

DESCRIÇÃO

Conceitos, representações, tipos e busca sobre grafos, além de algoritmos de caminho mínimo e algoritmo caixeiro-viajante.

PROPÓSITO

Apresentar os conceitos básicos para o entendimento de grafos a partir de exemplos, os diferentes tipos, representações e operações, a distinção entre os algoritmos de busca sobre grafos, o problema de custo mínimo e o problema do caixeiro-viajante.

OBJETIVOS

MÓDULO 1

Definir os conceitos básicos de grafos

MÓDULO 2

Identificar as diferentes representações de grafos

MÓDULO 3

Descrever os algoritmos de busca em grafos

MÓDULO 4

Descrever o problema do custo mínimo sobre um grafo ponderado

INTRODUÇÃO

A teoria dos grafos é estudada desde o século XVIII, antes da invenção do computador. No entanto, estes ainda são largamente aplicados para resolver diversos problemas, como a implementação de redes sociais.

Várias redes sociais utilizam fortemente grafos como abordagem para diversas funcionalidades apresentadas, como quais perfis amigos de um usuário curtiram um post ou sugestão de amigos para serem adicionados na rede de contatos.

Atualmente, existem bancos de dados criados para armazenamento e recuperação de grafos para resolver problemas das redes sociais e para extração de uma infinidade de informações da utilização da rede social por parte dos usuários.

Portanto, o estudo de grafos é bastante importante e, por isso, analisaremos os seus conceitos básicos, os diferentes tipos e as suas respectivas subclassificações, bem como os algoritmos para buscar elementos em um grafo e os algoritmos para se percorrer um grafo com o menor custo possível. Esses algoritmos são fundamentais para a resolução de problemas de rotas com menor custo, como definição de rotas de ônibus, de entregas, entre outros.

MÓDULO 1

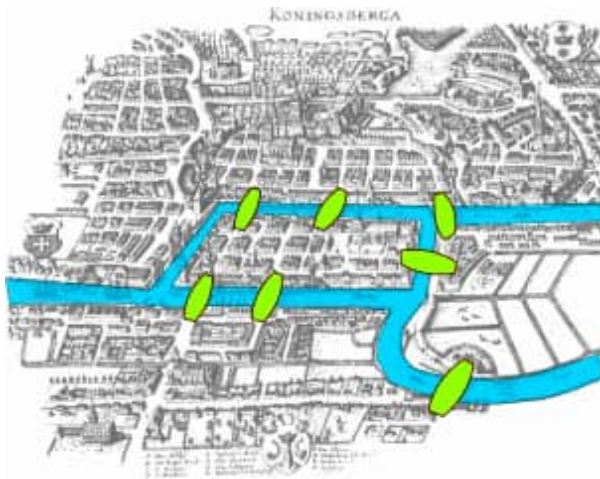
🕒 Definir os conceitos básicos de grafos

CONCEITOS DE GRAFOS

A teoria dos grafos é uma estrutura matemática que vem sendo estudada desde o século XVII na Matemática. Vários autores publicaram artigos nesse período, com destaque para o problema descrito por **Euler**, conhecido como **As Pontes de Königsberg**.

Numa cidade chamada Königsberg, sete pontes estabelecem ligações entre as margens do rio Pregel e duas de suas ilhas. Proposto em um artigo publicado em 1736, o problema consiste em, a partir de um determinado ponto, passar por todas as pontes somente uma vez e retornar ao ponto inicial.

Esse ficou conhecido como o problema do Caminho Euleriano, que ilustra o problema das Pontes de Königsberg, foi representado na Figura 1.1 (CARVALHO, 2005).



📷 Figura 1.1 – As Pontes de Königsberg

Fonte: Bogdan Giuşcă. Domínio Público. Wikipedia.

Com o passar dos séculos, foi obtido bastante conhecimento teórico e prático para aplicação da teoria dos grafos, como no problema anterior.

No entanto, com o surgimento da computação, a estrutura ganhou uma importância extraordinária por servir como base para modelagem de diversos processos e simulações matemáticas que são processados pelo computador.

Muitos problemas **NP-difícil** e **NP-completos** possuem a solução ótima obtida através de algoritmos de caminho em estruturas como **grafos**.

GRAFOS

Em algumas literaturas, o grafo é nomeado como um modelo matemático e, em outras, como uma estrutura. Vamos considerá-lo como uma estrutura.

Sua função é ajudar a simplificar problemas matemáticos complexos, e pode ser definido como:

"UM GRAFO É UM PAR ORDENADO DE CONJUNTOS DISJUNTOS (V, E) , ONDE V É UM CONJUNTO ARBITRÁRIO QUE SE DESIGNA POR CONJUNTO DOS VÉRTICES E E , UM MULTICONJUNTO DE PARES NÃO ORDENADOS DE ELEMENTOS (DISTINTOS) DE V QUE SE DESIGNA POR CONJUNTO DAS ARESTAS."

(ARAÚJO, 2020)

Neste tema, utilizaremos tanto o nome vértice como nós para referenciarmos os elementos de V .

Existem duas restrições para a definição:

Deve ser V um conjunto não vazio. Cada aresta é indivisível. Como exemplo, temos o conjunto V , onde, $V = \{v_0, v_1, v_2, v_3, v_4, v_5\}$ e $v_0 \dots v_n$ são vértices.

É um conjunto de arestas que, por sua vez, são pares de vértices distintos e não ordenados.

As arestas de um grafo podem ter um peso associado, podem ter direção, pode ser permitido ou não ligar um nó a ele mesmo. Se um grafo tem as arestas direcionadas, chamamos de **dígrafo**.

Para representar visualmente um grafo são, normalmente, empregados círculos para representarem os nós, ou vértices, e linhas para representar as arestas que interligam os nós.

Caso o grafo seja direcionado, a linha será substituída por uma seta.

COMENTÁRIO

Existem algumas outras formas de representar um grafo, conforme será apresentado mais adiante, mas, por enquanto, focaremos apenas nesse tipo de representação para facilitar a compreensão.

Para exemplificar, podemos apresentar um grafo para interligação entre os estados da região Sudeste do Brasil:

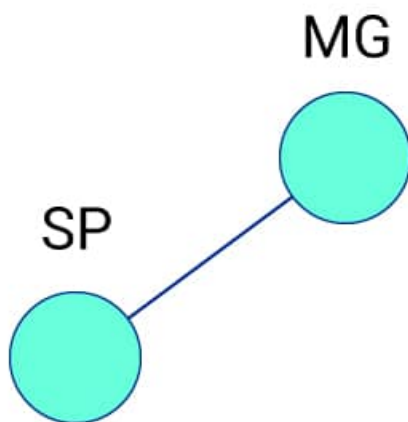
O grafo terá a seguinte representação na notação de grafos $G(V, E)$, onde:

$$V = \{ES, MG, SP, RJ\} \text{ e}$$

$$E = \{(MG, SP), (RJ, SP), (MG, ES), (RJ, MG), (RJ, ES)\}.$$

Consideramos como arestas apenas os estados que possuem fronteira geográfica entre si. Um grafo é construído considerando par de arestas “ligando” dois nós.

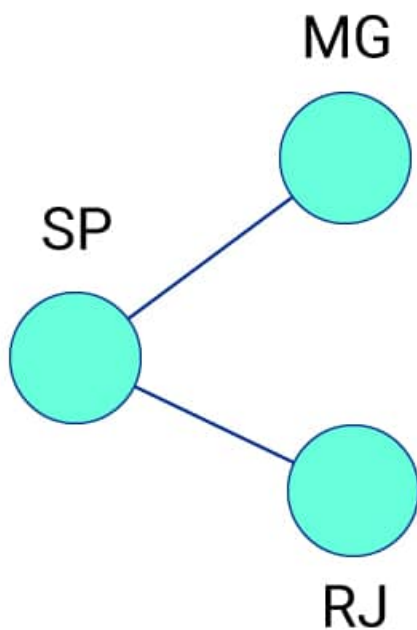
Adição da aresta $(SP, MG) \Rightarrow E = \{(MG, SP)\}$ representada na Figura 1.2.



📷 Figura 1.2 – Criação do grafo com aresta (SP, MG)

Fonte: EnsineMe.

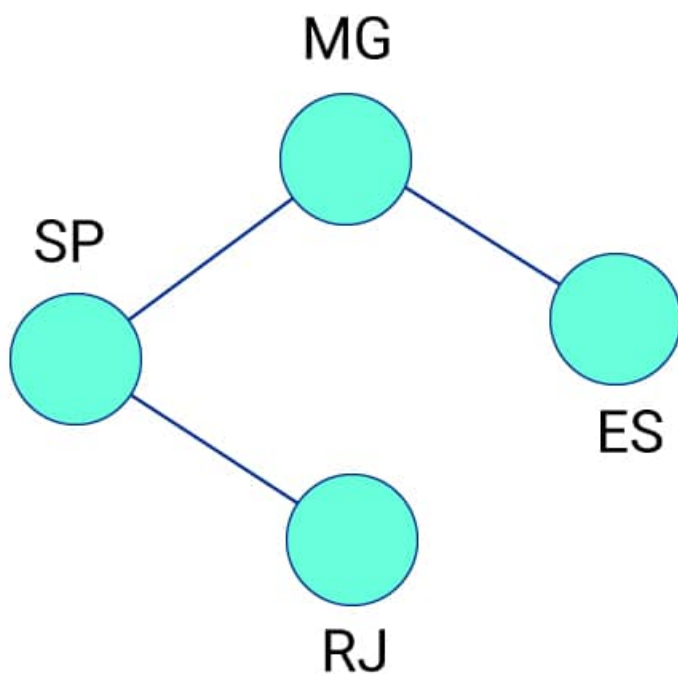
Adição da aresta $(RJ, SP) \Rightarrow E = \{(MG, SP), (RJ, SP)\}$ representada na Figura 1.3.



📷 Figura 1.3 – Adição da aresta (RJ, SP)

Fonte: EnsineMe.

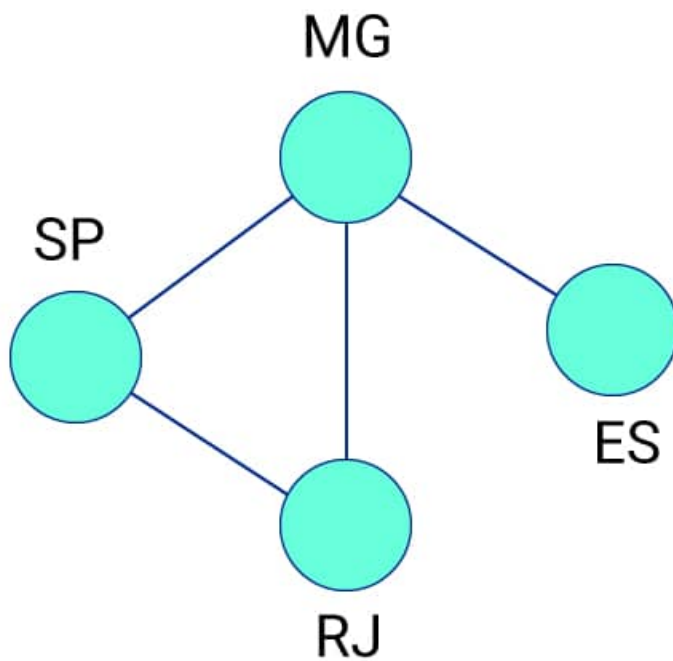
Adição da aresta (MG, ES) $\Rightarrow E = \{(MG, SP), (RJ, SP), (MG, ES)\}$ representada na Figura 1.4.



📷 Figura 1.4 – Adição da aresta (RJ, SP)

Fonte: EnsineMe.

Adição da aresta $(RJ, MG) \Rightarrow E = \{(MG, SP), (RJ, SP), (MG, ES), (RJ, MG)\}$ representada na Figura 1.5.



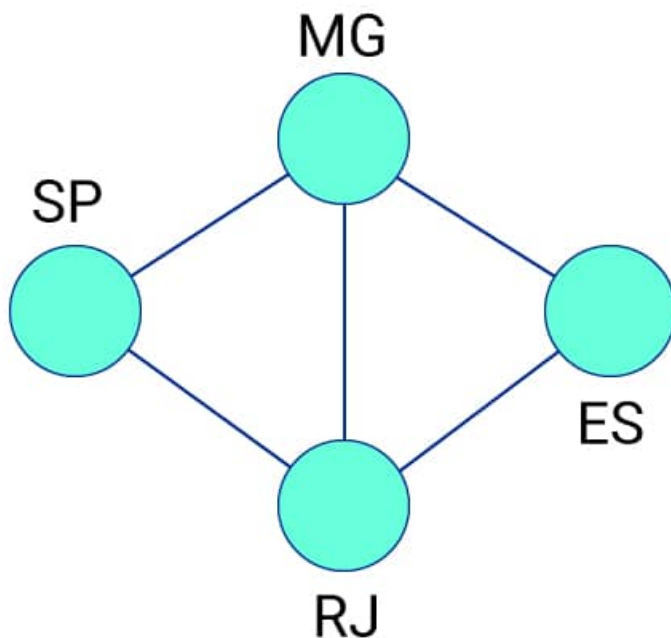
📷 Figura 1.5 – Adição da aresta (MG, ES)

Fonte: EnsineMe.

Finalização do grafo com adição da aresta

$(RJ, ES) \Rightarrow E = \{(MG, SP), (RJ, SP), (MG, ES), (RJ, MG), (RJ, ES)\}$

representada na Figura 1.6.



Assim, finaliza a construção do grafo-exemplo, que, nesse caso, representa a fronteira geográfica entre os estados.

ATENÇÃO

Importante ressaltar que o grafo é um conjunto não ordenado de pontos, portanto, não existe uma aresta que seja, obrigatoriamente, construída primeiro, segundo etc.

Isso significa que esse grafo pode ser construído de diferentes maneiras e o resultado será sempre o mesmo.

Partindo do grafo final, vários problemas podem ser resolvidos através dos algoritmos de caminhos sobre grafos, por exemplo: O menor caminho para se percorrer todos os estados sem repetir um nó; considerando um conjunto de produtos com determinado peso, como otimizar a entrega carregando a maior quantidade de produtos por cada trecho da viagem, entre outros. Alguns desses algoritmos serão vistos nos módulos 3 e 4.

DEFINIÇÕES SOBRE GRAFOS

Vamos considerar o seguinte conjunto de arestas e nós para montar um grafo:

1

Conjunto de nós $V(G) = (a, b, c, d)$.

2

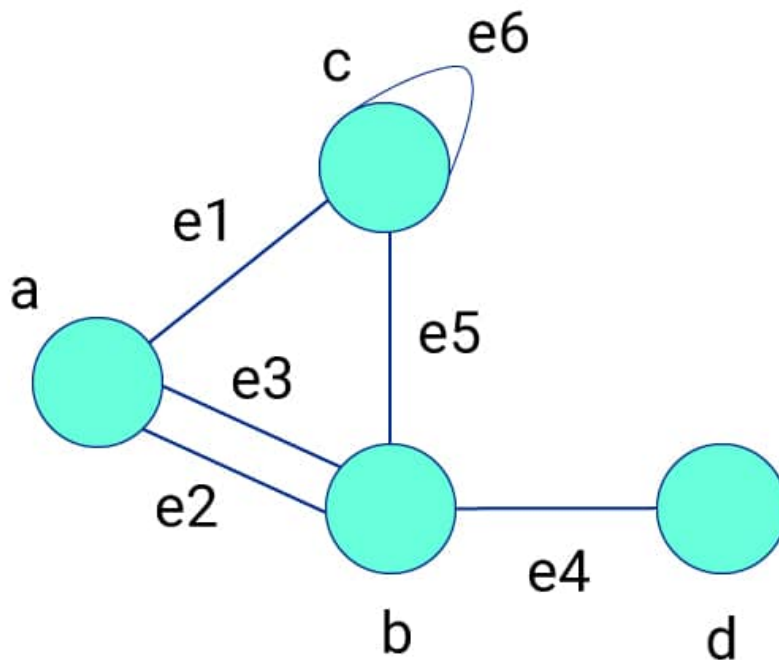
Conjunto de arestas $E(G) = \{e_1, e_2, e_3, e_4, e_5, e_6\}$.

3

Cada aresta é representada pelo par nomeado $\psi_G(e_1) = ac$, $\psi_G(e_2) = ab$, $\psi_G(e_3) = ab$, $\psi_G(e_4) = bd$, $\psi_G(e_5) = cb$ e $\psi_G(e_6) = cc$.

O conceito ψ_G é uma função que associa cada aresta de G a um par não ordenado de vértices de G .

No item 3, representamos o grafo da Figura 1.7 usando a função ψ .



📷 Figura 1.7 – Grafo exemplo.

Fonte: EnsineMe.

A partir desses grafos serão apresentadas algumas definições a respeito dos mesmos.

Definição 1 – Extremidades

Se e é uma aresta de G e $\psi_G(e) = uv$, então u e v são extremidades de e . Portanto, no grafo da figura 1.7, a e b são as extremidades de e_2 e e_3 .

Definição 2 – Vértices adjacentes

Se dois vértices são extremidades de uma mesma aresta, eles são ditos adjacentes. Algumas referências tratam como vértices vizinhos. Na figura 1.7, a e b ($\psi_G(e_2) = ab$) são adjacentes, enquanto c e d não são. Ou seja, não existe uma aresta que se inicie em c e finalize em d .

Definição 3 – Laço

É uma aresta cujas duas extremidades são idênticas, ou seja, as extremidades são o mesmo nó. Na figura 1.7, e_6 é um laço. Utilizando a definição de arestas $\psi_G(e_6) = cc$.

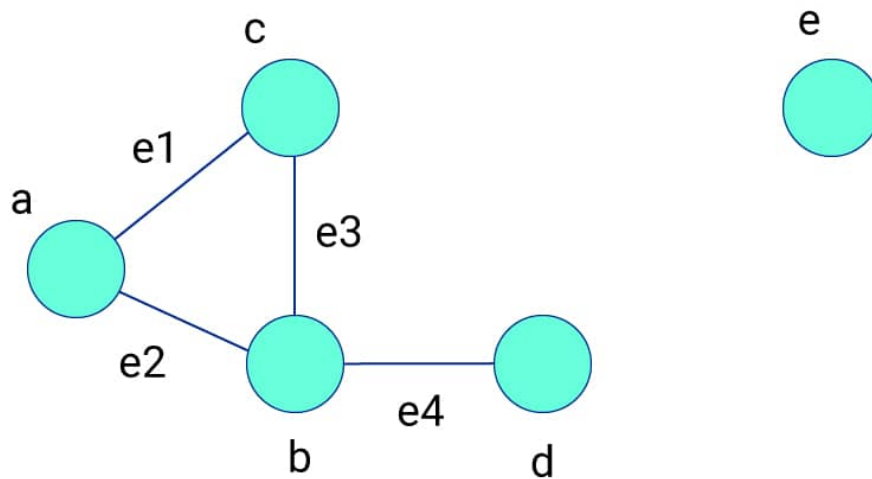
Definição 4 – Arestas paralelas

Essa definição consiste em duas arestas distintas com extremidades idênticas, ou seja, as duas chegam nos mesmos nós. Na figura 1.7, e_2 e e_3 são arestas paralelas. Utilizando a definição de arestas $\psi_G(e_2) = ab$, $\psi_G(e_3) = ab$.

Definição 5 – Grafos simples

Um grafo é dito simples se não contém arestas paralelas nem laços. O grafo da figura 1.7 não é

simples, pois, conforme as definições apresentadas anteriormente, possui um laço na aresta e_6 e arestas paralelas e_2 e e_3 . Ele pode, porém, ser redesenhado para se tornar um grafo simples, conforme a figura 1.8.



📷 Figura 1.8 – Transformação para grafo simples.

Fonte: EnsineMe.

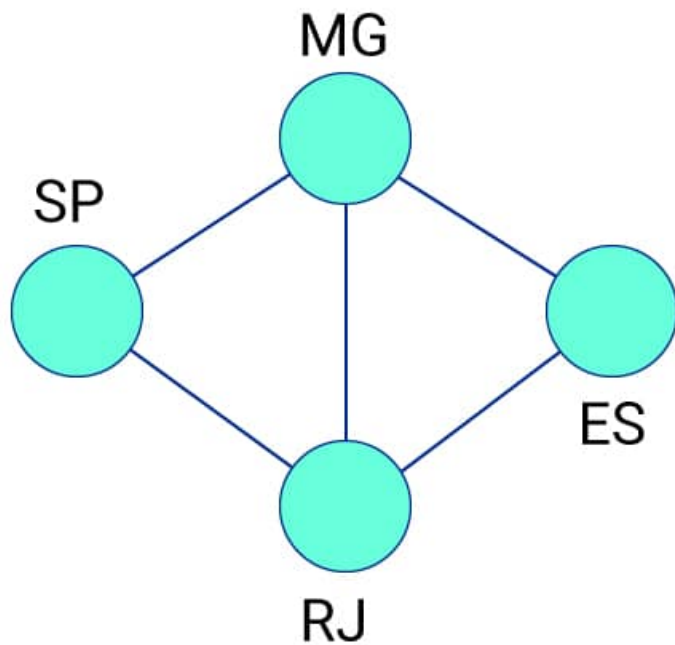
Outro conceito novo é o nó e , que não é vizinho a nenhum vértice do grafo, tornando-se um grafo isolado. Os grafos simples são mais utilizados de forma didática para simplificação dos problemas.

Definição 6 – Grau de vértices

Na figura 1.7, o vértice a tem 3 arestas ligadas a ele, o vértice c tem 3 arestas ligadas a ele e assim por diante. Dizemos que essas arestas são incidentes ao vértice.

O número de vezes que as arestas incidem sobre o vértice v é chamado grau do vértice v , simbolizado por $d(v)$. No nosso exemplo, $d(a) = 3$; $d(c) = 3$. Completando o grau do restante dos vértices, $d(b) = 4$ e $d(v) = 1$.

Considerando o grafo da figura 1.9 dos estados da região Sudeste.



📷 Figura 1.9 – Grafo dos estados da região Sudeste.

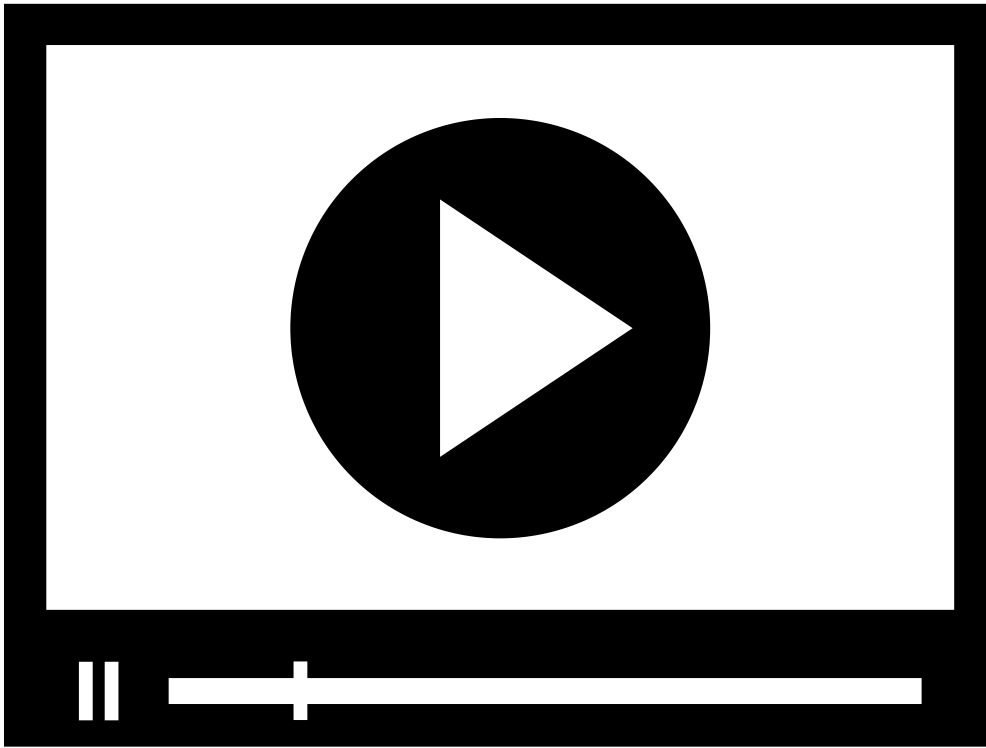
Fonte: EnsineMe.

O grau dos vértices: $d(\text{SP}) = 2$, $d(\text{MG}) = 2$, $d(\text{ES}) = 2$ e $d(\text{RJ}) = 3$. Se algum programa computacional interpretasse esse grafo, poderia indicar que o RJ é o estado que faz fronteira com 3 estados, porque 3 vértices incidem no nó RJ, e assim por diante.

Definição 7 – Grafo conexo

Um grafo $G = (V, E)$ é dito conexo quando existe um caminho entre quaisquer pares de vértices. Por exemplo, o grafo apresentado na figura 1.9 é considerado conexo porque é possível alcançar qualquer um dos estados a partir de qualquer nó.

Podemos concluir que caso haja um vértice não alcançável, este grafo será denominado desconexo. A figura 1.8 é um exemplo de um grafo desconexo.



CONHECENDO MELHOR A APLICAÇÃO DOS GRAFOS



VERIFICANDO O APRENDIZADO

MÓDULO 2

REPRESENTAÇÃO DE GRAFO

Para o entendimento e para a aplicação de grafos, é necessário conhecer alguns teoremas e tipos de grafos.

A partir dos tipos, existirão diversos algoritmos para se caminhar sobre os grafos.

Para essas representações, iremos considerar um grafo pela letra **G** e representaremos por **V(G)** e **A(G)**, respectivamente, os conjuntos de vértices e das arestas de G.

TEOREMA SOMA DOS GRAUS DOS VÉRTICES

Para todo grafo G, a soma dos graus dos vértices de um grafo é sempre o dobro do número de arestas.

Esse teorema pode ser representado pela fórmula:

$$\sum_{v \in V(G)} d(v) = 2 \cdot m$$



Atenção! Para visualização completa da equação utilize a rolagem horizontal

Onde:

$v \in V(G)$ – Todas as arestas dos grafos

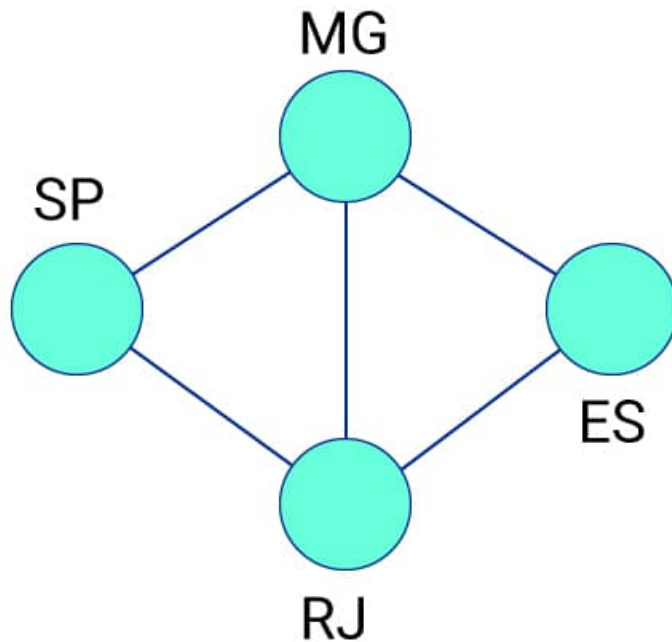
$d(v)$ – O grau de cada vértice

m – Arestas do vértice

Esse teorema pode ser demonstrado através da contagem dos vértices, pois quando contamos os graus dos vértices, estamos contando as extremidades das arestas uma vez.

Como cada aresta tem duas extremidades, cada aresta foi contada duas vezes.

Segue um exemplo para contagem na Figura 2.1, que representa o grafo de estados da região Sudeste que têm fronteira entre si.



📷 Figura 2.1 – Grafo para contagem dos vértices.

Fonte: EnsineMe.

Utilizando o teorema, temos que:

$$\sum_{\nu \in V(G)} d(\nu) = 2 \cdot m$$

📌 **Atenção!** Para visualização completa da equação utilize a rolagem horizontal

Onde:

$m = 5$ = número de arestas

$$d(SP) + d(MG) + d(RJ) + d(ES) = 2 + 3 + 3 + 2 = 10$$

Pela demonstração, realmente concluímos que em um grafo com 5 arestas a soma do grau de todos os vértices é igual a 10, ou seja, corresponde a $2.m = 10$.

Corolário: Todo grafo G possui um número par de vértices de grau ímpar.

Demonstração:

"SE TIVÉSSEMOS UM NÚMERO ÍMPAR DE VÉRTICES DE GRAU ÍMPAR A SOMA DOS GRAUS SERIA ÍMPAR. MAS A SOMA DOS GRAUS É O DOBRO DO NÚMERO DE ARESTAS E, PORTANTO, É UM NÚMERO PAR."

(JURKIEWICZ, 2009) .

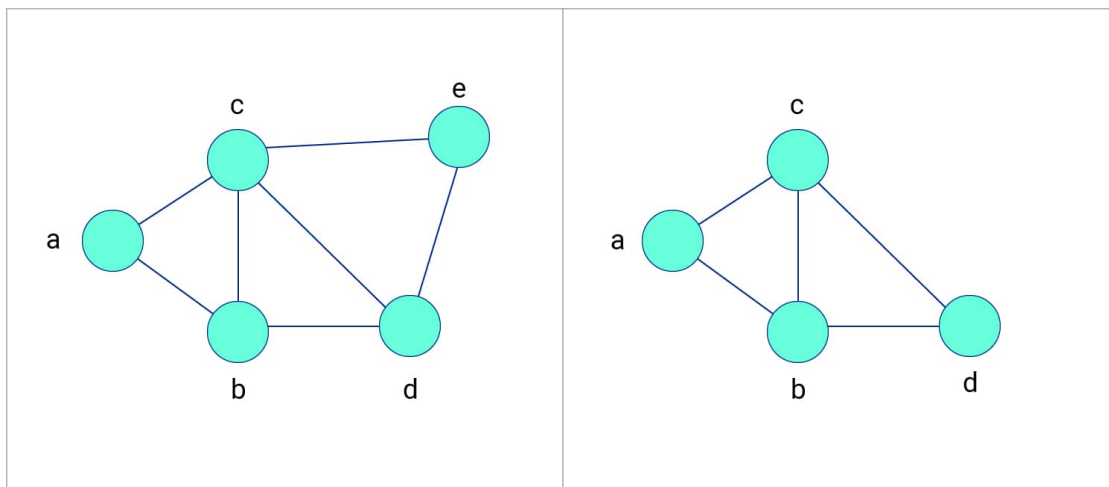
Novamente, pode ser utilizada a resolução do grafo da figura 2.1:

$d(SP) + d(MG) + d(RJ) + d(ES) = 2 + 3 + 3 + 2 = 10 \Rightarrow$ existem dois vértices ($d(MG)$ e $d(RJ)$) com grau ímpar de vértices (3).

OPERAÇÕES EM GRAFOS

a) Remoção de vértices

Quando se remove um vértice v , retira-se todas as arestas que têm v como extremidade. A partir da retirada do vértice, o novo grafo resultante é $V(G) - \{v\}$ pela remoção do vértice $v \in V(G)$, conforme apresentado na figura 2.2.



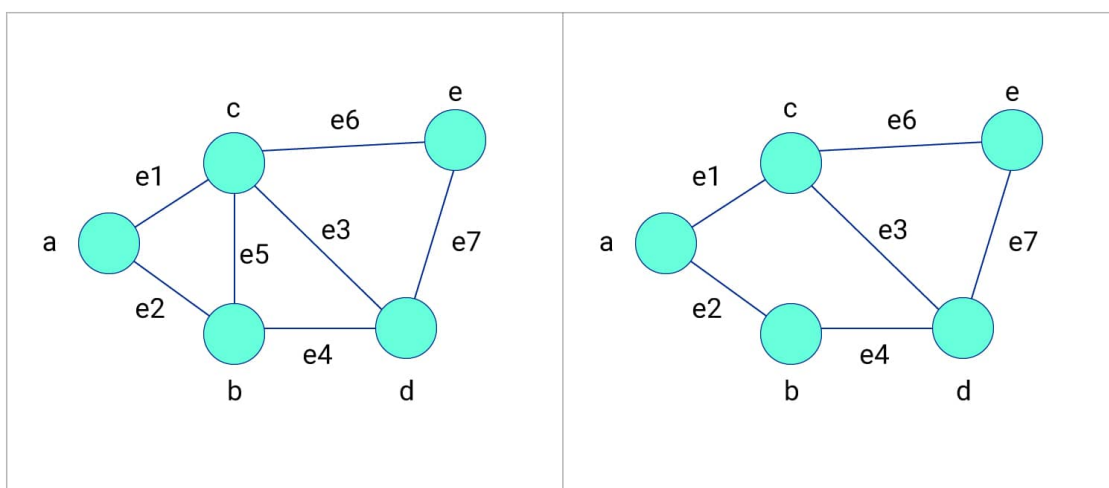
📷 Figura 2.2 – Remoção do vértice e do grafo.

Fonte: EnsineMe.

No grafo da esquerda, temos o grafo com todas os nós $V(G) = (a,b,c,d,e)$ e todas as arestas $E = \{(a,c), (a,b), (c,b), (b,d), (c,d), (c,e), (e,d)\}$. Com a retirada do vértice e, o novo grafo resultante será $V(G) = (a,b,c,d)$ e $E = \{(a,c), (a,b), (c,b), (b,d), (c,d)\}$, conforme exemplificado na representação do grafo à direita.

b) Remoção de arestas

Quando se remove uma aresta e, considerando um grafo $G = (V, E)$, obtém-se o grafo $G' = (V, E')$, onde $E' = E - \{e\}$.



📷 Figura 2.3 – Remoção da aresta e5 do grafo.

Fonte: EnsineMe.

O grafo representado à esquerda da Figura 2.3 está com todas as arestas. Com a retirada da aresta e5, o novo grafo resultante será $V(G) = (a,b,c,d,e)$ e $E = \{(a,c), (a,b), (c,b), (b,d), (c,d), (c,e), (e,d)\}$, conforme a Figura 2.3 à direita.

TIPOS ESPECIAIS DE GRAFOS

GRAFO COMPLETO

O grafo é dito completo quando existem arestas ligando todos os nós do mesmo.

Segundo Jurkiewicz (2009), um grafo completo é aquele no qual todo par de vértices é ligado por uma aresta. Um grafo completo com n vértices é denotado por K_n .

★ EXEMPLO

Considere um campeonato no qual todos os times devem jogar com todos os times. As equipes seriam formadas pelas respectivas turmas do ensino médio: 11A, 11B, 21A, 22B, 31A e 31 B. Esse tipo de exemplo apresenta uma situação que pode ser representada através de um grafo completo.

Na figura 2.4, foi representado o grafo da tabela desse campeonato.

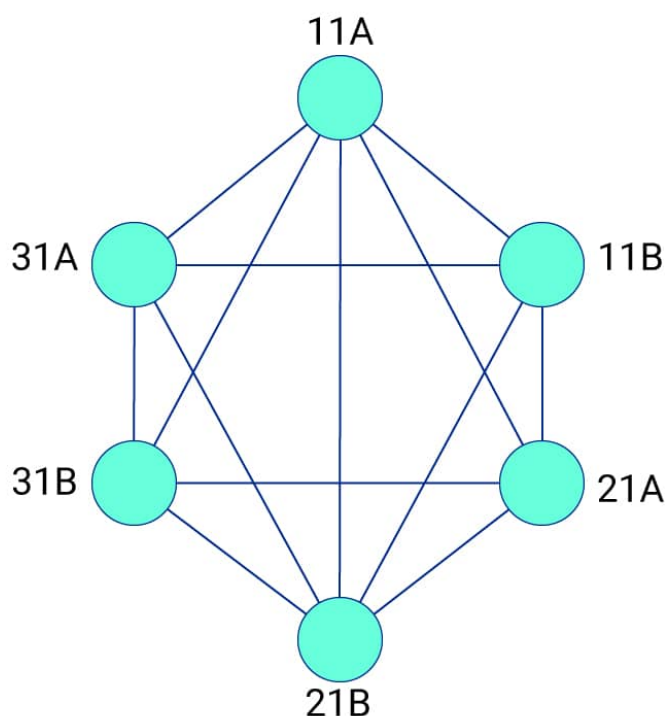


 Figura 2.4 – Grafo representativo da tabela do campeonato.

Fonte: EnsineMe.

Cada nó representa um time do campeonato, e cada aresta representa um jogo do campeonato. Nesse nosso exemplo é K_6 .

Pelo teorema de grafos, a quantidade de arestas em um grafo com n nós possui a seguinte fórmula:

$$\frac{n(n-1)}{2}$$

 **Atenção!** Para visualização completa da equação utilize a rolagem horizontal

Como podemos chegar a esse valor?

Uma maneira de chegar a esse resultado é considerar o fato que o número de arestas em um grafo completo de n vértices corresponde a todos os pares possíveis uv , onde u e v são vértices.

Assim, o número de vértices é um fatorial de n de 2 a 2, conforme a figura 2.5.

$$\frac{n!}{(n-2)!2!} = \frac{n(n-1)(n-2)!}{(n-2)!2} = \frac{n(n-1)}{2}$$

 **Atenção!** Para visualização completa da equação utilize a rolagem horizontal

 Figura 2.5 – Equação da quantidade de áreas de um grafo completo. Fonte: EnsineMe.

Considerando o grafo completo da Figura 2.3 com 6 nós, temos o seguinte resultado apresentado na Figura 2.6.

$$k_6 = \frac{n(n-1)}{2} = \frac{6(6-1)}{2} = \frac{6 \times 5}{2} = 15$$

 **Atenção!** Para visualização completa da equação utilize a rolagem horizontal

 Figura 2.6 – Equação da quantidade de áreas de um grafo completo. Fonte: EnsineMe.

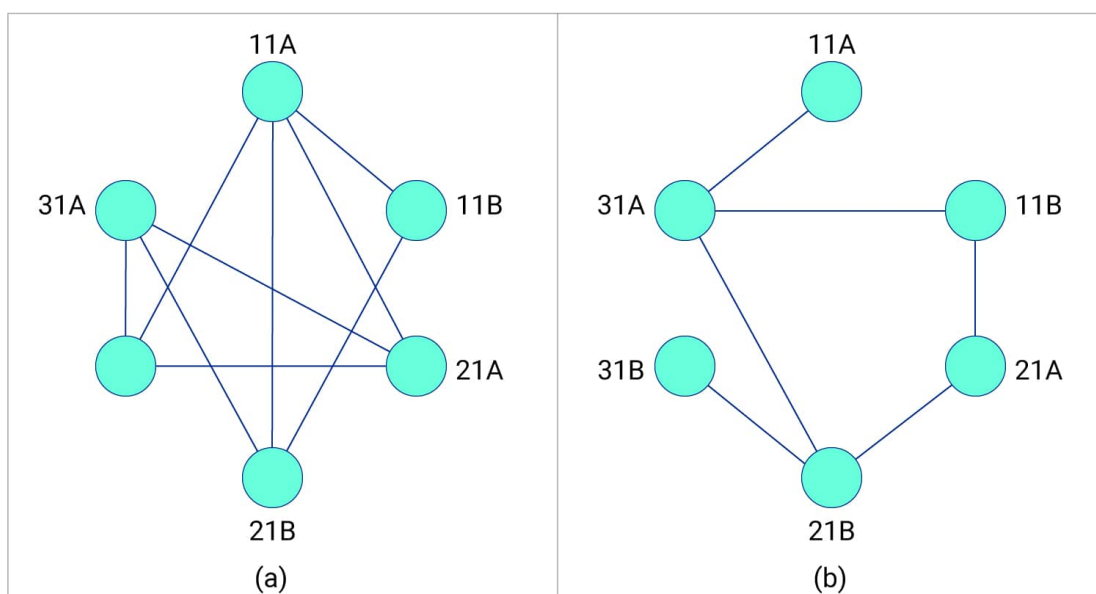
GRAFO COMPLEMENTAR

Considerando o exemplo apresentado anteriormente do campeonato, vamos supor que exista a necessidade de construir um grafo que represente os jogos do campeonato que ainda não foram disputados.

Na Figura 2.7(a), temos o grafo que apresenta todos os jogos disputados.

Para saber quais são os que não foram disputados, podemos inserir todas as arestas faltantes em (a) e remover todas as arestas que já estavam lá.

O resultado dessa operação é o grafo apresentado na Figura 2.7(b), que representa os jogos não disputados. Esse grafo resultante é complementar de (a).



📷 Figura 2.7 – Grafos complementares dos jogos restantes.

Fonte: EnsineMe.

Portanto, na Figura 2.7, o grafo (b) é chamado de grafo complementar do grafo G (a) denotado por \overline{G} .

Analisando os grafos (a) e (b) chega-se à conclusão que:

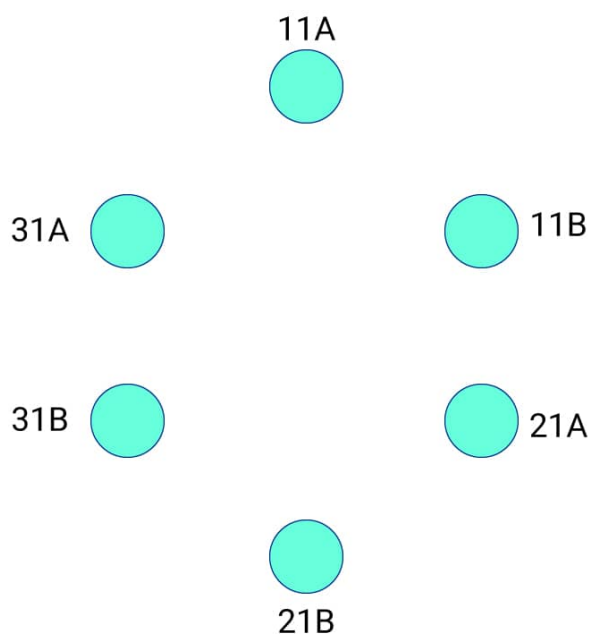
$V(G) = V(\overline{G})$ e que a união dos conjuntos de arestas dos dois grafos $(A(G) \cup A(\overline{G}))$ inclui todas as arestas de G.

Uma propriedade derivada é que o complementar de um **grafo sem arestas** é um **grafo completo e vice-versa**.

GRAFO NULO OU VAZIO

Um grafo G é nulo ou vazio quando o conjunto de arestas $A(G)$ é vazio. Considerando, ainda, o exemplo do campeonato, antes do seu começo nenhum jogo havia sido jogado.

O exemplo é apresentado na Figura 2.8.



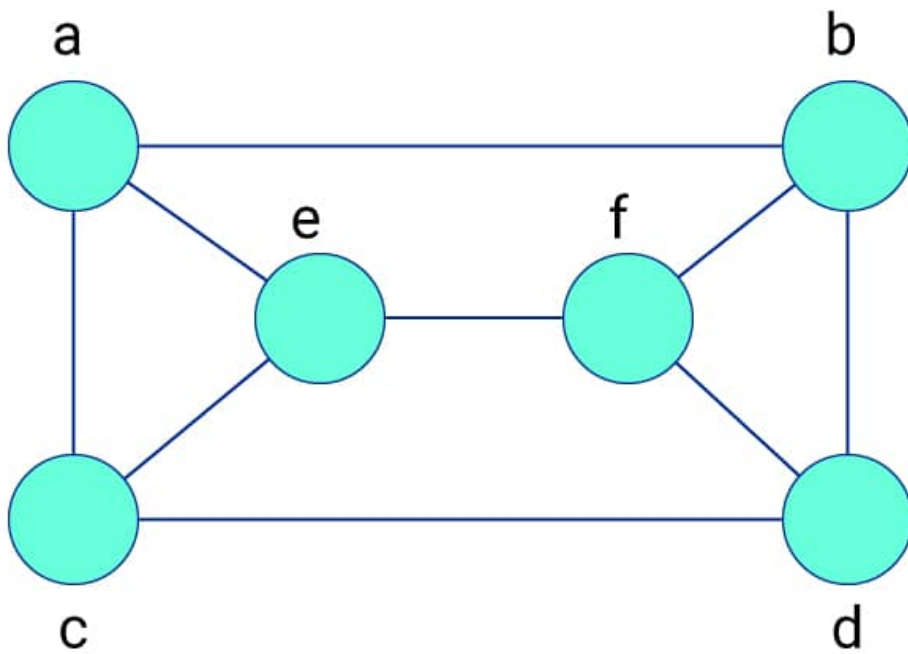
📷 Figura 2.8 – Grafo nulo ou vazio do campeonato.

Fonte: EnsineMe.

GRAFO CONEXO REGULAR

Um grafo é **regular** (de grau k , ou ainda k -regular) quando todos os seus vértices têm o mesmo grau (k), ou seja, possuem a mesma quantidade de arestas.

Na Figura 2.9, é representado um grafo 3-regular, isto é, todos os vértices têm grau 3.



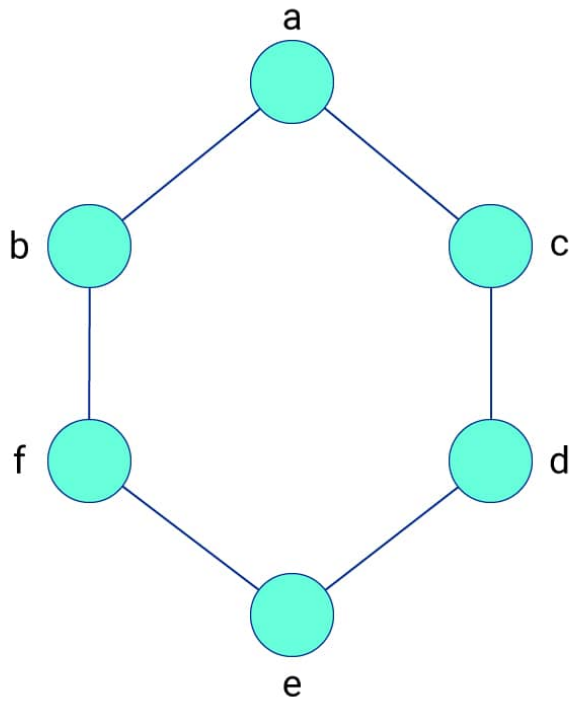
📷 Figura 2.9 – Grafo conexo regular.

Fonte: EnsineMe.

CICLO

O ciclo é considerado uma especialização do grafo conexo regular onde todos os grafos possuem vértice grau 2.

Na Figura 2.10, foi realizada uma transformação do grafo da Figura 2.9 para um considerado como **grafo ciclo**.



📷 Figura 2.10 – Grafo conexo regular.

Fonte: EnsineMe.

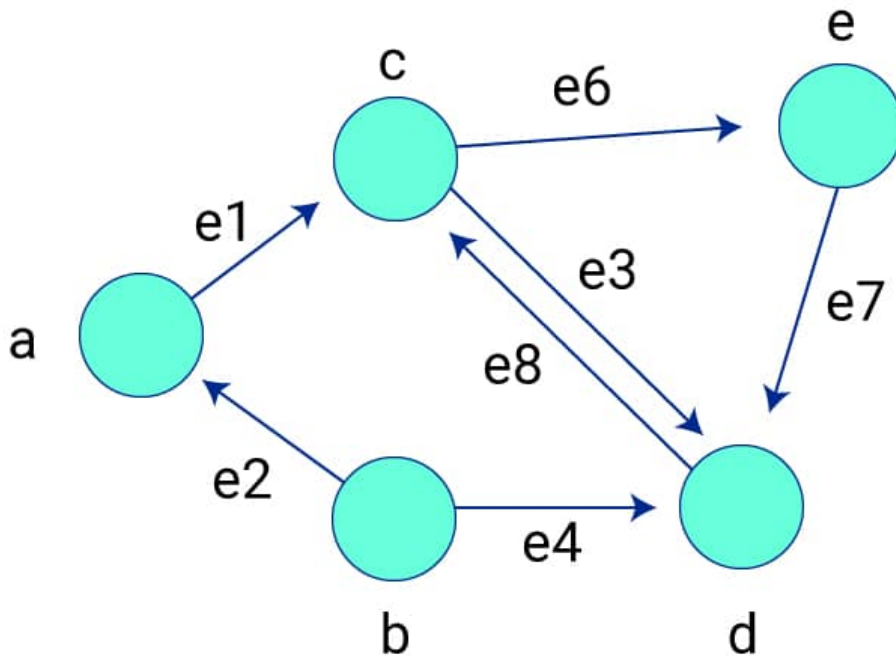
GRAFOS DIRECIONADOS

Um grafo $G = (V, E)$ é um conjunto não vazio V , cujos elementos são chamados vértices, e um conjunto E de arestas. Uma aresta é um par ordenado (v_j, v_k) .

Diz-se que cada aresta (v_j, v_k) possui uma única direção de v_j para v_k . Também a aresta (v_j, v_k) é dita divergente de v_j e convergente a v_k .

"O NÚMERO DE ARESTAS DIVERGENTES E CONVERGENTES DE UM VÉRTICE SÃO CHAMADOS GRAU DE SAÍDA E GRAU DE ENTRADA, RESPECTIVAMENTE."

A Figura 2.11 ilustra um exemplo de grafo direcionado.



📷 Figura 2.11 – Grafo Direcionado V.

Fonte: EnsineMe.

O grafo direcionado da Figura 2.11 pode ser representado com os seguintes nós e arestas:

$$V = (a,b,c,d,e)$$

$$E = \{(a,c),(b,a),(b,d),(e,d),(c,e),(c,d),(d,c)\}$$

📢 ATENÇÃO

É importante ressaltar a ordenação das arestas dos grafos. Por exemplo, a aresta e1 deve ser representada por (a,c), indicando que é divergente do nó a e convergente no nó c.

Teorema 1-6: A soma dos graus de saída (de entrada) de um grafo direcionado é igual ao número de arestas no grafo.

"O TEOREMA 1-6 É FÁCIL DE SER COMPREENDIDO, CONSIDERANDO QUE CADA ARESTA CONTRIBUI EXATAMENTE PARA UM GRAU DE ENTRADA E UM GRAU DE SAÍDA."

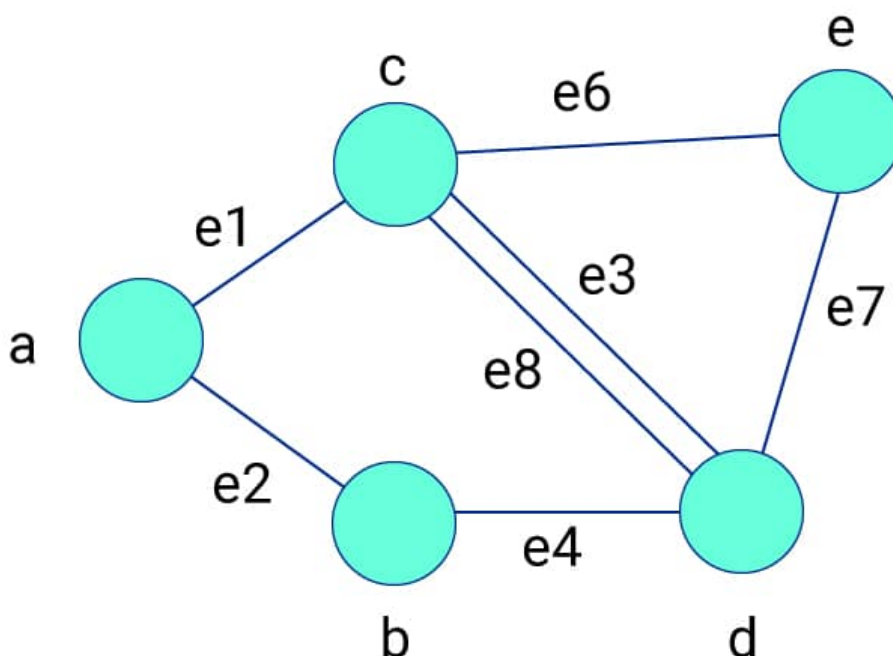
(FEOFIOFF; KOHAYAKAWA; WAKABAYASHI, 2011)

Partindo do grafo direcionado da Figura 2.11, obtém-se os seguintes valores:

Grau de Saída $V = 7 \Leftrightarrow$ Número de arestas do grafo

GRAFO SUBJACENTE DO GRAFO DIRECIONADO

O grafo obtido substituindo cada aresta de um grafo direcionado G por uma aresta não direcionada é chamado grafo subjacente de G . A Figura 2.12 ilustra o grafo subjacente do grafo direcionado da figura 2.11.



📷 Figura 2.12 – Grafo Subjacente V' .

Fonte: EnsineMe.

REPRESENTAÇÃO DOS GRAFOS

REPRESENTAÇÃO POR MATRIZES

As matrizes são um modelo matemático comum de “informar” uma estrutura de grafo para um computador.

Lembrando que uma matriz nada mais é do que uma tabela com linhas e colunas.

Uma das formas mais utilizadas para representar grafos é via a matriz de adjacência. **Mas o que é uma matriz de adjacência?**

Segundo explicação de Carvalho (2005), seja $A = [a_{ij}]$ uma matriz $n \times n$, onde n é o número de nós de um grafo $G = (V, E)$ qualquer. A matriz de adjacência A é construída da seguinte forma, apresentada na Figura 2.13:

$$A(i, j) = \begin{cases} 1 & \text{se existe uma aresta entre os vértices } i \text{ e } j \\ 0 & \text{caso contrário} \end{cases}$$

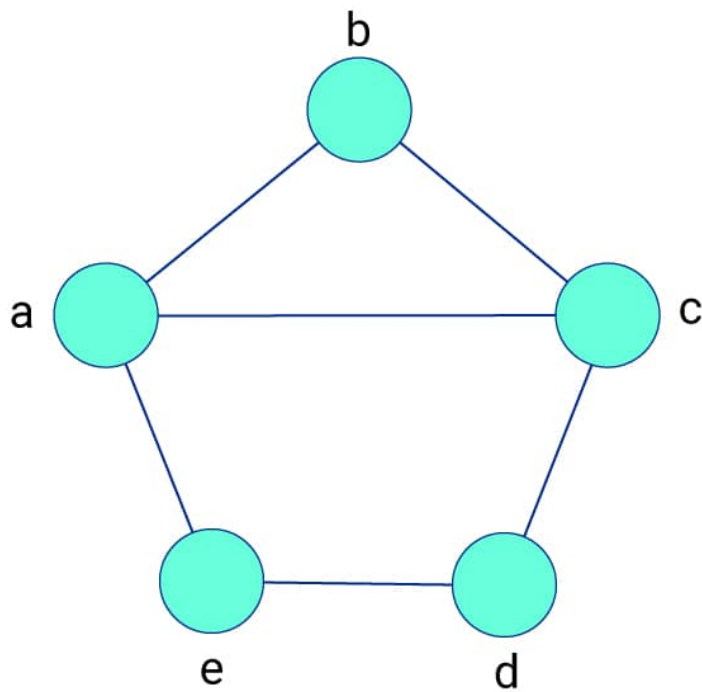



Atenção! Para visualização completa da equação utilize a rolagem horizontal



Figura 2.13 – Fórmula representação do grafo em matriz. Fonte: EnsineMe.

Considerando o grafo da Figura 2.14.



 Figura 2.14 – Grafo exemplo.

Fonte: EnsineMe.

Para que possamos montar a matriz de adjacência, é necessário que os nós sejam rotulados, ou seja, que sejam numerados e correspondam à linha e coluna.

Então, para montar a matriz de adjacência da Figura 2.14, vamos partir da matriz a seguir, rotulando cada linha e cada coluna com o nome do nó do grafo G.

$$A(i, j) = \begin{matrix} & \begin{matrix} a & \dots & e \end{matrix} \\ \begin{matrix} a \\ \vdots \\ e \end{matrix} & \begin{bmatrix} \square & \dots & \square \\ \vdots & \ddots & \vdots \\ \square & \dots & \square \end{bmatrix} \end{matrix}$$

 **Atenção!** Para visualização completa da equação utilize a rolagem horizontal

Como na Figura 2.14 existe uma aresta ligando o vértice a aos vértices b e e, as posições da matriz de adjacência (a,b) e (a,e) terão **valor 1**.

Na Figura 2.15, obtemos a matriz de adjacência para a Figura 2.14:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

📷 Figura 2.15 – Matriz de adjacência resultante.

Fonte: EnsineMe.

A complexidade em termos de espaço de memória de um algoritmo que use uma matriz de adjacência para armazenar o grafo é $O(n^2)$.

📢 ATENÇÃO

Uma observação importante é que apenas grafos que não possuem arestas paralelas é que podem ser representados através de uma matriz de adjacência.

Vale ressaltar ainda que as entradas ao longo da diagonal principal de X são todas nulas se, e somente se, o grafo não possuir laços.

MATRIZ DE INCIDÊNCIA

Seja G um grafo com n vértices e m arestas. Sua matriz de incidência é uma matriz de ordem $n \times m$.

Então, a matriz de incidência $B = [b_{ij}]$ de um grafo $G = (V, E)$, com $V = (v_1, v_2, \dots, v_n)$ e $E = (e_1, e_2, \dots, e_m)$, é definida da seguinte forma apresentada na Figura 2.16:

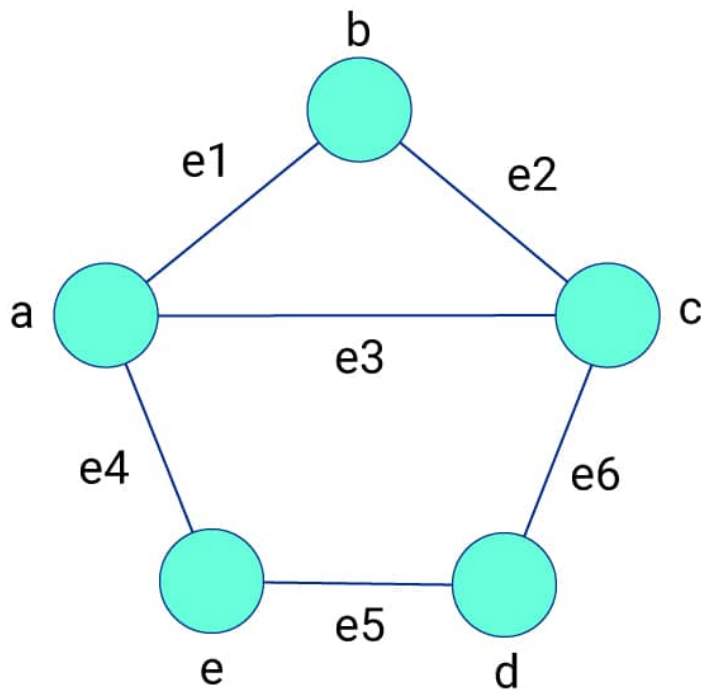
$$B(i, j) = \begin{cases} 1 & \text{se a aresta } e_j \text{ é incidente no vértice } i \\ 0 & \text{caso contrário} \end{cases}$$



Atenção! Para visualização completa da equação utilize a rolagem horizontal

📷 Figura 2.16 – Fórmula de representação do grafo incidente em matriz. Fonte: EnsineMe.

Vamos considerar o grafo da Figura 2.17 para montar nossa matriz de incidência.



📷 Figura 2.17 – Grafo exemplo incidente.

Fonte: EnsineMe.

Para que possamos montar a matriz de incidência, é necessário que os nós sejam rotulados, isto é, que sejam numerados e correspondam à linha e coluna.

Portanto, para montar a matriz de incidência da Figura 2.17, vamos partir da matriz a seguir, rotulando cada linha com o nome do nó do grafo e cada coluna com o nome da aresta.

$$A(i, j) = \begin{matrix} & \begin{matrix} e1 & \dots & e6 \end{matrix} \\ \begin{matrix} a \\ \vdots \\ e \end{matrix} & \begin{bmatrix} \square & \dots & \square \\ \vdots & \ddots & \vdots \\ \square & \dots & \square \end{bmatrix} \end{matrix}$$



Atenção! Para visualização completa da equação utilize a rolagem horizontal

Como na Figura 2.17 existem três arestas ($e1$, $e3$ e $e4$) incidindo no vértice a , então as posições da matriz de incidência $(e1,a)$, $(e3,a)$ e $(e4,a)$ terão **valor 1**.

Na Figura 2.18, temos a matriz de incidência resultante do grafo da Figura 2.17.

$$B = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

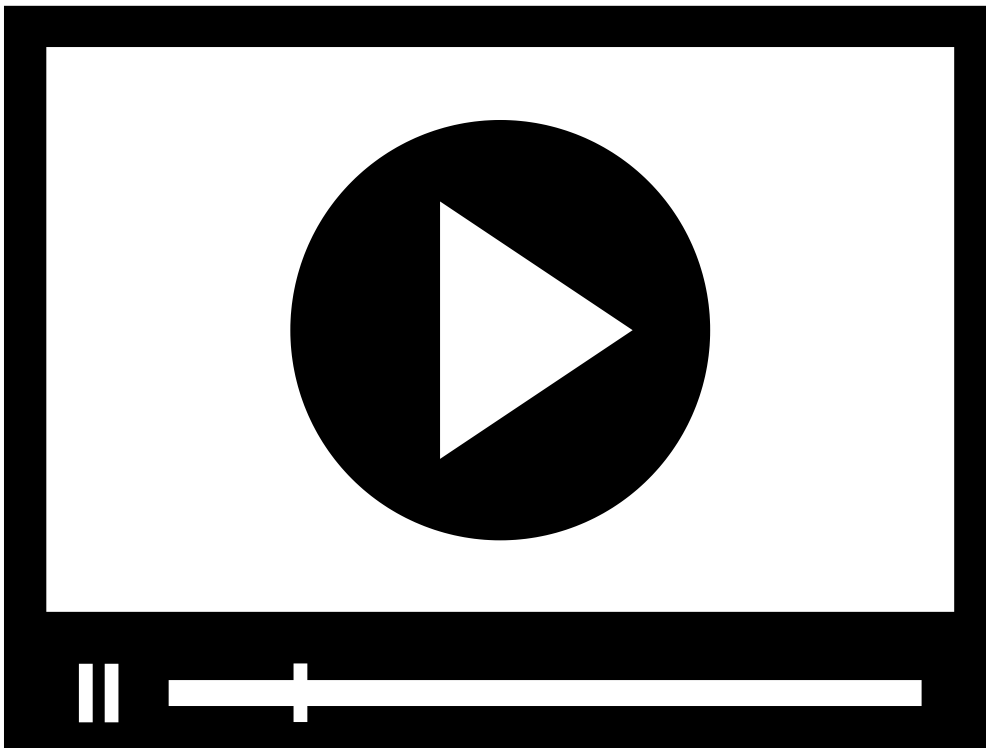
 Figura 2.18 – Matriz incidente resultante.

Fonte: EnsineMe.

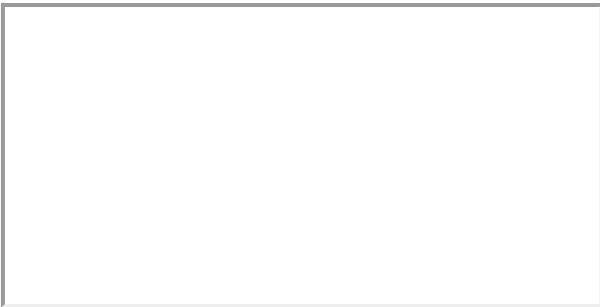
A complexidade em termos de espaço de memória de um algoritmo que use uma matriz de adjacência para armazenar o grafo é **$O(nm)$** .

ATENÇÃO

Uma observação importante é que somente grafos que não possuam laços podem ser representados através de uma matriz de incidência.



REPRESENTANDO OS GRAFOS COMPUTACIONALMENTE



VERIFICANDO O APRENDIZADO

MÓDULO 3

ALGORITMOS DE BUSCA

A busca visa a resolver uma questão básica: **como explorar um grafo?** Ou seja, deseja-se obter um processo de como caminhar pelos nós e arestas de um grafo.

Algoritmos de busca são bastante utilizados em aplicações da área de inteligência artificial e também para algoritmos de redes sociais.

★ EXEMPLO

Por exemplo, descobrir qual a distância entre dois perfis, ou seja, quantas arestas existem entre dois nós em um grafo representando a rede social. No caso, cada nó é um perfil e cada aresta representa um outro perfil que está adicionado à rede social do primeiro, ou seja, são dois perfis que se relacionam.

Basicamente, existem duas técnicas de busca em grafos, de acordo com Feofiloff (2019):

BUSCA EM PROFUNDIDADE (DEPTH-FIRST SEARCH – DFS).



BUSCA EM LARGURA (BREADTH-FIRST SEARCH).

Tipicamente, os dois exemplos são algoritmos recursivos, pois, a partir do resultado da visita de um nó, chama-se o próximo nó a ser visitado.

BUSCA EM PROFUNDIDADE

A busca em profundidade é um algoritmo para percorrer o grafo, com a estratégia de buscar o vértice “mais profundo” no mesmo, sempre que possível.

De forma simples, podemos dizer que uma busca em profundidade inicia selecionando um dos nós como raiz, e depois vai descendo em cada um dos ramos, até não ser mais possível.

Quando isso acontecer, o algoritmo retrocede até um nó que não tenha sido explorado. Cada nó visitado é considerado marcado e é atribuído um número a ele.

"UMA BUSCA É DITA EM PROFUNDIDADE QUANDO O CRITÉRIO DE ESCOLHA DE VÉRTICE MARCADO (VISITADO) OBEDECER A: DENTRE TODOS OS VÉRTICES MARCADOS E INCIDENTES A ALGUMA ARESTA AINDA NÃO EXPLORADA, ESCOLHER AQUELE MAIS RECENTEMENTE ALCANÇADO NA BUSCA."

(CARVALHO, 2005)

Caso o vértice não possua mais arestas a serem exploradas, deve-se voltar para o vértice anterior, até encontrar um que possua arestas a serem exploradas – conhecido como backtrack na área de inteligência artificial.

Vamos considerar um grafo $G = (V, E)$ que contém n vértices. Seja também uma representação que indica, para cada vértice, se ele foi visitado ou não.

Na Figura 3.1, apresentamos uma versão recursiva do algoritmo de busca em profundidade que visita todos os vértices de um grafo G :

procedimento Busca(G : Grafo, v :verticeInicial)

Para Cada vértice v de G :

Marque v como não visitado

Para Cada vértice v de G :

Se v não foi visitado:

Busca-Prof(v)

procedimento Busca-Prof(v : vértice)

Marque v como visitado

Para Cada vértice w adjacente a v :

Se w não foi visitado:

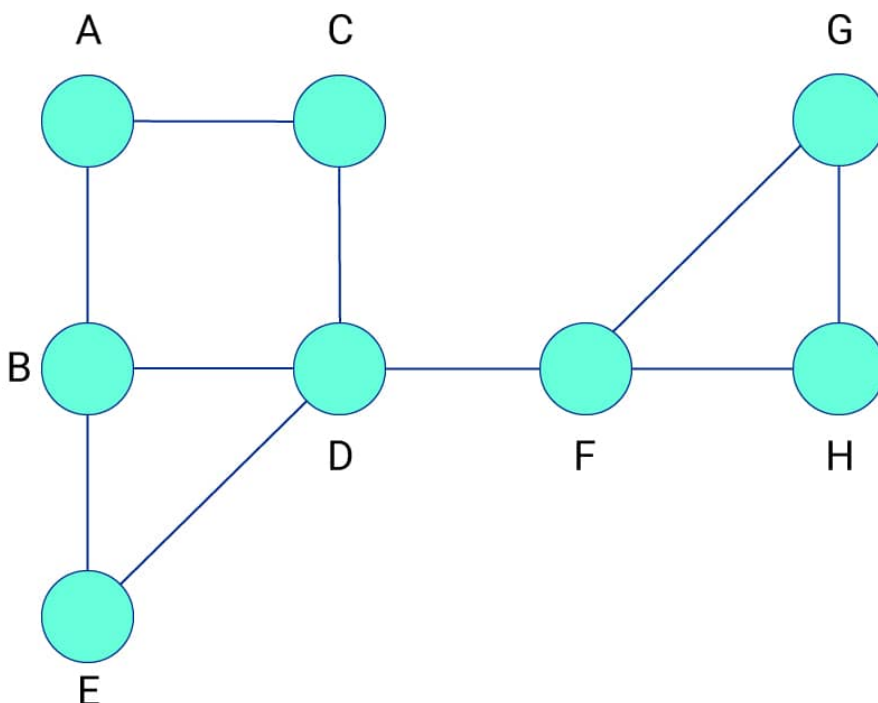
Busca-Prof(w)

📷 Figura 3.1 – Algoritmo de busca em profundidade. Fonte: CORMEN *et al.*, 2002.

Esse algoritmo funciona em grafos conexos, dessa forma, pode-se chamar diretamente a

função Busca-Prof, escolhendo arbitrariamente um vértice inicial e a partir desse percorrer os outros vértices.

Vamos agora visualizar como funciona a execução do algoritmo para o grafo da Figura 3.2, que é conexo.

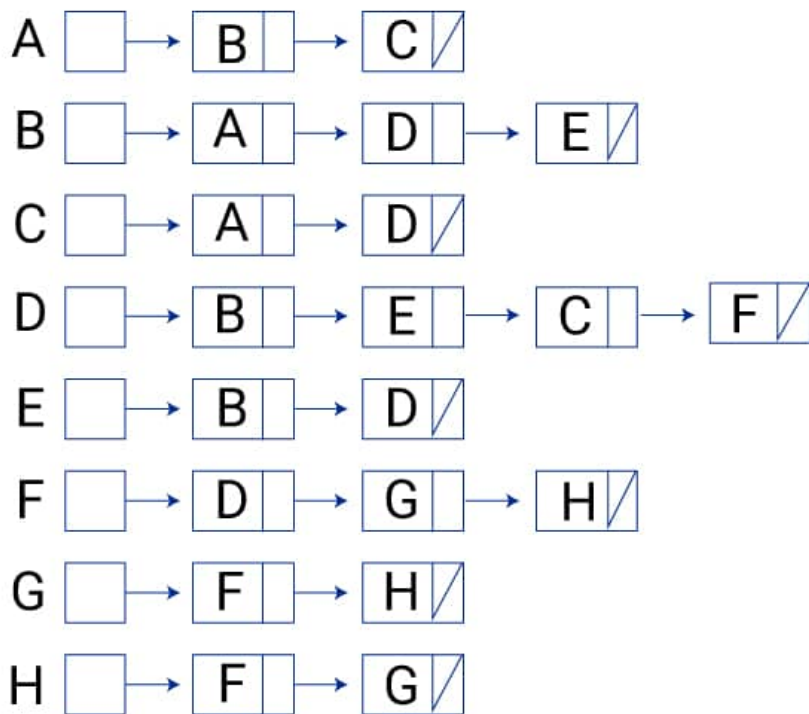


📷 Figura 3.2 – Grafo exemplo para busca em profundidade.

Fonte: EnsineMe.

Para facilitar a implementação do algoritmo, deve-se usar uma estrutura de adjacência como apresentada a seguir na Figura 3.3.

Essa estrutura de adjacência é montada criando uma lista para cada vértice do grafo indicando quem são seus vizinhos. Por exemplo, para o nó A nós temos os nós B e C como vizinhos, portanto, temos a lista encadeada A, B e C. De forma análoga, podemos analisar os outros nós.



📷 Figura 3.3 – Estrutura de adjacências representando o grafo exemplo.

Fonte: EnsineMe.

Considerando o grafo apresentado na Figura 3.2 e a estrutura de adjacência apresentada na Figura 3.3, vamos ver a simulação da execução do algoritmo de busca em profundidade, ilustrada na Figura 3.4:

Busca-Prof(A)

Busca-Prof(B)

Busca-Prof(D)

Busca-Prof(E)

Busca-Prof(C)

Busca-Prof(F)

Busca-Prof(G)

Busca-Prof(H)

Retorna H
Retorna G
Retorna F
Retorna C
Retorna E
Retorna D
Retorna B
Retorna A

📷 Figura 3.4 – Simulação da execução do algoritmo de busca em profundidade. Fonte: EnsineMe.

1

Inicialmente, o algoritmo percorre todos os vértices v do grafo G e os marca como não lidos.

Após essa etapa, a busca irá começar de forma efetiva a partir de um vértice do grafo.

2

3

Para nosso exemplo, iniciamos pelo vértice A. Como ele não foi visitado, ele vai chamar a função Busca-Prof(A).

Ao chamar a função Busca-Prof, o nó A vai ser marcado como visitado e, para cada nó não visitado de A, ele irá chamar a função Busca-Prof. Nesse caso, será a chamada Busca-Prof(B), porque é o primeiro nó na estrutura de adjacências para o vértice A.

4

5

A chamada Busca-Prof(B) vai percorrer a lista B da estrutura de adjacências. Como o nó A já foi visitado, ele agora irá visitar o nó D, chamando Busca-Prof(D).

Será percorrida então a lista do nó D. Como o nó B já foi visitado, será chamado Busca-Prof(E).

6

7

Quando o algoritmo chega ao vértice E, não há nenhum vértice vizinho que não tenha sido visitado pelo algoritmo. Este, então, realizará a estratégia de backtracking retornando dessa chamada para continuar a busca a partir do vértice anterior, que é o D.

A busca continua com o próximo vizinho de D que não foi visitado: o vértice C.

8

9

Como o vértice C não tem vizinhos que não foram visitados, o algoritmo retorna imediatamente ao D, para continuar a busca com o vértice F.

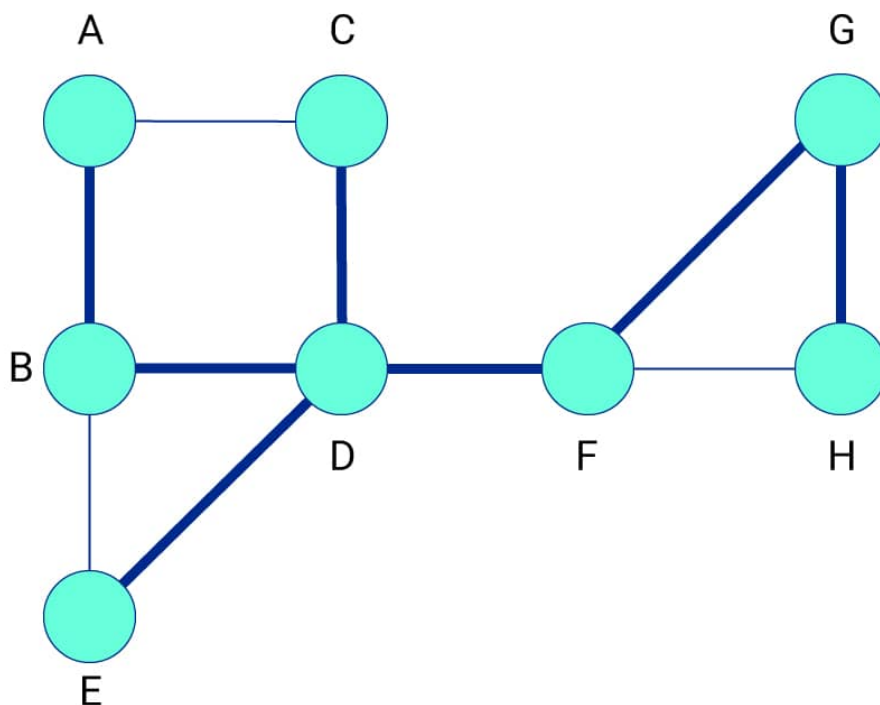
Por sua vez, o vértice F possui dois vértices vizinhos que não foram buscados: G e H.

10

11

A busca é feita no nó G e depois no nó H, chegando ao último vértice do grafo.

A árvore de busca obtida após a execução do algoritmo é apresentada na Figura 3.5.



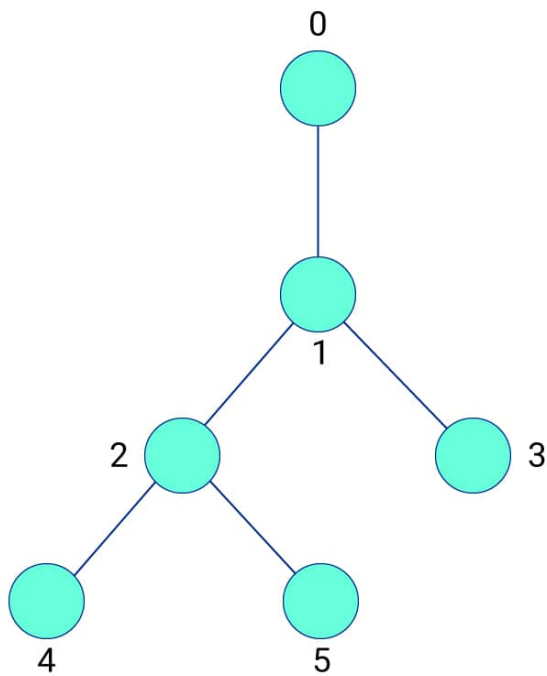
📷 Figura 3.5 – Árvore gerada após execução do algoritmo.

Fonte: EnsineMe.

Exemplo 1 – Busca em profundidade

Baseada no grafo da Figura 3.6, a simulação da execução do algoritmo da busca em

profundidade é apresentada na Figura 3.7:



📷 Figura 3.6 – Grafo para busca em profundidade.

Fonte: EnsineMe.

Busca-Prof(0)

Busca-Prof(1)

Busca-Prof(2)

Busca-Prof(4)

Retorna 4

Busca-Prof(5)

Retorna 5

Retorna 2

Busca-Prof(3)

Retorna 3

Retorna 1

Retorna 0

📷 Figura 3.7 – Simulação da execução do algoritmo de busca em profundidade – Exemplo 1.

Fonte: EnsineMe.

ANÁLISE DA COMPLEXIDADE DO ALGORITMO

Para implementar esse algoritmo, a estrutura de adjacência aparece como candidata ideal. Isso porque a principal operação efetuada pelo algoritmo é a escolha de um vértice adjacente.

Supondo então que a estrutura de adjacência é utilizada para implementar a busca, podemos ver que o tempo de execução do algoritmo é em $O(\max(a, n))$, onde **a** e **n** representam o número de arestas e de vértices, respectivamente.

Como cada vértice deve ser visitado, o algoritmo necessariamente fará n chamadas do procedimento Busca-Prof, muitas delas recursivas.

Em cada chamada, todos os vértices adjacentes serão testados.

No total, o número de testes realizados será igual ao número de arestas. Então, o tempo total gasto para as chamadas Busca-Prof é em $O(a)$.

A esse tempo devemos acrescentar o tempo gasto no procedimento busca: O tempo de inicialização e o tempo para testar, para cada vértice, se ele foi marcado. No total, isso dá um tempo em $O(2n) = O(n)$ – considerando a função assintótica.

Portanto, a busca em profundidade tem um tempo de execução em $O(a+n)$.

"NA VERDADE TEMOS UM TEMPO EM $O(\max(a, n))$: SE O GRAFO TEM MAIS ARESTAS QUE VÉRTICES, TEMOS UM TEMPO EM $O(a)$. NO CASO CONTRÁRIO, TEMOS UM TEMPO EM $O(n)$."

(CORMEN *et al.*, 2002)

BUSCA EM LARGURA

Em uma busca em largura a partir de um vértice v , esperamos que todos os vizinhos de v sejam visitados antes de continuar a busca mais profundamente.

Contrariamente à busca em profundidade, o algoritmo de busca em largura não é recursivo. Em outras palavras, a ideia da busca em largura consiste em processar todos os nós em um dado nível antes de caminhar para um nível mais alto.

Considerando um grafo $G = (V, E)$ que contém n vértices e uma representação que indica, para cada vértice, se ele foi visitado ou não.

Na Figura 3.8, é apresentado um algoritmo iterativo de busca em profundidade que visita todos os vértices:

procedimento Busca(G : Grafo)

Para cada vértice v de G :

Marcar v como não visitado

Para cada vértice v de G :

Se v não foi visitado:

Busca-Largura(v)

procedimento Busca-Largura(v : vértice)

Inicializar F

Marcar v como visitado

Colocar v no final de F

Enquanto F não vazio:

$u :=$ primeiro elemento de F

Retirar u de F

Para cada vértice w adjacente a u :

Se w não foi visitado:

Marcar w como visitado

Colocar w no final de F

 Figura 3.8 – Algoritmo de busca em largura. Fonte: CORMEN *et al.*, 2002.

Utilizando o algoritmo da Figura 3.8, vamos realizar a simulação da execução do algoritmo para o grafo apresentado na Figura 3.2, utilizando uma fila F .

Esta se comporta da seguinte forma de acordo com a execução do algoritmo:

A – Visita A e coloca A no final da Fila F

C B – Visita C e coloca C no final da Fila F

B E D – Visita B e coloca B no final da Fila F

E D – Visita E e coloca E no final da Fila F

D F – Visita D e coloca D no final da Fila F

F G H – Visita F e coloca F no final da Fila F

G H – Visita G e coloca G no final da Fila F

H – Visita H e coloca H no final da Fila F



Atenção! Para visualizaçãocompleta da tabela utilize a rolagem horizontal

Os valores retornados no algoritmo serão A, C, B, E, D, F, G, H.

A árvore de busca obtida após a execução do algoritmo é apresentada na Figura 3.9.

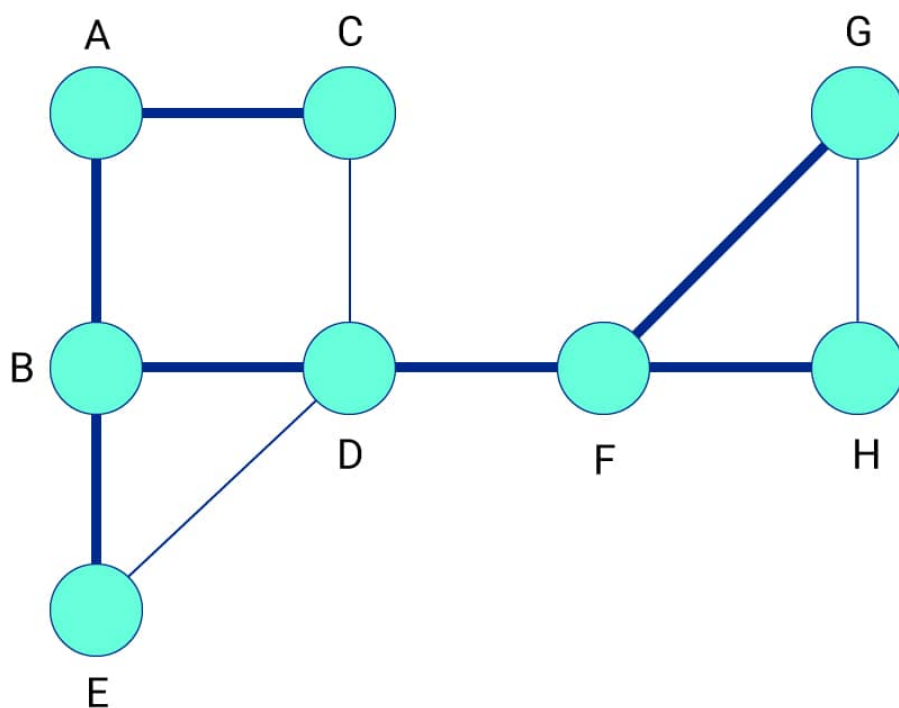
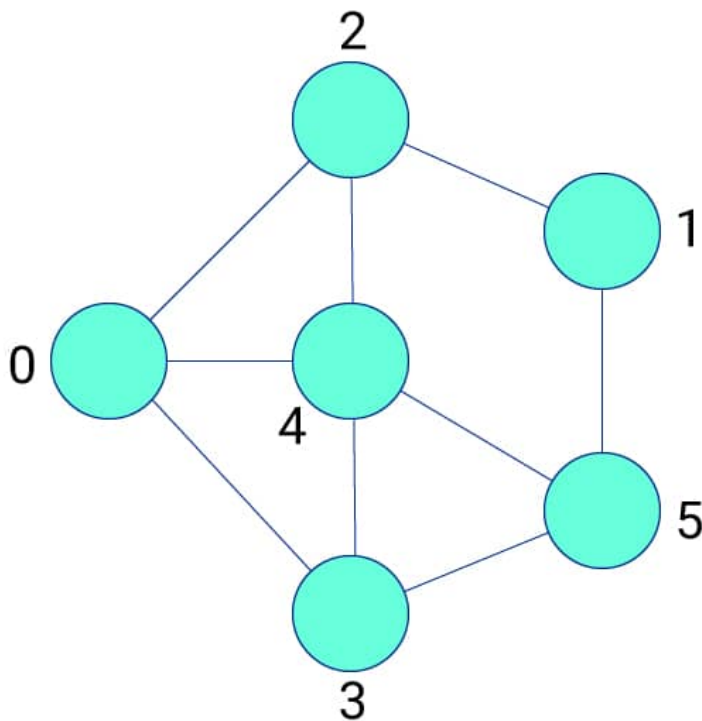


 Figura 3.9 – Árvore gerada após execução do algoritmo de busca em largura.

Fonte: EnsineMe.

Exemplo 2 – Busca em largura

Agora, vamos utilizar o algoritmo de busca em largura para percorrer o grafo da Figura 3.10.



📷 Figura 3.10 – Grafo para busca em largura – Exemplo 2.

Fonte: EnsineMe.

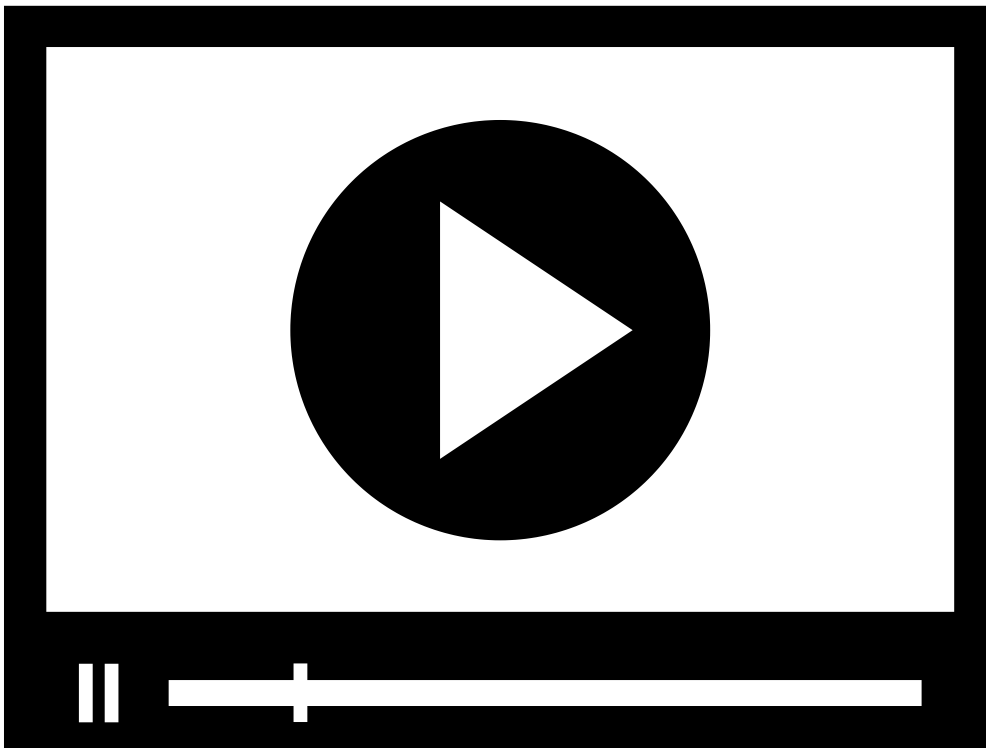
A fila F se comporta da seguinte forma, de acordo com a execução do algoritmo:

- 0 – Visita 0 e coloca 0 no final da Fila F
- 2 3 4 – Visita 2 e coloca 2 no final da Fila F
- 3 4 – Visita 3 e coloca 3 no final da Fila F
- 4 5 – Visita 4 e coloca 4 no final da Fila F
- 5 – Visita 5 e coloca 5 no final da Fila F
- 1 – Visita 1 e coloca 1 no final da Fila F

ANÁLISE DA COMPLEXIDADE DO ALGORITMO

É fácil perceber que o tempo de execução da busca em largura é o mesmo que o da busca em profundidade: $O(n+a)$, ou $O(\max(n,a))$. Mas, se consideramos a memória, o desempenho é pior.

Segundo Cormen *et al.* (2002), na busca em profundidade, somente um "ramo" da árvore é empilhado em qualquer momento, então a pilha não conterá mais de n elementos. Na busca em largura, porém, como todos os filhos de um nó são enfileirados a cada etapa, o espaço ocupado em memória tende a ser exponencial.



REALIZANDO BUSCAS EM GRAFOS



VERIFICANDO O APRENDIZADO

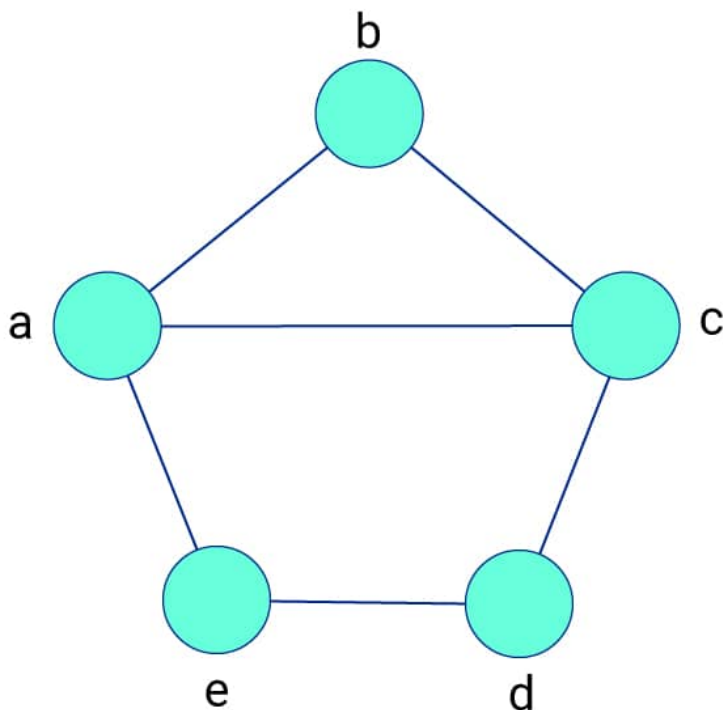
MÓDULO 4


CAMINHO

Um caminho em um grafo $G = (V, E)$ é uma sequência de vértices $V = (v_1, \dots, v_n)$, tal que (v_i, v_{i+1}) é um elemento de E , para $1 \leq i < n$. Essa sequência é dita um caminho de v_1 a v_n .

Segundo Feofiloff, Kohayakawa e Wakabayashi (2011), um caminho que não passa duas vezes pelo mesmo vértice é um caminho simples (ou elementar), enquanto um que não contém duas vezes a mesma aresta é um trajeto. Essa definição, porém, só funciona realmente com grafo simples.

No grafo ilustrado na Figura 4.1, a sequência $[a, b, c, d, e]$ é um caminho simples e também um trajeto. A sequência $[a, b, c, d, e, a, c]$ é um trajeto, mas não um caminho simples.



 Figura 4.1 – Grafo exemplo.

Fonte: EnsineMe.

Um caminho onde o primeiro vértice da sequência é igual ao último é um circuito, se todas as arestas são distintas. Se, além disso, não há vértice repetido no circuito, dizemos que ele é um ciclo.

Na Figura 4.1, a sequência $[a, b, c, d, e, a]$ é um ciclo: Nenhum vértice se repete, com a exceção do primeiro e do último.

A distância – outro conceito importante – entre dois vértices u e v ($d(u, v)$) em um grafo G é o comprimento do menor caminho entre u e v em G . A distância entre **a** e **d** na Figura 4.1 é 2.

CAMINHO MÍNIMO

O problema do caminho mínimo – ou mais curto – consiste em encontrar o melhor caminho entre dois nós. Isso pode significar o custo mínimo ou o menor tempo de viagem.

Numa rede qualquer, dependendo das suas características, podem existir vários caminhos entre um par de nós, definidos como origem e destino. Entre os vários existentes, aquele que possui o menor “peso” é chamado de **caminho mínimo**.

Esse peso representa a soma total dos valores dos arcos que compõem o caminho, cujos valores podem ser:

O tempo de viagem.

A distância percorrida.

Um custo qualquer do arco.

A seguir, as entradas e saídas do algoritmo:

Entrada: Grafo ponderado $G = (N, E)$ e nó origem $O \in N$, de modo que todos os custos das arestas sejam não negativos.

Saída: Comprimentos de caminhos mais curtos (ou os caminhos mais curtos em si) de um determinado nó origem $O \in N$ para todos os outros nós.

COMENTÁRIO

O **algoritmo de Dijkstra** é um dos algoritmos de busca de caminhos mais conhecidos. Dijkstra identifica, a partir do nó **O**, qual é o custo mínimo entre esse nó e todos os outros do grafo.

No início, o conjunto **S** contém somente esse nó, chamado origem.

A cada passo, selecionamos no conjunto de nós sobrando o que está mais perto da origem.

Depois atualizamos, para cada nó que está sobrando, a sua distância em relação à origem. Se passando pelo novo nó acrescentado a distância ficar menor, é essa nova distância que será

memorizada.

Escolhido um nó como origem da busca, esse algoritmo calcula, então, o custo mínimo deste para todos os demais do grafo.

O procedimento é iterativo, determinando, na iteração 1, o nó mais próximo do O; na segunda iteração, o segundo nó mais próximo do O, e assim sucessivamente, até que em alguma iteração todos os n nós sejam atingidos.

A seguir, os passos do algoritmo Dijkstra. Seja $G = (N, E)$ um grafo orientado e s um nó de G :

Atribua valor zero à estimativa do custo mínimo do nó O (a origem da busca) e infinito às demais estimativas.

Atribua um valor qualquer aos precedentes (o precedente de um nó t é o nó que precede t no caminho de custo mínimo de s para t).

Enquanto houver nó aberto:

- Seja k um nó ainda aberto cuja estimativa seja a menor dentre todos os nós abertos;.
- Feche o nó k . - Para todo nó j ainda aberto que seja sucessor de k faça:

Some a estimativa do nó k com o custo do arco que une k a j .

Caso essa soma seja melhor que a estimativa anterior para o nó j , substitua-a e anote k como precedente de j .

Entrada: Grafo conectado com pesos nos arcos (matriz w), nós a e z .

Saída: $L(z)$ - comprimento do menor caminho entre a e z .

$L(a) \leftarrow 0$

Para todo nó $x \in a$ Faça

$L(x) \leftarrow \infty$

Fim Para

$T \leftarrow$ conjunto de todos os nós cuja menor distância até a ainda não foi calculada.

Enquanto $z \in T$ Faça

Escolha $v \in T$ com menor $L(v)$

$T = T - \{v\}$

Para $x \in T$ vizinho a v Faça

$L(x) \leftarrow \min \{L(x), L(v) + w(v, x)\}$

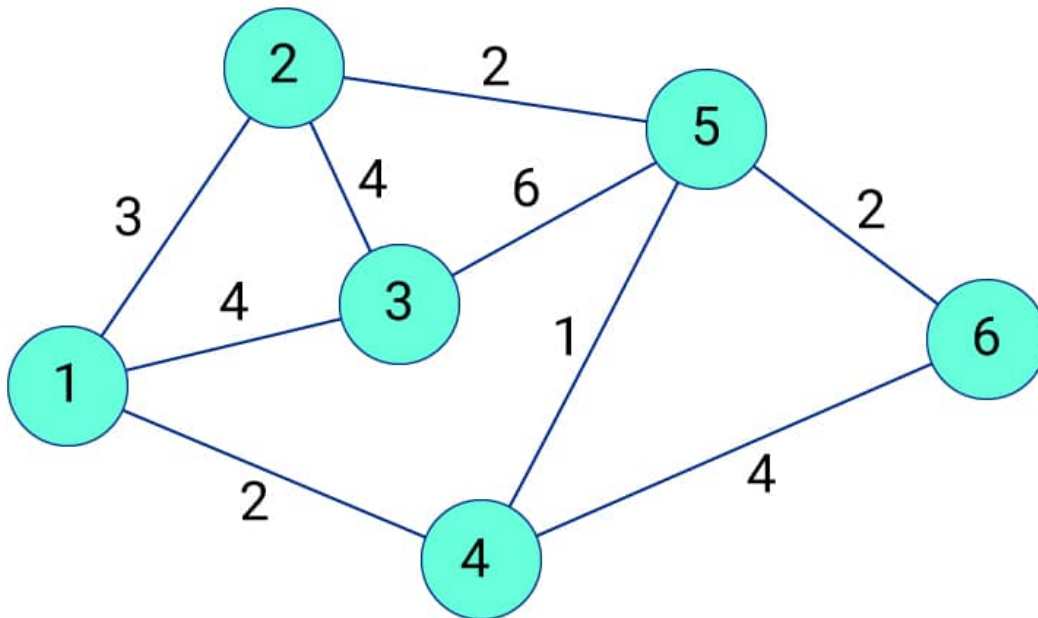
Fim Para

Fim Enquanto

📷 Figura 4.2 – Algoritmo Dijkstra.

Fonte: EnsineMe.

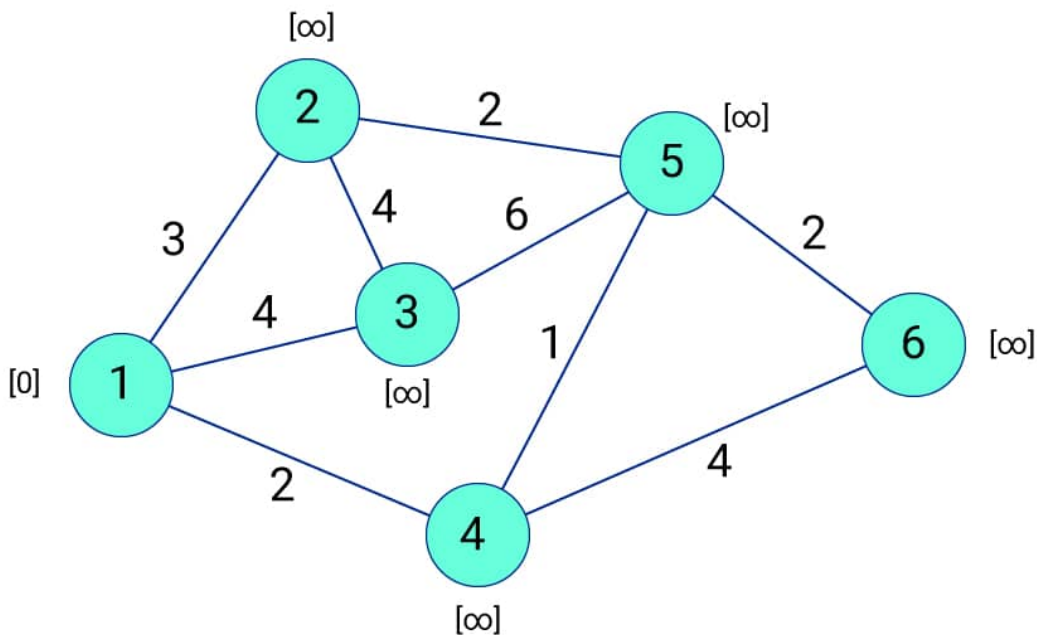
Utilizando o algoritmo de Dijkstra, vamos calcular o menor caminho entre os vértices 1 e 6 do grafo apresentado na Figura 4.3, que apresenta os vértices, as arestas e seus respectivos pesos.



📷 Figura 4.3 – Grafo exemplo.

Fonte: EnsineMe.

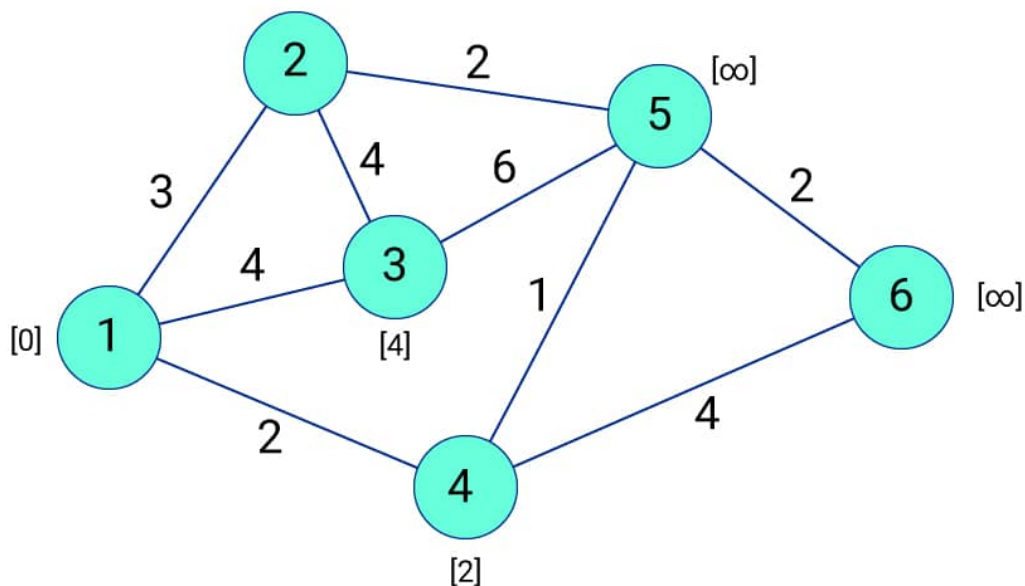
Inicialmente, vamos atribuir valor zero à estimativa do custo mínimo do nó 1 (a origem da busca) e infinito às demais estimativas, conforme Figura 4.4.



📷 Figura 4.4 – Passo 1 do algoritmo do caminho mínimo.

Fonte: EnsineMe.

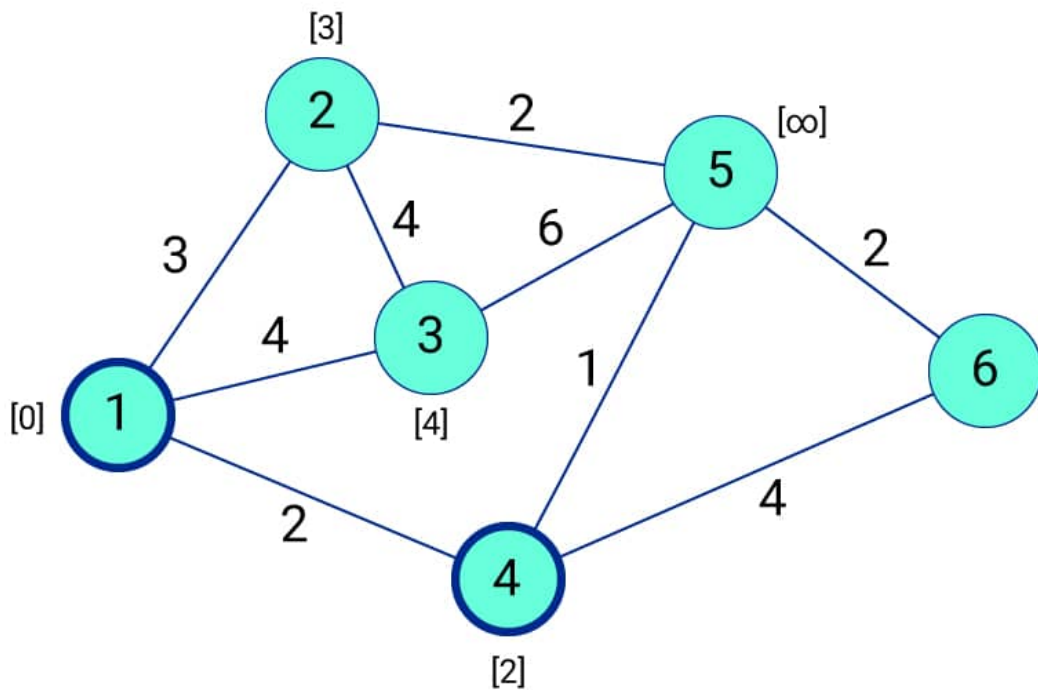
Em seguida, vamos avaliar o nó com menor custo no caminho entre 1 e 6. A partir de 1, iremos verificar o menor custo entre as arestas (1,2), (1,3) e (1,4), conforme a Figura 4.5.



📷 Figura 4.5 – Passo 2 do algoritmo do caminho mínimo.

Fonte: EnsineMe.

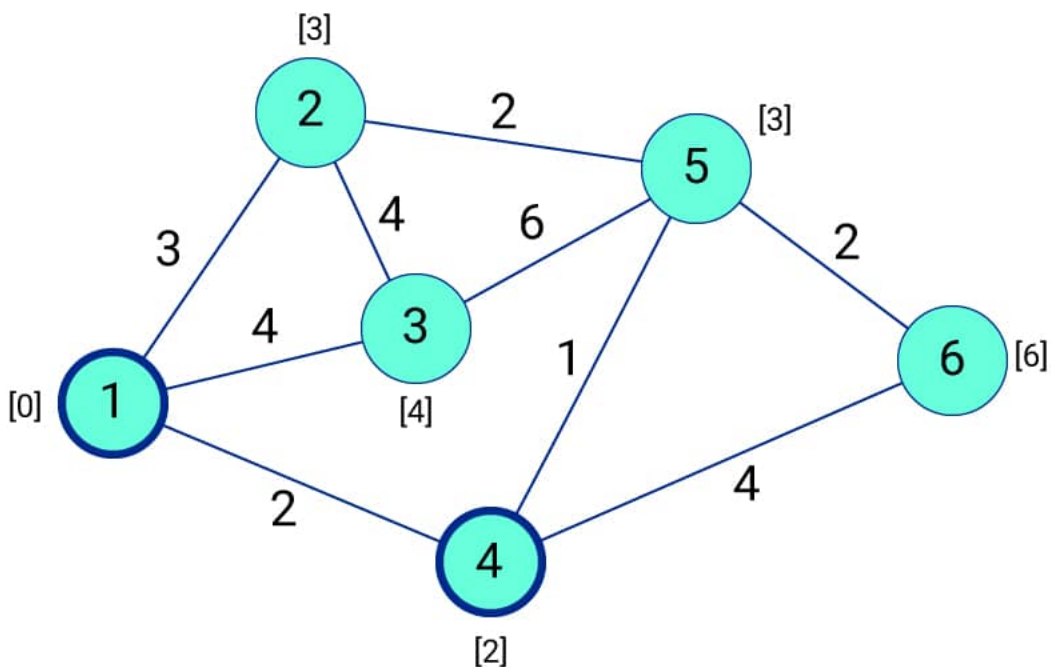
A aresta com o custo mínimo entre as arestas é (1,4). Portanto, o caminho com custo mínimo é inicialmente (1,4), conforme a Figura 4.6.



📷 Figura 4.6 – Passo 3 do algoritmo do caminho mínimo.

Fonte: EnsineMe.

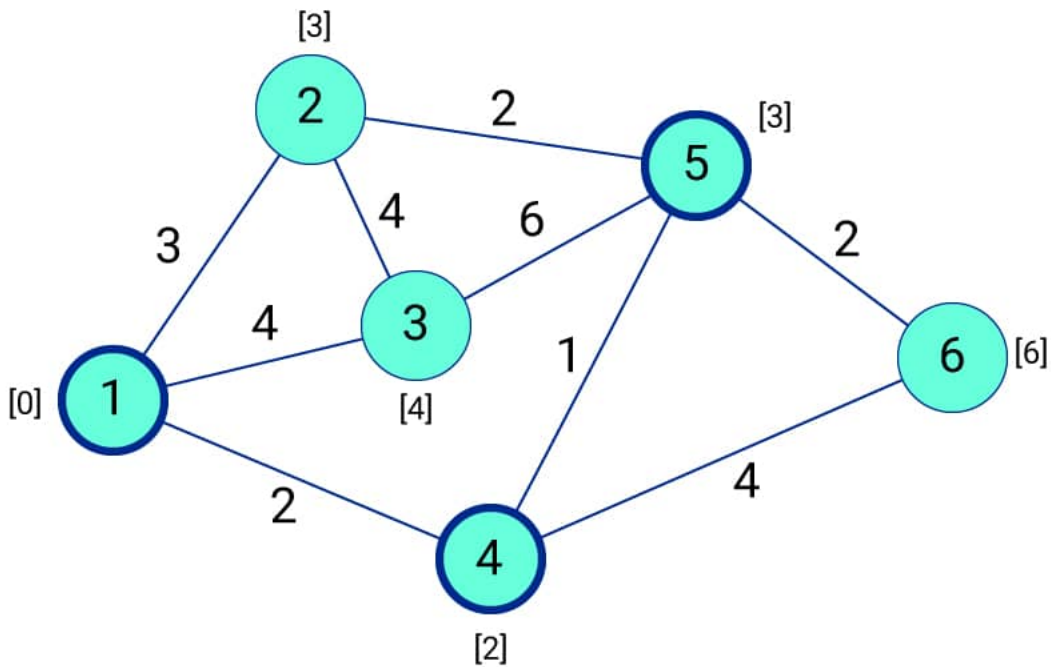
Realizamos, então, uma avaliação em todos os vizinhos do nó 4 para chegar ao valor mínimo. O menor custo é obtido entre as arestas (4,5), conforme a Figura 4.7. Vale ressaltar que já foi avaliada a aresta (4,6), que é o objetivo final. Porém, pelo algoritmo de Dijkstra, não é o valor mínimo a ser obtido:



📷 Figura 4.7 – Passo 4 do algoritmo do caminho mínimo.

Fonte: EnsineMe.

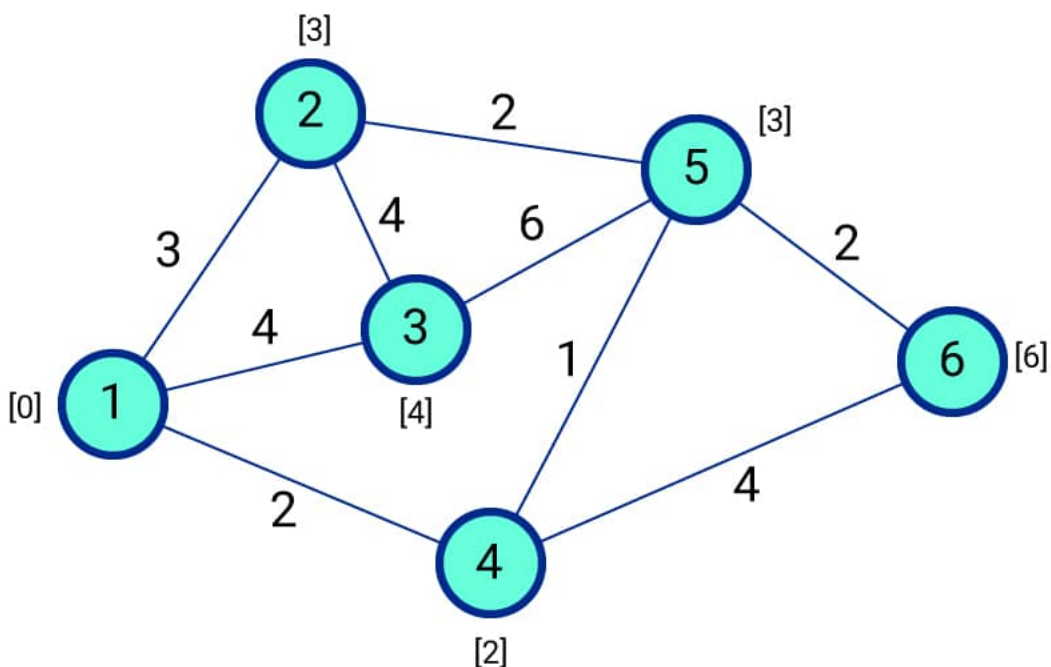
Uma avaliação em todos os vizinhos do nó 5 deve ser feita para chegar ao valor mínimo. Pela avaliação, o menor custo é obtido entre as arestas (5,6), conforme a Figura 4.8:



📷 Figura 4.8 – Passo 5 do algoritmo do caminho mínimo.

Fonte: EnsineMe.

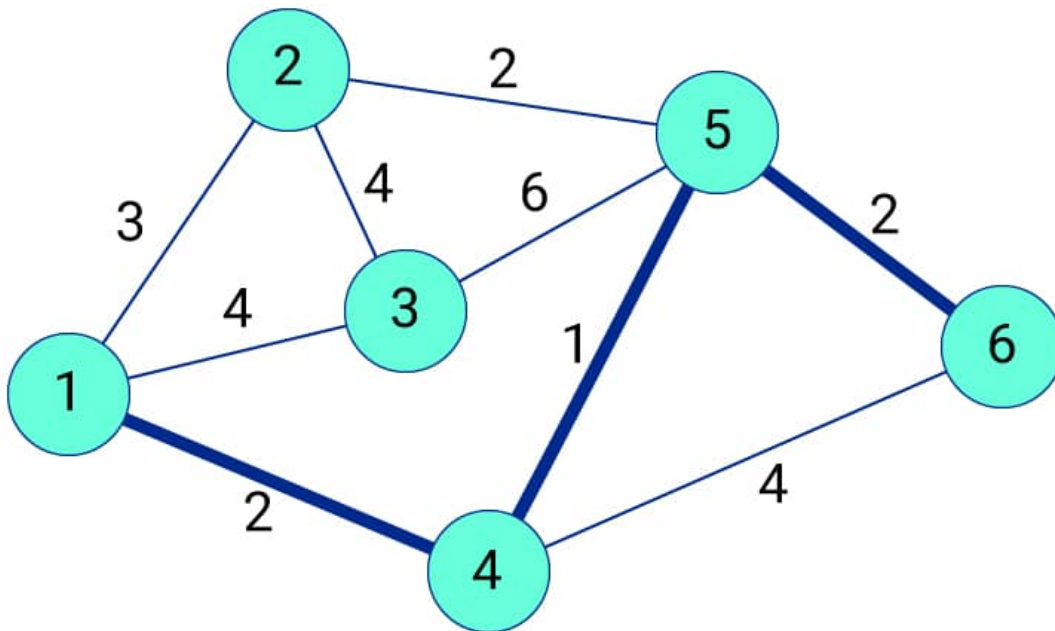
Chegamos ao caminho final com custo mínimo 5, após análise de todos os nós, conforme a Figura 4.9 que apresenta o custo mínimo para cada um dos nós do vértice:



📷 Figura 4.9 – Passo 6 do algoritmo do caminho mínimo.

Fonte: EnsineMe.

O caminho final pode ser analisado na Figura 4.10, a seguir:



📷 Figura 4.10 – Caminho final percorrido.

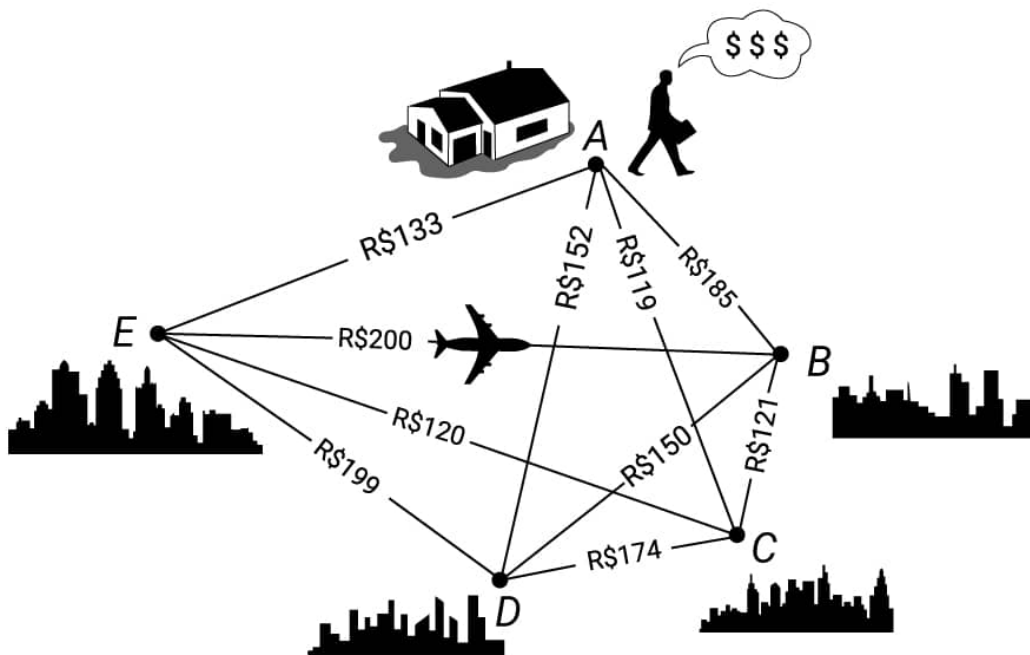
Fonte: EnsineMe.

PROBLEMA DO CAIXEIRO-VIAJANTE

Esse problema consiste na determinação da viagem do custo mínimo, partindo e regressando ao mesmo local.

Devido às inúmeras variáveis para sua resolução, este é considerado um **problema NP-Completo**, ou seja, a solução é obtida apenas através de algoritmos exponenciais ou fatoriais.

Para se encontrar o custo mínimo, deve-se considerar os pesos de cada aresta, que representam a distância entre as cidades, conforme grafo ilustrado na Figura 4.11.



📷 Figura 4.11 – Caminho final percorrido

Fonte: EnsineMe.

A resolução desse problema é importante para aplicações, como entrega de encomendas, recolha de objetos, planificação de viagens, leitura de contadores, entre outros. Alguns algoritmos mais simples são: da força bruta; e do vizinho mais próximo.

ALGORITMO DA FORÇA BRUTA

O algoritmo da força bruta tem as seguintes características:

1

Lista de todos os circuitos possíveis.



2

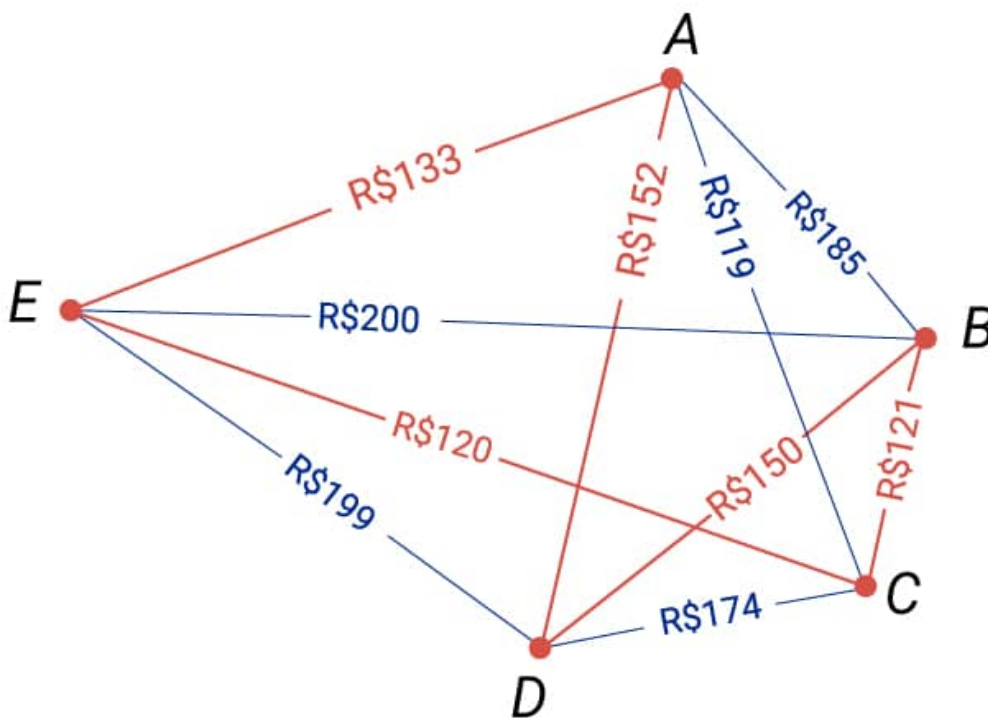
Cálculo do custo total de cada circuito.



3

Escolha de um circuito com o custo total mínimo.

O resultado obtido para a Figura 4.11 utilizando o algoritmo da força bruta é apresentado na Figura 4.12, a seguir, com custo total de R\$676, percorrendo os nós A, E, C, B, D e A.



📷 Figura 4.12 – Resultado da aplicação do algoritmo da força bruta.

Fonte: EnsineMe.

ALGORITMO DO VIZINHO MAIS PRÓXIMO

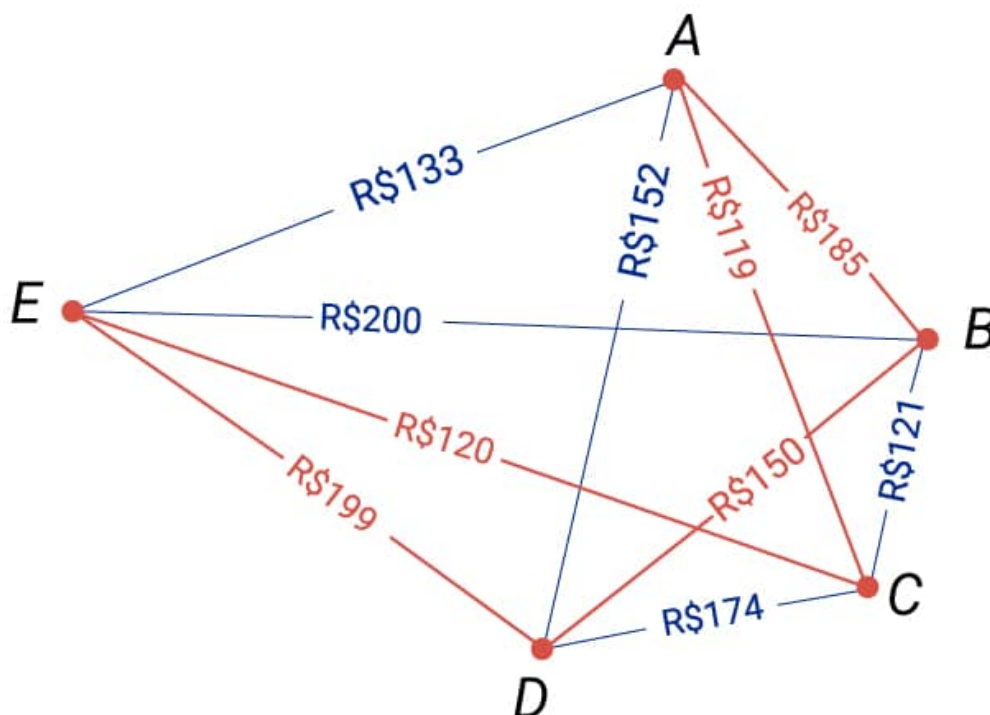
O algoritmo, de forma simplificada, pode ser executado da seguinte forma:

Selecionamos um ponto de partida (casa).

Das arestas adjacentes, escolhemos a que tem menor peso, e partimos para o vértice correspondente (vizinho mais próximo).

Continuamos a construir o circuito, indo sempre para o vizinho mais próximo, a não ser que a aresta que o liga feche um circuito havendo ainda vértices por visitar. Prosseguimos até que todos os vértices sejam visitados.

Chegando ao último vértice, regressamos a “casa”.



📷 Figura 4.13 – Resultado da aplicação do algoritmo do vizinho mais próximo.

Fonte: EnsineMe.

Utilizando o grafo definido na Figura 4.11, o algoritmo do vizinho mais próximo obteve o caminho A, C, E, D, B, A, com custo total de **R\$773**.

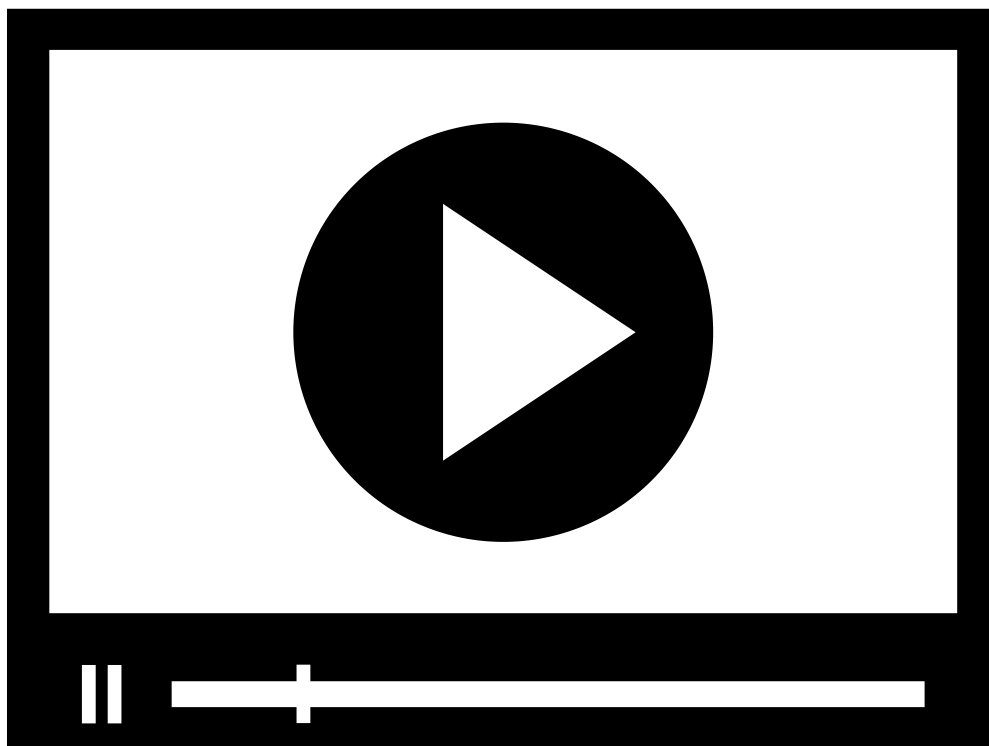
COMPARAÇÃO ENTRE OS ALGORITMOS

Realizando uma comparação entre os resultados obtidos pelos algoritmos, chegamos às seguintes conclusões:

Os resultados são diferentes para cada algoritmo.

O algoritmo da força bruta é conhecido como um algoritmo ineficiente, pois possui complexidade fatorial. Portanto, fica restrito a pequenos problemas. Para casos, por exemplo, com 10 cidades, teriam que ser avaliadas $10!$ possibilidades, o equivalente a 3.628.800 caminhos.

O algoritmo do vizinho mais próximo é um exemplo de algoritmo eficiente, pois se resolve $O(n)$ arestas. Contudo, o resultado pode estar longe de ser um circuito ótimo.



CALCULANDO CAMINHOS



VERIFICANDO O APRENDIZADO

CONCLUSÃO

CONSIDERAÇÕES FINAIS

O grafo ainda se apresenta com uma estrutura matemática bastante utilizada para resolução de problemas antigos, como o do custo mínimo ou o do menor caminho entre dois pontos em uma cidade ou entre um conjunto de cidades. São problemas NP-completos, largamente estudados devido à sua importância para resolução de dilemas práticos da indústria.

Para a compreensão dos grafos, este tema apresentou seus conceitos básicos e suas propriedades, além das formas de representação para que possa ser utilizado computacionalmente, utilizando matrizes de adjacências ou de incidência.

Vimos ainda os problemas clássicos relacionados a encontrar um caminho mais curto entre dois pontos, passando pelos algoritmos de Dijkstra e pelo problema do caixeiro-viajante.

Por fim, é importante enfatizar que as redes sociais ajudaram a alavancar mais ainda a utilização de grafos, pois toda a arquitetura das mesmas é baseada em grafos. Essa necessidade, bem como a facilidade da utilização da estrutura alavancaram a sua utilização.



PODCAST



PODCAST

REFERÊNCIAS

ARAÚJO, A. **As pontes de Königsberg**. Coimbra: Universidade de Coimbra, 2020.

CARVALHO, M. A. G. de. **Teoria dos grafos**: Uma introdução. Limeira: Ceset/Unicamp, 2005.

CORMEN, T. H. *et al.* **Algoritmos**: Teoria e prática. [S. l.]: Campus, 2002.

FEOFIOFF, P. **Busca em profundidade**. São Paulo: IME/USP, 2019.

FEOFIOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. **Uma introdução sucinta à teoria dos grafos**. São Paulo: IME/USP, 2011.

JURKIEWICZ, S. **Grafos**: Uma introdução. Rio de Janeiro: OBMEP, 2009.

WILHELM, V. E. **Problema do caminho mínimo**. Curitiba: UFPR, 2020.

EXPLORE+

Para saber mais sobre os assuntos tratados neste tema, leia:

Graph-based social media analysis, de Ioannis Pitas.

Graph databases in action, de Josh Perryman e Dave Bechberger.

Graph databases: New opportunities for connected data, de Emil Elfrem, Ian Robinson e Jim Webber.

CONTEUDISTA

Marcelo Nascimento Costa

 **CURRÍCULO LATTES**