UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Alberson Alison de Araújo

(GR)APHELIOS: UMA PLATAFORMA WEB INTERATIVA PARA O ENSINO E APRENDIZADO DE TEORIA DOS GRAFOS

Recife
2026

Alberson Alison de Araújo

(GR)APHELIOS: UMA PLATAFORMA WEB INTERATIVA PARA O ENSINO E APRENDIZADO DE TEORIA DOS GRAFOS

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal Rural de Pernambuco – UFRPE
Departamento de Estatística e Informática
Curso de Bacharelado em Sistemas de Informação

Orientador: Silvana Bocanegra

Recife
2026

A todos que exploram os fundamentos da Teoria dos Grafos.

AGRADECIMENTOS

Agradeço à minha mãe, Josivânia, e às minhas irmãs, Kesia e Eulalia, pelo apoio incondicional. Às minhas grandes amigas Taynah e Isis, e aos meus companheiros de vida Arion e Giuseppe, pelo afeto e incentivo diários. Estendo meus agradecimentos aos demais amigos e professores que acompanharam esta jornada. Por fim, expresso minha gratidão às instituições que foram fundamentais para a minha formação acadêmica e pessoal: o Instituto Federal do Rio Grande do Norte (IFRN) — Campus Caicó e a Universidade Federal Rural de Pernambuco (UFRPE).

“Enquanto eu estiver vivo, haverá infinitas possibilidades.”
(Eiichiro Oda)

RESUMO

A Teoria dos Grafos constitui uma das áreas fundamentais para a formação em Computação, sendo aplicada na modelagem de problemas que envolvem relações, conexões e percursos entre elementos. Apesar de sua relevância, o aprendizado de grafos e de seus algoritmos costuma apresentar dificuldades devido ao elevado nível de abstração exigido. Diante disso, foi proposta uma plataforma web interativa voltada ao ensino e à aprendizagem da Teoria dos Grafos, desenvolvida com base em React e estruturada com SVG para manipulação das representações visuais. A pesquisa envolveu a revisão da literatura, análise de ferramentas relacionadas e catalogação de recursos funcionais considerados fundamentais, como a presença de um canvas para construção dos grafos e o acompanhamento sequencial da execução dos algoritmos. A plataforma desenvolvida oferece suporte à criação e manipulação de grafos, controle de propriedades como peso e direcionamento, importação e exportação das estruturas desenvolvidas, validação automática de algoritmos compatíveis com o grafo criado e a execução passo a passo com apoio visual e pseudocódigo sincronizado. Como resultado, obteve-se um ambiente unificado e acessível via navegador conectado à internet, que integra os recursos essenciais para o trabalho com grafos, apoiando a experimentação prática e a compreensão das etapas envolvidas na resolução lógica dos problemas.

Palavras-chave: Algoritmos, Visualização de Algoritmos, Objeto de Aprendizagem, Grafos Computacionais, Simulação Visual.

ABSTRACT

Graph Theory is one of the fundamental areas in Computer Science education, as it is applied to the modeling of problems involving relationships, connections, and paths between elements. Despite its relevance, learning graphs and their algorithms often presents difficulties due to the high level of abstraction required. In this context, an interactive web platform was proposed to support the teaching and learning of Graph Theory, developed with React and structured with SVG for the manipulation of visual representations. The research involved a literature review, the analysis of related tools, and the cataloging of functional resources considered essential, such as the presence of a canvas for graph construction and the sequential monitoring of algorithm execution. The developed platform supports graph creation and manipulation, control of properties such as weight and direction, import and export of the developed structures, automatic validation of algorithms compatible with the created graph, and step-by-step execution with visual support and synchronized pseudocode. As a result, a unified environment accessible through an internet-connected browser was obtained, integrating the essential resources for working with graphs and supporting practical experimentation and the understanding of the steps involved in the logical resolution of problems.

Keywords: Algorithms, Algorithm Visualization, Learning Object, Computational Graphs, Visual Simulation.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

2D	Bidimensional
3D	Tridimensional
API	Application Programming Interface (Interface de Programação de Aplicações)
BFS	Breadth-First Search (Busca em Largura)
DFS	Depth-First Search (Busca em Profundidade)
CES	Câmara de Educação Superior
CNE	Conselho Nacional de Educação
DCNs	Diretrizes Curriculares Nacionais
DFS	Depth-First Search (Busca em Profundidade)
DOM	Document Object Model (Modelo de Objeto do Documento)
DOT	Linguagem DOT para descrição de grafos
ESM	ECMAScript Modules (Módulos ECMAScript)
HMR	Hot Module Replacement (Substituição de Módulos em Tempo Real)
HTML	HyperText Markup Language (Linguagem de Marcação de Hipertexto)
ID	Identifier (Identificador)
JSON	JavaScript Object Notation (Notação de Objetos JavaScript)
JUNG	Java Universal Network/Graph
MUI	Material UI
NP	Nondeterministic Polynomial Time (Tempo Polinomial Não Determinístico)
PNG	Portable Network Graphics
SVG	Scalable Vector Graphics (Gráficos Vetoriais Escaláveis)
TBC	Treinamento Baseado em Computador
UI	User Interface (Interface de Usuário)
XML	Extensible Markup Language (Linguagem de Marcação Extensível)

SUMÁRIO

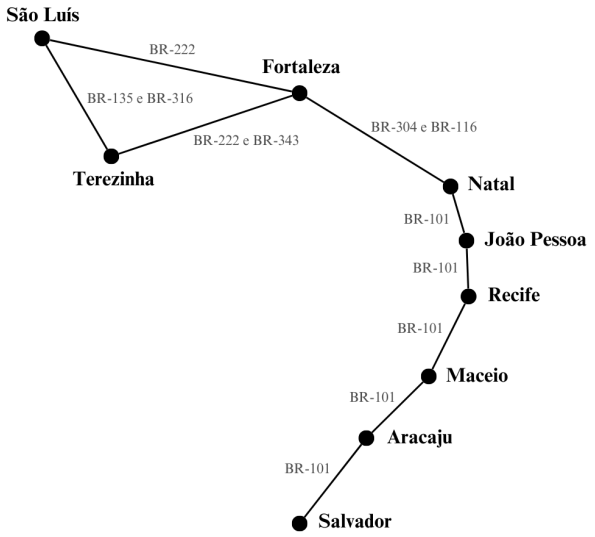
1 INTRODUÇÃO

O ensino de algoritmos e estruturas de dados constitui um pilar fundamental na formação de profissionais da área de Computação, ao proporcionar habilidades que lhes permitem analisar, modelar e resolver problemas de diferentes níveis de complexidade (CORMEN et al., 2012). Nesse contexto, a Teoria dos Grafos ganha destaque por oferecer ferramentas matemáticas que permitem representar e analisar relações entre elementos, contribuindo para a compreensão de estruturas complexas em múltiplas áreas do conhecimento (ROSEN, 2009).

O estudo dos grafos está presente principalmente em campos como redes de comunicação, transporte, esportes, bioinformática e engenharia (GOLDBARG; GOLDBARG, 2009). Ainda assim, a natureza abstrata dos conceitos envolvidos torna o aprendizado de grafos desafiador, especialmente para estudantes na fase inicial de sua formação acadêmica (MOCINECOVÁ; STEINGARTNER, 2020). Como exemplo ilustrativo de um grafo, apresenta-se na Figura 1 a representação das capitais da Região Nordeste do Brasil como pontos conectados pelas principais rodovias federais que as interligam.

A Figura 1 reforça como os grafos viabilizam a modelagem de problemas reais, sintetizando relações complexas em uma estrutura visualmente acessível. Essa capacidade de representação favorece não apenas a análise técnica, como também a comunicação de soluções em contextos multidisciplinares.

Figura 1 - Rodovias entre as capitais do nordeste brasileiro.



Fonte: Adaptado de BRASIL. Departamento Nacional de Infraestrutura de Transportes. Mapa do Brasil – Mapas Multimodais. Disponível em: https://www.gov.br/dnit/pt-br/assuntos/planejamento-e-pesquisa/dnit-geo/mapas-multimodais/mapa-do-brasil. Acesso em: 08 jul. 2025.

Adicionalmente, nas últimas décadas, a incorporação de tecnologias educacionais interativas têm promovido transformações significativas nas metodologias de ensino, favorecendo práticas pedagógicas mais visuais, dinâmicas e participativas (CHAVES; SANTOS, 2025). As ferramentas de visualização de algoritmos, por exemplo, permitem que os discentes acompanhem as transformações ocorridas nas estruturas de dados durante a execução de algoritmos em tempo real, potencializando a compreensão de conceitos e a fixação do conteúdo (LOURENÇO et al., 2022; SANTOS; COSTA, 2007). Plataformas baseadas na web se destacam nesse contexto por permitirem acesso direto via navegador, dispensando instalações locais e garantindo portabilidade ao funcionar em qualquer dispositivo conectado à internet.

Apesar dos avanços, muitos sistemas existentes ainda apresentam limitações, como interfaces pouco intuitivas, restrições na personalização de atividades e escassa interatividade para a experimentação ativa dos algoritmos (HAN et al., 2021). Além disso, nem sempre essas ferramentas reúnem, de forma integrada, a visualização gráfica da execução passo a passo de algoritmos, assim como a possibilidade de identificação e exploração das diferentes propriedades do grafo, o que dificulta a construção de uma experiência de aprendizagem completa, motivadora e adaptada às necessidades dos estudantes de Computação (MOCINECOVÁ; STEINGARTNER, 2020).

1.1 Motivação e Justificativa

A Teoria dos Grafos, pertencente à matemática discreta, constitui um pilar fundamental da Computação. No entanto, ela envolve conceitos abstratos e simbólicos que exigem do estudante habilidades de visualização espacial e de raciocínio bem desenvolvidas, o que a torna de difícil assimilação, especialmente nos estágios iniciais da formação acadêmica (MOCINECOVÁ; STEINGARTNER, 2020). Essa dificuldade pode gerar frustrações e inseguranças quanto à própria capacidade de aprendizado, além de reprovações e abandono das disciplinas, assim como do curso (SILVA; FRANCISCO, 2025).

Este cenário é corroborado por estudos que evidenciam as dificuldades enfrentadas por estudantes ao lidar com conteúdos que exigem abstração elevada, como algoritmos e estruturas de grafos. Deters et al. (2008) apontam que a disciplina de Algoritmos e Programação apresenta altas taxas de reprovação, especialmente entre alunos ingressantes, destacando que a complexidade dos conceitos e a ausência de abordagens pedagógicas mais adequadas contribuem para esse cenário. Além disso, Tilanterä et al. (2024) demonstram que,

mesmo após a instrução formal, estudantes continuam apresentando dificuldades em compreender os fundamentos dos algoritmo de caminhos mínimos, sobretudo no que se refere à representação do problema e à visualização de sua execução, o que reforça a necessidade de recursos que tornem o processo de aprendizagem mais tangível e acessível.

Outrossim, ferramentas digitais que têm como função elucidar os conceitos das estruturas e algoritmos em grafos e que estão disponíveis atualmente apresentam limitações significativas, como a falta de ambientes interativos, ausência de recursos integrados de execução de algoritmos, dificuldades na personalização de atividades e carência de etapas de exploração passo a passo (HAN et al., 2021). Diante dessas barreiras, tais serviços não conseguem oferecer o suporte necessário à autonomia do estudante, tampouco estimulam um processo de aprendizagem verdadeiramente ativo. É essencial, portanto, buscar soluções tecnológicas que sejam acessíveis, flexíveis e que promovam o protagonismo discente no estudo da Teoria dos Grafos (CHAVES; SANTOS, 2025).

1.2 Objetivos

Desenvolver uma plataforma web interativa voltada ao ensino da Teoria dos Grafos, reunindo de forma integrada os principais recursos identificados em ferramentas já existentes, ao mesmo tempo em que aprimora a experiência de aprendizagem por meio de uma abordagem centrada na clareza conceitual. A proposta busca oferecer um ambiente unificado, acessível e didático, que amplie as possibilidades de exploração dos conteúdos relacionados a grafos.

Para atingir esse objetivo, foram elaborados os seguintes objetivos específicos:

- Realizar uma revisão da literatura e análise de ferramentas existentes voltadas ao ensino da Teoria dos Grafos, identificando boas práticas e limitações funcionais;
- Catalogar os recursos fundamentais identificados nas ferramentas existentes;
- Analisar criticamente qual ferramenta é a mais adequada para a construção gráfica dos grafos, considerando critérios de integração e flexibilidade;
- Definir os algoritmos de grafos a serem contemplados na plataforma, com base em sua relevância didática;
- Desenvolver a plataforma com base nas tecnologias selecionadas e nos recursos definidos previamente.

1.3 Estrutura do Trabalho

Este trabalho está organizado em seis capítulos. O Capítulo 1 apresenta a introdução da pesquisa, contextualizando a importância da Teoria dos Grafos na formação em Computação, os desafios relacionados ao seu ensino e a motivação para o desenvolvimento de uma plataforma web interativa. Neste capítulo também são apresentados a justificativa, os objetivos geral e específicos e a organização geral do texto.

O Capítulo 2 reúne a fundamentação teórica utilizada como base para o desenvolvimento da proposta. Inicialmente, são apresentados conceitos fundamentais da Teoria dos Grafos, incluindo suas estruturas, representações e algoritmos. Em seguida, são discutidos desafios relacionados ao ensino desses conteúdos, o papel das tecnologias digitais no processo de aprendizagem e a análise de ferramentas e trabalhos relacionados ao ensino e à visualização de grafos.

O Capítulo 3 descreve a metodologia adotada para o desenvolvimento da plataforma. Nele são apresentadas as tecnologias utilizadas, a organização da arquitetura da aplicação, a interface, a estrutura dos dados manipulados, o gerenciamento de estados e a forma como os algoritmos foram implementados no sistema.

O Capítulo 4 apresenta os resultados obtidos com o desenvolvimento do (gr)Aphelios. Neste capítulo, são abordadas as funcionalidades da plataforma, como a construção e manipulação de grafos, os recursos de importação e exportação, a execução e visualização dos algoritmos e a discussão dos recursos implementados em relação às necessidades identificadas ao longo da pesquisa.

O Capítulo 5 aborda o desenvolvimento realizado e as conclusões acerca da plataforma desenvolvida. São sintetizadas as principais contribuições da pesquisa, desde o levantamento teórico até as decisões adotadas durante a implementação do sistema, destacando sua aplicação ao ensino da Teoria dos Grafos. Por fim, são abordadas as limitações do estudo, especialmente quanto à ausência de testes com usuários em cenário real.

Por fim, o Capítulo 6 apresenta as considerações finais e os trabalhos futuros. Nele, são discutidas as limitações que a plataforma possui e as ideias para a continuidade de seu desenvolvimento, que envolvem tanto questões relacionadas à ampliação dos recursos oferecidos pela ferramenta quanto a sua disponibilização em ambiente de produção, com validação em cenários reais de ensino da Teoria dos Grafos.

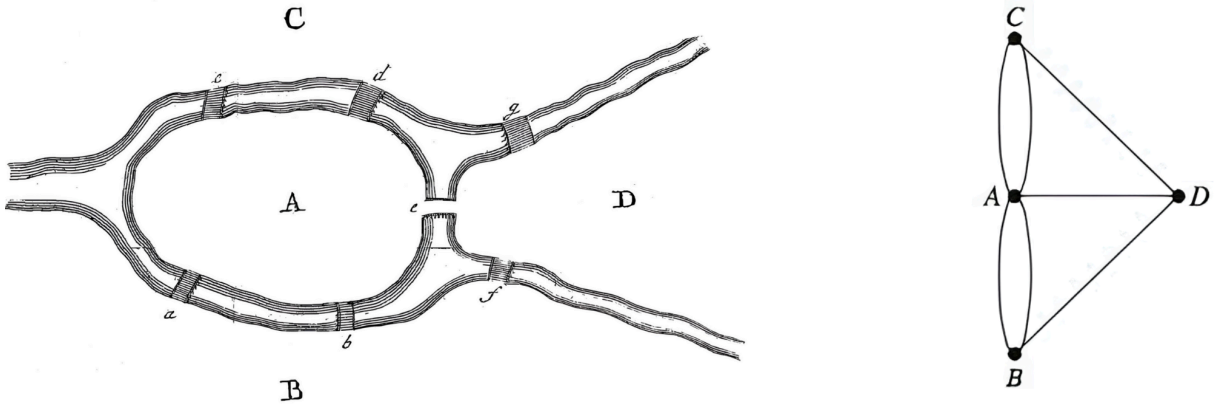
2 FUNDAMENTAÇÃO TEÓRICA

2.1 Teoria dos Grafos

A Teoria dos Grafos é um ramo da matemática discreta que estuda estruturas formadas por elementos chamados vértices ou nós e suas interconexões, denominadas arestas ou ligações. Essas estruturas, conhecidas como grafos, são amplamente utilizadas na modelagem e resolução de problemas que envolvem relações entre pares de entidades. Seja na análise de redes de transporte, na representação de interações sociais ou na construção de algoritmos eficientes, os grafos se destacam como ferramentas matemáticas de notável versatilidade e poder expressivo (GOLDBARG; GOLDBARG, 2009; ROSEN, 2009).

O marco inicial da Teoria dos Grafos é atribuído ao matemático suíço Leonhard Euler, que em 1735 resolveu o conhecido Problema das Sete Pontes de Königsberg, esquematizado na Figura 2. A questão propunha encontrar um caminho que cruzasse todas as pontes da cidade exatamente uma vez, retornando ao ponto de partida, e Euler demonstrou que o trajeto era impossível ao utilizar uma abstração baseada em pontos e conexões. Essa solução não apenas resolveu o problema de forma inovadora, como também inaugurou o campo da geometria de posição e introduziu a ideia de representar relações por meio de grafos (ALEXANDERSON, 2006).

Figura 2 - As sete pontes de Königsberg e seu modelo multigrafo.



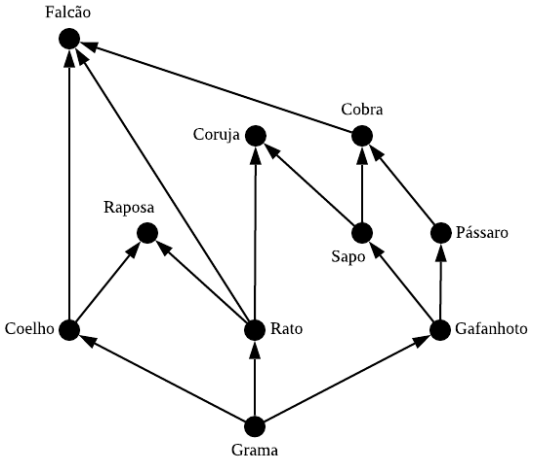
Fonte: Adaptado de Rosen (2009)

Com o passar do tempo, a Teoria dos Grafos foi formalizada no contexto matemático e, posteriormente, expandida. Consolidou-se como um dos pilares teóricos da ciência da computação pela sua capacidade de representar sistemas complexos, nos quais as relações

entre os elementos são tão relevantes quanto eles próprios, característica destacada por Goldbarg e Goldbarg (2009) ao abordarem a natureza relacional dos grafos.

Um exemplo prático de aplicação dos grafos pode ser dado na área ecológica, como no caso das cadeias alimentares, que representam as relações de predação entre diferentes espécies dentro de um ecossistema, ilustrado na Figura 3. A modelagem dessas interações por meio de grafos permite uma representação mais clara dos níveis tróficos e facilita a visualização das conexões entre os organismos.

Figura 3 - Cadeia alimentar do falcão.



Fonte: Adaptado de MUNDO EDUCAÇÃO. Teia alimentar. Disponível em: https://mundoeducacao.uol.com.br/biologia/teia-alimentar.htm. Acesso em: 07 jul. 2025.

A importância dos grafos também se evidencia em aplicações cotidianas, como o mapeamento de relações interpessoais, nas quais os vértices representam indivíduos e as arestas indicam vínculos de interação (ROSEN, 2009). Consoante a isso, em contextos computacionais contemporâneos, algoritmos sobre grafos são empregados na análise de redes sociais, possibilitando ações como a formação de agrupamentos e a avaliação de distâncias relacionais entre os indivíduos (LOURENÇO et al., 2022).

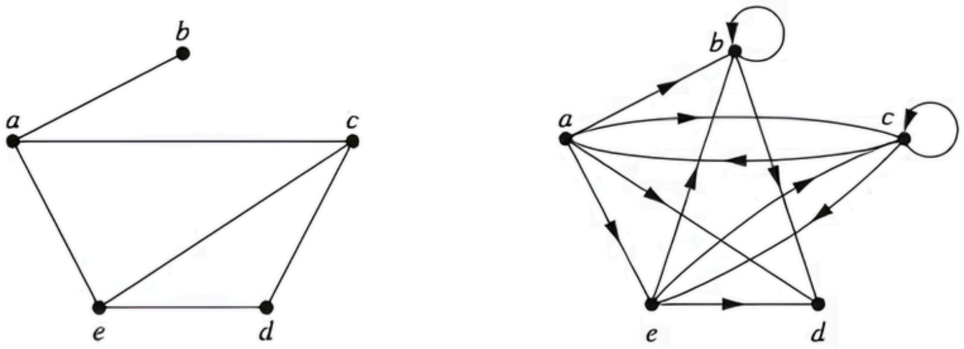
Dessa forma, a Teoria dos Grafos destaca-se não apenas por sua base matemática consolidada, mas também por sua relevância prática na modelagem de problemas em diferentes contextos. Sua presença é recorrente em diversas áreas da computação, especialmente naquelas que exigem o tratamento estruturado de relações e conexões entre elementos (GOLDBARG; GOLDBARG, 2009; ROSEN, 2009).

2.1.1 Estruturas e Definições Básicas

A formalização da Teoria dos Grafos fundamenta-se na definição de suas estruturas elementares. De acordo com Rosen (2009), um grafo é uma dupla ordenada $G = (V, E)$, em que V é um conjunto finito de vértices, ou nós, e E é um conjunto de arestas, ou ligações, que conectam pares de vértices. As arestas podem ser representadas como pares não ordenados $\{u, v\}$, no caso de grafos não direcionados, ou como pares ordenados (u, v), no caso de grafos direcionados. Essa estrutura permite modelar relações simétricas ou assimétricas entre entidades, o que torna os grafos altamente versáteis na representação de problemas reais (GOLDBARG; GOLDBARG, 2009).

A natureza das conexões entre os vértices é a base da definição, como ilustrado na Figura 4, o grafo $K1$, à esquerda, é um exemplo de grafo simples, no qual não há laços, isto é, arestas que ligam um vértice a ele mesmo, nem múltiplas arestas entre os mesmos pares de vértices. Já o grafo $K2$, à direita, ilustra um grafo direcionado com múltiplas conexões entre vértices distintos e com laços observados nos vértices b e c (ROSEN, 2009; GOLDBARG; GOLDBARG, 2009).

Figura 4 - Exemplos de grafos não direcionado e direcionado.

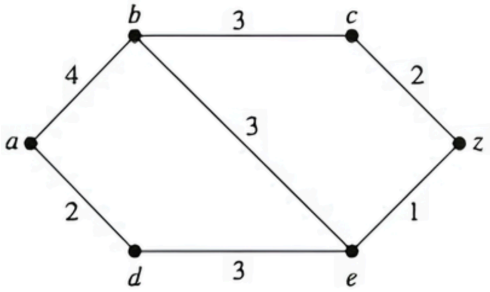


K1 K2

Fonte: Adaptado de Rosen (2009)

Além disso, os grafos podem ser classificados como ponderados, quando as arestas possuem valores numéricos associados que são equivalentes a custo, tempo ou distância (ROSEN, 2009), conforme representado na Figura 5 .

Figura 5 - Exemplo de grafo ponderado.

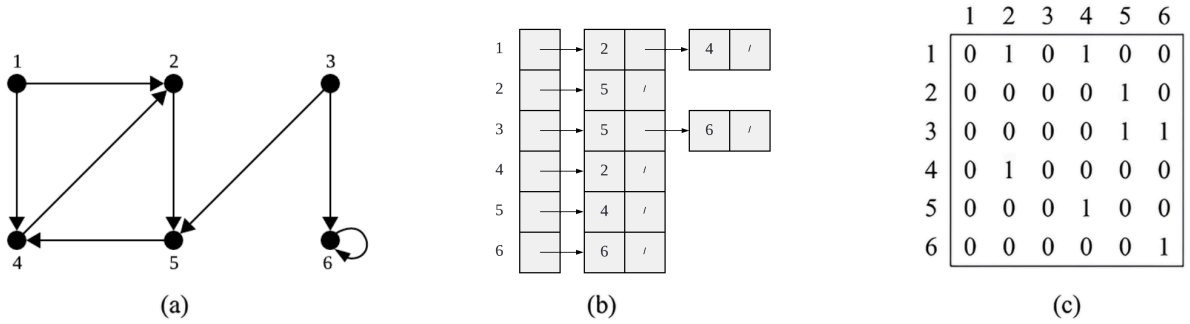


K3

Fonte: Adaptado de Rosen (2009)

Outrossim, existem formas estruturadas amplamente utilizadas no processamento computacional dos grafos. As duas principais incluem a lista de adjacência e a matriz de adjacência, ambas ilustradas na Figura 6. A lista de adjacência (b) organiza, para cada vértice, o conjunto de vértices aos quais ele está conectado, oferecendo uma representação mais eficiente em casos de grafos esparsos. Por sua vez, a matriz de adjacência (c) é uma tabela quadrada em que cada célula indica a existência de uma aresta entre dois vértices (ROSEN, 2009; CORMEN, 2012).

Figura 6 - Exemplo de lista e matriz de adjacências.



	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Fonte: Adaptado de Cormen et al. (2012)

Ademais, a Teoria dos Grafos abrange uma variedade de conceitos e propriedades que ampliam significativamente sua aplicabilidade prática. Dentre eles, destacam-se grafos conexos, bipartidos, completos, planos, árvores, ciclos, coloração, assim como diversas outras definições que possibilitam representar relações complexas e modelar problemas nos mais variados contextos (ROSEN, 2009). Essas definições formam a base conceitual sobre a qual se desenvolvem os algoritmos aplicados à Teoria dos Grafos, a clareza na distinção entre os tipos e suas propriedades é essencial para a correta compreensão dos algoritmos.

2.1.2 Algoritmos

Um algoritmo pode ser definido como uma sequência finita de instruções bem definidas que, partindo de um conjunto de entradas, produzem uma saída e resolvem um problema. Conforme CORMEN et al. (2012), algoritmos são procedimentos computacionais que transformam dados de entrada em resultados por meio de etapas logicamente estruturadas, podendo ser representados simplificadamente por meio de um pseudocódigo, uma linguagem intermediária entre a linguagem natural e a programação formal, usada para descrever a lógica de forma clara e independente de uma sintaxe específica.

Para que os algoritmos sejam corretamente expressos e executados, eles se apoiam em estruturas fundamentais que são intrinsecamente responsáveis pelo seu funcionamento. Apesar da variedade de mecanismos, os fundamentais são: variáveis, que guardam valores de diferentes naturezas; vetores, que armazenam sequências de valores; estruturas condicionais, como "se" e "se não", que executam comandos com base em verificações lógicas; e estruturas de repetição, como "para" e "enquanto", que mantêm a execução de um bloco até que determinada condição seja satisfeita. Além disso, é comum o uso de funções, que organizam conjuntos de instruções com finalidade específica. (CORMEN et al., 2012).

No contexto da Teoria dos Grafos, os algoritmos desempenham um papel essencial ao viabilizar a manipulação e a análise eficiente dessas estruturas. Por meio deles, é possível responder a diversas questões, como percorrer um grafo em busca de um vértice específico, verificar a existência de rotas entre dois pontos, identificar propriedades estruturais, determinar o menor caminho entre vértices, entre outros (GOLDBARG; GOLDBARG, 2009; CORMEN et al., 2012).

Dada a ampla gama de problemas que envolvem estruturas gráficas, uma variedade de algoritmos foi desenvolvida para atender a finalidades específicas dentro da Teoria dos Grafos. Entre as abordagens mais recorrentes estão os algoritmos de busca, que percorrem os vértices de forma sistemática; os algoritmos de caminhos mínimos, voltados à otimização de trajetos com base em pesos ou distâncias; e os algoritmos de árvores geradoras mínimas, utilizados na construção de estruturas conectadas com custo reduzido. Esses exemplos, comumente apresentados em obras como as de CORMEN et al. (2012) e GOLDBARG e GOLDBARG (2009), representam apenas pequena parte das diversas categorias existentes, que incluem ainda algoritmos como coloração, fluxo máximo, emparelhamento e outros.

2.1.2.1 Algoritmos de buscas

Os algoritmos de busca são alguns dos mais simples e se destacam por sua função essencial de percorrer todos os vértices de um grafo a partir de um ponto inicial. As abordagens de busca em largura (BFS) e busca em profundidade (DFS) realizam essa tarefa de formas diferentes: enquanto a BFS visita os vértices em camadas, avançando gradualmente para os níveis seguintes, a DFS explora cada caminho até o final antes de retroceder (CORMEN et al., 2012; GOLDBARG; GOLDBARG, 2009). Um pseudocódigo do algoritmo de busca em profundidade é mostrado no Algoritmo 1.

Algoritmo 1 - Pseudocódigo do algoritmo BFS.

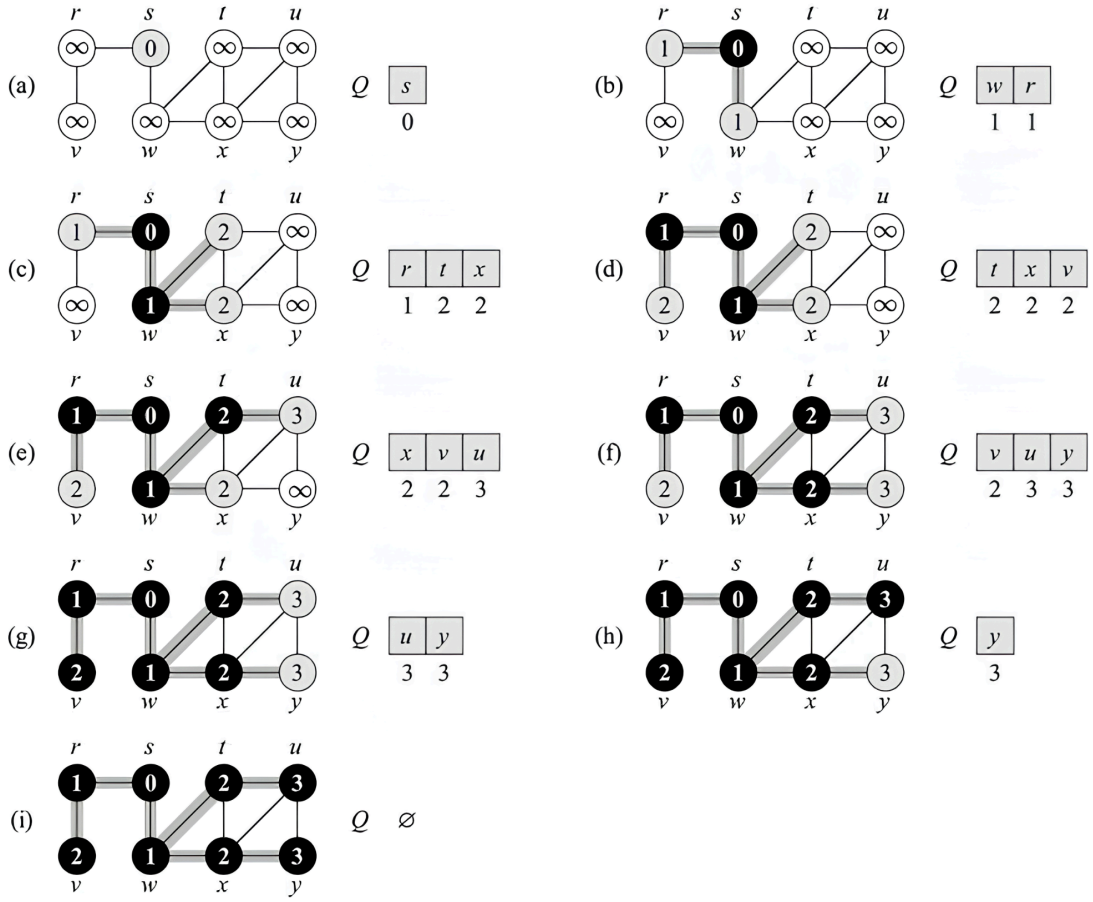
```
1  BFS(G, s)
2      para cada u ∈ V[G] − {s}
3          u.cor = branco
4          u.d = ∞
5          u.π = nulo
6      s.cor = cinza
7      s.d = 0
8      s.π = nulo
9      Q = ∅
10     Enfileirar(Q, s)
11     enquanto Q ≠ ∅
12         u = Desenfileirar(Q)
13         para cada v = Adjacentes[u]
14             se v.cor == branco
15                 v.cor == cinzento
16                 v.d = u.d + 1
17                 v.π = u
18                 Desenfileirar(Q, v)
19         u.cor = preto
```

Fonte: Adaptado de Cormen et al. (2012)

O funcionamento do algoritmo de busca em largura, ilustrado na Figura 7, parte de um grafo G = (V, E), direcionado ou não direcionado, e de um vértice inicial s, explorando o grafo em camadas, visitando primeiro todos os vértices a uma distância d=1, depois os de d=2, e assim sucessivamente. Inicialmente, todos os vértices são marcados como não visitados

(branco), com distância indefinida e sem predecessores. O vértice de origem s é então marcado como descoberto (cinza), recebe distância zero e é inserido em uma fila. A cada iteração, um vértice u é removido da fila e seus vizinhos ainda são marcados como descobertos e adquirem a distância $u.d + 1$, têm u como predecessor e são enfileirados. Após visitar todos os vizinhos de u, ele é marcado como totalmente processado (preto). Esse procedimento garante que os vértices sejam explorados na ordem em que são alcançados (CORMEN et al., 2012).

Figura 7 - Funcionamento da BFS em um grafo não direcionado.



Fonte: (CORMEN et al., 2012)

2.1.2.2 Algoritmos de caminhos mínimos

Enquanto os algoritmos de busca se concentram em percorrer os vértices de um grafo a fim de encontrar um destino, os algoritmos de caminhos mínimos não se preocupam apenas em alcançar o vértice-alvo, mas em determinar qual trajeto apresenta o menor custo até ele. Essa classe de algoritmos é fundamental para problemas de otimização em grafos ponderados

(CORMEN et al., 2012; GOLDBARG; GOLDBARG, 2009) e está presente em aplicações reais como o cálculo de rotas em sistemas de navegação e redes de comunicação

Entre os algoritmos clássicos para essa classe de problema, conforme apresentado por Cormen et al. (2012), o algoritmo de Dijkstra se destaca na resolução de caminhos mínimos em grafos ponderados. Sua proposta é encontrar o caminho de menor custo de um vértice de origem para os demais vértices de um grafo ponderado, desde que os pesos de todas as arestas sejam positivos (CORMEN et al., 2012). A lógica de sua execução consiste em manter um conjunto de vértices cujas distâncias mínimas já foram determinadas e, iterativamente, escolher o vértice de menor distância ainda não processado, conforme detalhado no pseudocódigo do Algoritmo 2.

Algoritmo 2 - Pseudocódigo do algoritmo de Dijkstra.

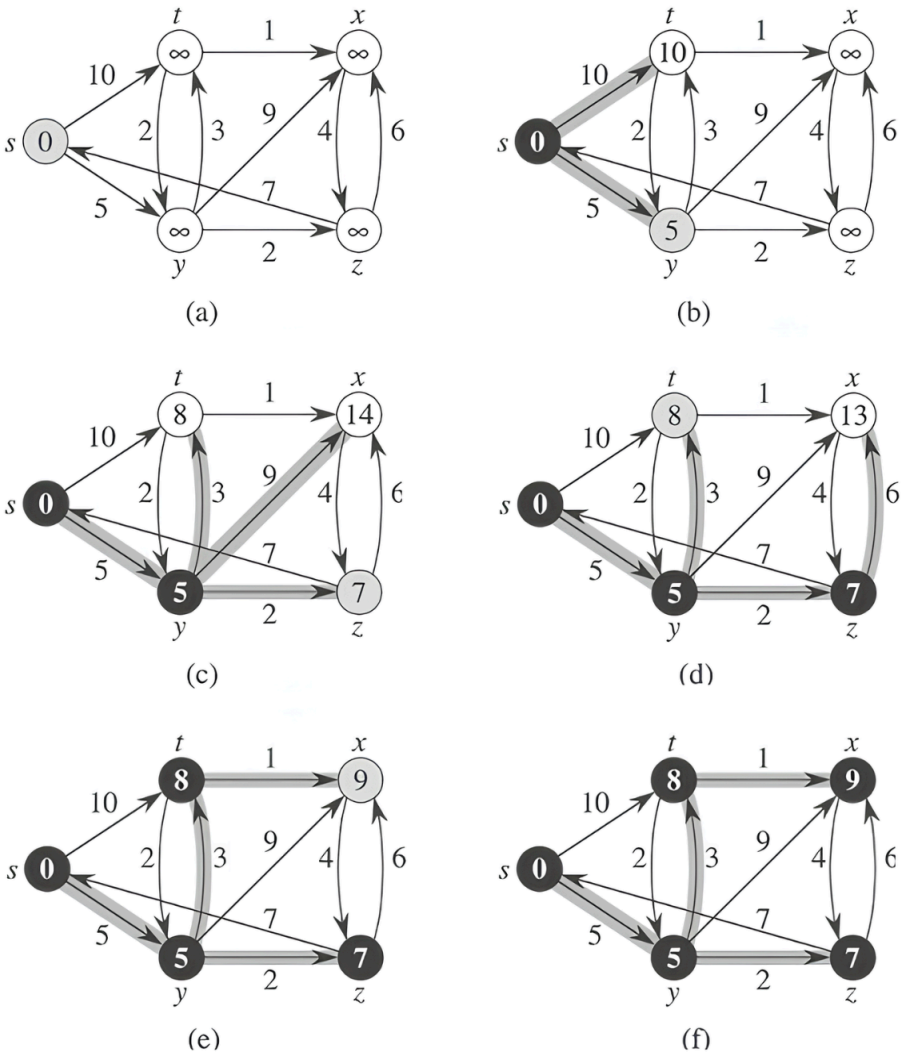
```
1 Dijkstra(G, w, s)
2       Inicializa(G, s)
3       A = ∅
4       Q = V[G]
5       enquanto Q ≠ ∅
6             u = Extrair-minimo(Q)
7             A = A ∪ {u}
8             para cada vertice v ∈ G.Adjacentes[u]
9                   Relaxa(u, v, w)
```

Fonte: Adaptado de Cormen et al. (2012)

O algoritmo recebe como entrada um grafo (G), uma função de pesos associada às arestas (w) e um vértice de origem (s), a partir do qual os menores caminhos serão calculados. O processo inicia com a chamada da função $Inicializa(G, s)$ que, semelhante ao BFS, irá inicialmente atribuir a todos os vértices, exceto s, uma distância infinita. Em seguida, todos os vértices do grafo são inseridos na fila de prioridade Q, e um conjunto A é criado para armazenar os vértices que já tiveram suas menores distâncias determinadas.

A cada iteração, o vértice u com menor distância estimada é removido da fila por meio da função $Extrair\text{-}minimo(Q)$, sendo adicionado ao conjunto A. Para cada vizinho v de u, a função $Relaxa(u, v, w)$ verifica se o caminho até v, por meio de u, oferece um custo menor que o já registrado, Quando isso ocorre, a distância e o predecessor de v são atualizados. Esse processo se repete até que todos os vértices tenham sido processados, mapeando as menores distâncias, conforme ilustrado na Figura 8.

Figura 8 - Funcionamento do algoritmo de Dijkstra.



Fonte: (CORMEN et al., 2012)

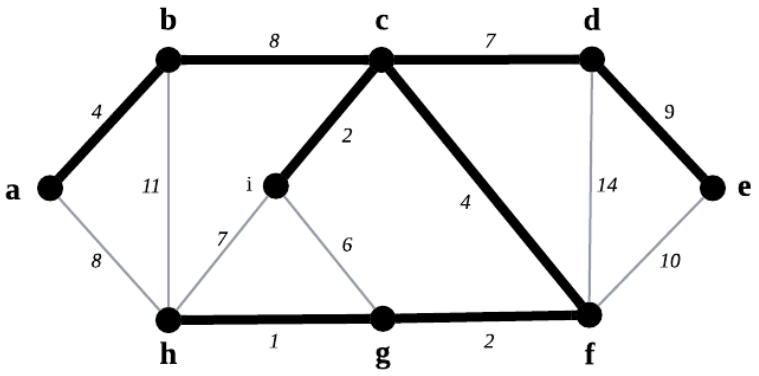
2.1.2.3 Algoritmos de árvores geradoras mínimas

Diferentemente dos algoritmos de busca e de caminhos mínimos, os algoritmos de árvores geradoras mínimas têm como objetivo construir uma subestrutura que conecte todos os vértices de um grafo de forma eficiente, sem formar ciclos e com o menor custo total possível (CORMEN et al., 2012). Essa estrutura, denominada árvore geradora mínima, é especialmente relevante em contextos que exigem a conexão entre elementos mantendo-se a simplicidade e a economia.

> Em projeto de circuitos eletrônicos, muitas vezes, é necessário que os pinos de vários componentes se tornem eletricamente equivalentes, o que é conseguido ligando-os uns aos outros. Para interconectar um conjunto de n pinos, podemos usar um arranjo de n - 1 fios, cada qual conectando dois pinos. De todos os arranjos possíveis, aquele que utiliza a mínima quantidade de fio é o mais desejável (CORMEN et al., 2012, p. 511)

A Figura 9 ilustra um exemplo simples de árvore geradora mínima, em que todas as conexões necessárias entre os vértices são mantidas com o menor somatório possível dos pesos das arestas, eliminando quaisquer ciclos. Os pesos estão indicados em cada uma das arestas e aquelas que fazem parte da composição da árvore encontram-se destacadas.

Figura 9 - Árvore geradora mínima de um grafo conexo.



Fonte: (CORMEN et al., 2012)

Entre os algoritmos clássicos para a construção de árvores geradoras mínimas, conforme apresentado por Cormen et al. (2012), o algoritmo de Prim destaca-se por construir a solução de forma incremental. O algoritmo recebe como entrada um grafo (G) ponderado e conexo, e produz como saída uma árvore geradora mínima. Ele parte de um vértice inicial e, a cada etapa, adiciona à árvore o vértice mais próximo ainda não incluído, considerando a aresta de menor peso que conecta um vértice pertencente à árvore a um vértice externo. Esse processo se repete até que todos os vértices estejam conectados, garantindo uma estrutura sem ciclos e com o custo total mínimo, como apresentado no pseudocódigo do Algorítmos 3.

Algoritmo 3 - Pseudocódigo do algoritmo de Prim.

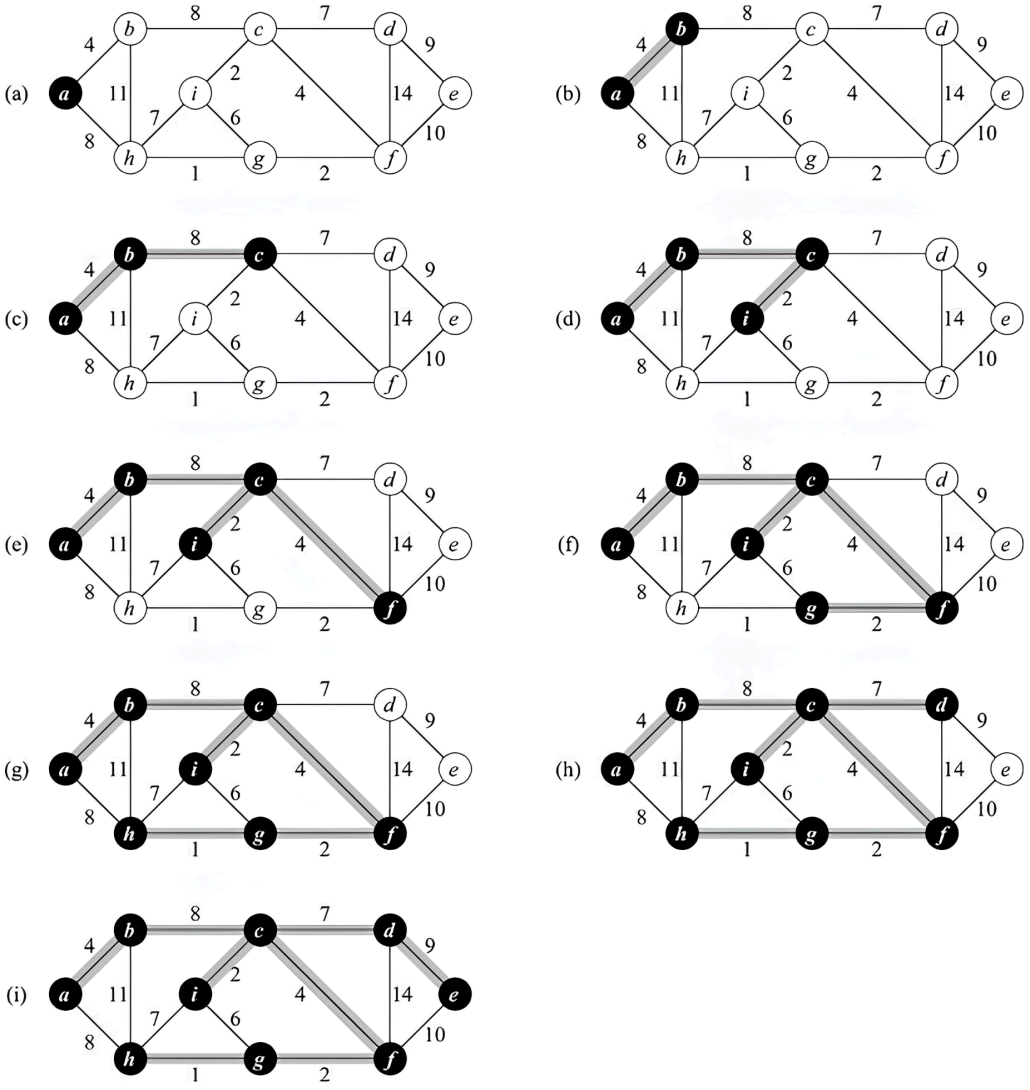
```
1  PRIM(G, w, r)
2      para cada u ∈ V[G]
3          u.chave = ∞
4          u.π = nulo
5      r.chave = 0
6      Q = V[G]
11     enquanto Q ≠ ∅
12         u = Extrair-minimo(Q)
13         para cada v ∈ G.Adjacentes[u]
14             se v ∈ Q e w(u, v) são < v.chave
17                 v.π = u
19                 v.chave = w(u, v)
```

Fonte: Adaptado de Cormen et al. (2012)

Assim como os dois algoritmos anteriores, o de Prim também inicia com a definição de um vértice de origem r, ao qual é atribuída a chave zero, enquanto todos os demais vértices recebem valor infinito e têm seus predecessores definidos como nulos. A partir desse ponto, o algoritmo percorre a fila de prioridade Q, sempre selecionando o vértice com menor valor de chave ainda não incluído na árvore. Para cada vizinho v do vértice u removido da fila, verifica-se se ele ainda permanece na fila e se o peso da aresta $w(u, v)$ que o conecta possui um custo inferior ao atualmente armazenado em sua chave. Se ambas as condições forem satisfeitas, os valores de chave e de predecessor desse vizinho são atualizados. Esse processo se repete até que todos os vértices tenham sido processados.

A Figura 10 ilustra uma execução do algoritmo de Prim sobre um grafo ponderado conexo, o mesmo grafo apresentado na Figura 9, evidenciando agora o crescimento progressivo da árvore geradora mínima.

Figura 10 - Funcionamento do algoritmo de Prim.



Fonte: (CORMEN et al., 2012)

Assim como nos algoritmos anteriores, a simulação visual do processo de execução do código contribui significativamente para a compreensão conceitual. Ao permitir que cada etapa seja acompanhada de forma ilustrada, a Figura 10, igualmente as Figuras 7 e 8, torna tangível o funcionamento dos mecanismos internos do algoritmo e favorece o entendimento correto da lógica por trás de cada decisão tomada.

Conclui-se, portanto, que mesmo com apenas três algoritmos da Teoria dos Grafos destacados, verifica-se que a experimentação por meio de métodos visuais contribui para o seu entendimento, auxilia no processo de aprendizagem e deve ser utilizada nos processos de ensino como ferramenta de suporte.

2.2 Desafios no Ensino dos Grafos

No Brasil, o ensino da Teoria dos Grafos nos cursos superiores da área de Computação é oficialmente respaldado pelas Diretrizes Curriculares Nacionais (DCNs). A Resolução CNE/CES nº 5, de 16 de novembro de 2016, determina que a formação acadêmica dessa área deve contemplar fundamentos teóricos e práticos relacionados a estruturas algorítmicas, matemáticas e de dados, com a Teoria dos Grafos destacada como componente essencial e obrigatório para o ensino (BRASIL, 2016).

Por mais que esteja formalmente assegurada nas diretrizes curriculares, a Teoria dos Grafos não possui uma trajetória de aprendizagem tranquila. A área da Computação, de modo geral, apresenta um grau elevado de complexidade e exige dos estudantes níveis progressivos de abstração, organização lógica e raciocínio formal, como é apresentado nas obras de Cormen et al. (2012) e Rosen (2009), que introduzem os algoritmos destacando a importância dessas habilidades cognitivas para a resolução de problemas complexos. Esses desafios cognitivos exigidos afetam transversalmente toda a formação e, segundo Deters et al. (2008), essas dificuldades contribuem para taxas de reprovação que, em alguns casos, superam 50%.

Os Grafos possuem características próprias que intensificam a complexidade do seu ensino. Para além da base matemática envolvida, sua compreensão exige que o estudante transite entre diferentes representações, como as matrizes de adjacência, listas de arestas e estruturas visuais, o que amplia significativamente a carga cognitiva necessária para entendimento dos conceitos, conforme discutido por Rosen (2009), ao tratar da complexidade envolvida na interpretação e transição entre as diferentes representações.

Alinhado a esse contexto, Wahyuningsih et al. (2020) destacam que muitos estudantes demonstram dificuldades para compreender os conceitos relacionados a grafos e sua aplicação na resolução de problemas, especialmente devido à falta de familiaridade prévia com esse conteúdo. Adicionalmente, Santos e Costa (2007) reforçam esse cenário ao apontarem que a ausência de ferramentas de visualização capazes de auxiliar os estudantes na construção do entendimento conceitual também constitui um fator crítico no processo de aprendizagem.

A relevância desse fator é evidenciada por diversos estudos que demonstram como a ausência de recursos visuais pode comprometer o aprendizado de estruturas complexas, como os grafos, de forma significativa. Silva e Francisco (2025), ao desenvolverem a ferramenta QuatiView, observaram que a maioria das ferramentas disponíveis apresenta baixa legibilidade para esse tipo de estrutura, pois utilizam esquemas de visualização que não preservam a organização lógica dos elementos. Essa limitação se intensifica diante da

necessidade de acompanhar o funcionamento dos algoritmos, o que, segundo Mocinecová e Steingartner (2020), é facilitado por ferramentas que dispõem de funcionalidades que vão além da visualização, como a possibilidade de manipular as etapas da execução.

Dessa forma, os desafios que envolvem o ensino da Teoria dos Grafos não se restringem à complexidade conceitual do conteúdo, mas abrangem também as limitações pedagógicas e tecnológicas que dificultam sua mediação em sala de aula. A ausência de recursos que ofereçam suporte visual e interativo, somada à carência de estratégias que aproximem teoria e prática por parte dos docentes, compromete o engajamento dos estudantes e sua capacidade de internalizar os conceitos fundamentais dos grafos.

2.3 Tecnologias Digitais no Ensino de Grafos

O uso de tecnologias digitais como apoio didático tem se mostrado essencial no ensino da Computação, especialmente quando se busca tornar conteúdos abstratos mais acessíveis aos estudantes. Santos e Costa (2007) argumentam que “os produtos de software que enfatizam a animação gráfica são de grande importância como facilitadores do processo de aprendizagem”, entende-se que eles tornam a apresentação dos conteúdos mais didática e favorecem a assimilação de tópicos complexos. No ensino de algoritmos em grafos, que demanda a compreensão de estruturas e processos dinâmicos, ferramentas interativas baseadas na web, como as discutidas por Lourenço et al. (2022), ampliam a aprendizagem ao oferecer uma interface acessível a partir de qualquer navegador com conexão à internet.

Nesse sentido, Chaves e Santos (2025) discutem como o uso de tecnologias digitais pode redefinir o papel de educadores e alunos, criando novas dinâmicas de ensino que possibilitam experiências mais autônomas, colaborativas e contextualizadas. Deters et al. (2008) e Chen, Lambert e Guidry (2010) corroboram esse ponto ao afirmarem que o uso de ferramentas computacionais, quando bem integradas ao currículo, contribui para reduzir índices de evasão e reprovação, especialmente em disciplinas com alto grau de dificuldade, como estruturas de dados e algoritmos, que são partes fundamentais da teoria dos grafos.

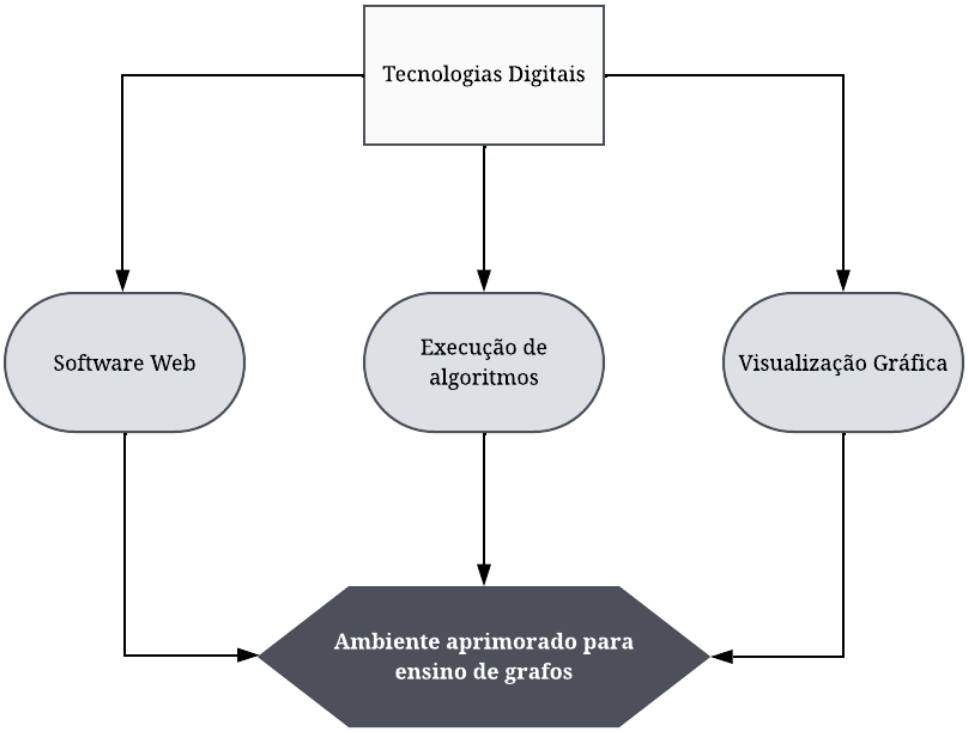
Dentro desse panorama, Quinn (2015) ressalta que as ferramentas gráficas desempenham um papel central na aprendizagem de temas complexos, pois permitem que os estudantes acompanhem visualmente as etapas e os processos executados por um algoritmo, o que contribui para o desenvolvimento da compreensão estrutural dos grafos. Adicionalmente, Mocinecová e Steingartner (2020) ressaltam que ferramentas interativas com suporte visual favorecem significativamente a compreensão dos algoritmos ao permitir que o usuário

acompanhe graficamente suas etapas de execução, como ocorre nos grafos, envolvendo a tomada de decisões, a definição das estruturas e dos percursos.

Esse tipo de ferramenta, quando disponibilizado em ambiente web, oferece ainda mais vantagens para o ensino ao disponibilizar acessibilidade e portabilidade em diferentes dispositivos, conforme apontado por Santos e Costa (2005). Consoante a isso, Lourenço et al. (2022) destacam que o uso de interfaces gráficas executadas no navegador facilita o acesso por parte dos estudantes e amplia as possibilidades de utilização em diferentes contextos educacionais, benefícios esses que também são reforçados por outros estudos, como os de Tilanterä et al. (2024) e Wahyuningsih et al. (2020).

Dessa forma, observa-se que as tecnologias digitais, especialmente aquelas em ambiente web com suporte visual dos grafos e da execução de seus algoritmos, representam uma estratégia pedagógica eficaz para o ensino da Teoria dos Grafos. Ao tornar visíveis os processos algorítmicos e permitir a exploração interativa dos conceitos, esse modelo de ferramentas, esquematizado na Figura 11, contribui para fortalecer o engajamento dos estudantes e promover a compreensão dos fundamentos estruturais dos grafos.

Figura 11 - Diagrama da estrutura de suporte digital para o ensino de grafos.



Fonte: O autor (2025)

2.4 Ferramentas Relacionadas

A avaliação de soluções digitais gratuitas voltadas à manipulação de grafos, disponibilizadas em plataformas web, mostra-se fundamental para compreender o cenário atual de recursos aplicados ao ensino da Teoria dos Grafos. Essas ferramentas oferecem meios práticos de representar visualmente conceitos abstratos, contribuindo diretamente para a mediação didática e a consolidação do aprendizado. Nesse contexto, busca-se identificar e analisar funcionalidades, padrões de interação e aspectos visuais que possam servir de referência para o desenvolvimento da solução proposta.

Recursos de visualização e simulação tornaram-se centrais para favorecer a compreensão de conceitos complexos, contribuindo significativamente para o processo de aprendizagem de algoritmos e estruturas de dados (CHEN et al., 2010). Hoje, diversas plataformas web disponibilizam ambientes interativos de desenho, execução e análise de grafos, constituindo referências importantes para professores, estudantes e desenvolvedores que desejam evitar a reinvenção de soluções já consolidadas. Assim como destacam Chaves e Santos (2025), essas ferramentas são fundamentais para potencializar a mediação pedagógica e fomentar práticas de ensino mais dinâmicas e acessíveis.

Para arquitetar a análise, foram mapeadas plataformas relevantes por meio de buscas exploratórias em navegadores de pesquisa, priorizando aplicações de acesso gratuito, executáveis diretamente via navegador e voltadas ao ensino ou à experimentação prática de grafos. Além disso, para apoiar a avaliação dessas plataformas, construiu-se um referencial de Recursos Funcionais, apresentado no Quadro 1, à medida que as ferramentas foram acessadas e testadas, reunindo características consideradas diferenciais ou fundamentais para subsidiar a solução proposta.

Quadro 1 - Levantamento dos recursos funcionais.

ID	Recursos Funcionais	Descrição
F01	Área de construção básica	Adicionar, editar, remover nós e arestas livremente
F02	Área de construção interativa e/ou personalizável	Drag-and-drop, menus contextuais, realce visual, zoom, etc.
F03	Diferentes tipos de visualização	Oferece opções como 2D, 3D, layout automático ou personalizado
F04	Suporte de visualização para árvores	Layouts e componentes adaptados para estruturas hierárquicas
F05	Execução de algoritmos de Grafos	Rodar algoritmos clássicos sobre o grafo criado
F06	Visualização passo a passo da execução	Apresenta cada etapa do algoritmo
F07	Visualização do pseudocódigo da execução	Exibe o código do algoritmo sendo executado
F08	Feedback visual da execução	Mensagens e/ou destaques sobre estados, decisões e etapas do algoritmo
F09	Suporte a múltiplos algoritmos	Oferece execução de pelo menos 5 algorítmos da teoria.
F10	Visualização do pseudocódigo do grafo	Mostra como o grafo atual pode ser representado em código ou linguagem
F11	Visualização da estrutura do grafo	Exibição de matriz de adjacência, lista de adjacência ou propriedades
F12	Reconhece propriedades do grafo	Identifica automaticamente se é completo, conexo, nulo, etc.
F13	Reconhece tipo do grafo	Consegue classificar automaticamente como árvore, multigrafo, ponderado, direcionado, etc.
F14	Suporte a grafos especiais	Permite testar e visualizar propriedades como planaridade, coloração, etc.
F15	Diagnóstico automático de algoritmos	Identifica quais algoritmos podem ser executados no grafo atual
F16	Geração automática de grafos	Recurso para fins didáticos e testes rápidos
F17	Exportação/Importação de grafos	Salvar e carregar grafos em diferentes formatos de arquivo
F18	Suporte a múltiplas linguagens de grafo	DOT, JSON, código nativo ou representações estruturadas
F19	Software unificado	Estrutura centralizada em uma mesma página/software

Fonte: O autor (2025)

A partir desse levantamento, as plataformas puderam ser analisadas de forma quantitativa e qualitativa, priorizando seu potencial pedagógico e o nível de interatividade oferecido com base nos recursos listados. Esse processo possibilitou uma visão mais completa sobre suas funcionalidades e limitações, alinhada ao contexto educacional pretendido. Os resultados mapeados são apresentados no Quadro 2, que sintetiza o identificador, nome e os recursos funcionais atendidos de cada ferramenta.

Quadro 2 - Características técnicas das plataformas web para grafos.

ID	Nome da ferramenta	Recursos Funcionais
A01	Graph Online (GraphOnline, 2025)	F01, F02, F03, F05, F06, F08, F09, F11, F14, F16, F17, F18, F19
A02	Graph IT (GraphIt, 2025)	F01, F02, F05, F06, F08, F17, F19
A03	Graph Editor - Netlify (GraphEditor, 2025)	F01, F02, F19
A04	zoomcharts (Zoomcharts, 2025)	F01, F02, F03, F17, F19
A05	Graph Editor - CS academy (CSAcademy, 2025)	F01, F02, F03, F04, F11, F13, F19
A06	VisuAlgo (Visualgo, 2025)	F01, F02, F03, F04, F05, F06, F07, F08, F09, F10, F11, F12, F13, F14, F15, F16
A07	Graph-Visualizer (GraphVisualizer, 2025)	F01, F02, F05, F06, F08, F19
A08	Edgitor (Edgitor, 2025)	F01, F02, F03, F04, F11, F17, F18, F19
A09	D3GT (D3GT, 2025)	F01, F02, F03, F04
A10	Tree Visualizer (TreeVisualizer, 2025)	F02, F03, F04, F05, F06, F08, F19
A11	Visualizations of Graph Algorithms (TechnicalUniversityOfMunich, 2025)	F01, F02, F05, F06, F07, F08, F09, F16, F17, F18
A12	Data Structure Visualizations (Galles, 2025)	F04, F05, F06, F08, F09, F11, F16
A13	Data Structure Animation (Yong, 2025)	F01, F04, F05, F08, F09
A14	Algorithm Visualizer (AlgoritmVisualizer, 2025)	F02, F04, F05, F06, F07, F08, F09, F10, F11, F19

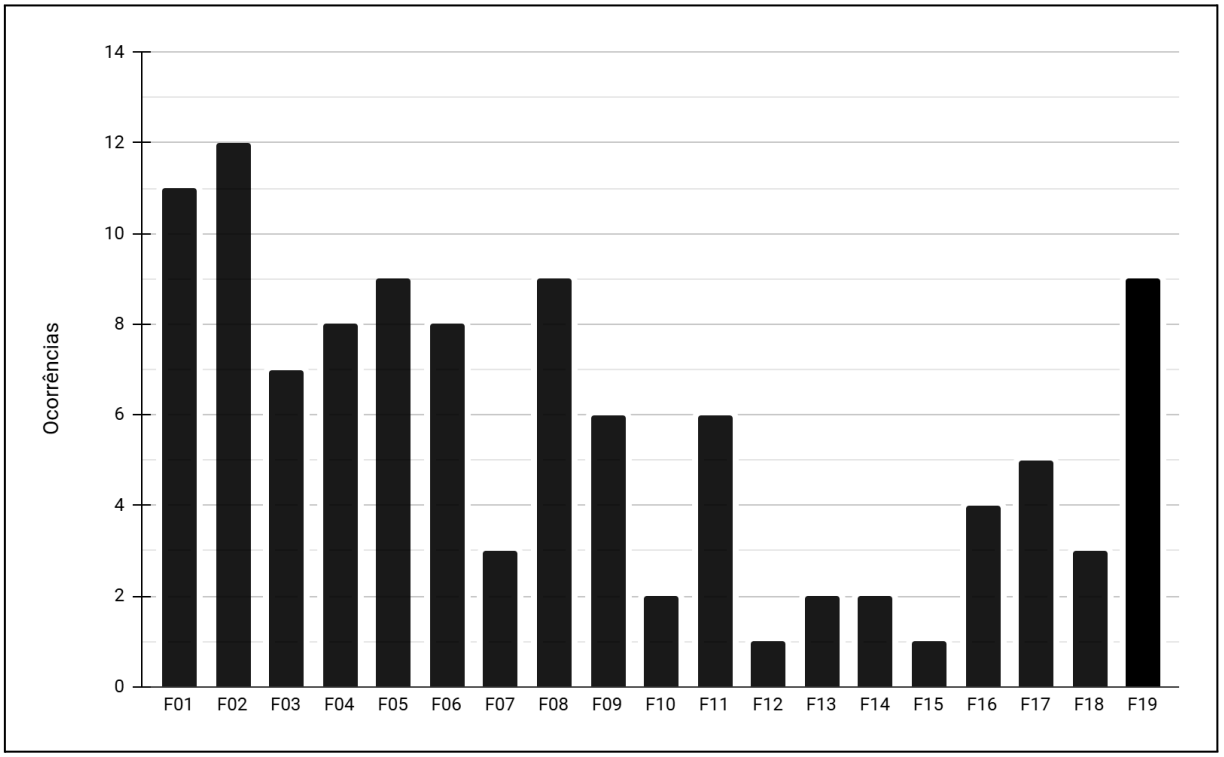
Fonte: O autor (2025)

É possível perceber que as ferramentas analisadas apresentam grande variação quanto ao atendimento dos recursos funcionais mapeados. Algumas plataformas oferecem apenas recursos básicos de edição de grafos, como a personalização de vértices e arestas (F01), enquanto outras se limitam à execução de algoritmos (F05) e à apresentação de suas etapas (F06). Desse modo, poucas soluções entregam experiências mais robustas, que consigam integrar essas capacidades junto a questões de visualização de pseudocódigo (F07, F10) e o acompanhamento detalhado do processo de execução. O VisuAlgo, por exemplo, destaca-se por abranger praticamente todos os recursos do quadro, funcionando como referência de completude funcional, porém com um foco bastante direcionado à execução e animação de algoritmos, com menor flexibilidade para o desenho livre do grafo, aspecto semelhante às limitações apontadas por Lourenço et al. (2022) em objetos de aprendizagem que priorizam a execução animada em detrimento da construção manual.

Essa heterogeneidade ressalta que, embora existam soluções relevantes disponíveis, nenhuma ferramenta analisada entrega plenamente todos os recursos considerados desejáveis, o que sinaliza oportunidades para inovações no desenvolvimento de plataformas didáticas. Esse cenário reforça a importância de projetos que considerem a integração de funcionalidades variadas e consistentes, alinhando-se às diretrizes de qualidade para o ensino de computação, como apontam Chen et al. (2010) e Santos e Costa (2005).

Para reforçar a perspectiva quantitativa e complementar a análise apresentada no Quadro 2, elaborou-se o gráfico exibido na Figura 12, que sumariza a frequência de ocorrência de cada recurso funcional entre as ferramentas mapeadas. Essa representação contribui para identificar quais recursos estão mais consolidados no ecossistema de soluções digitais e quais se mostram mais pontuais ou incipientes. Assim, essa síntese evidencia a importância de observar não apenas a distribuição e a recorrência de funcionalidades consideradas essenciais, mas também aquelas menos frequentes, que podem se tornar diferenciais inovadores ao apontar oportunidades de aprimoramento ainda pouco exploradas no cenário atual.

Figura 12 - Gráfico de recursos por quantidade de ocorrências.



Fonte: O autor (2025)

A análise do gráfico da Figura 12 reforça a percepção de que há uma concentração de

esforços no desenvolvimento de determinadas funcionalidades consideradas essenciais, como os recursos de canvas interativo (F02) e a centralização das ferramentas em um único ambiente (F19). Por outro lado, outras permanecem restritas a um número reduzido de plataformas, como o detalhamento dos pseudocódigos (F07, F10) e o reconhecimento de propriedades específicas dos grafos (F12, F13). Esse padrão sugere que, apesar de haver uma considerável gama de soluções web voltadas à prática da Teoria dos Grafos, existe um espaço significativo para explorar recursos que atendem às demandas pedagógicas.

O levantamento realizado oferece subsídios relevantes para nortear o desenvolvimento de soluções educacionais mais completas, entendendo que essas não devem se limitar apenas aos recursos comumente utilizados, tampouco focar exclusivamente em diferenciais isolados. É fundamental que consigam oferecer um conjunto equilibrado e abrangente de funcionalidades, capazes de atender diferentes perfis de usuários e de potencializar o processo de ensino-aprendizagem em ambientes digitais.

2.5 Trabalhos Relacionados

Durante o processo de revisão da literatura, foram identificados estudos que propõem o desenvolvimento de ferramentas computacionais voltadas ao ensino da Teoria dos Grafos, com ênfase na visualização de estruturas e na simulação do comportamento de algoritmos clássicos. Essas soluções se destacam por favorecer a compreensão conceitual e a ordenação do raciocínio lógico, reduzindo a barreira abstrata do conteúdo tradicional. Contudo, mesmo que um objetivo em comum, cada uma das propostas analisadas apresentaram escopos, públicos-alvo e estratégias próprias que as diferenciam e oferecem uma ampla perspectiva.

Santos e Costa (2007) desenvolveram o TBC-GRAFOS/WEB como um recurso de apoio ao ensino de algoritmos em grafos, especialmente pensado para alunos dos cursos de Computação e Informática. A ferramenta conta com um canvas que possibilita a construção manual dos grafos, incluindo funções para inserir e posicionar vértices, arestas e pesos, além de oferecer recursos de visualização passo a passo para algoritmos clássicos, como busca em largura, Dijkstra e Kruskal. Seu diferencial está na união de pseudocódigo e animações para ilustrar o processamento de cada etapa, promovendo um aprendizado mais ativo e contextualizado. Contudo, a solução apresentaria fortes desafios caso tentasse ser aplicada atualmente, pois depende de tecnologias ultrapassadas, como applets Java, que não contam mais com suporte em navegadores modernos. Além disso, a ausência de mecanismos automáticos de validação ou adaptação do grafo aos requisitos dos algoritmos limita a

confiabilidade da execução, exigindo do usuário conhecimento prévio da estrutura para evitar erros que inviabilizem o resultado correto, o que é uma forte negativa para sua aplicabilidade em contextos educacionais atuais.

Santana e Melo (2019) criaram o MÍNIMUS, um sistema interativo desenvolvido em JavaFX cujo objetivo central é potencializar o ensino e a prática de algoritmos de menor caminho, incluindo Dijkstra, Bellman-Ford e Floyd–Warshall. O software foi projetado para preencher lacunas identificadas no contexto educacional brasileiro, oferecendo uma interface intuitiva que permite a montagem de grafos personalizados, o ajuste de pesos e o acompanhamento detalhado da execução desses algoritmos. O trabalho destaca o potencial pedagógico da solução ao aproximar a abstração matemática do cotidiano dos alunos, porém reconhece que se trata de uma versão inicial, ainda carecendo de testes formais com turmas reais e de melhorias para ampliar a variedade de algoritmos e de estruturas de dados suportadas.

Mocinecová e Steingartner (2020) projetaram uma plataforma robusta de visualização de algoritmos em grafos, integrada ao contexto acadêmico de disciplinas de Estruturas de Dados na Universidade Técnica de Košice. A proposta se diferencia ao sincronizar o pseudocódigo com a animação gráfica do algoritmo, criando um elo direto entre a formalização teórica e sua execução visual, favorecendo o aprendizado por atingir dois pontos simultâneos. O sistema abrange algoritmos clássicos, incluindo busca em profundidade, busca em largura, Dijkstra e outros, além de permitir a criação livre dos grafos pelo usuário. Sua utilização foi direcionada ao apoio em aulas presenciais e a distância, reforçando o potencial de ambientes blended learning. Contudo, apesar da riqueza funcional, a ferramenta apresenta restrições claras de difusão: está vinculada ao contexto institucional, sem suporte multilíngue, documentação aberta ou licença pública, o que restringe sua adoção em outras universidades e dificulta sua customização.

Zanatto (2021) elaborou o GraphEDU, uma solução web orientada ao ensino de grafos, destacando-se pelo equilíbrio entre clareza didática e interação. A plataforma permite ao usuário construir grafos manualmente, selecionar algoritmos clássicos, como busca em profundidade, caminhos mínimos e árvores geradoras mínimas, além de permitir acompanhar a execução passo a passo e relacionar as ações do algoritmo com o pseudocódigo. A solução busca integrar teoria e prática de modo acessível, servindo como ponte entre a disciplina formal de algoritmos e a experiência de exploração livre pelo aluno. Contudo, a partir da análise técnica dos processos descritos, percebe-se que a aplicação apresenta uma estrutura fortemente acoplada ao seu modelo inicial, sem mecanismos de detecção automática de

propriedades dos grafos nem suporte à expansão de módulos, o que pode limitar futuras adaptações ou a incorporação de conteúdos mais avançados.

Lourenço et al. (2022) propuseram um objeto de aprendizagem direcionado ao ensino de algoritmos de caminho mínimo, contemplando métodos consagrados como Dijkstra, A*, Greedy Search e, de forma inovadora, aspectos heurísticos, algo ainda pouco explorado em outras ferramentas. Desenvolvido em Java com o uso da biblioteca JUNG, o recurso disponibiliza uma interface interativa que viabiliza a criação de grafos, configuração de pesos e visualização de resultados em tempo real, incluindo matriz de adjacências e detalhes do percurso calculado. Apesar desses avanços, o sistema não contém funcionalidade de visualização detalhada da execução dos algoritmos, apontada como uma carência pelos alunos participantes dos testes, além da pouca variedade de conceitos teóricos, o que dificulta sua utilização em cursos que contemplem outras estruturas clássicas da Teoria dos Grafos.

Burch et al. (2022) apresentaram uma abordagem diferenciada para análise de algoritmos de grafos, priorizando a exploração da dinâmica temporal durante sua execução. O sistema combina visualizações do tipo tempo-a-tempo (animações) e tempo-a-espaço (visões estáticas), permitindo uma visão macro e micro do comportamento do algoritmo em diferentes estágios. A proposta é especialmente interessante para contextos de análise de desempenho e investigação científica, uma vez que fornece dados sobre tempo de processamento e consumo de memória ao longo das etapas. Por outro lado, não incorpora funcionalidades de canvas para a criação ou edição de grafos, dependendo do carregamento prévio de arquivos externos e de sua arquitetura automática, o que limita sua aplicabilidade em contextos educacionais introdutórios que demandam maior liberdade na modelagem de exemplos e experimentações.

Os trabalhos descritos nesta seção evidenciam a relevância das ferramentas digitais no processo de ensino e aprendizagem da Teoria dos Grafos, ao favorecerem a compreensão de suas estruturas, a organização do raciocínio lógico e o entendimento dos algoritmos envolvidos. Com base nesses aspectos, o desenvolvimento da ferramenta (gr)Aphelios foi orientado pela visualização passo a passo das soluções, pela oferta de uma área de construção interativa de grafos, pela possibilidade de importação e exportação de dados e pela sincronização entre pseudocódigo e simulação visual.

O nome (gr)Aphelios foi idealizado pela união entre as palavras graph, termo em inglês para grafo, e Hélio, divindade associada ao Sol na mitologia grega. Essa composição representa a proposta da ferramenta de contribuir para uma compreensão mais clara dos conceitos de Teoria dos Grafos, utilizando recursos visuais e interativos como apoio ao processo de aprendizagem.

3 METODOLOGIA

3.1 Ferramentas de Desenvolvimento

Para o desenvolvimento do sistema (gr)Aphelio, foi necessária a utilização de ferramentas capazes de atender às necessidades de construção de uma aplicação web interativa voltada para manipulação de grafos e execução visual de algoritmos. Dessa forma, a seleção das tecnologias utilizadas no projeto considerou principalmente fatores como integração entre ferramentas, desempenho na renderização e manipulação dos elementos gráficos, facilidade de desenvolvimento e disponibilidade de documentação.

A partir desses critérios, foram selecionadas tecnologias pertencentes majoritariamente ao ecossistema JavaScript, permitindo a construção de uma aplicação modular, reativa e adequada às necessidades de interação e atualização dinâmica exigidas pelo sistema. As subseções a seguir apresentam as principais ferramentas que foram escolhidas e utilizadas para a construção do sistema.

3.1.1 JavaScript

O desenvolvimento da aplicação foi realizado utilizando JavaScript, que é uma linguagem nativa dos navegadores web amplamente utilizada para construção de aplicações interativas e dinâmicas. Por operar diretamente no ambiente do navegador, o JavaScript permite manipular elementos da página em tempo real, controlar eventos de interação e executar lógicas de processamento no lado do cliente (W3SCHOOLS, 2026).

Sua utilização no projeto ocorre além da robustez e versatilidade oferecidas pela linguagem, pela ampla disponibilidade de bibliotecas e ferramentas do ecossistema web voltadas para construção de interfaces interativas, manipulação de objetos e atualização dinâmica de elementos gráficos em aplicações web, como o React.

3.1.2 React

O React é uma biblioteca JavaScript de código aberto utilizada para o desenvolvimento de interfaces de usuário em aplicações web. Desenvolvida e mantida pelo Facebook, atual Meta, sua principal proposta baseia-se no conceito de componentes, estruturas independentes e reutilizáveis que encapsulam lógica e interface (REACT, 2026).

Um dos principais mecanismos do React, e parte do conjunto de fatores determinantes

para sua utilização neste projeto, é o Virtual DOM, uma representação em memória do Document Object Model (DOM) utilizada para otimizar o processo de renderização da interface. A partir da detecção de alterações no estado da aplicação, o React compara o Virtual DOM com o DOM real e realiza apenas as atualizações necessárias, em vez de renderizar toda a interface desnecessariamente (REACT, 2026). Além disso, o sistema de gerenciamento de estados da biblioteca possibilita o controle reativo dos dados da aplicação, permitindo que as alterações nos objetos sejam refletidas dinamicamente nos componentes da interface.

Por ser uma biblioteca e não de um framework completo, o React apresenta alta flexibilidade na integração com ferramentas externas. Essa característica contribuiu para sua escolha no desenvolvimento do projeto, possibilitando a utilização conjunta com a biblioteca responsáveis pelo gerenciamento global de estados da aplicação e pela atualização dinâmica dos elementos renderizados na interface do sistema.

3.1.3 Material UI (MUI)

O MUI é uma biblioteca de componentes React desenvolvida com base nas diretrizes do Material Design, especificação visual criada pelo Google para o desenvolvimento de aplicações web e mobile. A biblioteca disponibiliza uma ampla coleção de componentes reutilizáveis e personalizáveis, como botões, menus, caixas de diálogo e barras de navegação, permitindo a construção de interfaces padronizadas. Essa abordagem reduz significativamente a necessidade de desenvolvimento manual de elementos visuais e contribui para uma maior organização estrutural da interface da aplicação (MATERIAL UI, 2026).

Sua integração nativa com o React e a consistência visual proporcionada pelos componentes foram fatores determinantes para sua utilização no desenvolvimento do sistema, permitindo a construção de uma interface coesa e reutilizável alinhada à arquitetura baseada em componentes adotada no projeto.

3.1.4 Vite

O Vite é uma ferramenta de automação de compilação (build tool) voltada para o desenvolvimento de aplicações front end modernas. Seu ecossistema é composto principalmente por um servidor de desenvolvimento baseado em módulos ECMAScript Modules (ESM) e por um sistema de compilação responsável por gerar arquivos otimizados para produção (VITE, 2026). Durante o desenvolvimento, o Vite utiliza o Hot Module Replacement (HMR), mecanismo que permite refletir instantaneamente as alterações

realizadas no código-fonte na aplicação, sem a necessidade de recarregar completamente a página, proporcionando uma experiência de desenvolvimento mais rápida e eficiente.

A adoção do Vite neste projeto ocorreu principalmente devido à sua integração nativa com o React e à disponibilização de uma estrutura inicial oficial voltada para aplicações desenvolvidas com a biblioteca. Além disso, seu desempenho durante a etapa de desenvolvimento foi um fator relevante para sua escolha, especialmente pela rapidez no carregamento da aplicação e pela atualização dinâmica das alterações realizadas na interface e nos componentes do sistema.

3.1.5 SVG

Foi utilizado o Scalable Vector Graphics (SVG) para a renderização visual dos grafos na interface do sistema, por se tratar de uma tecnologia baseada em Extensible Markup Language (XML) e suportada nativamente pelos principais navegadores web. O seu principal diferencial é permitir a definição de formas geométricas diretamente no HTML por meio de um sistema de coordenadas cartesianas, que possibilita a construção de superfícies gráficas interativas (W3SCHOOLS, 2026).

A utilização do SVG mostrou-se fundamental para o desenvolvimento do projeto devido à forma como seus elementos são mantidos individualmente no DOM da página. Dessa forma, cada vértice, aresta e elemento gráfico presente na interface pode ser manipulado dinamicamente via JavaScript, permitindo alterações visuais em tempo real, como movimentação, rotação, coloração e atualização de propriedades durante a interação do usuário e a execução dos algoritmos implementados no sistema.

3.1.6 Zustand

Para coordenar as informações que mudam ao longo do uso da aplicação, como quais pontos estão selecionados, quais conexões existem e qual devem estar sinalizadas no grafo, o Zustand mostrou-se uma solução adequada ao escopo do projeto. Trata-se de uma biblioteca baseada em hooks nativos do React, desenvolvida para fornecer uma API simples e escalável para manipulação de estados globais, sem a necessidade de configurações excessivamente verbosas ou estruturas altamente opinativas (ZUSTAND, 2026).

Todo o estado da aplicação é definido por meio de uma store, que funciona como um hook customizado que armazena dados primitivos, objetos e funções responsáveis pelas atualizações (ZUSTAND, 2026). Essas alterações são realizadas de forma reativa, permitindo

que apenas os componentes afetados pelas modificações sejam renderizados.

A integração entre Zustand, React e Vite se torna fundamental para o desempenho da aplicação, especialmente na manipulação dos elementos gráficos do SVG. Como apenas os elementos alterados são atualizados no DOM, operações como movimentação, criação e atualização dos grafos tornam-se mais eficientes. Adicionalmente, o gerenciamento centralizado dos estados permite controlar as execuções dos algoritmos implementados, refletindo visualmente cada alteração realizada no grafo durante a execução.

3.1.7 Git e GitHub

Para o versionamento das versões do projeto e gerenciamento das alterações realizadas durante o desenvolvimento da aplicação, foram utilizadas as ferramentas Git e GitHub. Em conjunto, ambas permitem não apenas o controle e acompanhamento das modificações realizadas no código-fonte, mas também funcionalidades relacionadas ao armazenamento remoto, sincronização e disponibilização do projeto em ambiente colaborativo para a comunidade DEV.

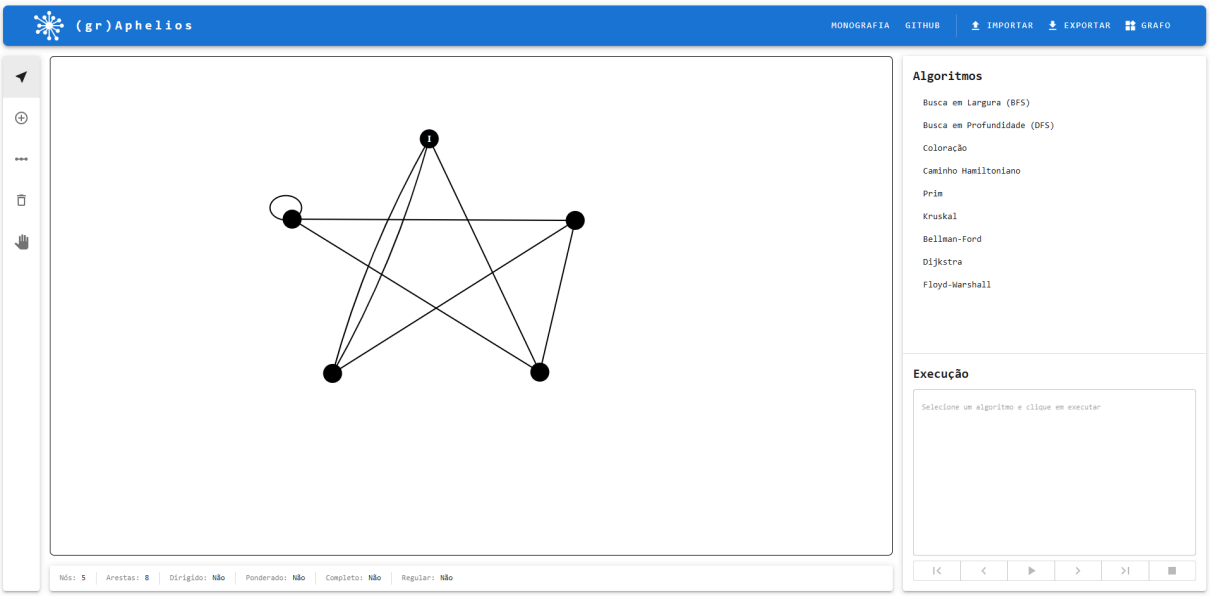
O Git consiste em um sistema de controle de versão distribuído responsável por registrar o histórico de alterações do projeto, possibilitando a criação de diferentes versões e ramificações de desenvolvimento que não comprometem a estrutura principal da aplicação (GIT, 2026). Já o GitHub atua como uma plataforma de hospedagem de repositórios baseada em Git, ele permite o armazenamento remoto do código-fonte e o gerenciamento centralizado do projeto (GITHUB, 2026).

3.2 Arquitetura da Aplicação

3.2.1 Interface do Sistema

A aplicação adota uma arquitetura de tela única, visível na Figura 13, concentrando em nível estrutural a proposta central do sistema: um ambiente onde a construção do grafo e a execução dos algoritmos coexistem no mesmo espaço de interação, sem rupturas de contexto ou quebras de fluxo. Todas as operações relacionadas à manipulação e à visualização do grafo permanecem acessíveis de forma integrada ao longo de toda a utilização da plataforma.

Figura 13 - Tela do sistema.



Fonte: O autor (2026)

Nessa interface, reúnem-se de maneira centralizada as principais áreas responsáveis pelo funcionamento do sistema, incluindo ferramentas de construção e manipulação do grafo, renderização visual, seleção de algoritmos, controles de execução e um painel textual destinado à exibição do pseudocódigo das etapas de execução em andamento. Essa organização busca manter todos os recursos necessários acessíveis simultaneamente, evitando interrupções no fluxo de interação durante a utilização da plataforma.

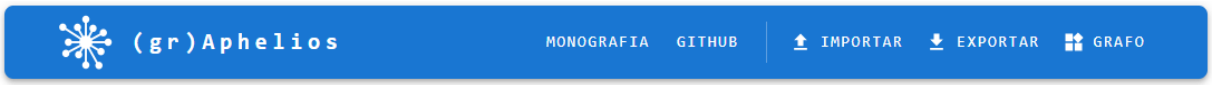
A organização da interface foi desenvolvida para favorecer a interação direta com a área de construção e o acompanhamento contínuo das alterações realizadas no grafo. Essa dinâmica é proporcionada pela disposição dos componentes da interface, que mantém a área de construção em posição central, enquanto os controles de manipulação, seleção de algoritmos e o painel de pseudocódigo permanecem acessíveis sem a necessidade de rolagem da página ou abertura de novas janelas. Dessa forma, operações como criação, remoção e movimentação dos elementos, assim como as modificações visuais decorrentes da execução dos algoritmos, permanecem constantemente visíveis ao usuário ao longo de cada etapa do processo de utilização do sistema.

Embora a plataforma não imponha restrições em relação à quantidade de vértices e arestas disponíveis para a construção dos grafos, a área de construção possui tamanho limitado, cabendo ao usuário controlar a quantidade de elementos inseridos para manter a legibilidade da representação e evitar a sobreposição de elementos na interface.

3.2.1.1 Navbar

A barra de navegação superior, apresentada na Figura 14, concentra as funcionalidades globais do sistema. Sua posição foi definida de forma a manter essas ações constantemente acessíveis ao usuário, independentemente das demais áreas da interface que são responsáveis por funções específicas da manipulação e execução dos grafos. Ocupando uma faixa horizontal no topo da aplicação, a barra não compete visualmente com o canvas ou com os painéis laterais, estabelecendo uma separação estrutural entre funções gerais do sistema.

Figura 14 - Barra de navegação da aplicação.



Fonte: O autor (2026)

As funcionalidades disponíveis na navbar incluem o acesso à documentação da plataforma e ao repositório de versionamento do projeto, opções para exportação do grafo nos formatos .png e .json, importação de estruturas via arquivos .json e, por fim, uma função para realizar a limpeza completa do canvas, excluindo todos os elementos.

3.2.1.2 Barra de Ferramentas

Os recursos responsáveis pela manipulação direta dos elementos dos grafos no canvas ficam na barra de ferramentas. Posicionada lateralmente à área principal de renderização, sua estrutura foi construída para manter as funções de edição constantemente acessíveis durante a utilização da aplicação, permitindo alternar rapidamente entre diferentes modos de interação sem interromper o fluxo de construção do grafo.

Figura 15 - Barra de ferramentas

Fonte: O autor (2026)

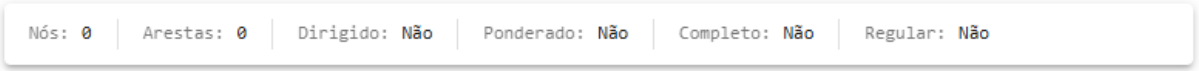
Conforme apresentado na Figura 15, a barra reúne, em sequência vertical, as ferramentas de: seleção de objetos, criação de vértices, criação de arestas, remoção de componentes e movimentação de elementos. A organização de cima para baixo segue uma lógica de uso progressivo, partindo das ações de seleção e inspeção até as de manipulação estrutural do grafo.

Cada uma das ferramentas funciona como um modo de operação exclusivo. Ao ser ativada, ela passa a determinar como os eventos disparados sobre o canvas serão interpretados pelo sistema. Dessa forma, interações realizadas em regiões vazias, vértices ou arestas produzem comportamentos distintos de acordo com a ferramenta selecionada, estabelecendo um modelo de interação baseado em estados bem definidos e previsíveis para o usuário.

3.2.1.3 Barra de Status

A barra de status possui um objetivo específico dentro da aplicação: acompanhar e apresentar as propriedades básicas dos grafos construídos no canvas, conforme apresentado na Figura 16. Sua posição inferior, localizada abaixo da área principal de interação, foi definida para manter essas informações constantemente visíveis sem competir visualmente com o canvas. Dessa forma, o usuário consegue acompanhar as características estruturais do grafo à medida que modificações são realizadas durante sua construção.

Figura 16 - Barra de status.



Fonte: O autor (2026)

As informações exibidas na barra possuem relevância especialmente em contextos educacionais, auxiliando usuários com menor familiaridade na identificação das propriedades do grafo construído. Além da quantidade de vértices e arestas, informações sobre se ele é direcionado, ponderado, completo e regular fornecem suporte à compreensão estrutural do grafo e auxiliam na identificação de possíveis estratégias e soluções para os problemas.

3.2.1.4 Barra de Algoritmos e Execução

O painel de algorítmos, apresentado na Figura 17, concentra os recursos responsáveis pelo controle e acompanhamento da execução dos algoritmos implementados no sistema. Sua estrutura foi organizada em três funcionalidades principais: seleção do algoritmo, área de exibição do pseudocódigo e controles de execução. Esses elementos foram projetados de forma integrada para permitir que o usuário acompanhe simultaneamente o comportamento visual do grafo e o estado lógico da execução em andamento.

Figura 17 - Barra de algoritmos e execução.

Fonte: O autor (2026)

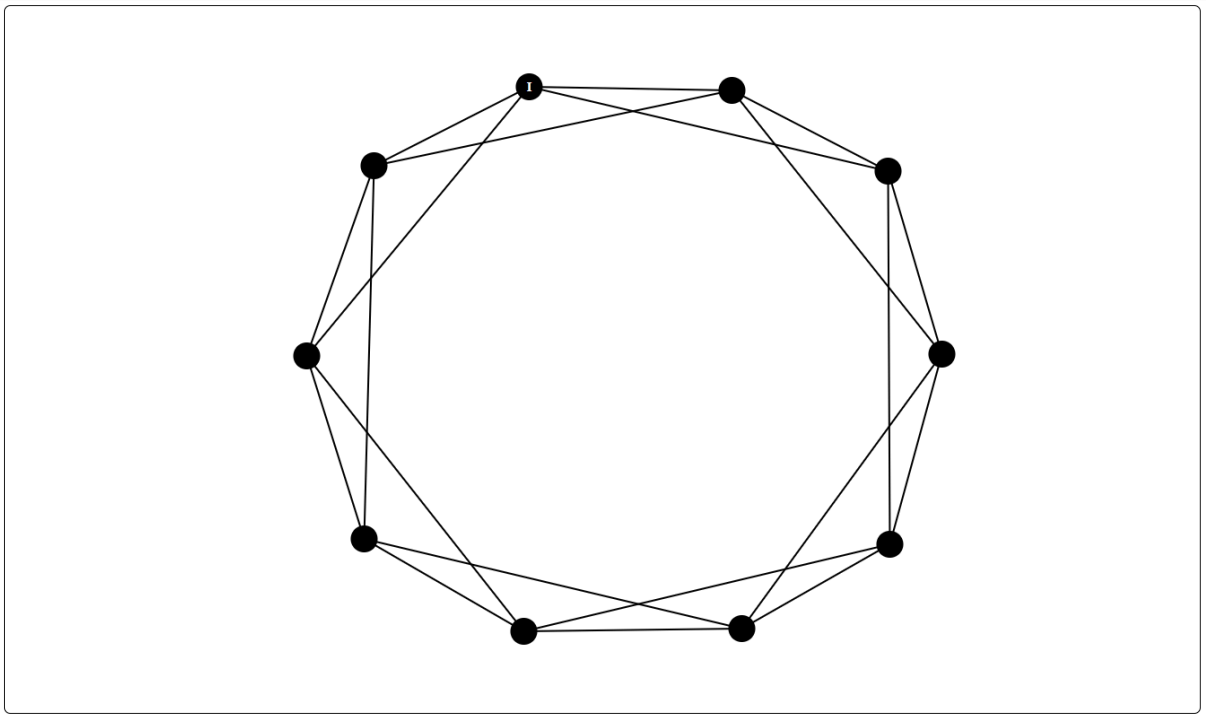
A separação entre o canvas e a área textual do pseudocódigo busca reduzir a sobrecarga visual durante a execução dos algoritmos, permitindo que a atenção do usuário permaneça voltada às alterações gráficas realizadas no grafo enquanto, em paralelo, o painel textual apresenta o fluxo lógico da execução passo a passo, relacionando cada comportamento visual exibido no canvas às instruções executadas pelo algoritmo.

A seleção dos algoritmos segue um modelo convencional de listagem, permitindo ao usuário escolher qual procedimento será executado sobre o grafo construído. Já os controles de execução foram organizados seguindo a lógica presente em reprodutores de áudio e vídeo, contendo opções para retornar ao primeiro passo, retornar ao passo anterior, iniciar a execução, avançar para o próximo passo, ir diretamente ao último passo e interromper completamente a execução em andamento.

3.2.1.5 Canvas

O canvas, apresentado nas Figuras 18 e 19, representa a principal área de interação da aplicação, sendo o espaço responsável pela construção dos grafos e pela execução visual dos algoritmos. O componente é estruturado em SVG, onde ocorrem a inserção, manipulação e renderização dinâmica, concentrando as principais interações do usuário com a plataforma.

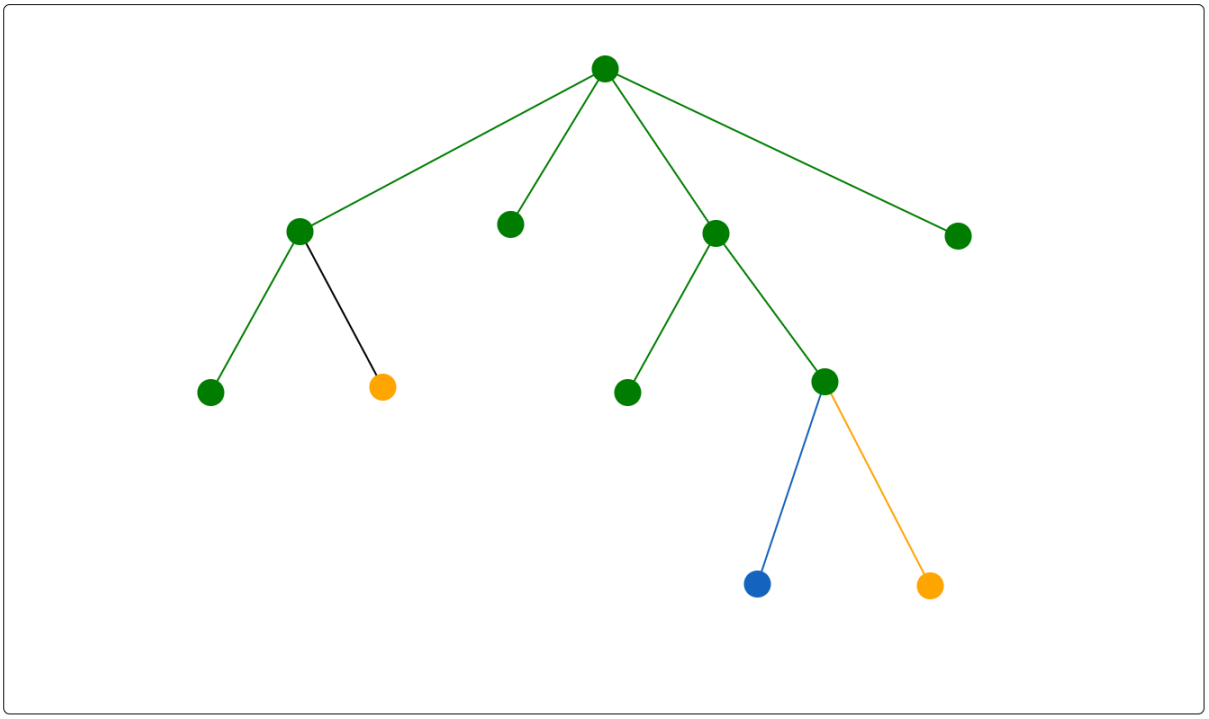
Figura 18 - Canvas.



Fonte: O autor (2026)

O canvas também atua como área de exibição dos estados de execução dos algoritmos. Mudanças de estado, seleções de elementos, percursos e colorações são refletidos visualmente em tempo real, permitindo acompanhar o comportamento da execução diretamente sobre a estrutura do grafo. A Figura 19 demonstra a aplicação de alterações visuais e colorações utilizadas para representar os diferentes estados assumidos pelos elementos durante a execução dos algoritmos.

Figura 19 - Canvas durante execução.



Fonte: O autor (2026)

3.2.2 Estrutura dos Dados

Não foi utilizado banco de dados externo para armazenamento das informações da aplicação. Todo o estado do sistema é mantido em memória por meio de uma store global gerenciada pelo Zustand, que atua como single source of truth da aplicação, centralizando os dados estruturais do grafo.

Os vértices são armazenados por meio de um array de objetos, onde cada elemento representa individualmente um nó do grafo. Cada vértice armazena um identificador, suas coordenadas e um campo textual utilizado para identificação visual na interface, conforme apresentado no Quadro 3.

Quadro 3 - Propriedades dos vértices.

Campo	Tipo	Descrição
ID	Integer	Identificador único do vértice
X	Float	Coordenada horizontal no plano cartesiano
Y	Float	Coordenada vertical no plano cartesiano
Label	String	Texto utilizado para identificação visual do nó

Fonte: O autor (2026)

As arestas também são armazenadas em um array de objetos, onde cada elemento representa uma conexão entre dois vértices. Cada aresta possui um identificador, referências para os vértices de origem e destino, um campo de peso e uma propriedade booleana responsável por indicar se a conexão será tratada como direcionada dentro da aplicação, descritas no Quadro 4.

Quadro 4 - Propriedades das arestas.

Campo	Tipo	Descrição
ID	Integer	Identificador único da aresta
Source	Float	Identificador do vértice de origem
Target	Float	Identificador do vértice de destino
Weight	Float/Null	Peso atribuído à aresta
Directed	Boolean	Define se a aresta é direcionada

Fonte: O autor (2026)

O valor *null* atribuído ao campo *weight* diferencia intencionalmente a ausência de peso do valor numérico zero, permitindo que a interface trate ambas as situações de forma distinta durante a renderização e a execução dos algoritmos.

3.2.3 Gerenciamento de Estados

A *store* global implementada com Zustand não atua apenas no armazenamento dos dados estruturais do grafo, mas na centralização de todo o comportamento da aplicação. Cada interação realizada pelo usuário, como adicionar um vértice, selecionar uma ferramenta ou iniciar a execução de um algoritmo, resulta em uma mutação de estado dentro do sistema.

Nesse contexto, o Zustand garante que o controle dessas informações permaneça centralizado e sincronizado, fazendo com que a interface reflita informações consistentes dos estados.

As ações de manipulação dos grafos abrangem todas as operações de criação, atualização e remoção dos elementos estruturais. Essas ações, apresentadas no Quadro 5, são responsáveis por controlar todo o ciclo de vida dos elementos do grafo e recebem como parâmetros informações necessárias para identificação, posicionamento e atualização dos objetos manipulados na interface. Entre os comportamentos gerenciados por essas operações estão definições estruturais relevantes para execução dos algoritmos, como a atribuição do nó de origem utilizado durante o processamento dos grafos e, na exclusão de um nó, ocorre simultaneamente a exclusão de todas as arestas que estavam conectadas a ele.

Quadro 5 - Ações de alteração de estado dos elementos do grafo.

Ação	Parâmetros	Descrição
Adicionar nó	X, Y	Adiciona um vértice na posição cartesiana
Remover nó	ID	Remove o vértice e todas as arestas conectadas a ele
Mover nó	ID, X, Y	Atualiza as coordenadas do vértice no canvas
Atualizar nó	ID, Label	Atualiza o rótulo textual do vértice
Adicionar aresta	Source, Target	Adiciona uma aresta entre dois vértices
Remover aresta	ID	Remove a aresta pelo identificador
Atualizar aresta	ID, Weight, Directed	Atualiza o peso e a orientação de uma aresta
Limpar canvas	-	Remove todos os elementos do canvas
Definir nó de origem	ID	Define o vértice selecionado como nó de origem

Fonte: O autor (2026)

Além dos dados estruturais do grafo, a aplicação mantém um conjunto de estados globais responsáveis por controlar características gerais da interface e do comportamento do sistema. Conforme apresentado no Quadro 6, esses estados são de propriedades relacionadas ao direcionamento e à ponderação do grafo, bem como a ferramenta atualmente selecionada pelo usuário, responsável por definir a forma como as interações realizadas no canvas serão interpretadas pela aplicação.

Quadro 6 - Estados gerais de configuração do sistema.

Campo	Tipo	Descrição
Direcionado	Boolean	Indica se o grafo é tratado como dirigido
Ponderado	Boolean	Indica se o grafo opera no modo ponderado
Ferramenta ativa	String	Ferramenta atualmente ativa na interface

Fonte: O autor (2026)

A execução dos algoritmos também é controlada por um conjunto dedicado de campos de estado. Conforme apresentado no Quadro 7, esses estados são responsáveis por armazenar o algoritmo selecionado pelo usuário, a informação referente ao status da execução, indicando se ela está em andamento ou não, a sequência de passos produzida durante o processamento do código e o índice correspondente à etapa atualmente exibida na interface.

Quadro 7 - Estados de execução dos algoritmos.

Campo	Tipo	Descrição
Algoritmo selecionado	String/Null	Qual algoritmo está selecionado
Em execução	Boolean	Indica se uma execução está em andamento
Passos	Array	Sequência de passos da execução
Passo atual	Integer	Índice do passo atualmente em curso

Fonte: O autor (2026)

Assim como ocorre com a manipulação dos estados dos grafos, a aplicação também mantém um conjunto de ações responsáveis por controlar o ciclo de execução dos algoritmos. Essas ações, descritas no Quadro 8, permitem iniciar, interromper e navegar entre os passos gerados durante o processamento. Essa navegação pode ser realizada de forma sequencial, avançando para o próximo passo ou retornando ao anterior, assim como diretamente para o primeiro estado ou para o último passo processado. Caso a execução seja interrompida, os estados relacionados ao processamento são reinicializados, retornando ao estado inicial.

Quadro 8 - Ações de alteração de estado da execução dos algoritmos.

Ação	Parâmetros	Descrição
Selecionar algoritmo	Algoritmo	Define o algoritmo selecionado
Iniciar execução	Passos	Inicia a execução a partir da sequência de passos fornecida
Parar execução	-	Interrompe a execução e reinicia integralmente o estado
Próximo passo	Passo atual	Avança um passo na execução
Passo anterior	Passo atual	Retrocede um passo na execução
Primeiro passo	Passos	Navega diretamente ao primeiro passo
Último passo	Passos	Navega diretamente ao último passo

Fonte: O autor (2026)

A separação realizada entre a manipulação estrutural do grafo, configurações globais da interface e os controles de execução representa uma decisão arquitetural adotada para reduzir o acoplamento entre as diferentes responsabilidades do sistema.

3.3 Implementação dos Algoritmos

Os algoritmos foram desenvolvidos considerando como poderiam ser representados visualmente na área de construção, mesmo que suas implementações tenham sido realizadas de forma separada. O principal problema dessa abordagem ocorre no fato de que, mesmo que os algoritmos consigam operar de forma independente, é a representação visual o ponto chave para que os usuários consigam compreender o que está em execução.

Cada algoritmo foi construído como um módulo independente, responsável por receber o estado atual do grafo, seus nós e arestas, e processar sua estrutura de acordo com as regras específicas de sua execução. Suas implementações foram fortemente influenciadas pelas ideias apresentadas por Cormen et al. (2012) e Rosen (2009), sendo realizados os ajustes necessários para viabilizar a representação visual de suas etapas na interface da plataforma, preservando a lógica e a essência do funcionamento de cada algoritmo.

Conforme apresentado no Quadro 9, os algoritmos selecionados abrangem diferentes áreas clássicas da Teoria dos Grafos, englobando busca, percurso, caminho mínimo, árvore geradora mínima e coloração. A escolha desses algoritmos busca contemplar diferentes cenários e permitir a visualização de estratégias algorítmicas amplamente utilizadas no contexto acadêmico e computacional.

Quadro 9 - Algoritmos implementados no sistema.

Algoritmo	Categoria	Descrição e Detalhes da Implementação
Busca em Largura (BFS)	Busca e percurso	Realiza a travessia do grafo em níveis de proximidade a partir do nó inicial (CORMEN et al., 2012). A implementação utilizou uma fila do tipo "primeiro a entrar, primeiro a sair" para garantir que os vértices sejam processados na ordem em que foram descobertos, preservando a exploração em camadas.
Busca em Profundidade (DFS)	Busca e percurso	Executa o trajeto do grafo explorando em profundidade cada ramo que parte do nó inicial (CORMEN et al., 2012). Na implementação, foi utilizada uma pilha do tipo "último a entrar, primeiro a sair" para priorizar os vértices descobertos mais recentemente, permitindo que a busca avance por um único caminho até o seu limite antes de explorar alternativas.
Dijkstra	Caminho mínimo	Determina os menores caminhos entre o nó inicial e os demais nós de um grafo que não possua arestas com pesos negativos (CORMEN et al., 2012). Para sua implementação, foi utilizada uma estrutura baseada na seleção do vértice não visitado com menor distância conhecida, atualizando as distâncias sempre que um novo caminho encontrado apresenta custo menor que o anterior avaliado.
Bellman-Ford	Caminho mínimo	Define os caminhos mínimos entre o nó inicial e os demais nós de um grafo, mesmo na presença de arestas com pesos negativos (CORMEN et al., 2012). Sua implementação foi realizada com passagens sucessivas pelas conexões, atualizando as distâncias sempre que um novo caminho encontrado apresenta custo menor que o anterior calculado.
Prim	Árvore geradora mínima	A partir do nó inicial, constrói uma árvore geradora mínima conectando todos os vértices do grafo com o menor custo total possível (CORMEN et al., 2012). Para implementação, o nó de origem caracteriza-se como uma árvore que cresce gradualmente, selecionando a aresta de menor peso que conecta a árvore a um novo vértice.
Kruskal	Árvore geradora mínima	Constrói uma árvore geradora mínima conectando todos os vértices do grafo com o menor custo total possível (CORMEN et al., 2012). Na implementação, todas as arestas são ordenadas por peso e analisadas da menor para a maior, sendo adicionadas à solução apenas quando conectam partes diferentes da árvore sem formar ciclos.
Ciclo Euleriano	Ciclos e percursos	Verifica a existência de um ciclo capaz de visitar todas as arestas exatamente uma vez e retornar ao vértice de origem (CORMEN et al., 2012). Para realizar a implementação, foi utilizada uma adaptação da DFS, marcando as arestas já percorridas e retrocedendo para um ponto anterior sempre que o caminho atual não permite completar o circuito.
Ciclo Hamiltoniano	Ciclos e percursos	Analisa a existência de um ciclo que consegue visitar todas os vértices exatamente uma vez, retornando ao nó de inicial (CORMEN et al., 2012). No processo de implementação, foi empregada uma adaptação do DFS, mantendo o controle dos nós já visitados e retrocedendo a um estado anterior quando o caminho atual não completar o ciclo.
Coloração de Grafos	Coloração	Determina a coloração dos vértices do grafo com cores únicas para os vértices adjacentes e utilizando o mínimo de cores possível (CORMEN et al., 2012). A implementação foi

		construída por meio de um algoritmo guloso, verificando as cores já utilizadas pelos vizinhos de um nó e escolhendo a primeira cor disponível que não gera repetições.

Fonte: O autor (2026)

No sistema, os algoritmos foram desenvolvidos para responder dinamicamente às alterações de estado realizadas no canvas. Sempre que ocorre uma modificação estrutural no grafo, como adição, remoção ou atualização de vértices e arestas, mecanismos de validação são acionados para todos os algoritmos implementados no sistema. As informações utilizadas durante esse processamento estão apresentadas no Quadro 10 e representam os principais dados estruturais necessários para validação e execução dos algoritmos. Cada implementação possui uma função inicial de verificação responsável por analisar se o estado atual do grafo atende aos requisitos necessários para sua execução, como quantidade mínima de vértices, existência de nó inicial e restrições relacionadas ao direcionamento das arestas.

Quadro 10 - Informações recebidas pelos algoritmos para execução.

Informação	Tipo	Descrição
Vértices	Array	Estrutura contendo os vértices do grafo e suas propriedades
Arestas	Array	Estrutura contendo as conexões entre os vértices, incluindo informações de peso e direcionamento
Nó inicial	Integer/Null	Identificador do vértice definido como origem

Fonte: O autor (2026)

Caso essa validação inicial seja atendida, o algoritmo é executado integralmente sobre a estrutura atual do grafo para verificar se existe uma solução válida para o problema proposto. Essa abordagem se justifica pelo fato de que certos problemas, como o Ciclo Hamiltoniano (Quadro 9), pertencem à classe NP, na qual validações estruturais prévias são insuficientes para determinar a existência de uma solução, o que só pode ser constatado mediante a execução do algoritmo (CORMEN et al., 2012). Dessa forma, as verificações iniciais atuam como mecanismos de filtragem para reduzir execuções desnecessárias, embora ainda seja necessário executar completamente o algoritmo para confirmação dos resultados. Além disso, considerando o contexto educacional da aplicação, a quantidade de vértices e arestas manipulados tende a permanecer reduzida quando comparada à capacidade de processamento de computadores convencionais. Assim, mesmo com a reexecução dos

algoritmos após alterações estruturais no grafo, não há impacto perceptível no desempenho da interface ou prejuízo à experiência de utilização do sistema.

Após a execução do processamento, os algoritmos retornam uma sequência estruturada de passos intermediários contendo as informações necessárias para representar visualmente o estado atual da execução. Cada passo dessa sequência armazena dados relacionados ao estado corrente do processamento, conforme detalhado no Quadro 8, elementos atualmente em análise e informações já consolidadas pelo algoritmo, como vértices visitados, arestas selecionadas e estruturas já processadas. Toda a sequência produzida é então devolvida ao componente responsável pela reprodução visual, permitindo o controle progressivo e regressivo das etapas da execução diretamente sobre o grafo renderizado no canvas.

4 RESULTADOS

4.1 Visão Geral do Protótipo

Desenvolvido como um ambiente web interativo, o protótipo da plataforma (gr)Aphelio tem como objetivo auxiliar no ensino e aprendizagem da Teoria dos Grafos. Para isso, a ferramenta reúne recursos para a construção de grafos e a execução de algoritmos com visualização interativa, estando disponível para acesso em https://gr-aphelios.vercel.app/.

Conforme apresentado na Figura 13 e detalhado na Seção 3.2.1, a aplicação foi organizada em uma tela única, sem rolamento, com regiões delimitadas e destinadas a funcionalidades específicas: ferramentas de construção e edição do grafo, área de construção, acompanhamento de definições e o controle da execução dos algoritmos. Dessa maneira, o usuário pode consultar informações, alterar configurações e controlar a execução dos algoritmos sem a necessidade de alternar entre diferentes páginas, abas ou janelas.

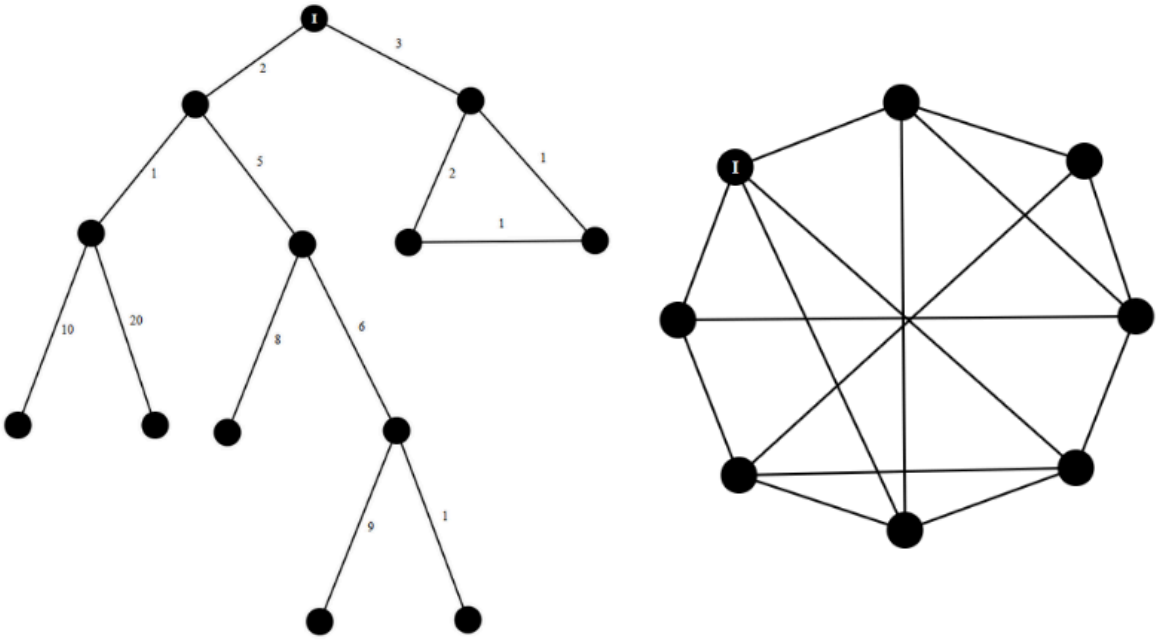
Foi elaborado um guia inicial para auxiliar os usuários na utilização da ferramenta que pode ser consultado através do link https://github.com/AlisonDLovegood/-gr-aphelios/blob/225ac12eb245192a2436a01304738e0a2306ab28/docs/Guia%20inicial%20-%20(gr)aphelios.pdf. Nele, foram mapeadas questões relacionadas ao funcionamento da plataforma, às funcionalidades de suas seções e seus respectivos botões, assim como orientações sobre como construir os grafos, selecionar os algoritmos e executar as principais ações de controle de fluxo de passos.

A plataforma busca proporcionar um espaço de experimentação no qual seja possível representar grafos, modificar suas propriedades e visualizar o comportamento dos algoritmos durante a execução. Assim, o sistema aproxima os conceitos teóricos de suas representações práticas, favorecendo a compreensão das estruturas e dos processos envolvidos na resolução de problemas em grafos.

4.2 Construção e Manipulação de Grafos

O gerenciamento dos grafos corresponde ao núcleo da plataforma. Por meio das ações de construção e manipulação, o usuário pode representar visualmente diferentes tipos de grafo, criando vértices e arestas conforme a necessidade da atividade proposta ou de um cenário em análise, como ilustrado na Figura 20.

Figura 20 - Diferentes construções de grafos.

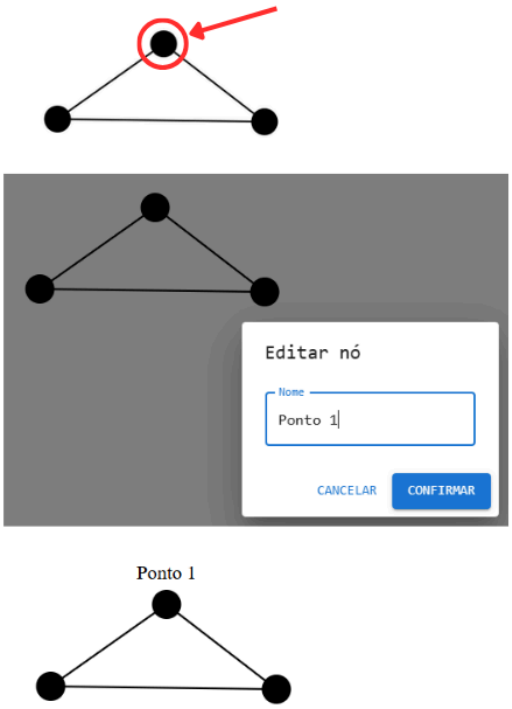


Fonte: O autor (2026)

A construção dos modelos é realizada por meio da barra de ferramentas, que contém as funcionalidades de criação, edição e remoção dos elementos. Com esses recursos, o usuário pode adicionar novos vértices, estabelecer conexões entre os elementos existentes, selecionar componentes do grafo, alterar suas informações e excluir estruturas.

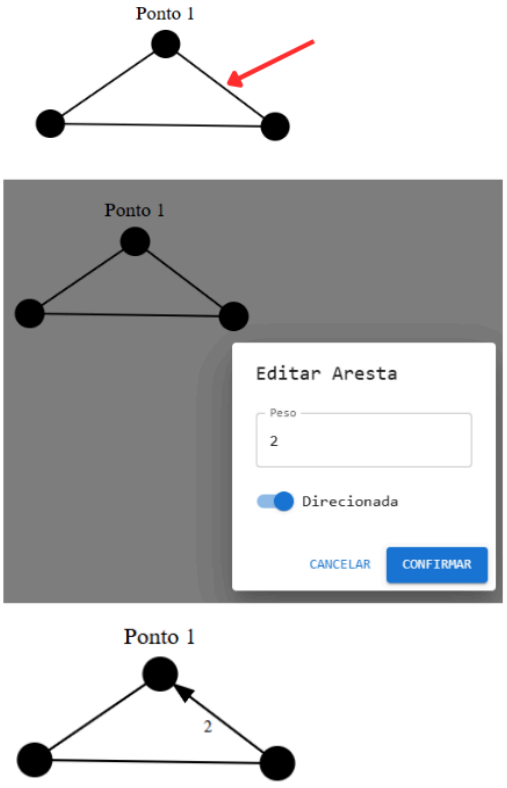
Além da criação dos elementos de ligação, a definição de suas propriedades é utilizada como base para a representação e o processamento do grafo pelos algoritmos. No caso dos vértices, é possível alterar a informação de rótulo, já nas arestas podem ser configurados os atributos de peso e direcionamento, características que desempenham papeis fundamentais na execução dos algoritmos. Ambas as edições estão ilustradas nas figuras 21 e 22.

Figura 21 - Edição de propriedades do vértice.



Fonte: O autor (2026)

Figura 22 - Edição de propriedades da aresta.



Fonte: O autor (2026)

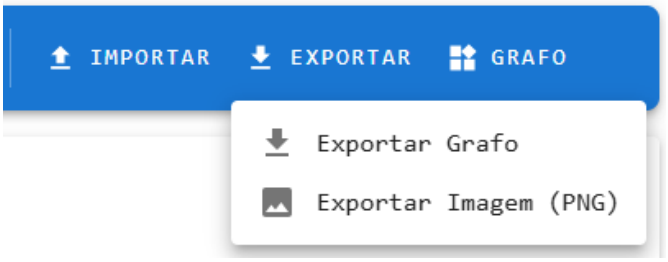
As operações de manipulação permitem que o usuário construa grafos sem depender de modelos previamente definidos ou limitações fixas de posicionamento. Assim, a plataforma possibilita a criação de diferentes estruturas que podem ser utilizadas para a exploração livre dos conceitos da teoria.

4.3 Importação e Exportação

As funcionalidades de importação e exportação de grafos complementam o processo de construção, permitindo que estruturas criadas no sistema sejam armazenadas e reutilizadas posteriormente. Dessa forma, grafos previamente analisados podem ser recuperados sem a necessidade de realizar a reconstrução manual de sua estrutura.

A exportação, ilustrada na Figura 23, pode ser realizada de duas formas: como um arquivo de imagem, em PNG, ou em formato de arquivo de dados, em JSON. O formato PNG permite salvar uma representação visual direta do grafo, possibilitando utilização em apresentações, documentos ou materiais didáticos. Enquanto que o formato JSON armazena todas as informações estruturais do grafo, incluindo os vértices, as arestas, os pesos, os direcionamentos e as posições cartesianas dos elementos.

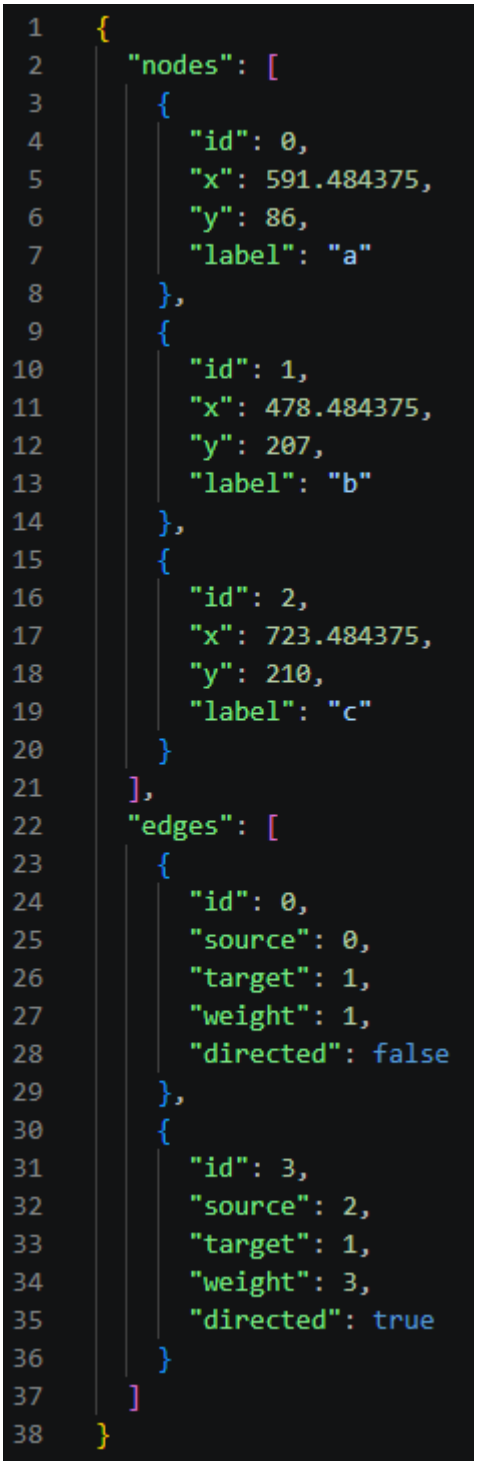
Figura 23 - Exportar o grafo.



Fonte: O autor (2026)

Por meio da exportação em JSON, o usuário pode preservar um grafo para utilização posterior na plataforma. Conforme ilustrado na Figura 24, o arquivo gerado permite reconstruir automaticamente a estrutura previamente criada, possibilitando o reaproveitamento de exemplos de estudos sem a necessidade de uma nova modelagem.

Figura 24 - Exemplo de JSON exportado de um grafo.



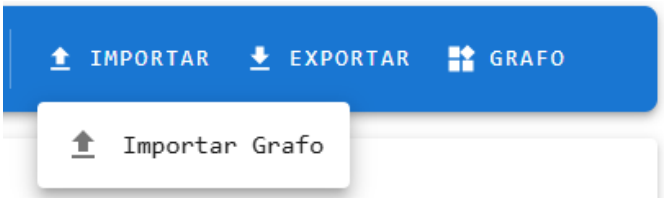
```json
{
  "nodes": [
    {
      "id": 0,
      "x": 591.484375,
      "y": 86,
      "label": "a"
    },
    {
      "id": 1,
      "x": 478.484375,
      "y": 207,
      "label": "b"
    },
    {
      "id": 2,
      "x": 723.484375,
      "y": 210,
      "label": "c"
    }
  ],
  "edges": [
    {
      "id": 0,
      "source": 0,
      "target": 1,
      "weight": 1,
      "directed": false
    },
    {
      "id": 3,
      "source": 2,
      "target": 1,
      "weight": 3,
      "directed": true
    }
  ]
}
```

Fonte: O autor (2026)

De forma complementar, a importação, ilustrada na Figura 25, realiza o caminho inverso, interpretando um arquivo previamente salvo em formato JSON e reconstruindo a estrutura no canvas da plataforma. Como essa funcionalidade depende dos dados estruturais do grafo, arquivos exportados em PNG não podem ser utilizados nesse processo. Após a

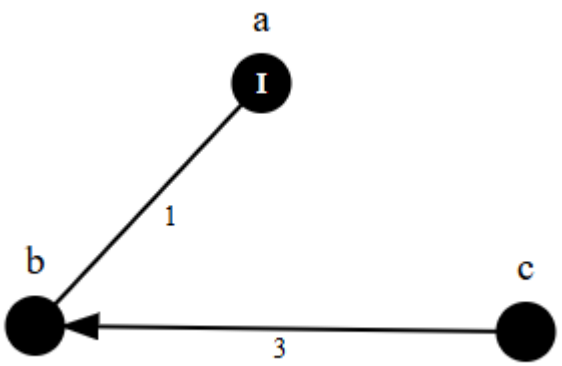
importação ser submetida no sistema, os elementos voltam a estar disponíveis para edição, alteração de propriedades e execução dos algoritmos compatíveis com a estrutura carregada. O resultado desse processo pode ser visualizado na Figura 26, que reproduz o grafo exportado anteriormente na Figura 24.

Figura 25 - Importar o grafo.



Fonte: O autor (2026)

Figura 26 - Exemplo de JSON importado de um grafo.



Fonte: O autor (2026)

Esses recursos favorecem a continuidade das atividades realizadas na plataforma, uma vez que permite salvar, compartilhar e retomar diferentes configurações de grafo. Além disso, a possibilidade de exportar a imagem do grafo amplia seu uso fora do sistema, especialmente em contextos acadêmicos e didáticos que solicitam representações visuais.

4.4 Execução e Visualização dos Algoritmos

A execução dos algoritmos corresponde à funcionalidade responsável por aplicar, sobre o grafo construído, os procedimentos definidos para cada uma das lógicas contempladas na plataforma. A partir dessa funcionalidade, o usuário deixa de atuar na construção da estrutura e passa a observar como os algoritmos se comportam diante da estrutura.

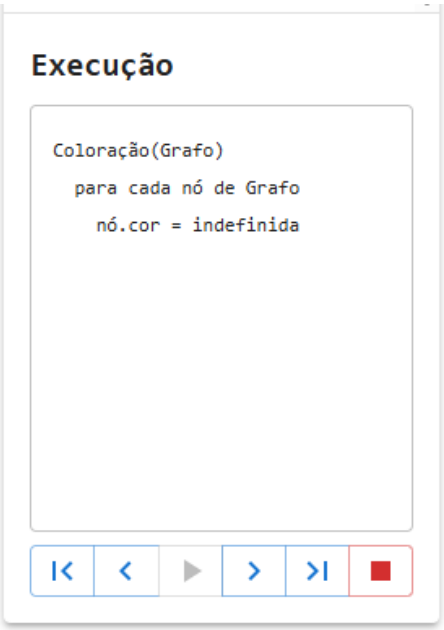
A funcionalidade de seleção de algoritmos está presente na barra de algoritmos, por meio da qual o usuário pode selecionar o algoritmo que deseja executar. O sistema possui mecanismos de validação automática, verificando quais algoritmos conseguem ser executados com sucesso no grafo construído e retornando na barra apenas as opções compatíveis, conforme ilustrado na Figura 27. Essa validação evita que o usuário tente executar algoritmos em grafos que não atendem aos requisitos mínimos dos algoritmos, como pesos, direcionamento ou conectividade.

Figura 27 - Algoritmos compatíveis com o grafo.

Fonte: O autor (2026)

Durante a execução, o sistema organiza o processamento em etapas, permitindo que o usuário acompanhe a sequência de operações realizadas pelo algoritmo. Esse funcionamento possibilita observar, com maior precisão, como os elementos do grafo são percorridos, permitindo controlar o avanço ou retrocesso dessas etapas por meio dos controles de execução dispostos em detalhes na Figura 28.

Figura 28 - Controle de execução.



Fonte: O autor (2026)

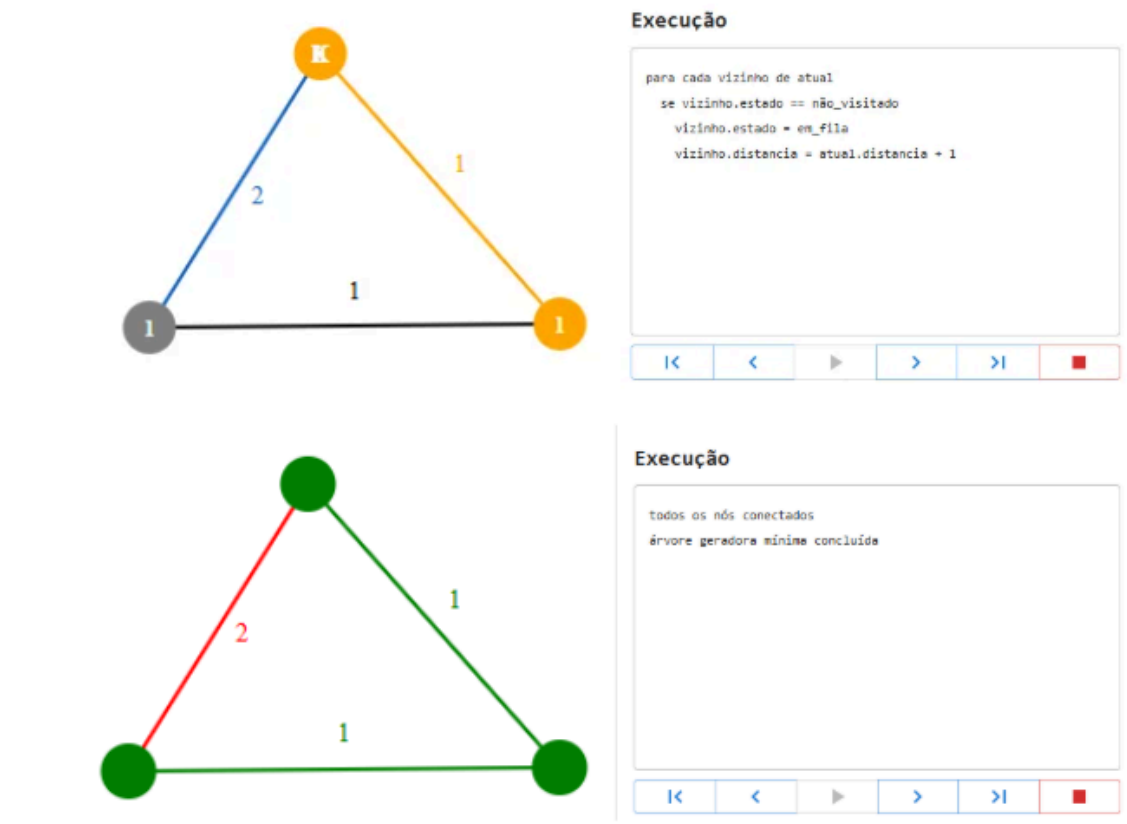
Toda a visualização da execução ocorre diretamente no grafo disposto no canvas, utilizando um esquema de cores para representar os diferentes estados dos elementos. Conforme ilustrado na Figura 29, cada cor possui uma definição que segue as especificações dispostas no Quadro 11.

Quadro 11 - Cores utilizadas na execução.

Cor	Descrição
Preto	O elemento não está em execução
Cinza	O elemento está em fila
Amarelo	O elemento está em execução
Azul	O elemento foi executado mas ainda está em análise
Verde	O elemento foi executado e aceito na solução
Vermelho	O elemento foi executado e descartado

Fonte: O autor (2026)

Figura 29 - Coloração da execução de um algoritmo.



Fonte: O autor (2026)

De forma complementar à visualização gráfica, o campo de pseudocódigo, também ilustrado na Figura 29, localizado acima dos controles de execução, destaca a instrução correspondente à etapa atual do algoritmo, permitindo associar a mudança observada no canvas à lógica executada naquele momento.

Dessa forma, a execução dos algoritmos não se limita apenas a uma apresentação de um resultado final numérico ou visual. A plataforma permite acompanhar o processo de construção da solução, oferecendo ao usuário meios para analisar cada etapa e revisar movimentos anteriores.

4.5 Discussão dos Recursos Implementados

Os recursos implementados na plataforma (gr)Aphelios foram organizados para compor um fluxo contínuo de interação com grafos e algoritmos. A proposta não se restringe à visualização estática de estruturas, característica observada em parte das ferramentas analisadas, ela busca realizar a integração dos recursos considerados relevantes para a

plataforma, conforme levantado nas seções 2.4 e 2.5, que incluem questões como a existência de uma área de construção do grafo, pseudocódigos, execução de algoritmos e retorno visual de suas operações lógicas.

A construção manual do grafo representa um dos requisitos fundamentais para uma solução adequada ao propósito educacional, principalmente quando comparada à ferramentas que utilizam apenas exemplos previamente definidos. Com essa funcionalidade, amplia-se o caráter exploratório da Teoria dos Grafos pelos estudantes, possibilitando testar diferentes configurações de estrutura e observar seus efeitos durante a execução.

Essa característica aproxima o (gr)Aphelio da plataforma da proposta de Santos e Costa (2007), na qual o usuário também podia desenhar o grafo antes da execução do algoritmo. No entanto, o (gr)Aphelios atualiza essa abordagem ao se apresentar como uma solução web moderna, que não depende de instaladores, applets Java ou conectores locais, requerendo para seu funcionamento apenas um navegador e acesso à internet.

Dentre os recursos discutidos como necessários e diferenciais, a importação e exportação de grafos também assume papel relevante. Esse recurso foi identificado em apenas 5 das 14 plataformas avaliadas no levantamento, o que demonstra que não se trata de uma funcionalidade recorrente nas soluções. No (gr)Aphelios, ela foi implementada com o objetivo de apoiar tanto a geração de arquivos de imagem, em formato PNG, para utilização em documentações e apresentações, quanto o armazenamento e a recuperação dos grafos por meio de arquivos JSON, permitindo preservar a estrutura construída para posterior reutilização no próprio sistema.

A validação dos algoritmos disponíveis a partir da estrutura do grafo construído é um ponto de extrema importância na usabilidade da plataforma. No levantamento realizado, esse recurso foi identificado apenas no VisuAlgo (Visualgo, 2025) dentre todas as ferramentas analisadas, destacando-se como uma funcionalidade pouco explorada, mas com potencial significativo. Sua implementação permite que seja exibido somente os algoritmos compatíveis com o grafo, reduzindo tentativas de execução que não solucionam o problema e tornando mais clara a questão dos requisitos de cada algoritmo.

Incorporar a execução passo a passo no sistema uniu aspectos visuais e lógicos entre os componentes de canvas e pseudocódigo. De forma semelhante ao objeto de aprendizagem proposto por Lourenço et al. (2024), que utiliza recursos gráficos para demonstrar a resolução de algoritmos, o sistema (gr)Aphelios busca apresentar a execução como um processo dividido em etapas, e não apenas um resultado final.

Para isso, as colorações e marcações aplicadas durante a execução permitem

identificar vértices e arestas em análise, processados ou pertencentes ao resultado final. Ao mesmo tempo, o pseudocódigo destacado permite relacionar cada alteração visual à instrução lógica correspondente, auxiliando o usuário a acompanhar a execução tanto pela representação gráfica quanto pela sequência formal do algoritmo. A união desses comportamentos organizam a execução do algoritmo como uma sequência visual de decisões, favorecendo a identificação de como a estrutura é percorrida e de como a solução é alcançada.

Dessa forma, os recursos implementados não aparecem como funcionalidades isoladas, mas como partes de um fluxo integrado de uso. A construção livre do grafo, a manipulação de propriedades, a importação e exportação, o diagnóstico automático de algoritmos compatíveis, a execução passo a passo e o pseudocódigo sincronizado compõem uma estrutura voltada à experimentação, permitindo que o usuário transite entre criação, análise e visualização dentro da mesma plataforma.

5 CONCLUSÃO

A construção do (gr)Aphelios teve como objetivo desenvolver uma plataforma web interativa voltada ao ensino da Teoria dos Grafos, que reúne em um único ambiente recursos fundamentais buscando aprimorar a experiência de aprendizagem por meio de uma abordagem centrada na clareza. A proposta nasceu a partir da necessidade de oferecer um sistema didático, capaz de apoiar a exploração de grafos e algoritmos de forma visual.

A revisão da literatura permitiu compreender que o ensino dos grafos envolve desafios relacionados à abstração dos conteúdos, à transição entre representações teóricas e visuais e à compreensão lógica dos funcionamentos dos seus algoritmos. Adicionalmente, a análise das ferramentas e trabalhos relacionados evidenciou limitações das soluções existentes, especialmente quanto à integração entre construção de grafos, execução sequencial, pseudocódigo e validação dos algoritmos compatíveis com a estrutura construída.

Outrossim, foi realizada uma análise crítica das alternativas para a construção gráfica das estruturas dos grafos. A escolha pela utilização de recursos próprios baseados em SVG, integrados ao React e ao gerenciamento de estados da aplicação, mostrou-se adequada aos objetivos do trabalho por permitir maior controle sobre os elementos visuais, suas propriedades e os estados assumidos durante a execução dos algoritmos.

Além disso, a definição dos algoritmos contemplados no sistema considerou sua relevância didática no estudo da Teoria dos Grafos. Foram selecionadas lógicas que abrangem diferentes categorias de problemas, como busca, ciclos, caminhos mínimos, árvores geradoras mínimas e coloração. Desse modo, a plataforma busca oferecer um leque de opções capaz de apoiar a compreensão de conceitos fundamentais da teoria.

O (gr)Aphelios reúne, em um ambiente web interativo, os principais recursos definidos ao longo deste trabalho. O sistema permite a construção manual de grafos, a edição de propriedades de vértices e arestas, a importação e exportação de estruturas, a validação automática dos algoritmos compatíveis e a execução passo a passo com apoio visual e pseudocódigo sincronizado. A integração dessas funcionalidades em uma interface busca facilitar a exploração dos conceitos dos algoritmos da Teoria dos Grafos, reunindo em um mesmo espaço recursos que, em soluções semelhantes, frequentemente se encontram dispersos ou indisponíveis.

Assim, o desenvolvimento da plataforma atende à proposta estabelecida para este trabalho ao integrar os recursos levantados durante a pesquisa e aplicá-los em um ambiente voltado ao ensino da Teoria dos Grafos. Como resultado, foi desenvolvido o protótipo

funcional da plataforma (gr)Aphelios, contemplando os mecanismos identificados nessa etapa. Entretanto, o escopo deste trabalho esteve concentrado apenas em seu desenvolvimento, não tendo sido incluídas etapas de avaliação com usuários ou de validação formal de usabilidade, aspectos que são necessários para uma análise mais aprofundada de sua efetividade em contextos educacionais.

6 TRABALHOS FUTUROS

Como continuidade do trabalho, uma das principais etapas futuras consiste na preparação da plataforma para publicação em ambiente de produção, sendo disponibilizada na internet e podendo ser acessada por qualquer usuário. Embora o sistema tenha sido desenvolvido como uma solução funcional, sua disponibilização pública exige etapas adicionais, como configuração de hospedagem, testes em diferentes navegadores e ajustes de responsividade. Apenas com essa publicação o sistema poderá ser acessado de forma estável por estudantes, professores e demais interessados na experimentação da Teoria dos Grafos.

Outro ponto de continuidade está relacionado à ampliação dos recursos de representação e exportação dos grafos. A exportação da matriz de adjacência e da lista de adjacência ampliaria as possibilidades de estudo oferecidas aos usuários, permitindo relacionar a representação visual do grafo com suas formas matemáticas e computacionais de armazenamento. Dessa forma, o usuário poderia observar não apenas a estrutura desenhada no canvas, mas também sua organização em formatos utilizados no processamento de algoritmos.

Em complemento, visando ampliar a compreensão do funcionamento interno dos algoritmos, mostra-se interessante a implementação de um painel de rastreio durante as execuções. Esse recurso permitiria visualizar, em tempo real, estruturas auxiliares como filas, pilhas, conjuntos de vértices visitados e valores de custo ou distância, tornando explícitas as alterações de estado realizadas a cada etapa do algoritmo. Paralelamente, a possibilidade de alternar o pseudocódigo atualmente exibido para implementações equivalentes em linguagens de programação, como Python, Java e C++, aproximaria a representação conceitual da implementação prática, facilitando a compreensão de como os procedimentos observados na interface são traduzidos para código.

No âmbito da usabilidade, a implementação de mecanismos de desfazer e refazer ações proporcionaria maior segurança durante a construção dos grafos. Por meio de um histórico de alterações gerenciado pelo Zustand, o usuário poderia recuperar modificações realizadas anteriormente, minimizando os impactos de exclusões ou edições acidentais.

Em complemento às funcionalidades que auxiliam na visualização sequencial dos algoritmos, a inclusão de um controle de velocidade para a execução automática permitiria adequar o ritmo das simulações às necessidades de cada usuário. A possibilidade de reduzir ou acelerar as animações favoreceria tanto a análise detalhada de cada etapa quanto a revisão mais dinâmica de algoritmos já conhecidos, tornando a plataforma mais flexível para a aprendizagem.

Sob a perspectiva da acessibilidade e da inclusão, a incorporação de recursos de síntese de voz contribuiria para ampliar o alcance da plataforma. A narração automática das ações realizadas durante a execução dos algoritmos, como a visita de vértices, a atualização de estruturas auxiliares e as mudanças de estado, disponibilizaria um canal adicional de acompanhamento das etapas executadas. Além de beneficiar usuários com deficiência visual, essa funcionalidade também poderia servir como um recurso pedagógico complementar, reforçando a compreensão do funcionamento dos algoritmos.

Adicionalmente, destaca-se a possibilidade de modelagem automática dos grafos como recurso funcional. Esse tipo de ferramenta poderia auxiliar na organização visual de estruturas construídas manualmente e na criação automática de disposições mais legíveis para os vértices no plano cartesiano. Critérios como distância entre os elementos, angulação das arestas e distribuição espacial poderiam ser utilizados para gerar alternativas de organização visual, reduzindo sobreposições e facilitando a interpretação da estrutura.

Como sugestão principal, a realização de avaliações com usuários em contextos reais de ensino é fundamental para a validação da ferramenta. Executar testes com estudantes e professores permitiria verificar aspectos-chave da ferramenta, como clareza da interface, facilidade de uso, compreensão dos algoritmos durante e após a execução e, principalmente, a possibilidade da plataforma ser efetivamente utilizada como apoio no ensino.

REFERÊNCIAS

ALEXANDERSON, Gerald L.. About the cover: euler and königsberg's bridges. Bulletin Of The American Mathematical Society, [S.L.], v. 43, n. 04, p. 567-574, 18 jul. 2006. American Mathematical Society (AMS). http://dx.doi.org/10.1090/s0273-0979-06-01130-x.

ALGORITHM VISUALIZER. Disponível em: https://algorithm-visualizer.org/. Acesso em: 25 jun. 2025.

BRASIL. Ministério da Educação. Conselho Nacional de Educação. Câmara de Educação Superior. Resolução CNE/CES nº 5, de 16 de novembro de 2016. Institui as Diretrizes Curriculares Nacionais para os cursos de graduação na área da Computação. Diário Oficial da União: seção 1, Brasília, DF, p. 22-24, 17 nov. 2016.

BURCH, Michael; WETERING, Huub van de; WALLNER, Günter; ROOKS, Freek; MORRA, Olof. Exploring the dynamics of graph algorithms. Journal Of Visualization, [S.L.], v. 26, n. 2, p. 477-492, 14 out. 2022. Springer Science and Business Media LLC. http://dx.doi.org/10.1007/s12650-022-00885-0.

CHAVES, Rosenilda Alves dos Santos; SANTOS, Maria Pricila Miranda dos. DESAFIOS E PERSPECTIVAS DA TECNOLOGIA NA EDUCAÇÃO. Revista Ibero-Americana de Humanidades, Ciências e Educação, [S.L.], v. 11, n. 3, p. 1074-1084, 18 mar. 2025. Revista Ibero-Americana de Humanidades, Ciencias e Educacao. http://dx.doi.org/10.51891/rease.v11i3.18213.

CHEN, Pu-Shih Daniel; LAMBERT, Amber D.; GUIDRY, Kevin R.. Engaging online learners: the impact of web-based learning technology on college student engagement. Computers & Education, [S.L.], v. 54, n. 4, p. 1222-1232, maio 2010. Elsevier BV. http://dx.doi.org/10.1016/j.compedu.2009.11.008.

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. Algoritmos: teoria e prática. 3. ed. rev. e ampl. Rio de Janeiro: Elsevier, 2012.

CS ACADEMY. Graph Editor. Disponível em: https://csacademy.com/app/graph_editor. Acesso em: 18 jun. 2025.

D3GT. D3 Graph Theory. Disponível em: https://d3gt.com. Acesso em: 16 jun. 2025.

DETERS, Janice Inês et al. O desafio de trabalhar com alunos repetentes na disciplina de Algoritmos e Programação. Simpósio Brasileiro de Informática na Educação, 2008.

EDGITOR. Disponível em: https://edgitor.chunlaw.io/. Acesso em: 03 jun. 2025.

GALLES, David. Data Structure Visualizations. Disponível em: https://www.cs.usfca.edu/~galles/visualization/Algorithms.html. Acesso em: 10 jun. 2025.

GITHUB. GitHub Docs. Disponível em: https://docs.github.com/pt. Acesso em: 03 jan. 2026.

GIT. --everything-is-local. Disponível em: https://git-scm.com. Acesso em: 03 jan. 2026.

GOLDBARG, Marco A.; GOLDBARG, Elizabeth. Grafos: conceitos, algoritmos e aplicações. Rio de Janeiro: Elsevier, 2009.

GRAPH EDITOR. Disponível em: https://graph-editor.netlify.app. Acesso em: 12 jun. 2025.

GRAPHIT. Disponível em: https://graphit.web.app/app. Acesso em: 09 jun. 2025.

GRAPHONLINE. Disponível em: https://graphonline.top/pt. Acesso em: 05 jun. 2025.

GRAPH-VISUALIZER. Disponível em: https://www.graph-visualizer.io. Acesso em: 22 jun. 2025.

HAN, Dongming; PAN, Jiacheng; ZHAO, Xiaodong; CHEN, Wei. NetV.js: a web-based library for high-efficiency visualization of large-scale graphs and networks. Visual Informatics, [S.L.], v. 5, n. 1, p. 61-66, mar. 2021. Elsevier BV. http://dx.doi.org/10.1016/j.visinf.2021.01.002.

LOURENÇO, Wilson da Silva; ALVES, Wonder Alexandre Luz; LIMA, Stanley Jefferson de Araujo; ARAUJO, Sidnei Alves de. Objeto de aprendizagem para o ensino de algoritmos para solução do problema de caminho mínimo. Exacta, [S.L.], v. 22, n. 3, p. 926-939, 13 out. 2022. University Nove de Julho. http://dx.doi.org/10.5585/exactaep.2022.22247.

MATERIAL UI. Getting Started – Material UI. Disponível em: https://mui.com/material-ui/getting-started. Acesso em: 08 jan. 2026.

MOCINECOVÁ, Karina; STEINGARTNER, William. Software Support for Visualizing of the Graph Algorithms in a Novel Approach in Educating of Young IT Experts. Ipsi Transactions On Internet Research, [s. l], v. 16, n. 2, p. 14-23, jul. 2020.

QUINN, Anne. Using Apps to Visualize Graph Theory. The Mathematics Teacher, [S.L.], v. 108, n. 8, p. 626-631, abr. 2015. National Council of Teachers of Mathematics. http://dx.doi.org/10.5951/mathteacher.108.8.0626.

REACT. The library for web and native user interfaces. Disponível em: https://react.dev. Acesso em: 03 jan. 2026.

ROSEN, Kenneth H. Matemática discreta e suas aplicações. 6. ed. São Paulo: McGraw-Hill, 2009.

SANTANA, Kawan; DE MELO, Maxmilian Jaderson. MÍNIMUS: uma ferramenta para o ensino e prática de grafos com enfoque nos algoritmos de menor caminho. Encontro Internacional de Gestão, Desenvolvimento e Inovação (EIGEDIN), v. 3, n. 1, 2019.

SANTOS, Rodrigo Pereira dos; COSTA, Heitor Augustus Xavier. TBC-AED e TBC-AED/Web: Um desafio no ensino de algoritmos, estruturas de dados e programação. In: IV Workshop Em Educação Em Computação E Informática Do Estado De Minas Gerais. 2005.

SANTOS, Rodrigo Pereira dos; COSTA, Heitor Augustus Xavier. TBC-GRAFOS/WEB–treinamento baseado em computador para algoritmos em grafos via web. 2007.

SILVA, Rômulo César; FRANCISCO, Giovani Rubim. QuatiView: ferramenta de visualização de algoritmos e estrutura de dados. Observatório de La Economía Latinoamericana, [S.L.], v. 23, n. 3, p. 37, 10 mar. 2025. Brazilian Journals. http://dx.doi.org/10.55905/oelv23n3-039.

TECHNICAL UNIVERSITY OF MUNICH. Visualizations of Graph Algorithms. Disponível em: https://algorithms.discrete.ma.tum.de/. Acesso em: 07 jun. 2025.

TILANTERÄ, Artturi; SORVA, Juha; SEPPÄLÄ, Otto; KORHONEN, Ari. Students Struggle with Concepts in Dijkstra's Algorithm. Proceedings Of The 2024 Acm Conference On International Computing Education Research - Volume 1, [S.L.], p. 154-165, 12 ago. 2024. ACM. http://dx.doi.org/10.1145/3632620.3671096.

TREE VISUALIZER. Disponível em: https://tree-visualizer.netlify.app/. Acesso em: 19 jun. 2025.

VISUALGO. Disponível em: https://visualgo.net/en. Acesso em: 20 jun. 2025.

VITE. A Ferramenta de Construção para a Web. Disponível em: https://pt.vite.dev. Acesso em: 03 jan. 2026.

WAHYUNINGSIH, Sapti; SATYANANDA, Darmawan; QOHAR, Abd; ATAN, Noor Azean. An Integration of "Online Interactive Apps" for Learning Application of Graph Theory to Enhance Creative Problem Solving of Mathematics Students. International Journal Of Interactive Mobile Technologies (Ijim), [S.L.], v. 14, n. 12, p. 97, 31 jul. 2020. International Association of Online Engineering (IAOE). http://dx.doi.org/10.3991/ijim.v14i12.15583.

W3SCHOOLS. SVG Introduction. Disponível em: https://www.w3schools.com/graphics/svg_intro.asp. Acesso em: 23 fev. 2026.

W3SCHOOLS. JavaScript Tutorial. Disponível em: https://www.w3schools.com/js. Acesso em: 03 jan. 2026.

YONG, Daniel Liang. Data Structures Animation. Disponível em: https://yongdanielliang.github.io/animation/animation.html. Acesso em: 15 jun. 2025.

ZANATTO, Nícolas Erciro. GRAPHEDU: SISTEMA WEB PARA VISUALIZAÇÃO DE ALGORITMOS DE GRAFOS. 2021. 74 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade de Caxias do Sul, Caxias do Sul, 2021.

ZHAO, Xin; WANG, Xuan; ZOU, Xianzhe; LIANG, Huiming; BAI, Genghuai; ZHANG, Ning; HUANG, Xin; ZHOU, Fangfang; ZHAO, Ying. Graph visualization efficiency of popular web-based libraries. Visual Computing For Industry, Biomedicine, And Art, [S.L.], v. 8, n. 1, p. 1-17, 8 maio 2025. Springer Science and Business Media LLC. http://dx.doi.org/10.1186/s42492-025-00193-y.

ZOOMCHARTS. Disponível em: https://apps.zoomcharts.com/graph-editor. Acesso em: 14 jun. 2025.

ZUSTAND. Zustand Documentation. Disponível em: https://zustand.docs.pmnd.rs. Acesso em: 20 fev. 2026.