Alexandre Ferreira Gomes
Diego de Souza Nicodemos
José Wilson Coura Pinto

LIGHTS OUT EM GRAFOS: MANUAL DE USO DO PRODUTO EDUCACIONAL

Rio de Janeiro, 2026

LIGHTS OUT EM GRAFOS: MANUAL DE USO DO PRODUTO EDUCACIONAL

COLÉGIO PEDRO II

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

BIBLIOTECA PROFESSORA SILVIA BECHER

CATALOGAÇÃO NA FONTE

G633 Gomes, Alexandre Ferreira
O jogo lights out em grafos caminhos, ciclos e sólidos de Platão / Alexandre Ferreira Gomes; Diego de Souza Nicodemos; José Wilson Coura Pinto. – Rio de Janeiro: Imperial Editora, 2026.

113f.

Bibliografia: p. 14.

ISBN: 978-65-5930-388-5

1. Matemática - Estudo e ensino. 2. Jogos educativos. 3. Aprendizagem. 4.Educação básica. 5. Ensino fundamental. I. Nicodemos, Diego de Souza. II. Pinto, José Wilson Coura. III. Colégio Pedro II. IV. Título.

CDD 510

Ficha catalográfica elaborada pela Bibliotecária Simone Alves – CRB-7: 5026.

Alexandre Ferreira Gomes
Diego de Souza Nicodemos
José Wilson Coura Pinto

LIGHTS OUT EM GRAFOS: MANUAL DE USO DO PRODUTO EDUCACIONAL

1ª Edição

IMPERIAL editora

Rio de Janeiro, 2026

RESUMO

Este produto educacional apresenta o manual de uso do jogo digital interativo Lights Out em Grafos, desenvolvido no âmbito do Mestrado Profissional em Matemática em Rede Nacional (PROFMAT) do Colégio Pedro II. O recurso reúne grafos caminho, grafos ciclo e os grafos dos cinco sólidos de Platão, permitindo explorar, de forma visual e interativa, relações de adjacência, mudanças de estado, estratégias de resolução e comparação do número de movimentos. Nos caminhos e ciclos, o usuário pode escolher a quantidade de vértices e o número de estados; nos sólidos de Platão, o produto disponibiliza versões com dois e três estados. A cada acionamento, o estado do vértice selecionado e o de seus vizinhos é alterado ciclicamente, e o objetivo consiste em conduzir todos os vértices ao estado-alvo. O manual descreve as funções da interface, orienta a navegação, apresenta possibilidades de exploração das diferentes famílias de grafos e propõe articulações com habilidades da Base Nacional Comum Curricular do 6º ano do Ensino Fundamental ao 3º ano do Ensino Médio. Também são apresentadas uma sequência didática, orientações ao professor e o código-fonte da versão utilizada, favorecendo sua consulta e adaptação. O produto foi concebido como recurso para apoiar atividades de resolução de problemas, reconhecimento de padrões, raciocínio lógico, elaboração de estratégias, investigação e argumentação matemática.

Palavras-chave: Lights Out; teoria de grafos; jogos digitais; ensino de matemática; resolução de problemas.

SUMÁRIO

APRESENTAÇÃO....6
1 COMO NAVEGAR PELO JOGO....7
2 EXPLORAÇÃO POR FAMÍLIA DE GRAFOS....8
2.1 Caminhos....8
2.2 Ciclos....9
2.3 Sólidos de Platão....10
3 POSSIBILIDADES DE ARTICULAÇÃO COM A BNCC....10
3.1 6º ano do Ensino Fundamental....10
3.2 7º ano do Ensino Fundamental....11
3.3 8º ano do Ensino Fundamental....11
3.4 9º ano do Ensino Fundamental....11
3.5 1º ano do Ensino Médio....11
3.6 2º ano do Ensino Médio....12
3.7 3º ano do Ensino Médio....12
4 SUGESTÃO DE SEQUÊNCIA DIDÁTICA....12
5 ORIENTAÇÕES AO PROFESSOR....13
REFERÊNCIAS....14
APÊNDICE A – CÓDIGO-FONTE DA VERSÃO ATUAL DO PRODUTO EDUCACIONAL....15

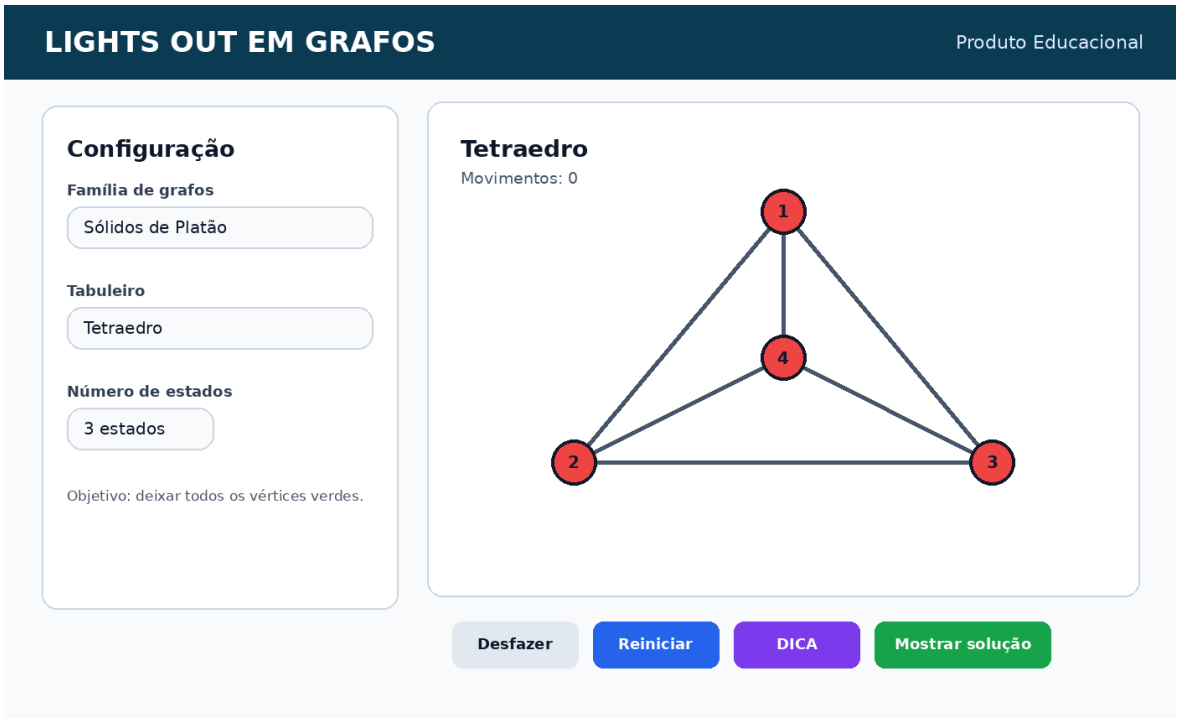
APRESENTAÇÃO

Este manual apresenta o jogo digital interativo desenvolvido como produto educacional da dissertação O Jogo Lights Out em Grafos: Caminhos, Ciclos e Sólidos de Platão. O ambiente reúne grafos caminho, grafos ciclo e os grafos dos cinco sólidos de Platão. Nos caminhos e ciclos, o usuário pode escolher a quantidade de vértices e o número de estados; nos sólidos de Platão, o produto oferece versões com dois e três estados.

A regra central é simples: ao acionar um vértice, o estado desse vértice e o de todos os seus vizinhos é alterado ciclicamente. Na versão de três estados, as cores seguem vermelho → amarelo → verde → vermelho. O objetivo é levar todos os vértices ao estado-alvo, procurando também comparar estratégias e reduzir o número de movimentos.

O produto foi concebido para favorecer resolução de problemas, reconhecimento de padrões, raciocínio lógico, elaboração de estratégias, investigação e argumentação matemática. As possibilidades didáticas descritas neste manual são propostas de uso; não correspondem a resultados de aplicação empírica com turmas.

Figura 1 – Visão geral da interface da versão atual do produto educacional.



Fonte: Os autores, 2026.

1 COMO NAVEGAR PELO JOGO

A tela foi organizada para que o professor ou estudante escolha primeiro a família de grafos, depois o tabuleiro e, quando disponível, a quantidade de estados. O tabuleiro ocupa a região central e os comandos ficam próximos ao grafo.

1. Escolher a família

No campo “Família de grafos”, selecione Sólidos de Platão, Ciclos ou Caminhos.

2. Escolher o tabuleiro

Selecione o sólido desejado ou, nos caminhos e ciclos, a quantidade de vértices que será explorada.

3. Definir os estados

Nos caminhos e ciclos, escolha a quantidade de estados. Nos sólidos de Platão, utilize as versões de 2 ou 3 estados.

4. Jogar

Clique ou toque em um vértice. O próprio vértice e todos os vértices adjacentes mudam de estado.

5. Acompanhar movimentos

O contador registra o número de acionamentos realizados, permitindo comparar estratégias.

6. Desfazer

O botão “Desfazer” retorna uma jogada, sendo útil para testar hipóteses sem reiniciar a partida.

7. Reiniciar

O botão “Reiniciar” restaura a configuração inicial do tabuleiro.

8. Pedir uma dica

O botão “DICA” sugere um vértice para continuar a resolução, preservando o caráter investigativo.

9. Mostrar solução

O botão “Mostrar solução” exibe uma solução possível; na configuração inicial, a interface identifica a solução mínima quando essa informação está disponível no produto.

Figura 2 – Comandos de navegação e área de solução.



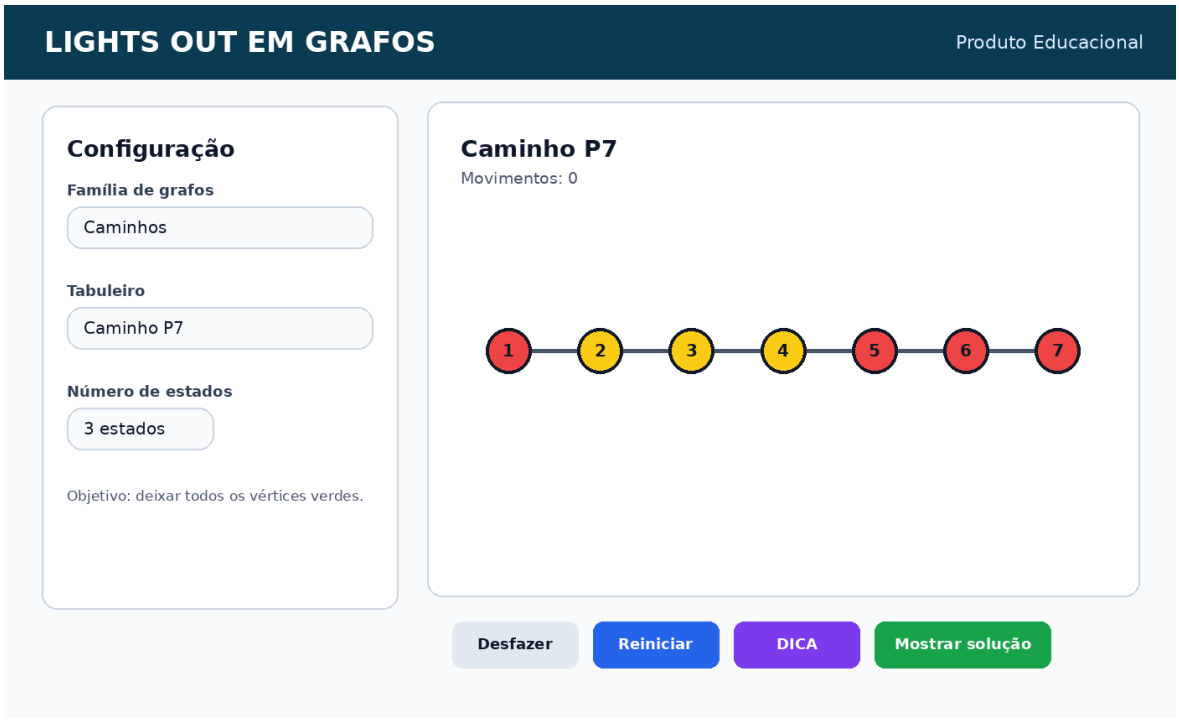
Fonte: Os autores, 2026.

2 EXPLORAÇÃO POR FAMÍLIA DE GRAFOS

2.1 Caminhos

Os caminhos são especialmente adequados para observar propagação local, sequências de acionamentos e regularidades. O professor pode começar com poucos vértices e aumentar gradualmente o tamanho do grafo, pedindo que os estudantes registrem quais vértices foram acionados e comparem estratégias.

Figura 3 – Exemplo de exploração de um grafo caminho na interface atual.

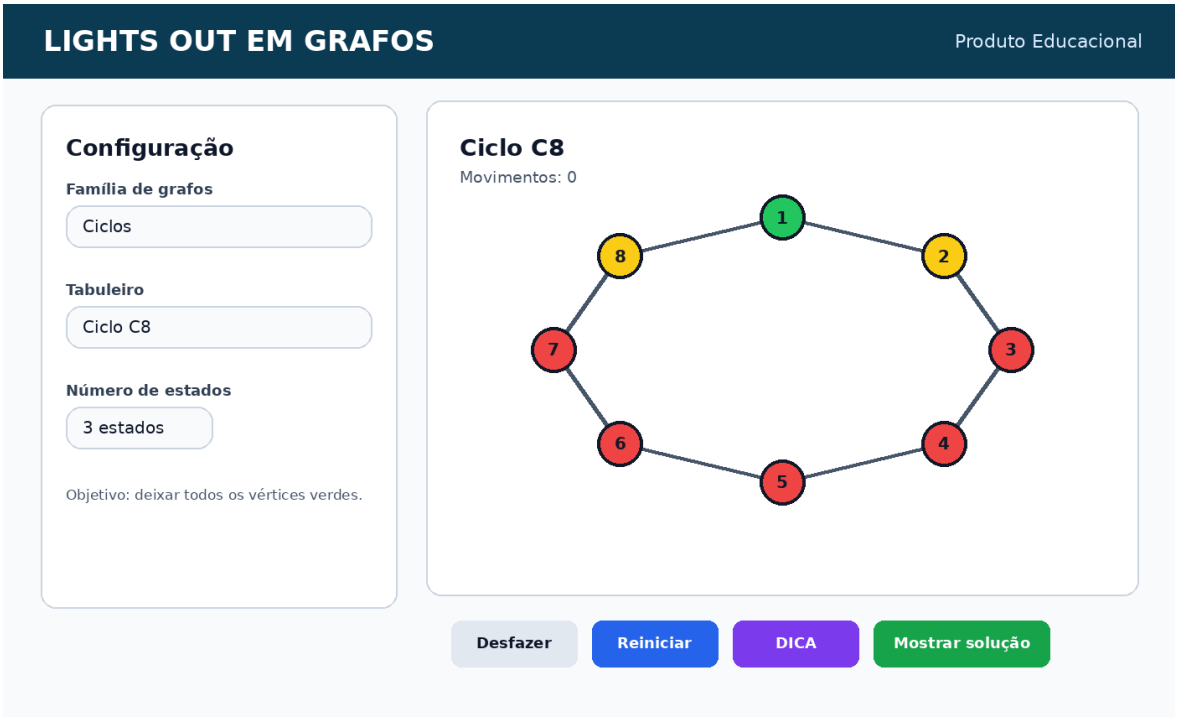


Fonte: Os autores, 2026.

2.2 Ciclos

Nos ciclos, a ligação entre o primeiro e o último vértice altera a dinâmica observada nos caminhos. Isso permite discutir padrões, possibilidade ou impossibilidade de determinadas transformações e comparação entre ciclos com diferentes números de vértices.

Figura 4 – Exemplo de exploração de um grafo ciclo na interface atual.



Fonte: Os autores, 2026.

2.3 Sólidos de Platão

Os cinco sólidos de Platão permitem relacionar o jogo a vértices, arestas, vizinhanças, regularidade e visualização geométrica. No produto, tetraedro, hexaedro, octaedro, dodecaedro e icosaedro podem ser explorados com dois ou três estados.

3 POSSIBILIDADES DE ARTICULAÇÃO COM A BNCC

O Lights Out e a Teoria de Grafos não são apresentados aqui como conteúdos obrigatórios da BNCC. O jogo é utilizado como recurso para mobilizar habilidades previstas no currículo. A seleção abaixo combina as habilidades já relacionadas na dissertação com outras habilidades oficiais particularmente compatíveis com algoritmos, regularidades, geometria e resolução de problemas.

3.1 6º ano do Ensino Fundamental

Quadro 1 – Articulação com a BNCC: 6º ano do Ensino Fundamental

Habilidades	EF06MA17; EF06MA23
Foco	Vértices, faces e arestas; construção de algoritmos passo a passo.
Sugestão de uso	Explorar os sólidos de Platão, contar vértices/arestas, identificar vizinhos e pedir que cada grupo escreva um procedimento de resolução para um caminho pequeno.

Fonte: Elaborado pelos autores, 2026.

3.2 7º ano do Ensino Fundamental

Quadro 2 – Articulação com a BNCC: 7º ano do Ensino Fundamental

Habilidades	EF07MA04; EF07MA05; EF07MA07; EF07MA14; EF07MA15
Foco	Operações com inteiros; diferentes algoritmos; fluxogramas; sequências recursivas; expressão de regularidades.
Sugestão de uso	Codificar as cores por 0, 1 e 2, comparar duas estratégias para o mesmo tabuleiro, registrar o algoritmo de jogadas e investigar padrões em P_n.

Fonte: Elaborado pelos autores, 2026.

3.3 8º ano do Ensino Fundamental

Quadro 3 – Articulação com a BNCC: 8º ano do Ensino Fundamental

Habilidades	EF08MA10; EF08MA11
Foco	Regularidades em sequências recursivas e não recursivas e construção de algoritmos/fluxogramas.
Sugestão de uso	Registrar as sequências de acionamentos em caminhos de tamanhos sucessivos e construir um fluxograma que descreva uma estratégia observada.

Fonte: Elaborado pelos autores, 2026.

3.4 9º ano do Ensino Fundamental

Quadro 4 – Articulação com a BNCC: 9º ano do Ensino Fundamental

Habilidades	EF09MA15
Foco	Descrição escrita e por fluxograma de um algoritmo, com uso de software.
Sugestão de uso	Transformar uma estratégia de jogo em algoritmo formal: entrada, decisões, acionamentos e condição de término; comparar eficiência pelo número de movimentos.

Fonte: Elaborado pelos autores, 2026.

3.5 1º ano do Ensino Médio

Quadro 5 – Articulação com a BNCC: 1º ano do Ensino Médio

Habilidades	EM13MAT315
Foco	Reconhecimento, formulação e solução de problemas algorítmicos.
Sugestão de uso	Formalizar uma estratégia para caminhos ou ciclos como algoritmo e discutir por que ela funciona em exemplos escolhidos.

Fonte: Elaborado pelos autores, 2026.

3.6 2º ano do Ensino Médio

Quadro 6 – Articulação com a BNCC: 2º ano do Ensino Médio

Habilidades	EM13MAT310; EM13MAT315
Foco	Problemas de contagem e estratégias algorítmicas.
Sugestão de uso	Contar possibilidades de acionamento em tabuleiros pequenos, comparar estratégias, estudar número de movimentos e introduzir a ideia de solução mínima.

Fonte: Elaborado pelos autores, 2026.

3.7 3º ano do Ensino Médio

Quadro 7 – Articulação com a BNCC: 3º ano do Ensino Médio

Habilidades	EM13MAT310; EM13MAT315
Foco	Contagem, algoritmos e argumentação sobre estratégias.
Sugestão de uso	Aprofundar a modelagem: representar estados numericamente, discutir aritmética modular, grafos e, conforme a turma, vetores/matrizes; justificar existência de solução e minimalidade em casos acessíveis.

Fonte: Elaborado pelos autores, 2026.

Observação sobre o Ensino Médio

Na BNCC, códigos iniciados por EM13 abrangem o bloco das três séries do Ensino Médio, e não uma série isolada. A separação em 1º, 2º e 3º ano feita neste manual é, portanto, uma proposta de progressão pedagógica: parte de algoritmos e estratégias, avança para contagem e comparação e culmina em uma formalização matemática mais profunda.

4 SUGESTÃO DE SEQUÊNCIA DIDÁTICA

1. Jogar

Permitir uma primeira exploração sem fornecer a solução.

2. Registrar

Anotar vértices acionados, estados observados e quantidade de movimentos.

3. Comparar

Colocar lado a lado estratégias de diferentes estudantes ou grupos.

4. Conjecturar

Identificar padrões: repetições, periodicidades, vizinhanças e casos que parecem mais difíceis.

5. Formalizar

Adequar ao ano: fluxograma, sequência, codificação 0-1-2, aritmética modular, grafo ou matriz.

6. Justificar

Pedir que o estudante explique por que sua estratégia funciona e, quando apropriado, por que não há outra com menos movimentos.

5 ORIENTAÇÕES AO PROFESSOR

O botão de solução não precisa ser utilizado imediatamente. Para preservar o caráter investigativo, recomenda-se que o estudante primeiro jogue, registre tentativas e compare estratégias. A dica pode funcionar como intervenção intermediária. A solução completa pode ser usada depois para análise, conferência e discussão sobre eficiência.

Em turmas do Ensino Fundamental, a linguagem pode permanecer concreta: cores, pontos, ligações e vizinhos. A formalização em termos de grafos, congruências, vetores e matrizes pode ser introduzida progressivamente, conforme os objetivos da aula e o nível da turma.

Atividades em duplas ou pequenos grupos são particularmente adequadas: um estudante pode operar o jogo enquanto outro registra a sequência de movimentos; depois, os papéis são invertidos e as estratégias são comparadas.

REFERÊNCIAS

BRASIL. Ministério da Educação. Base Nacional Comum Curricular. Brasília, DF: Ministério da Educação, 2018.

GOMES, Alexandre Ferreira. O jogo Lights Out em grafos caminhos, ciclos e sólidos de Platão. 2026. Dissertação (Mestrado Profissional em Matemática) – Colégio Pedro II, Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura, Rio de Janeiro, 2026.

Observação: as habilidades da BNCC foram selecionadas como possibilidades de articulação curricular com o produto, e não como afirmação de que Teoria de Grafos ou Lights Out sejam conteúdos obrigatórios da Base.

APÊNDICE A – CÓDIGO-FONTE DA VERSÃO ATUAL DO PRODUTO EDUCACIONAL

A seguir é reproduzido o arquivo HTML completo utilizado nesta versão do produto educacional. Por ser um arquivo autocontido, ele pode ser salvo com extensão .html e aberto em um navegador moderno. O código pode também servir de base para adaptação em outras plataformas, observadas as características técnicas do ambiente de destino.

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta
name="description"
content="Lights Out em grafos com 2 ou 3 estados nos Sólidos de Platão e quantidade livre de
estados em Ciclos e Caminhos."
>
<title>Lights Out em Grafos — estados variáveis</title>
<style>
:root {
--bg1: #eef4ff;
--bg2: #f9fbff;
--panel: rgba(255,255,255,0.88);
--panel-solid: #ffffff;
--text: #172033;
--muted: #667085;
--border: #d7dce5;
--accent: #2563eb;
--accent-dark: #1d4ed8;
--accent-soft: #dbeafe;
--hint: #7c3aed;
--success: #16a34a;
--shadow: 0 20px 45px rgba(37, 99, 235, 0.10);
}
* { box-sizing: border-box; }
body {
margin: 0;
min-height: 100vh;
font-family: Arial, Helvetica, sans-serif;
color: var(--text);
background:
radial-gradient(circle at top left, #dbeafe 0%, transparent 35%),
radial-gradient(circle at bottom right, #e9d5ff 0%, transparent 25%),
linear-gradient(180deg, var(--bg1), var(--bg2));
}
body.modal-open { overflow: hidden; }
.screen {
width: min(1120px, calc(100% - 24px));
margin: 18px auto;
padding: 18px;
border: 1px solid rgba(255,255,255,0.45);
border-radius: 28px;
background: rgba(255,255,255,0.34);
backdrop-filter: blur(14px);
box-shadow: var(--shadow);
}
.hero {
padding: 20px 16px 8px;
text-align: center;
}
.hero h1 {
margin: 0 0 10px;
font-size: clamp(1.9rem, 4vw, 2.7rem);
line-height: 1.1;
}
.hero p {
max-width: 860px;
margin: 0 auto;
color: var(--muted);
line-height: 1.55;
```

```
font-size: 1rem;
}
.family-grid {
display: grid;
grid-template-columns: repeat(3, minmax(0, 1fr));
gap: 16px;
margin-top: 22px;
}
.family-card {
padding: 18px;
border: 1px solid #dbe3f2;
border-radius: 22px;
background: var(--panel-solid);
box-shadow: 0 10px 28px rgba(15, 23, 42, 0.06);
cursor: pointer;
transition: transform 180ms ease, box-shadow 180ms ease, border-color 180ms ease;
text-align: left;
}
.family-card:hover {
transform: translateY(-3px);
box-shadow: 0 18px 36px rgba(37, 99, 235, 0.12);
border-color: #9dc1ff;
}
.family-card.active {
border-color: var(--accent);

background: linear-gradient(180deg, #ffffff, #f3f7ff);
box-shadow: 0 18px 36px rgba(37, 99, 235, 0.17);
}
.family-badge {
display: inline-flex;
align-items: center;
gap: 8px;
margin-bottom: 12px;
padding: 7px 12px;
border-radius: 999px;
background: var(--accent-soft);
color: var(--accent-dark);
font-size: 0.88rem;
font-weight: 700;
}
.family-card h3 {
margin: 0 0 8px;
font-size: 1.2rem;
}
.family-card p {
margin: 0;
color: var(--muted);
line-height: 1.5;
font-size: 0.96rem;
}
.game-panel {
margin-top: 22px;
padding: 18px;
border: 1px solid rgba(255,255,255,0.56);
border-radius: 24px;
background: rgba(255,255,255,0.76);
box-shadow: 0 18px 36px rgba(15, 23, 42, 0.06);
}
.topbar {
display: flex;
flex-wrap: wrap;
align-items: center;
justify-content: space-between;
gap: 10px 14px;
margin-bottom: 16px;
}
.title-group h2 {
margin: 0 0 4px;
font-size: 1.35rem;
}
.title-group p {
margin: 0;
color: var(--muted);
font-size: 0.95rem;
}
.pill {
```

```
display: inline-flex;
align-items: center;
gap: 8px;
padding: 9px 14px;
border-radius: 999px;
border: 1px solid #dbe3f2;
background: #fff;
font-size: 0.92rem;
font-weight: 700;
}
.controls {
display: grid;
grid-template-columns: minmax(0, 1.05fr) minmax(0, 1.05fr) minmax(180px, 0.55fr) auto;
gap: 12px;
align-items: end;
margin-bottom: 14px;
}
label {
display: block;
font-weight: 700;
}
select,
input[type="number"],
button {
font: inherit;
}
select,
input[type="number"] {
width: 100%;
margin-top: 6px;
padding: 12px 13px;
border: 1px solid var(--border);
border-radius: 14px;
color: var(--text);
background: #fff;
}
.state-help {
display: block;
margin-top: 6px;
color: var(--muted);
font-size: 0.80rem;
font-weight: 400;
line-height: 1.35;
}
.button-row {
display: flex;
flex-wrap: wrap;
gap: 8px;
}
button {
min-height: 46px;
padding: 10px 16px;

border: 1px solid var(--border);
border-radius: 14px;
cursor: pointer;
color: var(--text);
background: #fff;
transition: border-color 160ms ease, transform 140ms ease, box-shadow 160ms ease;
}
button:hover:not(:disabled) {
border-color: var(--accent);
transform: translateY(-1px);
box-shadow: 0 8px 20px rgba(37, 99, 235, 0.08);
}
button:focus-visible,
select:focus-visible,
input[type="number"]:focus-visible,
.family-card:focus-visible {
outline: 3px solid rgba(37, 99, 235, 0.26);
outline-offset: 2px;
}
button.primary {
color: #ffffff;
border-color: var(--accent);
background: linear-gradient(180deg, var(--accent), var(--accent-dark));
}
```

```
button.secondary {
color: #fff;
border-color: #7c3aed;
background: linear-gradient(180deg, #8b5cf6, #7c3aed);
}
button.success {
color: #fff;
border-color: #16a34a;
background: linear-gradient(180deg, #22c55e, #16a34a);
}
button:disabled {
opacity: 0.48;
cursor: not-allowed;
transform: none;
box-shadow: none;
}
.stats {
display: grid;
grid-template-columns: repeat(5, minmax(0, 1fr));
gap: 12px;
margin-bottom: 14px;
}
.stat {
padding: 14px;
border: 1px solid #e0e7f1;
border-radius: 18px;
background: #fff;
box-shadow: 0 8px 18px rgba(15, 23, 42, 0.04);
}
.stat-label {
display: block;
margin-bottom: 4px;
color: var(--muted);
font-size: 0.84rem;
}
.stat-value {
font-weight: 700;
font-size: 1.07rem;
}
.notice,
.rules {
margin-bottom: 14px;
padding: 14px 16px;
border-radius: 16px;
}
.notice {
border-left: 5px solid var(--accent);
background: #eaf2ff;
}
.rules {
border: 1px solid #dbe3f2;
background: #fff;
line-height: 1.6;
}
.legend {
display: flex;
flex-wrap: wrap;
gap: 10px 16px;
margin-bottom: 14px;
}
.legend-item {
display: flex;
align-items: center;
gap: 7px;
font-size: 0.9rem;
padding: 6px 10px;
border-radius: 999px;
background: rgba(255,255,255,0.82);
border: 1px solid #e3eaf5;
}
.legend-dot {
width: 18px;
height: 18px;
flex: 0 0 auto;
border: 2px solid #111827;
border-radius: 50%;
}
```

```
.board-wrap {

overflow: auto;
padding: 10px;
border: 1px solid #dbe3f2;
border-radius: 22px;
background: linear-gradient(180deg, #ffffff, #f7faff);
box-shadow: inset 0 1px 0 rgba(255,255,255,0.7);
}
#board {
display: block;
width: 100%;
height: auto;
color: #111827;
user-select: none;
}
.edge {
stroke: currentColor;
stroke-width: 4;
stroke-linecap: round;
}
.vertex {
cursor: pointer;
outline: none;
}
.hit-area { fill: transparent; }
.focus-ring {
fill: none;
stroke: var(--accent);
stroke-width: 4;
opacity: 0;
}
.vertex:focus .focus-ring { opacity: 1; }
.node {
stroke: #111827;
stroke-width: 3;
transition: r 140ms ease, fill 140ms ease;
}
.vertex:hover .node,
.vertex:focus .node {
r: 18;
stroke-width: 4;
}
.vertex.hint .focus-ring {
opacity: 1;
stroke: var(--hint);
stroke-width: 6;
animation: hintPulse 900ms ease-in-out infinite alternate;
}
.vertex.hint .node {
stroke: var(--hint);
stroke-width: 5;
}
.state-number {
font-family: Arial, Helvetica, sans-serif;
font-size: 14px;
font-weight: 700;
dominant-baseline: middle;
text-anchor: middle;
pointer-events: none;
}
.vertex-label {
fill: currentColor;
font-family: Arial, Helvetica, sans-serif;
font-size: 18px;
font-weight: 700;
pointer-events: none;
}
.message-row {
display: flex;
align-items: center;
justify-content: space-between;
gap: 12px;
margin-top: 12px;
}
#message {
min-height: 24px;
```

```
margin: 0;
color: var(--muted);
flex: 1 1 auto;
}
.mini-actions {
display: flex;
flex-wrap: wrap;
gap: 8px;
}
.footer {
margin-top: 16px;
color: var(--muted);
font-size: 0.85rem;
text-align: center;
}
.overlay {
position: fixed;
inset: 0;
z-index: 1000;
display: none;
align-items: center;
justify-content: center;
padding: 20px;
background: rgba(15, 23, 42, 0.68);
}
.overlay.visible { display: flex; }

.dialog {
width: min(100%, 560px);
max-height: min(88vh, 780px);
overflow: auto;
padding: 26px;
border: 1px solid var(--border);
border-radius: 22px;
text-align: left;
background: #ffffff;
box-shadow: 0 24px 70px rgba(15, 23, 42, 0.32);
}
.dialog.centered { text-align: center; }
.dialog-icon {
display: grid;
width: 76px;
height: 76px;
margin: 0 auto 14px;
place-items: center;
border-radius: 50%;
font-size: 2.6rem;
background: #dcfce7;
}
.dialog h2 {
margin: 0 0 10px;
font-size: 1.65rem;
}
.dialog p {
margin: 0 0 14px;
color: var(--muted);
line-height: 1.55;
}
.dialog-actions {
display: flex;
flex-wrap: wrap;
justify-content: center;
gap: 9px;
margin-top: 10px;
}
.dialog-actions button {
flex: 1 1 150px;
}
.solution-box {
padding: 14px 16px;
border: 1px solid #dbe3f2;
border-radius: 16px;
background: #f8fbff;
margin: 12px 0;
line-height: 1.7;
}
.solution-list {
```

```
margin: 8px 0 0;
padding-left: 18px;
}
.solution-list li {
margin: 3px 0;
}
.solution-summary {
margin-top: 12px;
padding: 12px 14px;
border-radius: 14px;
background: #f0fdf4;
border: 1px solid #bbf7d0;
color: #166534;
}
@keyframes hintPulse {
from { r: 22; opacity: 0.65; }
to { r: 29; opacity: 1; }
}
@media (max-width: 980px) {
.family-grid {
grid-template-columns: 1fr;
}
.controls {
grid-template-columns: 1fr 1fr;
}
.controls > label:nth-child(3) {
grid-column: 1 / span 1;
}
.button-row {
grid-column: 1 / -1;
}
.stats {
grid-template-columns: repeat(2, minmax(0, 1fr));
}
.message-row {
flex-direction: column;
align-items: stretch;
}
}
@media (max-width: 620px) {
.screen {
width: calc(100% - 14px);
margin: 10px auto;
padding: 10px;
border-radius: 22px;
}
.controls,
.stats {
grid-template-columns: 1fr;

}
.button-row button,
.mini-actions button {
flex: 1 1 125px;
}
}
</style>
<style>
:root {
--navy: #12213f;
--navy-light: #1d315d;
--surface: #ffffff;
--page: #eef1f6;
}
body {
background:
linear-gradient(180deg, #e8edf5 0, #f5f7fb 240px, #f5f7fb 100%);
}
.screen {
width: min(1120px, calc(100% - 28px));
margin: 20px auto;
padding: 0;
overflow: hidden;
border: 1px solid #d9e0eb;
border-radius: 24px;
background: var(--surface);
box-shadow: 0 22px 55px rgba(18, 33, 63, 0.12);
```

```
backdrop-filter: none;
}
.app-header {
display: flex;
align-items: center;
gap: 18px;
padding: 24px 28px;
color: #ffffff;
background:
radial-gradient(circle at 88% 18%, rgba(255,255,255,0.13), transparent 24%),
linear-gradient(135deg, var(--navy), var(--navy-light));
}
.app-symbol {
display: grid;
width: 64px;
height: 64px;
flex: 0 0 auto;
place-items: center;
border: 2px solid rgba(255,255,255,0.75);
border-radius: 18px;
font-size: 1.15rem;
font-weight: 700;
letter-spacing: 0.08em;
background: rgba(255,255,255,0.08);
}
.app-header h1 {
margin: 2px 0 6px;
font-size: clamp(1.65rem, 4vw, 2.35rem);
line-height: 1.08;
}
.app-header p {
margin: 0;
color: rgba(255,255,255,0.78);
line-height: 1.45;
}
.app-kicker {
font-size: 0.76rem;
font-weight: 700;
letter-spacing: 0.16em;
color: #a9c7ff !important;
}
.game-panel {
margin: 0;
padding: 24px 28px 28px;
border: 0;
border-radius: 0;
background: #ffffff;
box-shadow: none;
}
.topbar {
padding-bottom: 14px;
border-bottom: 1px solid #e3e8f0;
}
.title-group h2 {
color: var(--navy);
}
.pill {
color: var(--navy);
border-color: #cdd8e8;
background: #f4f7fb;
}
.controls {
margin-top: 18px;
padding: 16px;
border: 1px solid #dfe5ee;
border-radius: 18px;
background: #f7f9fc;
}
select,
input[type="number"] {
border-color: #ccd5e2;
border-radius: 12px;
background: #ffffff;
}
button {

border-radius: 12px;
```

```
}
.stats {
margin-top: 16px;
}
.stat {
border-color: #e0e6ef;
border-radius: 14px;
box-shadow: none;
background: #ffffff;
}
.notice {
border-left-color: var(--navy-light);
background: #eef3fb;
}
.rules {
background: #fbfcfe;
}
.board-wrap {
border-color: #d9e1ec;
border-radius: 18px;
background: #ffffff;
box-shadow: none;
}
.legend-item {
border-color: #dfe5ee;
background: #ffffff;
}
.message-row {
padding: 12px 2px 0;
border-top: 1px solid #eef1f5;
}
@media (max-width: 620px) {
.screen {
width: calc(100% - 12px);
margin: 6px auto;
border-radius: 18px;
}
.app-header {
align-items: flex-start;
padding: 20px 16px;
}
.app-symbol {
width: 52px;
height: 52px;
border-radius: 14px;
}
.game-panel {
padding: 18px 12px 22px;
}
.controls {
padding: 12px;
}
}
</style>
<style>
.board-actions {
display: flex;
flex-wrap: wrap;
gap: 10px;
justify-content: center;
margin-top: 14px;
margin-bottom: 6px;
padding: 12px;
border: 1px solid #dfe5ee;
border-radius: 16px;
background: #f7f9fc;
}
.board-actions button {
min-width: 170px;
font-weight: 700;
}
.message-row {
display: block;
padding-top: 10px;
}
#message {
margin: 0;
```

```
}
@media (max-width: 620px) {
.board-actions {
justify-content: stretch;
}
.board-actions button {
flex: 1 1 100%;
min-width: 0;
}
}
</style>
<style>
.button-row {
display: none !important;
}
.board-actions {
justify-content: center;
align-items: center;

gap: 10px;
}
.board-actions button {
min-width: 165px;
}
@media (max-width: 620px) {
.board-actions button {
flex: 1 1 100%;
}
}
</style>
<style>
/* A solução fica abaixo do tabuleiro, sem cobrir o grafo. */
#solutionOverlay {
position: static;
inset: auto;
z-index: auto;
display: none;
padding: 0;
margin: 12px 0 0;
background: transparent;
}
#solutionOverlay.visible {
display: block;
}
#solutionOverlay .dialog {
width: 100%;
max-height: none;
overflow: visible;
margin: 0;
padding: 18px;
border-radius: 16px;
box-shadow: none;
background: #ffffff;
}
#solutionOverlay .dialog-actions {
justify-content: flex-end;
}
#solutionOverlay .dialog-actions button {
flex: 0 0 auto;
min-width: 130px;
}
</style>
<style>
.state-change-panel {
margin-top: 14px;
padding: 14px;
border: 1px solid #dfe5ee;
border-radius: 16px;
background: #f7f9fc;
}
.state-change-panel h3 {
margin: 0 0 4px;
color: var(--navy);
font-size: 1rem;
}
.state-change-panel p {
margin: 0 0 10px;
```

```
color: var(--muted);
font-size: 0.86rem;
}
.state-change-grid {
display: grid;
grid-template-columns: repeat(auto-fit, minmax(78px, 1fr));
gap: 8px;
}
.state-change-item {
padding: 9px 8px;
border: 1px solid #dbe3f2;
border-radius: 12px;
background: #ffffff;
text-align: center;
}
.state-change-vertex {
display: block;
margin-bottom: 3px;
color: var(--muted);
font-size: 0.78rem;
font-weight: 700;
}
.state-change-value {
display: block;
color: var(--text);
font-size: 1rem;
font-weight: 700;
}
</style>
</head>
<body>
<main class="screen">
<header class="app-header">
<div class="app-symbol" aria-hidden="true">LO</div>
<div>
<p class="app-kicker">PROJETO INTERATIVO</p>
<h1>Lights Out em Grafos</h1>
<p>
Sólidos de Platão, ciclos e caminhos em um único jogo, com
quantidade variável de estados.
</p>
</div>
</header>

<section class="game-panel">
<div class="topbar">
<div class="title-group">
<h2 id="panelTitle">Sólidos de Platão</h2>
<p id="panelSubtitle">Escolha o tabuleiro, a quantidade de estados e comece a jogar.</p>
</div>
<div class="pill" id="currentModePill">Modo atual: Sólidos de Platão</div>
</div>
<section class="controls" aria-label="Controles do jogo">
<label for="familySelect">
Tipo de tabuleiro
<select id="familySelect">
<option value="solids">Sólidos de Platão</option>
<option value="cycles">Ciclos</option>
<option value="paths">Caminhos</option>
</select>
</label>
<label id="boardControlLabel">
<span id="boardControlTitle">Escolha o tabuleiro</span>
<select id="boardSelect"></select>
<input id="boardSizeInput" type="number" min="3" step="1" value="3" inputmode="numeric" hidden
aria-describedby="boardSizeHelp">
<span id="boardSizeHelp" class="state-help" hidden></span>
</label>
<label for="stateCountSelect">
Quantidade de estados
<input id="stateCountSelect" type="number" min="2" step="1" value="3" inputmode="numeric"
aria-describedby="stateCountHelp">
<span id="stateCountHelp" class="state-help">Nos sólidos de Platão: 2 ou 3 estados. Em
caminhos e ciclos: qualquer inteiro a partir de
2.</span>
</label>
<div class="button-row">
```

```
</div>
</section>
<section class="stats" aria-label="Informações da partida">
<div class="stat">
<span class="stat-label">Família</span>
<span id="familyName" class="stat-value">Sólidos de Platão</span>
</div>
<div class="stat">
<span class="stat-label">Tabuleiro atual</span>
<span id="boardName" class="stat-value">Tetraedro</span>
</div>
<div class="stat">
<span class="stat-label">Estados</span>
<span id="stateCountValue" class="stat-value">3</span>
</div>
<div class="stat">
<span class="stat-label">Jogadas</span>
<span id="moves" class="stat-value">0</span>
</div>
<div class="stat">
<span class="stat-label">Vértices no estado final</span>
<span id="greenCount" class="stat-value">0/4</span>
</div>
</section>
<div id="solvabilityNotice" class="notice">
Esta configuração inicial possui solução.
</div>
<div id="legend" class="legend" aria-label="Legenda dos estados"></div>
<div class="board-wrap">
<svg
id="board"
viewBox="0 0 700 560"
role="img"
aria-label="Tabuleiro interativo do jogo Lights Out"
>
<g id="edges"></g>
<g id="vertices"></g>
</svg>
</div>
<section class="state-change-panel" aria-label="Contador de mudanças de estado por vértice">
<h3>Contador de mudanças de estado</h3>
<p>Quantidade de vezes que cada vértice teve seu estado alterado.</p>
<div id="stateChangeCounter" class="state-change-grid"></div>
</section>
<div class="board-actions">
<button id="undoBtn" type="button">Desfazer</button>
<button id="resetBtn" class="primary" type="button">Reiniciar</button>
<button id="hintBtn" class="secondary" type="button">DICA</button>
<button id="solutionBtn" class="success" type="button">Mostrar solução</button>
</div>
<!-- Solução exibida abaixo do grafo para não sobrepor o tabuleiro. -->
<div id="solutionOverlay" class="overlay" aria-hidden="true">
<section class="dialog" role="dialog" aria-modal="true" aria-labelledby="solutionTitle" aria-
describedby="solutionDescription">
<h2 id="solutionTitle">Solução mínima</h2>
<p id="solutionDescription">
Abaixo está uma sequência mínima de acionamentos, calculada a partir da configuração inicial,
que leva todos os vértices ao estado
final.
</p>
<div id="solutionContent" class="solution-box"></div>
<div class="dialog-actions">
<button id="closeSolutionBtn" type="button">Fechar</button>
</div>
</section>
</div>
<div class="message-row">
<p id="message" aria-live="polite">Todos os vértices começam no estado vermelho.</p>
</div>
</section>

</main>
<div id="victoryOverlay" class="overlay" aria-hidden="true">
<section class="dialog centered" role="dialog" aria-modal="true" aria-
labelledby="victoryTitle" aria-describedby="victoryMessage">
<div class="dialog-icon" aria-hidden="true">■</div>
<h2 id="victoryTitle">Parabéns!</h2>
```

Você levou todos os vértices ao estado final.

<button id="closeVictoryBtn" type="button">Ver tabuleiro</button>
<button id="playAgainBtn" class="primary" type="button">Jogar novamente</button>

</section>

<script>
const boards = {"solid:tetra":{"family":"solids","familyName":"Sólidos de
Platão","name":"Tetraedro","positions":{"1":[350,65],"2":[115,470],
"3":[585,470],"4":[350,300]},"adjacency":{"1":[2,3,4],"2":[1,3,4],"3":[1,2,4],"4":
[1,2,3]},"snf":{"d":[1,0,0,0],"s":[[1,0,0,0],[-1,0,0,1],[-
1,0,1,0],[-1,1,0,0]],"t":[[1,-1,-1,-1],[0,0,0,1],[0,0,1,0],[0,1,0,0]]}},"solid:hexa":
{"family":"solids","familyName":"Sólidos de Platão","name":"
Hexaedro","positions":{"1":[115,75],"2":[585,75],"3":[585,485],"4":[115,485],"5":
[245,185],"6":[455,185],"7":[455,375],"8":[245,375]},"adjacency"
:{"1":[2,4,5],"2":[1,3,6],"3":[2,4,7],"4":[1,3,8],"5":[1,6,8],"6":[2,5,7],"7":[3,6,8],"8":
[4,5,7]},"snf":{"d":[1,1,1,1,2,0,0,0],"s":[[1,0,0,0,0,0
,0,0],[0,0,1,0,0,0,0,0],[-1,1,0,0,0,0,0,0],[0,-1,1,0,1,0,0,0],[-1,-1,2,-1,3,0,0,0],[0,1,-1,0,-
1,0,0,1],[1,0,-1,0,-1,0,1,0],[0,0,-1,1,-
1,1,0,0]],"t":[[1,-1,1,1,-1,0,1,0],[0,1,-1,-2,1,1,0,0],[0,0,1,1,0,-1,-1,-1],[0,0,0,1,-
1,0,0,1],[0,0,0,0,1,-1,-1,-
1],[0,0,0,0,0,0,0,1],[0,0,0,0,0,0,1,0],[0,0,0,0,0,1,0,0]]}},"solid:octa":
{"family":"solids","familyName":"Sólidos de Platão","name":"Octaedro","p
ositions":{"1":[364,49],"2":[634,499],"3":[93,499],"4":[290,281],"5":[436,281],"6":
[363,400]},"labelPositions":{"1":[364,22,"middle"],"2":[650,50
7,"start"],"3":[79,507,"end"],"4":[276,273,"end"],"5":[450,273,"start"],"6":
[363,432,"middle"]},"adjacency":{"1":[2,3,4,5],"2":[1,3,5,6],"3":[1,2
,4,6],"4":[1,3,5,6],"5":[1,2,4,6],"6":[2,3,4,5]},"snf":{"d":[1,1,1,1,1,5],"s":[[1,0,0,0,0,0],
[1,0,0,-1,0,0],[1,0,0,0,-1,0],[1,-1,0,0,0,0],[1,0,-
1,0,0,0],[-4,1,1,1,1,1]],"t":[[1,-1,-1,-1,-1,-
4],[0,1,0,0,0,1],[0,0,1,0,0,1],[0,0,0,1,0,1],[0,0,0,0,1,1],[0,0,0,0,0,1]]}},"solid:dodeca":
{"family":"solids","familyName":"Sólidos de Platão","n
ame":"Dodecaedro","positions":{"1":[342,35],"2":[614,225],"3":[510,515],"4":[190,515],"5":
[70,225],"6":[342,120],"7":[424,168],"8":[515,230],"9":
[485,350],"10":[455,455],"11":[342,455],"12":[235,455],"13":[205,348],"14":[175,230],"15":
[258,168],"16":[290,225],"17":[394,225],"18":[427,330],
"19":[342,390],"20":[258,330]},"adjacency":{"1":[2,5,6],"2":[1,3,8],"3":[2,4,10],"4":
[3,5,12],"5":[1,4,14],"6":[1,7,15],"7":[6,8,17],"8":[2,7,9],
"9":[8,10,18],"10":[3,9,11],"11":[10,12,19],"12":[4,11,13],"13":[12,14,20],"14":
[5,13,15],"15":[6,14,16],"16":[15,17,20],"17":[7,16,18],"18":[9,1
7,19],"19":[11,18,20],"20":[13,16,19]},"snf":{"d":
[1,1,1,1,1,1,1,1,1,1,1,1,1,2,4,4,4,4,16],"s":[[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[-1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[1,-
1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[1,-1,0,1,-
1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0],[-1,2,-1,-1,2,-
3,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0],[1,1,-3,2,-3,1,-
2,1,0,0,0,4,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0],[0,0,0,-1,1,-1,1,0,-
1,1,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0],[1,-4,2,-11,21,-18,17,1,-
14,13,-1,-12,24,-
11,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0],[1449,-965,241,-965,1447,-
1931,2654,-725,-964,1689,-966,-723,2654,-1931,-
2172,2,2,0,0,0,0],[0,2,-1,1,-2,0,-4,-1,3,-2,0,2,-3,1,4,-5,5,0,0,0],[2,-4,0,-1,3,-1,4,2,-2,5,-3,-
2,6,-4,-5,10,-5,-5,0,0],[-1,2,0,0,-1,0,-1,-1,0,-
2,2,1,-3,2,2,-4,2,3,-1,0],[-2,4,-1,-1,-1,-1,-1,-1,-2,-2,5,3,-7,4,4,-7,3,5,-6,0],[1,-
3,1,1,1,1,1,1,1,1,-3,-3,5,-3,-3,5,-3,-3,5,1]],"t":[[1,-
1,1,1,-2,3,-1,1,-1,-4,8,18,-14,-1348550,-447,864002,2806844,6477332,-1079555,1],[0,1,-1,-1,1,-
2,1,-3,3,12,-24,-53,41,3949108,1309,-2530152,-
8219590,-18968284,3161380,-3],[0,0,1,0,1,-1,0,1,-1,-4,8,17,-13,-1252009,-
415,802149,2605905,6013627,-1002271,1],[0,0,0,1,-2,3,-1,2,-2,-9,17,38,-
30,-2893196,-959,1853640,6021837,13896547,-2316091,1],[0,0,0,0,1,-2,1,-3,3,13,-25,-
56,44,4244762,1407,-2719574,-8834958,-
20388365,3398061,1],[0,0,0,0,0,1,-1,5,-5,-21,41,91,-71,-6845320,-
2269,4385724,14247704,32879317,-5479886,1],[0,0,0,0,0,0,1,-6,6,25,-49,-
109,85,8193869,2716,-5249725,-17054546,-39356645,6559441,1],[0,0,0,0,0,0,0,1,-1,-4,8,18,-14,-
1348549,-447,864001,2806841,6477325,-
1079554,1],[0,0,0,0,0,0,0,0,0,1,3,-7,-16,12,1152452,382,-738364,-2398689,-5535436,922573,1],
[0,0,0,0,0,0,0,0,0,0,1,-1,-2,2,196097,65,-125637,-
408152,-941890,156982,1],[0,0,0,0,0,0,0,0,0,0,1,1,-1,-96540,-32,61852,200936,463699,-77284,-
3],[0,0,0,0,0,0,0,0,0,0,0,1,-1,-99557,-
33,63785,207216,478191,-79699,-3],[0,0,0,0,0,0,0,0,0,0,0,0,1,102573,34,-65717,-213493,-
492677,82114,5],[0,0,0,0,0,0,0,0,0,0,0,0,0,-3016,-
1,1932,6277,14486,-2415,-3],[0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,-1,-2,-4,0,-
3],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,5],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,3,-1,-3],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,-1,-

```
3],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]]}},"solid:icosa":
{"family":"solids","familyName":"Sólidos
de Platão","name":"Icosaedro","positions":{"1":[355,55],"2":[632,509],"3":[82,509],"4":
[355,198],"5":[419,255],"6":[420,354],"7":[356,412],"8":[2
91,354],"9":[291,255],"10":[321,288],"11":[390,288],"12":[355,353]},"labelPositions":{"1":
[355,24,"middle"],"2":[646,520,"start"],"3":[68,520,"en
d"],"4":[372,188,"start"],"5":[434,248,"start"],"6":[411,383,"start"],"7":
[356,441,"middle"],"8":[278,379,"end"],"9":[278,247,"end"],"10":[310,28
9,"end"],"11":[404,305,"start"],"12":[371,341,"start"]},"adjacency":{"1":[2,3,4,5,9],"2":
[1,3,5,6,7],"3":[1,2,7,8,9],"4":[1,5,9,10,11],"5":[1,2,4
,6,11],"6":[2,5,7,11,12],"7":[2,3,6,8,12],"8":[3,7,9,10,12],"9":[1,3,4,8,10],"10":
[4,8,9,11,12],"11":[4,5,6,10,12],"12":[6,7,8,10,11]},"snf":{"d"
:[1,1,1,1,1,1,2,0,0,0,0,0],"s":[[1,0,0,0,0,0,0,0,0,0,0,0],[1,0,0,-1,0,0,0,0,0,0,0,0],
[1,0,0,0,-1,0,0,0,0,0,0,0],[1,-1,0,0,0,0,0,0,0,0,0,0],[0,1,-
1,0,0,0,0,0,0,0,0,0],[0,1,-1,-1,1,-1,0,0,0,0,0,0],[-1,-1,1,2,-1,1,1,0,0,0,0,0],[1,0,0,-1,0,0,-
1,0,0,0,0,1],[0,0,1,-1,0,0,-1,0,0,0,1,0],[0,1,0,-
1,0,0,-1,0,0,1,0,0],[0,0,0,-1,0,1,-1,0,1,0,0,0],[0,0,0,-1,1,0,-1,1,0,0,0,0]],"t":[[1,-1,0,-1,-
1,0,-1,1,0,0,0,0],[0,1,-1,0,0,-1,-
1,0,0,1,0,0],[0,0,1,0,0,1,1,0,1,0,0,0],[0,0,0,1,0,1,2,-1,-1,-1,-1,-1],[0,0,0,0,1,-1,-
1,0,0,0,0,1],[0,0,0,0,0,1,1,0,0,0,1,0],[0,0,0,0,0,0,1,-1,-
1,-1,-1,-
1],[0,0,0,0,0,0,0,0,0,0,0,1],[0,0,0,0,0,0,0,0,0,0,1,0],[0,0,0,0,0,0,0,0,0,1,0,0],
[0,0,0,0,0,0,0,0,1,0,0,0],[0,0,0,0,0,0,0,1,0,0,0,0]]}},"cycle:3"
:{"family":"cycles","familyName":"Ciclos","name":"Ciclo C3","positions":{"1":[350.0,65.0],"2":
[536.1954618136543,387.49999999999994],"3":[163.804
53818634572,387.50000000000006]},"adjacency":{"1":[3,2],"2":[1,3],"3":[2,1]},"snf":{"d":
[1,0,0],"s":[[1,0,0],[-1,0,1],[-1,1,0]],"t":[[1,-1,-
1],[0,0,1],[0,1,0]]}},"cycle:4":{"family":"cycles","familyName":"Ciclos","name":"Ciclo
C4","positions":{"1":[350.0,65.0],"2":[565.0,280.0],"3":[3
50.0,495.0],"4":[135.0,280.0]},"adjacency":{"1":[4,2],"2":[1,3],"3":[2,4],"4":[3,1]},"snf":
{"d":[1,1,1,3],"s":[[1,0,0,0],[0,0,1,0],[-
1,1,0,0],[1,-2,1,1]],"t":[[1,-1,1,1],[0,1,-1,-2],[0,0,1,1],[0,0,0,1]]}},"cycle:5":
{"family":"cycles","familyName":"Ciclos","name":"Ciclo C5","pos
itions":{"1":[350.0,65.0],"2":[554.477151003458,213.56134620938633],"3":
[476.3738292428817,453.93865379061367],"4":[223.6261707571183,453.9386537
9061367],"5":[145.52284899654197,213.56134620938633]},"adjacency":{"1":[5,2],"2":[1,3],"3":
[2,4],"4":[3,5],"5":[4,1]},"snf":{"d":[1,1,1,1,3],"s":
[[1,0,0,0,0],[0,0,1,0,0],[-1,1,0,0,0],[1,-1,0,1,0],[2,-1,-1,2,-1]],"t":[[1,-1,1,1,-2],[0,1,-
1,-1,1],[0,0,1,0,1],[0,0,0,1,-
2],[0,0,0,0,1]]}},"cycle:6":{"family":"cycles","familyName":"Ciclos","name":"Ciclo
C6","positions":{"1":[350.0,65.0],"2":[536.1954618136543,172.5
],"3":[536.1954618136543,387.49999999999994],"4":[350.0,495.0],"5":
[163.80453818634572,387.50000000000006],"6":[163.8045381863457,172.49999999999
997]},"adjacency":{"1":[6,2],"2":[1,3],"3":[2,4],"4":[3,5],"5":[4,6],"6":[5,1]},"snf":{"d":
[1,1,1,1,0,0],"s":[[1,0,0,0,0,0],[0,0,1,0,0,0],[-
1,1,0,0,0,0],[1,-1,0,1,0,0],[-1,0,1,-1,0,1],[-1,1,0,-1,1,0]],"t":[[1,-1,1,1,-1,-1],[0,1,-1,-
1,0,1],[0,0,1,0,1,0],[0,0,0,1,-1,-
1],[0,0,0,0,0,1],[0,0,0,0,1,0]]}},"cycle:7":
{"family":"cycles","familyName":"Ciclos","name":"Ciclo C7","positions":{"1":[350.0,65.0],"2":
[518.093
7687306264,145.9496926003723],"3":[559.6095011190921,327.8420008006076],"4":
[443.285003910275,473.70830659902015],"5":[256.714996089725,473.70830
659902015],"6":[140.39049888090793,327.8420008006076],"7":
[181.90623126937356,145.94969260037232]},"adjacency":{"1":[7,2],"2":[1,3],"3":[2,4],"4"
:[3,5],"5":[4,6],"6":[5,7],"7":[6,1]},"snf":{"d":[1,1,1,1,1,1,3],"s":[[1,0,0,0,0,0,0],
[0,0,1,0,0,0,0],[-1,1,0,0,0,0,0],[1,-
1,0,1,0,0,0],[0,0,0,0,0,1,0],[-1,1,0,-1,1,0,0],[1,-2,1,1,-2,1,1]],"t":[[1,-1,1,1,-1,1,1],
[0,1,-1,-1,1,-1,-2],[0,0,1,0,0,0,1],[0,0,0,1,-
1,1,1],[0,0,0,0,1,-1,-2],[0,0,0,0,0,1,1],[0,0,0,0,0,0,1]]}},"cycle:8":
{"family":"cycles","familyName":"Ciclos","name":"Ciclo C8","positions":{"1"
:[350.0,65.0],"2":[502.0279579551077,127.9720420448923],"3":[565.0,280.0],"4":
[502.0279579551077,432.0279579551077],"5":[350.0,495.0],"6":[197.97
20420448923,432.0279579551077],"7":[135.0,280.0],"8":
[197.97204204489225,127.9720420448923]},"adjacency":{"1":[8,2],"2":[1,3],"3":[2,4],"4":[3,5]
,"5":[4,6],"6":[5,7],"7":[6,8],"8":[7,1]},"snf":{"d":[1,1,1,1,1,1,1,3],"s":[[1,0,0,0,0,0,0,0],
[0,0,1,0,0,0,0,0],[-1,1,0,0,0,0,0,0],[1,-
1,0,1,0,0,0,0],[0,0,0,0,0,1,0,0],[-1,1,0,-1,1,0,0,0],[1,-1,0,1,-1,0,1,0],[2,-1,-1,2,-1,-1,2,-
1]],"t":[[1,-1,1,1,-1,1,1,-2],[0,1,-1,-1,1,-1,-
1,1],[0,0,1,0,0,0,0,1],[0,0,0,1,-1,1,1,-2],[0,0,0,0,1,-1,-1,1],[0,0,0,0,0,1,0,1],
[0,0,0,0,0,0,1,-
2],[0,0,0,0,0,0,0,1]]}},"cycle:9":{"family":"cycles","familyName":"Ciclos","name":"Ciclo
C9","positions":{"1":[350.0,65.0],"2":[488.1993360826059
5,115.30044472941972],"3":[561.7336668976247,242.66564180160998],"4":
[536.1954618136543,387.49999999999994],"5":[423.5343308150188,482.0339134689
703],"6":[276.4656691849812,482.03391346897035],"7":
[163.80453818634572,387.50000000000006],"8":[138.26633310237526,242.66564180161004],"9":[211.
```

```
80066391739402,115.30044472941975]},"adjacency":{"1":[9,2],"2":[1,3],"3":[2,4],"4":[3,5],"5":
[4,6],"6":[5,7],"7":[6,8],"8":[7,9],"9":[8,1]},"snf"
:{"d":[1,1,1,1,1,1,1,0,0],"s":[[1,0,0,0,0,0,0,0,0],[0,0,1,0,0,0,0,0,0],[-1,1,0,0,0,0,0,0,0],
[1,-1,0,1,0,0,0,0,0],[0,0,0,0,0,1,0,0,0],[-1,1,0,-
1,1,0,0,0,0],[1,-1,0,1,-1,0,1,0,0],[-1,0,1,-1,0,1,-1,0,1],[-1,1,0,-1,1,0,-1,1,0]],"t":[[1,-
1,1,1,-1,1,1,-1,-1],[0,1,-1,-1,1,-1,-
1,0,1],[0,0,1,0,0,0,0,1,0],[0,0,0,1,-1,1,1,-1,-1],[0,0,0,0,1,-1,-1,0,1],[0,0,0,0,0,1,0,1,0],
[0,0,0,0,0,0,1,-1,-
1],[0,0,0,0,0,0,0,0,1],[0,0,0,0,0,0,0,1,0]]}},"path:3":
{"family":"paths","familyName":"Caminhos","name":"Caminho P3","positions":{"1":[75.0,240.0
],"2":[350.0,240.0],"3":[625.0,240.0]},"adjacency":{"1":[2],"2":[1,3],"3":[2]},"snf":{"d":
[1,1,1],"s":[[1,0,0],[0,0,1],[-1,1,0]],"t":[[1,-
1,1],[0,1,-1],[0,0,1]]}},"path:4":{"family":"paths","familyName":"Caminhos","name":"Caminho
P4","positions":{"1":[75.0,240.0],"2":[258.3333333333
3337,240.0],"3":[441.6666666666667,240.0],"4":[625.0,240.0]},"adjacency":{"1":[2],"2":
[1,3],"3":[2,4],"4":[3]},"snf":{"d":[1,1,1,1],"s":[[1,0,0,0
],[0,0,1,0],[-1,1,0,0],[1,-1,0,1]],"t":[[1,-1,1,1],[0,1,-1,-
1],[0,0,1,0],[0,0,0,1]]}},"path:5":{"family":"paths","familyName":"Caminhos","name":"Caminho
P5","positions":{"1":[75.0,240.0],"2":[212.5,240.0],
"3":[350.0,240.0],"4":[487.5,240.0],"5":[625.0,240.0]},"adjacency":{"1":[2],"2":[1,3],"3":
[2,4],"4":[3,5],"5":[4]},"snf":{"d":[1,1,1,1,0],"s":[[1
,0,0,0,0],[0,0,1,0,0],[-1,1,0,0,0],[1,-1,0,1,0],[-1,1,0,-1,1]],"t":[[1,-1,1,1,-1],[0,1,-1,-
1,1],[0,0,1,0,0],[0,0,0,1,-
1],[0,0,0,0,1]]}},"path:6":{"family":"paths","familyName":"Caminhos","name":"Caminho
P6","positions":{"1":[75.0,240.0],"2":[185.0,240.0],"3":[295
.0,240.0],"4":[405.0,240.0],"5":[515.0,240.0],"6":[625.0,240.0]},"adjacency":{"1":[2],"2":
[1,3],"3":[2,4],"4":[3,5],"5":[4,6],"6":[5]},"snf":{"d"
:[1,1,1,1,1,1],"s":[[1,0,0,0,0,0],[0,0,1,0,0,0],[-1,1,0,0,0,0],[1,-1,0,1,0,0],[0,0,0,0,0,1],
[-1,1,0,-1,1,0]],"t":[[1,-1,1,1,-1,1],[0,1,-1,-1,1,-
1],[0,0,1,0,0,0],[0,0,0,1,-1,1],[0,0,0,0,1,-1],[0,0,0,0,0,1]]}},"path:7":
{"family":"paths","familyName":"Caminhos","name":"Caminho P7","positions
":{"1":[75.0,240.0],"2":[166.66666666666669,240.0],"3":[258.33333333333337,240.0],"4":
[350.0,240.0],"5":[441.6666666666667,240.0],"6":[533.333333
3333334,240.0],"7":[625.0,240.0]},"adjacency":{"1":[2],"2":[1,3],"3":[2,4],"4":[3,5],"5":
[4,6],"6":[5,7],"7":[6]},"snf":{"d":[1,1,1,1,1,1,1],"s":
[[1,0,0,0,0,0,0],[0,0,1,0,0,0,0],[-1,1,0,0,0,0,0],[1,-1,0,1,0,0,0],[0,0,0,0,0,1,0],[-1,1,0,-
1,1,0,0],[1,-1,0,1,-1,0,1]],"t":[[1,-1,1,1,-
1,1,1],[0,1,-1,-1,1,-1,-1],[0,0,1,0,0,0,0],[0,0,0,1,-1,1,1],[0,0,0,0,1,-1,-
1],[0,0,0,0,0,1,0],[0,0,0,0,0,0,1]]}},"path:8":
{"family":"paths","familyName":"Caminhos","name":"Caminho P8","positions":{"1":
[75.0,240.0],"2":[1
53.57142857142856,240.0],"3":[232.14285714285714,240.0],"4":[310.7142857142857,240.0],"5":
[389.2857142857143,240.0],"6":[467.85714285714283,240.0
],"7":[546.4285714285714,240.0],"8":[625.0,240.0]},"adjacency":{"1":[2],"2":[1,3],"3":
[2,4],"4":[3,5],"5":[4,6],"6":[5,7],"7":[6,8],"8":[7]},"snf
":{"d":[1,1,1,1,1,1,1,1,0],"s":[[1,0,0,0,0,0,0,0,0],[0,0,1,0,0,0,0,0,0],[-1,1,0,0,0,0,0,0,0],[1,-
1,0,1,0,0,0,0,0],[0,0,0,0,0,0,1,0,0],[-1,1,0,-
1,1,0,0,0],[1,-1,0,1,-1,0,1,0],[-1,1,0,-1,1,0,-1,1]],"t":[[1,-1,1,1,-1,1,1,-1],[0,1,-1,-1,1,-
1,-1,1],[0,0,1,0,0,0,0,0,0],[0,0,0,1,-1,1,1,-
1],[0,0,0,0,1,-1,-1,1],[0,0,0,0,0,1,0,0],[0,0,0,0,0,0,0,1,-
1],[0,0,0,0,0,0,0,0,1]]}},"path:9":{"family":"paths","familyName":"Caminhos","name":"Caminho
P9","positions":{"1":[75.0,240.0],"2":[143.75,240.0],"

3":[212.5,240.0],"4":[281.25,240.0],"5":[350.0,240.0],"6":[418.75,240.0],"7":
[487.5,240.0],"8":[556.25,240.0],"9":[625.0,240.0]},"adjacency":{"1"
:[2],"2":[1,3],"3":[2,4],"4":[3,5],"5":[4,6],"6":[5,7],"7":[6,8],"8":[7,9],"9":[8]},"snf":
{"d":[1,1,1,1,1,1,1,1,1],"s":[[1,0,0,0,0,0,0,0,0],[0,0,
1,0,0,0,0,0,0],[-1,1,0,0,0,0,0,0,0],[1,-1,0,1,0,0,0,0,0],[0,0,0,0,0,1,0,0,0],[-1,1,0,-
1,1,0,0,0,0],[1,-1,0,1,-1,0,1,0,0],[0,0,0,0,0,0,0,0,1],[-
1,1,0,-1,1,0,-1,1,0]],"t":[[1,-1,1,1,-1,1,1,-1,1],[0,1,-1,-1,1,-1,-1,1,-1],
[0,0,1,0,0,0,0,0,0],[0,0,0,1,-1,1,1,-1,1],[0,0,0,0,1,-1,-1,1,-
1],[0,0,0,0,0,1,0,0,0],[0,0,0,0,0,0,0,1,-1,1],[0,0,0,0,0,0,0,1,-1],[0,0,0,0,0,0,0,0,1]]}}};
const familyOptions = {
solids: {
name: "Sólidos de Platão",
subtitle: "Explore os cinco sólidos estudados no trabalho, com 2 ou 3 estados.",
keys: ["solid:tetra", "solid:hexa", "solid:octa", "solid:dodeca", "solid:icosa"]
},
cycles: {
name: "Ciclos",
subtitle: "Escolha qualquer ciclo C■, com n ≥ 3, e qualquer quantidade inteira de estados a
partir de 2.",
keys: []
},
paths: {
name: "Caminhos",
```

```
subtitle: "Escolha qualquer caminho P■, com n ≥ 2, e qualquer quantidade inteira de estados a
partir de 2.",
keys: []
}
};
const svgNS = "http://www.w3.org/2000/svg";
const familySelect = document.getElementById("familySelect");
const boardSelect = document.getElementById("boardSelect");
const boardSizeInput = document.getElementById("boardSizeInput");
const boardSizeHelp = document.getElementById("boardSizeHelp");
const boardControlTitle = document.getElementById("boardControlTitle");
const stateCountSelect = document.getElementById("stateCountSelect");
const panelTitle = document.getElementById("panelTitle");
const panelSubtitle = document.getElementById("panelSubtitle");
const currentModePill = document.getElementById("currentModePill");
const familyNameEl = document.getElementById("familyName");
const boardNameEl = document.getElementById("boardName");
const stateCountValueEl = document.getElementById("stateCountValue");
const movesEl = document.getElementById("moves");
const greenCountEl = document.getElementById("greenCount");
const solvabilityNotice = document.getElementById("solvabilityNotice");
const legendEl = document.getElementById("legend");
const messageEl = document.getElementById("message");
const undoBtn = document.getElementById("undoBtn");
const hintBtn = document.getElementById("hintBtn");
const solutionBtn = document.getElementById("solutionBtn");
const resetBtn = document.getElementById("resetBtn");
const boardSvg = document.getElementById("board");
const edgesLayer = document.getElementById("edges");
const verticesLayer = document.getElementById("vertices");
const stateChangeCounterEl = document.getElementById("stateChangeCounter");
const victoryOverlay = document.getElementById("victoryOverlay");
const victoryMessage = document.getElementById("victoryMessage");
const closeVictoryBtn = document.getElementById("closeVictoryBtn");
const playAgainBtn = document.getElementById("playAgainBtn");
const solutionOverlay = document.getElementById("solutionOverlay");
const solutionContent = document.getElementById("solutionContent");
const closeSolutionBtn = document.getElementById("closeSolutionBtn");
const sessions = {};
const solvabilityCache = {};
let currentFamily = "solids";
let currentKey = "solid:tetra";
let stateCount = 3;
const familySizes = { cycles: 3, paths: 3 };
function generatePathBoard(n) {
const spacing = 60;
const margin = 70;
const width = Math.max(700, margin * 2 + spacing * (n - 1));
const y = 240;
const positions = {};
const adjacency = {};
for (let vertex = 1; vertex <= n; vertex++) {
positions[vertex] = [margin + spacing * (vertex - 1), y];
const neighbors = [];
if (vertex > 1) neighbors.push(vertex - 1);
if (vertex < n) neighbors.push(vertex + 1);
adjacency[vertex] = neighbors;
}
return {
family: "paths",
familyName: "Caminhos",
name: `Caminho P${n}`,
positions,
adjacency,
dynamic: true,
center: [width / 2, 240],
viewBox: [0, 0, width, 480],
pixelWidth: width
};
}
function generateCycleBoard(n) {
const rx = Math.max(215, n * 10);
const ry = 205;
const width = Math.max(700, 2 * rx + 120);
const height = 560;
const centerX = width / 2;
const centerY = 280;
```

```
const positions = {};
const adjacency = {};
for (let vertex = 1; vertex <= n; vertex++) {
const angle = -Math.PI / 2 + 2 * Math.PI * (vertex - 1) / n;
positions[vertex] = [
centerX + rx * Math.cos(angle),
centerY + ry * Math.sin(angle)
];
adjacency[vertex] = [
vertex === 1 ? n : vertex - 1,

vertex === n ? 1 : vertex + 1
];
}
return {
family: "cycles",
familyName: "Ciclos",
name: `Ciclo C${n}`,
positions,
adjacency,
dynamic: true,
center: [centerX, centerY],
viewBox: [0, 0, width, height],
pixelWidth: width
};
}
function ensureDynamicBoard(family, n) {
const key = `${family === "cycles" ? "cycle" : "path"}:${n}`;
boards[key] = family === "cycles" ? generateCycleBoard(n) : generatePathBoard(n);
return key;
}
function sessionKey() {
return `${currentKey}|${stateCount}`;
}
function ensureSession() {
const key = sessionKey();
const n = Object.keys(boards[currentKey].positions).length;
if (!sessions[key]) {
sessions[key] = {
state: Array(n + 1).fill(0),
changes: Array(n + 1).fill(0),
history: [],
moves: 0,
celebrationShown: false
};
} else if (!sessions[key].changes || sessions[key].changes.length !== n + 1) {
sessions[key].changes = Array(n + 1).fill(0);
}
return sessions[key];
}
function mod(value, modulus = stateCount) {
return ((value % modulus) + modulus) % modulus;
}
function gcd(a, b) {
a = Math.abs(a);
b = Math.abs(b);
while (b !== 0) [a, b] = [b, a % b];
return a;
}
function extendedGcd(a, b) {
if (b === 0) return [a, 1, 0];
const [g, x1, y1] = extendedGcd(b, a % b);
return [g, y1, x1 - Math.trunc(a / b) * y1];
}
function inverseMod(value, modulus) {
if (modulus === 1) return 0;
const normalized = mod(value, modulus);
const [divisor, x] = extendedGcd(normalized, modulus);
if (Math.abs(divisor) !== 1) throw new Error("Sem inverso modular.");
return mod(x, modulus);
}
function matrixVectorMultiply(matrix, vector) {
return matrix.map(row => {
let total = 0;
for (let index = 0; index < row.length; index++) total += row[index] * vector[index];
return total;
});
```

```
}
function solveWithSmith(board, target) {
const { d, s, t } = board.snf;
const transformedTarget = matrixVectorMultiply(s, target).map(value => mod(value));
const y = Array(d.length).fill(0);
for (let index = 0; index < d.length; index++) {
const diagonal = d[index];
const value = transformedTarget[index];
if (diagonal === 0) {
if (value !== 0) return null;
y[index] = 0;
continue;
}
const divisor = gcd(diagonal, stateCount);
if (value % divisor !== 0) return null;
const reducedModulus = stateCount / divisor;
if (reducedModulus === 1) {
y[index] = 0;
continue;
}
const reducedDiagonal = diagonal / divisor;
const reducedValue = value / divisor;
y[index] = mod(
inverseMod(reducedDiagonal, reducedModulus) * reducedValue,
reducedModulus
);
}
return matrixVectorMultiply(t, y).map(value => mod(value));
}
function solveLinearCongruence(a, b, modulus = stateCount) {
a = mod(a, modulus);
b = mod(b, modulus);
const divisor = gcd(a, modulus);

if (b % divisor !== 0) return null;
const reducedModulus = modulus / divisor;
if (reducedModulus === 1) return 0;
return mod(
inverseMod(a / divisor, reducedModulus) * (b / divisor),
reducedModulus
);
}
function solvePath(board, target) {
const n = target.length;
if (n < 2) return null;
const A = Array(n + 1).fill(0);
const C = Array(n + 1).fill(0);
A[1] = 1;
A[2] = -1;
C[2] = mod(target[0]);
for (let i = 2; i < n; i++) {
A[i + 1] = -A[i - 1] - A[i];
C[i + 1] = mod(target[i - 1] - C[i - 1] - C[i]);
}
const coefficient = A[n - 1] + A[n];
const rhs = mod(target[n - 1] - C[n - 1] - C[n]);
const t = solveLinearCongruence(coefficient, rhs);
if (t === null) return null;
const solution = [];
for (let i = 1; i <= n; i++) {
solution.push(mod(A[i] * t + C[i]));
}
return solution;
}
function solveCycle(board, target) {
const n = target.length;
if (n < 3) return null;
const T = Array(n + 1).fill(0);
const U = Array(n + 1).fill(0);
const C = Array(n + 1).fill(0);
T[1] = 1;
U[2] = 1;
for (let i = 2; i < n; i++) {
T[i + 1] = -T[i - 1] - T[i];
U[i + 1] = -U[i - 1] - U[i];
C[i + 1] = mod(target[i - 1] - C[i - 1] - C[i]);
}
```

```
const r1 = mod(target[0] - C[n]);
const r2 = mod(target[n - 1] - C[n - 1] - C[n]);
let t;
let u;
if (n % 3 === 0) {
if (r1 !== 0 || r2 !== 0) return null;
t = 0;
u = 0;
} else if (n % 3 === 1) {
t = solveLinearCongruence(3, mod(r1 + r2));
if (t === null) return null;
u = mod(r1 - 2 * t);
} else {
u = solveLinearCongruence(3, mod(2 * r1 - r2));
if (u === null) return null;
t = mod(r1 - 2 * u);
}
const solution = [];
for (let i = 1; i <= n; i++) {
solution.push(mod(T[i] * t + U[i] * u + C[i]));
}
return solution;
}
function solveBoard(board, target) {
if (board.family === "paths") return solvePath(board, target);
if (board.family === "cycles") return solveCycle(board, target);
return solveWithSmith(board, target);
}
function solveWithSmithMinimum(board, target) {
const { d, s, t } = board.snf;
const transformedTarget = matrixVectorMultiply(s, target).map(value => mod(value));
const choices = [];
for (let index = 0; index < d.length; index++) {
const valid = [];
for (let y = 0; y < stateCount; y++) {
if (mod(d[index] * y) === transformedTarget[index]) valid.push(y);
}
if (!valid.length) return null;
choices.push(valid);
}
let bestSolution = null;
let bestWeight = Infinity;
const y = Array(d.length).fill(0);
function search(position) {
if (position === choices.length) {
const candidate = matrixVectorMultiply(t, y).map(value => mod(value));
const weight = candidate.reduce((sum, value) => sum + value, 0);
if (weight < bestWeight) {
bestWeight = weight;
bestSolution = candidate;
}
return;
}

for (const value of choices[position]) {
y[position] = value;
search(position + 1);
}
}
search(0);
return bestSolution;
}
function minimumInitialPathSolution(board) {
const n = Object.keys(board.positions).length;
const solution = Array(n).fill(0);
const amount = stateCount - 1;
let start;
if (n % 3 === 1) start = 0;
else start = 1;
for (let index = start; index < n; index += 3) {
solution[index] = amount;
}
return solution;
}
function minimumInitialCycleSolution(board) {
const n = Object.keys(board.positions).length;
const amount = stateCount - 1;
```

```
if (n % 3 === 0) {
const solution = Array(n).fill(0);
for (let index = 0; index < n; index += 3) {
solution[index] = amount;
}
return solution;
}
if (stateCount % 3 === 0) return null;
const x = stateCount % 3 === 1
? (stateCount - 1) / 3
: (2 * stateCount - 1) / 3;
return Array(n).fill(x);
}
function getInitialMinimumSolution() {
const board = boards[currentKey];
const n = Object.keys(board.positions).length;
const target = Array(n).fill(stateCount - 1);
if (board.family === "paths") return minimumInitialPathSolution(board);
if (board.family === "cycles") return minimumInitialCycleSolution(board);
return solveWithSmithMinimum(board, target);
}
function getTarget(board, session) {
const finalState = stateCount - 1;
const n = Object.keys(board.positions).length;
const target = [];
for (let vertex = 1; vertex <= n; vertex++) {
target.push(mod(finalState - session.state[vertex]));
}
return target;
}
function isInitialConfigurationSolvable() {
const key = sessionKey();
if (!(key in solvabilityCache)) {
const board = boards[currentKey];
const n = Object.keys(board.positions).length;
const initialSession = { state: Array(n + 1).fill(0) };
solvabilityCache[key] = solveBoard(board, getTarget(board, initialSession)) !== null;
}
return solvabilityCache[key];
}
function getCurrentSolution() {
const board = boards[currentKey];
const session = ensureSession();
return solveBoard(board, getTarget(board, session));
}
function getStateColor(index) {
if (stateCount === 3) return ["#ef4444", "#facc15", "#22c55e"][index];
if (index === 0) return "#ef4444";
if (index === stateCount - 1) return "#22c55e";
const hue = 120 * index / (stateCount - 1);
return `hsl(${hue} 78% 48%)`;
}
function getStateTextColor(index) {
if (stateCount === 3) return index === 1 ? "#111827" : "#ffffff";
if (index === 0 || index === stateCount - 1) return "#ffffff";
const hue = 120 * index / (stateCount - 1);
if (hue >= 40 && hue <= 75) return "#111827";
return "#ffffff";
}
function getStateLabel(index) {
if (index === 0) return "Estado 0 — vermelho";
if (index === stateCount - 1) return `Estado ${index} — verde`;
if (stateCount === 3 && index === 1) return "Estado 1 — amarelo";
return `Estado ${index}`;
}
function renderLegend() {
legendEl.replaceChildren();
const appendLegendItem = (index, customLabel = null) => {
const item = document.createElement("span");
item.className = "legend-item";
const dot = document.createElement("span");
dot.className = "legend-dot";
dot.style.background = getStateColor(index);
const label = document.createElement("span");

label.textContent = customLabel ?? getStateLabel(index);
item.append(dot, label);
```

```
legendEl.appendChild(item);
};
// Para valores grandes de k, não criamos milhares de itens na legenda.
// O jogo continua funcionando normalmente; a legenda é apenas resumida.
if (stateCount <= 20) {
for (let index = 0; index < stateCount; index++) appendLegendItem(index);
return;
}
appendLegendItem(0);
const middle = Math.floor((stateCount - 1) / 2);
appendLegendItem(middle, `Estados 1 a ${stateCount - 2} — cores intermediárias`);
appendLegendItem(stateCount - 1);
}
function uniqueEdges(adjacency) {
const edges = [];
for (const [uText, neighbors] of Object.entries(adjacency)) {
const u = Number(uText);
for (const v of neighbors) if (u < v) edges.push([u, v]);
}
return edges;
}
function createSvgElement(name, attributes = {}) {
const element = document.createElementNS(svgNS, name);
for (const [key, value] of Object.entries(attributes)) {
element.setAttribute(key, String(value));
}
return element;
}
function clearHint() {
verticesLayer.querySelectorAll(".vertex.hint").forEach(vertex =>
vertex.classList.remove("hint"));
}
function radialLabelPosition(x, y, board) {
const [centerX, centerY] = board.center || [350, 280];
let dx = x - centerX;
let dy = y - centerY;
const length = Math.hypot(dx, dy);
if (length < 25) return [x + 25, y + 7, "start"];
dx /= length;
dy /= length;
const labelX = x + dx * 28;
const labelY = y + dy * 28 + 6;
return [labelX, labelY, labelX < x ? "end" : "start"];
}
function getLabelPosition(board, vertex, x, y) {
if (board.labelPositions && board.labelPositions[vertex]) {
return board.labelPositions[vertex];
}
if (board.family === "paths") return [x, y + 56, "middle"];
return radialLabelPosition(x, y, board);
}
function createVertex(board, vertex, x, y) {
const group = createSvgElement("g", {
class: "vertex",
"data-v": vertex,
role: "button",
tabindex: 0
});
const hitRadius = board.family === "paths" ? 38 : 29;
const focusRadius = board.family === "paths" ? 29 : 22;
const nodeRadius = board.family === "paths" ? 18 : 14;
const hitArea = createSvgElement("circle", {
class: "hit-area",
cx: x,
cy: y,
r: hitRadius
});
const focusRing = createSvgElement("circle", {
class: "focus-ring",
cx: x,
cy: y,
r: focusRadius
});
const node = createSvgElement("circle", {
class: "node",
cx: x,
cy: y,
```

```
r: nodeRadius
});
const stateNumber = createSvgElement("text", {
class: "state-number",
x: x,
y: y + 1
});
stateNumber.textContent = "0";
const [labelX, labelY, anchor] = getLabelPosition(board, vertex, x, y);
const label = createSvgElement("text", {
class: "vertex-label",
x: labelX,
y: labelY,
"text-anchor": anchor
});
label.textContent = String(vertex);
group.append(hitArea, focusRing, node, stateNumber, label);
group.addEventListener("click", () => pressVertex(vertex));

group.addEventListener("keydown", event => {
if (event.key === "Enter" || event.key === " ") {
event.preventDefault();
pressVertex(vertex);
}
});
verticesLayer.appendChild(group);
}
function renderStateChangeCounter() {
if (!stateChangeCounterEl) return;
const board = boards[currentKey];
const session = ensureSession();
const n = Object.keys(board.positions).length;
stateChangeCounterEl.replaceChildren();
for (let vertex = 1; vertex <= n; vertex++) {
const item = document.createElement("div");
item.className = "state-change-item";
const vertexLabel = document.createElement("span");
vertexLabel.className = "state-change-vertex";
vertexLabel.textContent = `Vértice ${vertex}`;
const value = document.createElement("span");
value.className = "state-change-value";
value.textContent = String(session.changes[vertex] || 0);
item.append(vertexLabel, value);
stateChangeCounterEl.appendChild(item);
}
}
function buildBoard() {
edgesLayer.replaceChildren();
verticesLayer.replaceChildren();
const board = boards[currentKey];
const viewBox = board.viewBox || [0, 0, 700, 560];
boardSvg.setAttribute("viewBox", viewBox.join(" "));
if (board.pixelWidth) {
boardSvg.style.width = `${board.pixelWidth}px`;
boardSvg.style.maxWidth = "none";
} else {
boardSvg.style.width = "100%";
boardSvg.style.maxWidth = "100%";
}
boardSvg.style.height = "auto";
if (board.family === "cycles") {
const n = Object.keys(board.positions).length;
for (let vertex = 1; vertex <= n; vertex++) {
const next = vertex === n ? 1 : vertex + 1;
const [x1, y1] = board.positions[vertex];
const [x2, y2] = board.positions[next];
edgesLayer.appendChild(createSvgElement("line", { class: "edge", x1, y1, x2, y2 }));
}
} else if (board.family === "paths") {
const n = Object.keys(board.positions).length;
for (let vertex = 1; vertex < n; vertex++) {
const [x1, y1] = board.positions[vertex];
const [x2, y2] = board.positions[vertex + 1];
edgesLayer.appendChild(createSvgElement("line", { class: "edge", x1, y1, x2, y2 }));
}
} else {
for (const [u, v] of uniqueEdges(board.adjacency)) {
```

```
const [x1, y1] = board.positions[u];
const [x2, y2] = board.positions[v];
edgesLayer.appendChild(createSvgElement("line", { class: "edge", x1, y1, x2, y2 }));
}
}
for (const [vertexText, position] of Object.entries(board.positions)) {
createVertex(board, Number(vertexText), position[0], position[1]);
}
renderLegend();
render();
}
function pressVertex(vertex) {
clearHint();
const board = boards[currentKey];
const session = ensureSession();
const finalState = stateCount - 1;
if (session.state.slice(1).every(value => value === finalState)) return;
session.history.push({
state: session.state.slice(),
changes: session.changes.slice()
});
for (const affected of [vertex, ...board.adjacency[vertex]]) {
session.state[affected] = mod(session.state[affected] + 1);
session.changes[affected] += 1;
}
session.moves++;
const neighbors = board.adjacency[vertex];
let neighborText = "";
if (neighbors.length === 1) neighborText = `Também mudou o vértice ${neighbors[0]}.`;
else if (neighbors.length === 2) neighborText = `Também mudaram os vértices ${neighbors[0]} e
${neighbors[1]}.`;
else neighborText = `Também mudaram os vértices ${neighbors.join(", ")}.`;
messageEl.textContent = `Vértice ${vertex} acionado. ${neighborText}`;
render();
}
function chooseBestSuggestion(solution) {
const board = boards[currentKey];

const session = ensureSession();
const finalState = stateCount - 1;
let suggestion = null;
let bestScore = -Infinity;
for (let index = 0; index < solution.length; index++) {
if (solution[index] === 0) continue;
const vertex = index + 1;
let score = 0;
for (const affected of [vertex, ...board.adjacency[vertex]]) {
const before = session.state[affected];
const after = mod(before + 1);
if (after === finalState && before !== finalState) score += 3;
if (before === finalState && after !== finalState) score -= 2;
if (after > before) score += 0.25;
}
if (score > bestScore) {
bestScore = score;
suggestion = vertex;
}
}
return suggestion;
}
function showHint() {
clearHint();
const board = boards[currentKey];
const session = ensureSession();
const finalState = stateCount - 1;
if (session.state.slice(1).every(value => value === finalState)) {
messageEl.textContent = "O tabuleiro já está resolvido.";
return;
}
const solution = getCurrentSolution();
if (!solution) {
messageEl.textContent = `Esta configuração não pode ser levada ao estado final com $
{stateCount} estados.`;
return;
}
const suggestion = chooseBestSuggestion(solution);
if (suggestion === null) {
```

```
messageEl.textContent = "O tabuleiro já está resolvido.";
return;
}
verticesLayer.querySelector(`[data-v="${suggestion}"]`)?.classList.add("hint");
const neighbors = board.adjacency[suggestion];
let desc = "";
if (neighbors.length === 1) desc = `Seu vizinho é o vértice ${neighbors[0]}.`;
else if (neighbors.length === 2) desc = `Seus vizinhos são ${neighbors[0]} e $
{neighbors[1]}.`;
else desc = `Seus vizinhos são ${neighbors.join(", ")}.`;
messageEl.textContent = `Dica: experimente acionar o vértice ${suggestion}. ${desc}`;
}
function showSolution() {
const board = boards[currentKey];
const session = ensureSession();
const isInitial = session.moves === 0 && session.state.slice(1).every(value => value === 0);
const solution = isInitial
? getInitialMinimumSolution()
: getCurrentSolution();
solutionTitle.textContent = isInitial ? "Solução mínima" : "Uma solução possível";
solutionDescription.textContent = isInitial
? "Abaixo está uma solução com o menor número de acionamentos para a configuração inicial."
: "Abaixo está uma sequência de acionamentos que resolve o jogo a partir da configuração
atual.";
if (!solution) {
solutionContent.innerHTML = `
<p>Para a configuração ${isInitial ? "inicial" : "atual"} de <strong>${board.name}</strong>
com <strong>${stateCount}</strong> estados,
não existe solução que leve todos os vértices ao estado final.</p>
`;
} else {
const nonzero = [];
let total = 0;
for (let index = 0; index < solution.length; index++) {
const count = solution[index];
total += count;
if (count !== 0) {
nonzero.push(`<li>Acione o vértice <strong>${index + 1}</strong> exatamente <strong>${count}
</strong> vez(es).</li>`);
}
}
const listHtml = nonzero.length
? `<ol class="solution-list">${nonzero.join("")}</ol>`
: `<p>Nenhum acionamento adicional é necessário.</p>`;
const compact = solution
.map((count, index) => `x${index + 1} = ${count}`)
.join(", ");
solutionContent.innerHTML = `
<p><strong>${board.name}</strong> com <strong>${stateCount}</strong> estados.</p>
${listHtml}
<div class="solution-summary">
<strong>Resumo algébrico:</strong><br>
${compact}<br><br>
<strong>${isInitial ? "Número mínimo de acionamentos" : "Total de acionamentos desta
solução"}:</strong> ${total}.
</div>
`;
}
solutionOverlay.classList.add("visible");
solutionOverlay.setAttribute("aria-hidden", "false");
closeSolutionBtn.focus();

}
function closeSolution() {
solutionOverlay.classList.remove("visible");
solutionOverlay.setAttribute("aria-hidden", "true");
}
function showVictory() {
const board = boards[currentKey];
const session = ensureSession();
victoryMessage.textContent =
`Você resolveu ${board.name.toLowerCase()} com ${stateCount} estados e levou todos os vértices
ao verde em ${session.moves} jogadas.`;
victoryOverlay.classList.add("visible");
victoryOverlay.setAttribute("aria-hidden", "false");
document.body.classList.add("modal-open");
closeVictoryBtn.focus();
```

```
}
function closeVictory() {
victoryOverlay.classList.remove("visible");
victoryOverlay.setAttribute("aria-hidden", "true");
document.body.classList.remove("modal-open");
}
function resetCurrentGame() {
clearHint();
closeVictory();
closeSolution();
const board = boards[currentKey];
const n = Object.keys(board.positions).length;
sessions[sessionKey()] = {
state: Array(n + 1).fill(0),
changes: Array(n + 1).fill(0),
history: [],
moves: 0,
celebrationShown: false
};
messageEl.textContent = `${board.name} reiniciado com ${stateCount} estados. Todos os vértices
estão vermelhos.`;
render();
}
function render() {
const board = boards[currentKey];
const session = ensureSession();
const n = Object.keys(board.positions).length;
const finalState = stateCount - 1;
for (let vertex = 1; vertex <= n; vertex++) {
const group = verticesLayer.querySelector(`[data-v="${vertex}"]`);
if (!group) continue;
const currentState = session.state[vertex];
group.querySelector(".node").setAttribute("fill", getStateColor(currentState));
group.querySelector(".state-number").setAttribute("fill", getStateTextColor(currentState));
group.querySelector(".state-number").textContent = String(currentState);
group.setAttribute("aria-label", `Vértice ${vertex}, estado ${currentState} de ${finalState}
`);
}
const finalCount = session.state.slice(1).filter(value => value === finalState).length;
const solved = finalCount === n;
const initiallySolvable = isInitialConfigurationSolvable();
familyNameEl.textContent = familyOptions[currentFamily].name;
boardNameEl.textContent = board.name;
stateCountValueEl.textContent = String(stateCount);
movesEl.textContent = String(session.moves);
greenCountEl.textContent = `${finalCount}/${n}`;
undoBtn.disabled = session.history.length === 0;
renderStateChangeCounter();
panelTitle.textContent = familyOptions[currentFamily].name;
panelSubtitle.textContent = familyOptions[currentFamily].subtitle;
currentModePill.textContent = `Modo atual: ${familyOptions[currentFamily].name}`;
if (initiallySolvable) {
solvabilityNotice.textContent = `${board.name}: a configuração inicial possui solução com $
{stateCount} estados.`;
} else {
solvabilityNotice.textContent = `${board.name}: a configuração inicial não possui solução com
${stateCount} estados. O tabuleiro pode ser
usado para investigação matemática.`;
}
if (solved) {
messageEl.textContent = "";
session.celebrationShown = true;
} else {
session.celebrationShown = false;
}
}
function normalizeBoardSize(rawValue) {
let value = Number(rawValue);
const minimum = currentFamily === "cycles" ? 3 : 2;
if (!Number.isFinite(value)) value = minimum;
value = Math.trunc(value);
if (value < minimum) value = minimum;
return value;
}
function updateBoardControl() {
if (currentFamily === "solids") {
boardControlTitle.textContent = "Escolha o tabuleiro";
```

```
boardSelect.hidden = false;
boardSizeInput.hidden = true;
boardSizeHelp.hidden = true;
boardSelect.replaceChildren();
for (const key of familyOptions.solids.keys) {
const option = document.createElement("option");
option.value = key;
option.textContent = boards[key].name;
boardSelect.appendChild(option);

}
if (!familyOptions.solids.keys.includes(currentKey)) {
currentKey = familyOptions.solids.keys[0];
}
boardSelect.value = currentKey;
return;
}
boardControlTitle.textContent = "Tamanho do tabuleiro";
boardSelect.hidden = true;
boardSizeInput.hidden = false;
boardSizeHelp.hidden = false;
const minimum = currentFamily === "cycles" ? 3 : 2;
const symbol = currentFamily === "cycles" ? "C■" : "P■";
boardSizeInput.min = String(minimum);
boardSizeInput.step = "1";
boardSizeInput.removeAttribute("max");
boardSizeInput.value = String(familySizes[currentFamily]);
boardSizeInput.title = `Digite qualquer inteiro n ≥ ${minimum}. Não há limite máximo
predefinido.`;
boardSizeHelp.textContent =
`${symbol}: digite qualquer inteiro n ≥ ${minimum}. Sem limite máximo predefinido; tamanhos
muito grandes dependem do desempenho do
navegador.`;
}
function normalizeStateCount(rawValue) {
let value = Number(rawValue);
if (!Number.isFinite(value)) value = 2;
value = Math.trunc(value);
if (value < 2) value = 2;
if (currentFamily === "solids" && value > 3) value = 3;
return value;
}
function updateStateCountControl() {
stateCountSelect.min = "2";
stateCountSelect.step = "1";
if (currentFamily === "solids") {
stateCountSelect.max = "3";
stateCountSelect.title = "Nos sólidos de Platão estão disponíveis apenas 2 ou 3 estados.";
if (stateCount > 3) stateCount = 3;
} else {
stateCountSelect.removeAttribute("max");
stateCountSelect.title = "Digite qualquer número inteiro de estados a partir de 2.";
}
stateCountSelect.value = String(stateCount);
}
function setFamily(family) {
clearHint();
closeVictory();
closeSolution();
currentFamily = family;
familySelect.value = currentFamily;
updateStateCountControl();
if (currentFamily === "solids") {
currentKey = familyOptions.solids.keys[0];
} else {
currentKey = ensureDynamicBoard(currentFamily, familySizes[currentFamily]);
}
updateBoardControl();
ensureSession();
buildBoard();
messageEl.textContent = `Família alterada para ${familyOptions[currentFamily].name}.`;
}
familySelect.addEventListener("change", event => {
setFamily(event.target.value);
});
boardSelect.addEventListener("change", event => {
clearHint();
```

```
closeVictory();
closeSolution();
currentKey = event.target.value;
ensureSession();
buildBoard();
messageEl.textContent = `Tabuleiro alterado para ${boards[currentKey].name}.`;
});
boardSizeInput.addEventListener("change", event => {
if (currentFamily === "solids") return;
clearHint();
closeVictory();
closeSolution();
const requested = Number(event.target.value);
const normalized = normalizeBoardSize(requested);
const minimum = currentFamily === "cycles" ? 3 : 2;
const wasAdjusted = normalized !== Math.trunc(requested);
familySizes[currentFamily] = normalized;
currentKey = ensureDynamicBoard(currentFamily, normalized);
updateBoardControl();
ensureSession();
buildBoard();
if (wasAdjusted) {
messageEl.textContent =
`Use um número inteiro n ≥ ${minimum}. O tabuleiro foi ajustado para $
{boards[currentKey].name}.`;
} else {
messageEl.textContent = `Tabuleiro alterado para ${boards[currentKey].name}.`;
}
});
stateCountSelect.addEventListener("change", event => {
clearHint();

closeVictory();
closeSolution();
const requested = Number(event.target.value);
const normalized = normalizeStateCount(requested);
const wasAdjusted = normalized !== Math.trunc(requested);
stateCount = normalized;
updateStateCountControl();
ensureSession();
buildBoard();
if (currentFamily === "solids" && requested > 3) {
messageEl.textContent = "Nos sólidos de Platão, o jogo está disponível apenas com 2 ou 3
estados.";
} else if (wasAdjusted) {
messageEl.textContent = `Use um número inteiro de estados a partir de 2. O jogo foi ajustado
para ${stateCount} estados.`;
} else {
messageEl.textContent = `A quantidade foi alterada para ${stateCount} estados.`;
}
});
undoBtn.addEventListener("click", () => {
clearHint();
const session = ensureSession();
if (!session.history.length) return;
const previous = session.history.pop();
if (Array.isArray(previous)) {
// Compatibilidade com sessões antigas.
session.state = previous;
session.changes = Array(session.state.length).fill(0);
} else {
session.state = previous.state;
session.changes = previous.changes;
}
session.moves = Math.max(0, session.moves - 1);
session.celebrationShown = false;
messageEl.textContent = "A última jogada foi desfeita.";
render();
});
hintBtn.addEventListener("click", showHint);
solutionBtn.addEventListener("click", showSolution);
resetBtn.addEventListener("click", resetCurrentGame);
closeVictoryBtn.addEventListener("click", closeVictory);
playAgainBtn.addEventListener("click", resetCurrentGame);
closeSolutionBtn.addEventListener("click", closeSolution);
victoryOverlay.addEventListener("click", event => {
if (event.target === victoryOverlay) closeVictory();
```

```
});
solutionOverlay.addEventListener("click", event => {
if (event.target === solutionOverlay) closeSolution();
});
document.addEventListener("keydown", event => {
if (event.key === "Escape") {
if (victoryOverlay.classList.contains("visible")) closeVictory();
if (solutionOverlay.classList.contains("visible")) closeSolution();
}
});
updateStateCountControl();
updateBoardControl();
ensureSession();
buildBoard();
</script>
</body>
</html>
```