sobre a finalidade da pesquisa e a síntese dos resultados

∞ (EF06MA03) Resolver e elaborar problemas que envolvam cálculos (mentais ou escritos, exatos ou aproximados) com números naturais, por meio de estratégias variadas, com compreensão dos processos neles envolvidos com e sem uso de calculadora

∞ (EF06MA04) Construir algoritmo em linguagem natural e representá-lo por fluxograma que indique a resolução de um problema simples (por exemplo, se um número natural qualquer é par).

∞ (EF06MA23) Construir algoritmo para resolver situações passo a passo (como na construção de dobraduras ou na indicação de deslocamento de um objeto no plano segundo pontos de referência e distâncias fornecidas etc.).

∞ (EF06MA34) Interpretar e desenvolver fluxogramas simples, identificando as relações entre os objetos representados (por exemplo, posição de cidades considerando as estradas que as unem, hierarquia dos funcionários de uma empresa etc.).

∞ (EF06MA33) Planejar e coletar dados de pesquisa referente a práticas sociais escolhidas pelos alunos e fazer uso de planilhas eletrônicas para registro, representação e interpretação das informações, em tabelas, vários tipos de gráficos e texto.

∞ (EF07MA05) Resolver um mesmo problema utilizando diferentes algoritmos.

∞ (EF07MA06) Reconhecer que as resoluções de um grupo de problemas que têm a mesma estrutura, podem ser obtidas utilizando os mesmos procedimentos.

∞ (EF07MA07) Representar por meio de um fluxograma os passos utilizados para resolver um grupo de problemas.

∞ (EF08MA11) Identificar a regularidade de uma sequência numérica recursiva e construir um algoritmo por meio de um fluxograma que permita indicar os números seguintes.

∞ (EM13MAT106) Identificar situações da vida cotidiana nas quais seja necessário fazer escolhas levando-se em conta os riscos probabilísticos (usar este ou aquele método contraceptivo, optar por um tratamento médico em detrimento de outro etc.)

∞ (EM13MAT310) Resolver e elaborar problemas de contagem envolvendo agrupamentos ordenáveis ou não de elementos, por meio dos princípios multiplicativo e aditivo,

recorrendo a estratégias diversas, como o diagrama de árvore.

∞ (EM13MAT315) Investigar e registrar, por meio de um fluxograma, quando possível, um algoritmo que resolve um problema.

∞ (EM13MAT507) Identificar e associar progressões aritméticas (PA) a funções afins de domínios discretos, para análise de propriedades, dedução de algumas fórmulas e resolução de problemas.

∞ (EM13MAT508) Identificar e associar progressões geométricas (PG) a funções exponenciais de domínios discretos, para análise de propriedades, dedução de algumas fórmulas e resolução de problemas.

∞ (EM13MAT405) Utilizar conceitos iniciais de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática.

APÊNDICE C – HABILIDADES DA BNCC – CONEXÕES ENTRE MATEMÁTICA E PENSAMENTO COMPUTACIONAL

- (EF07MA26) Descrever, por escrito e por meio de um fluxograma, um algoritmo para a construção de um triângulo qualquer, conhecidas as medidas dos três lados e
- (EF07MA28) Descrever, por escrito e por meio de um fluxograma, um algoritmo para a construção de um polígono regular (como quadrado e triângulo equilátero), conhecida a medida de seu lado e
- (EF08MA16) Descrever, por escrito e por meio de um fluxograma, um algoritmo para a construção de um hexágono regular de qualquer área, a partir da medida do ângulo central e da utilização de esquadros e compasso.
- (EF09MA15) Descrever, por escrito e por meio de um fluxograma, um algoritmo para a construção de um polígono regular cuja medida do lado é conhecida, utilizando régua e compasso, como também softwares.
- (EM13MAT403) Analisar e estabelecer relações, com ou sem apoio de tecnologias digitais, entre as representações de funções exponencial e logarítmica expressas em tabelas e em plano cartesiano, para identificar as características fundamentais (domínio, imagem, crescimento) de cada função.