EAD

EDUCAÇÃO A DISTÂNCIA

UNIDADE I

Teoria dos Grafos

Profa. Dra. Miryam de Moraes

- Objetivo;
- Definições;
- Aplicações de grafos;
- Terminologia da Teoria dos Grafos;
- Grafos isomorfos;
- Grafos planares;
- Representação de grafos no computador;
- Caminho de Euler e hamiltoniano;
- Árvores;
- Algoritmos de percurso: em nível, largura e profundidade;
- Ordenação topológica;
- Componentes fortemente conexas.

Definições de um grafo

- Informalmente: um **grafo** é um conjunto não vazio de **nós** (vértices) e um conjunto de **arcos** (arestas) tais que cada arco conecta dois nós.
- O número de nós e de arestas é sempre finito.
- Grafos.

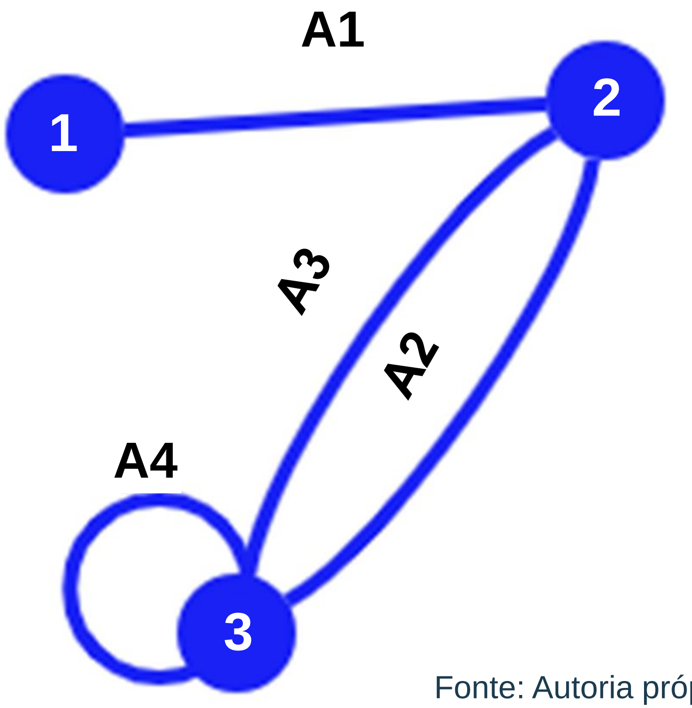
Fonte: https://l1nq.com/ofmG7

Fonte: https://l1nq.com/DuvPR

Definição (formal) de grafos

Um grafo é uma tripla ordenada (N, A, g), em que:

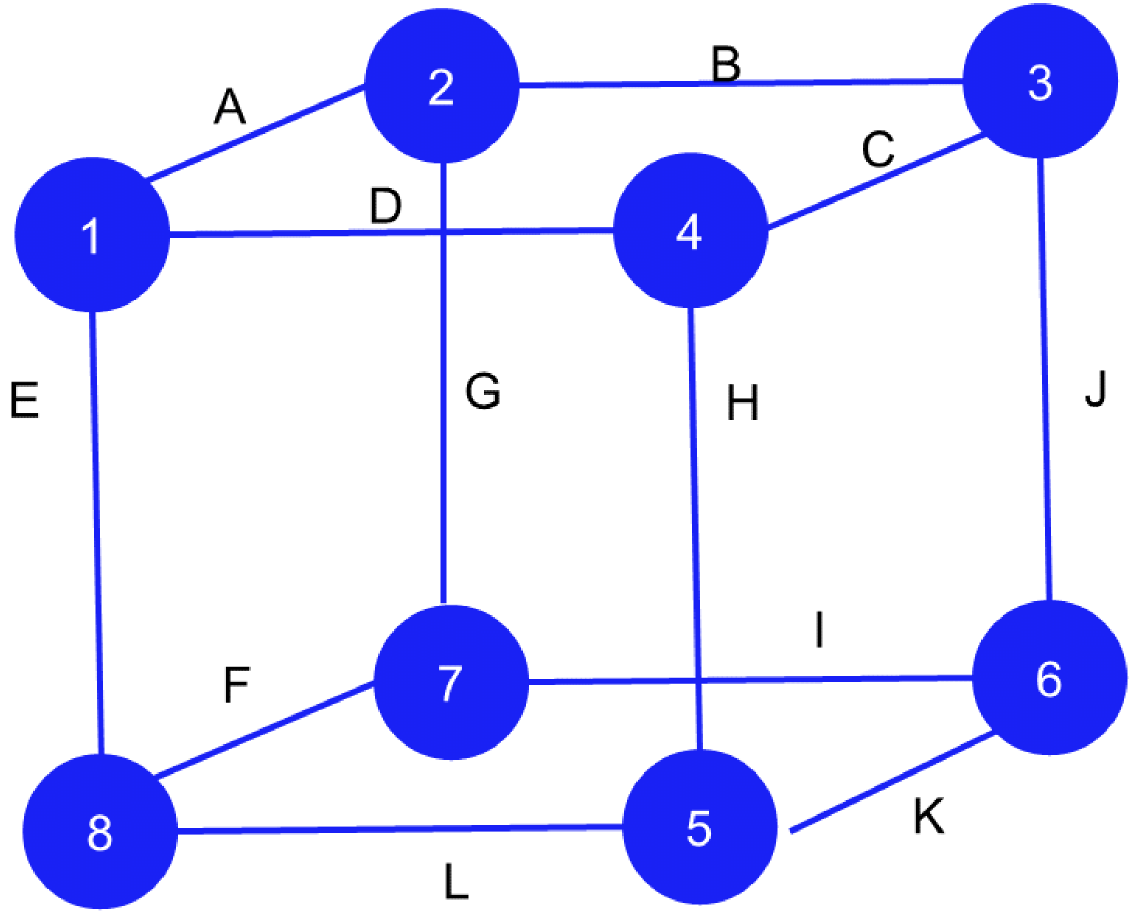
- N = um conjunto não vazio de nós (vértices).
- A = um conjunto de arcos (arestas).
- g = uma função que associa cada arco a um par não ordenado x-y de nós, chamadas as extremidades de:
- N = {1,2,3};
- A = {A1, A2, A3, A4};
- g(A1) = 1-2, g(A2) = (2-3), g(A3) = 2-3; g(A4) = 3-3.



Fonte: Autoria própria.

Definição (formal) de grafos

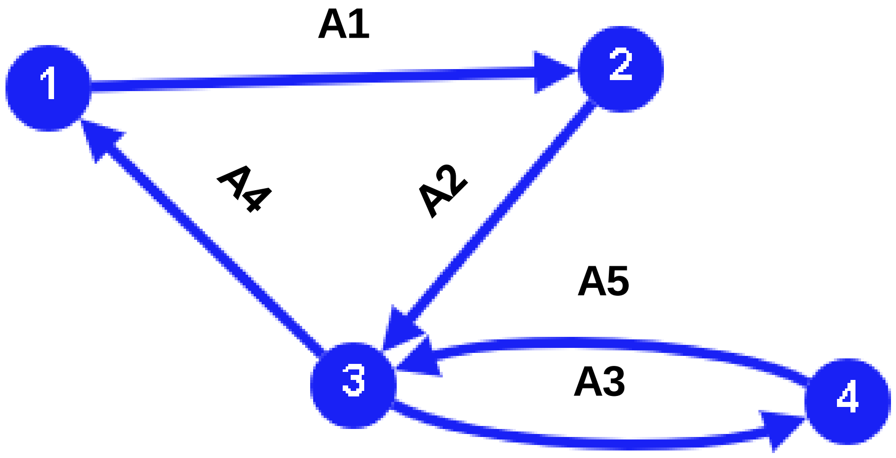
- Para o grafo ao lado, tem-se:
- N = {1,2,3,4,5,6,7,8};
- A = {A, B, C, D, E, F, G, H, I, J, K, L};
- g(A)= 1-2, g(B) = (2-3), g(C) = 3-4;
- G(D)= 1-4 e assim por diante...



Fonte: Autoria própria.

Definição – Grafo direcionado

- Um **grafo direcionado (dígrafo)** é uma tripla ordenada (N, A, g), em que:
- N = um conjunto não vazio de nós;
- A = um conjunto de arcos;
- g = uma função associa a cada arco um par ordenado (x,y) de nós, em que x é o **ponto inicial** e y é **o ponto final** de a.
- Em um arco direcionado, cada arco tem um sentido ou orientação.
- N = {1,2,3,4}; A = {A1, A2, A3, A4, A5};
- g(A1) = (1,2), g(A2) = (2,3), g(A3) = (3,4); g(A4) = (3,1); g(A5) = (4,3).



Fonte: Autoria própria.

Aplicações de grafos – Redes neurais

- Redes neurais são um exemplo da aplicação de grafos.

Fonte: Gottlich; Totzeck (2021).

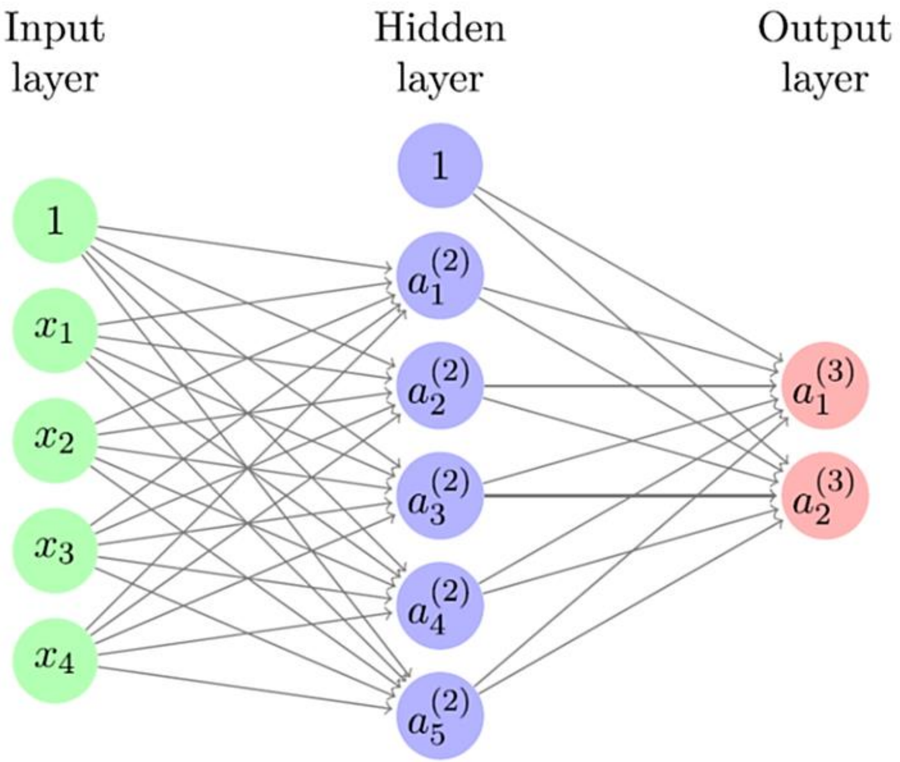


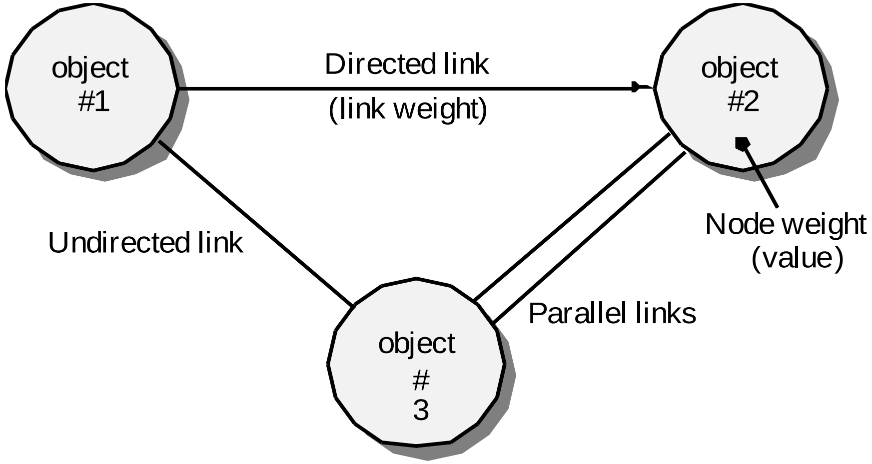
Fig. 1 Illustration of a feed-forward artificial network with 4 inputs and one bias input, one hidden layer with one bias unit and 5 neurons and two outputs. This corresponds to $n^{(1)} = 4$, $n^{(2)} = 5$, $n^{(3)} = 2$, $L = 3$

Springer

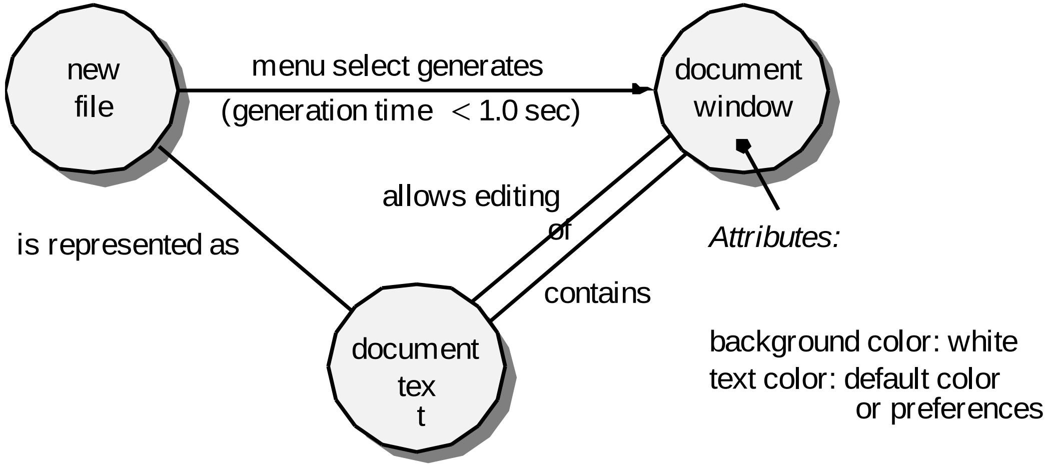
Aplicações de grafos

- Uma interessante e importante aplicação da Teoria dos Grafos diz respeito à testabilidade de *software* de aplicações convencionais.

Testabilidade de *software*:



(a)



(b)

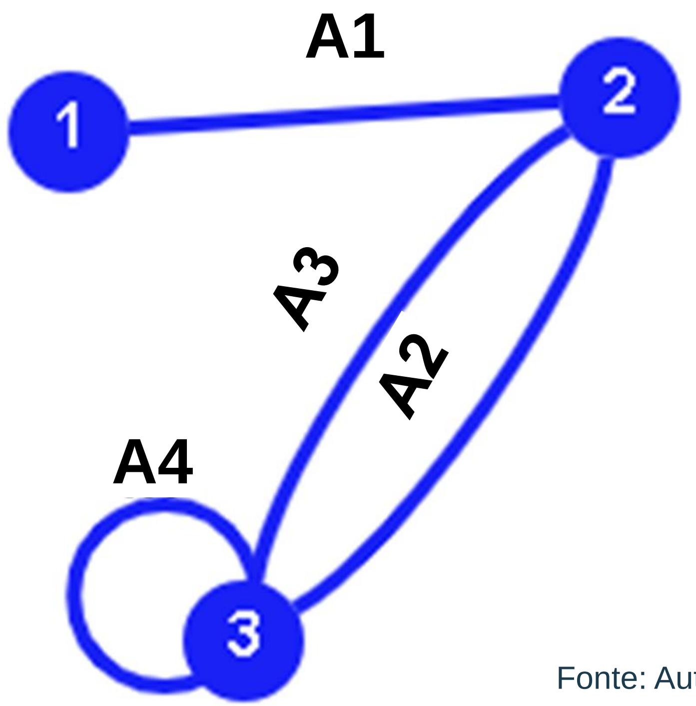
Fonte: Pressman; Maxim (2016).

Terminologia da Teoria dos Grafos

- Dois nós em um grafo são ditos **adjacentes** se ambos são extremidades de algum arco.
- Um **laço** em um arco é um arco com extremidades n-n para algum nó n.
- Dois arcos com as mesmas extremidades são ditos **arcos paralelos.**
- Um **grafo simples** é um arco que não tem laços nem arcos paralelos.
  - Um **nó isolado** é um nó que não é adjacente a nenhum outro.
  - O **grau de um nó** é o número de arcos que terminam naquele nó.

Terminologia da Teoria dos Grafos – Exemplo 1

- Na figura ao lado, o nó 2 é adjacente aos nós 1 e 3.
- A2 e A3 são arcos paralelos e A4 é um laço.
- O grafo não é simples.
- grau(1)=1; grau(2)=3; grau(3)=4.



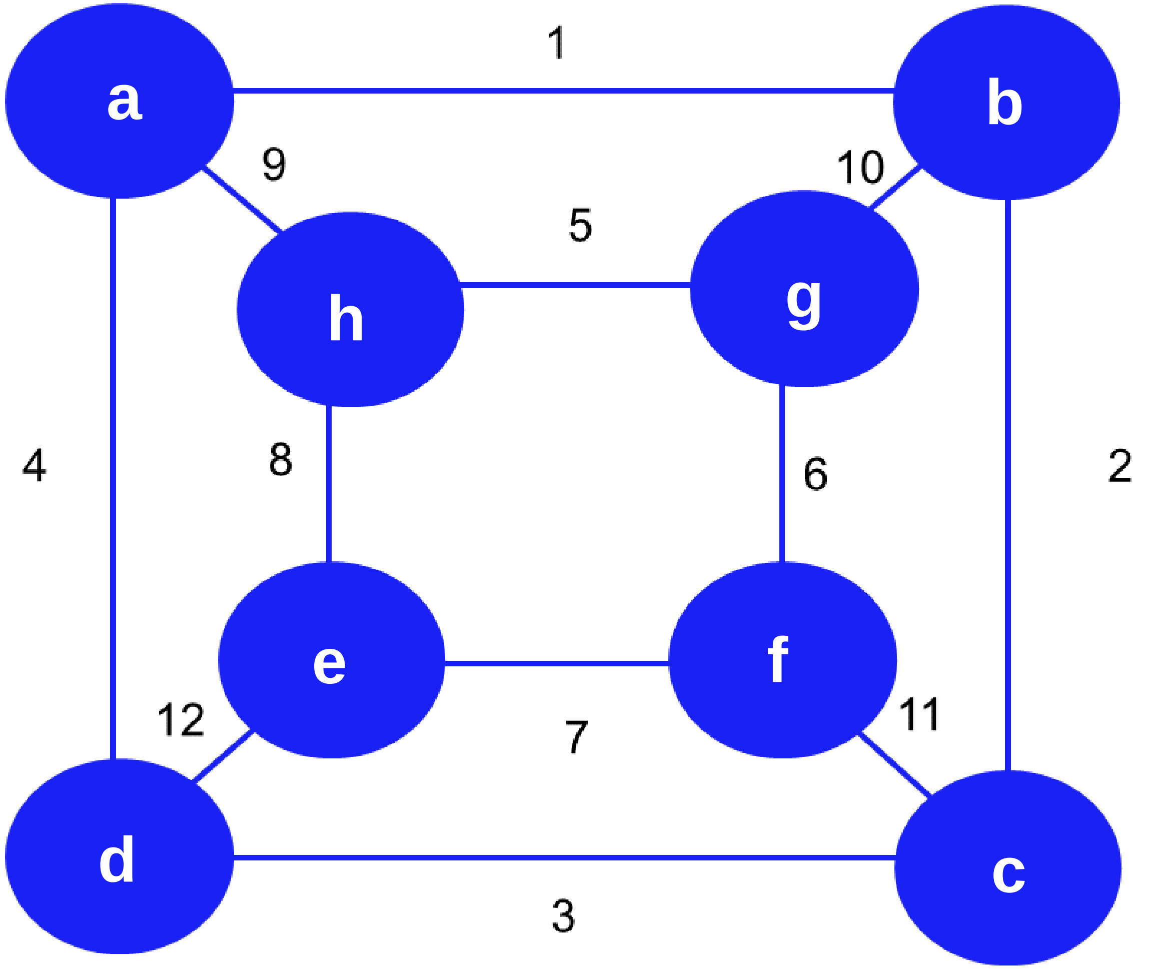
Fonte: Autoria própria.

Terminologia da Teoria dos Grafos – Exemplo 2

- Na figura ao lado, o nó **a** é adjacente aos nós **b**, **h** e **d**.

- O grafo é simples.

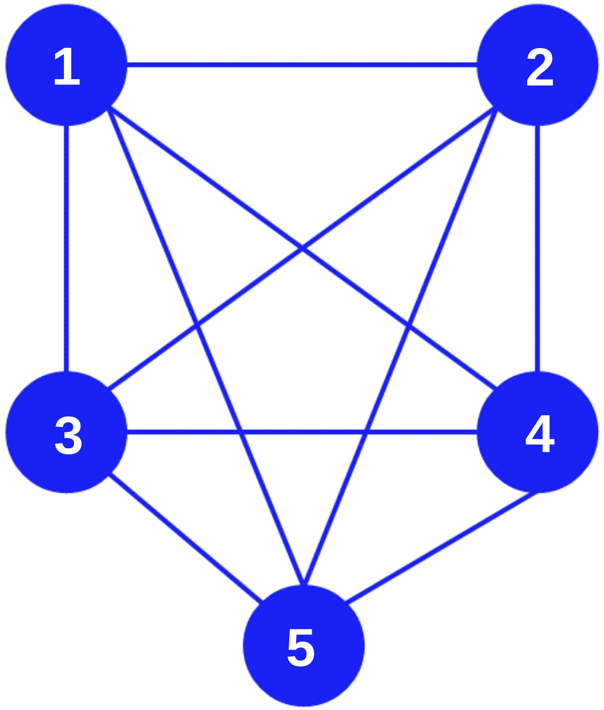
- O grau de todos os nós é 3.



Fonte: Autoria própria.

Terminologia da Teoria dos Grafos

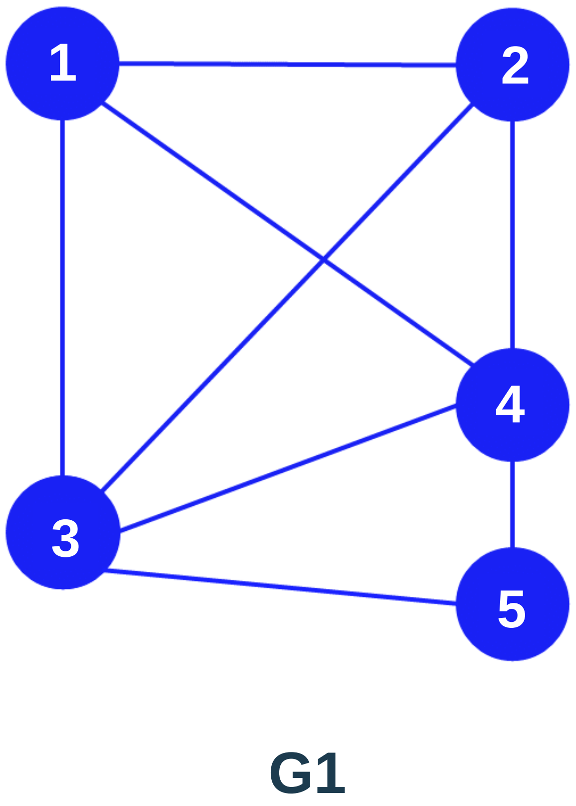
- Um **grafo completo** é um grafo no qual dois nós distintos quaisquer são adjacentes. Nesse caso, g é uma função sobrejetora (todo par x-y de nós distintos é a imagem, sob g, de algum arco), mas não há necessidade de se ter um laço em cada nó. Portanto, pares de forma x-x podem não ter uma imagem inversa.
- O grafo simples completo com n vértices é denotado por Kn. Exemplo: K1, K2, K5.

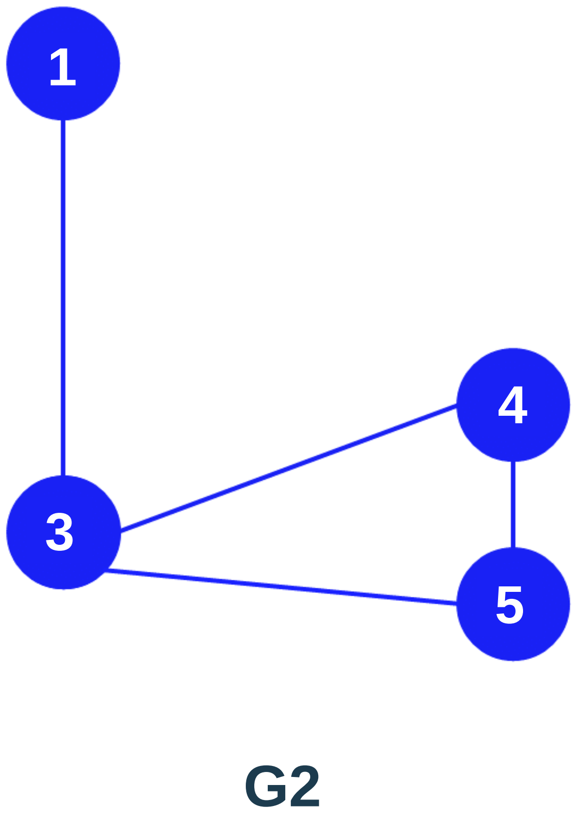


Fonte: Autoria própria.

Terminologia da Teoria dos Grafos

- Um **subgrafo** de um grafo consiste em um conjunto de nós e um conjunto de arcos que são subconjuntos do conjunto original de nós e arcos, respectivamente, nos quais as extremidades de um arco têm que ser as mesmas que no grafo original.
- G2 é subgrafo de G1.

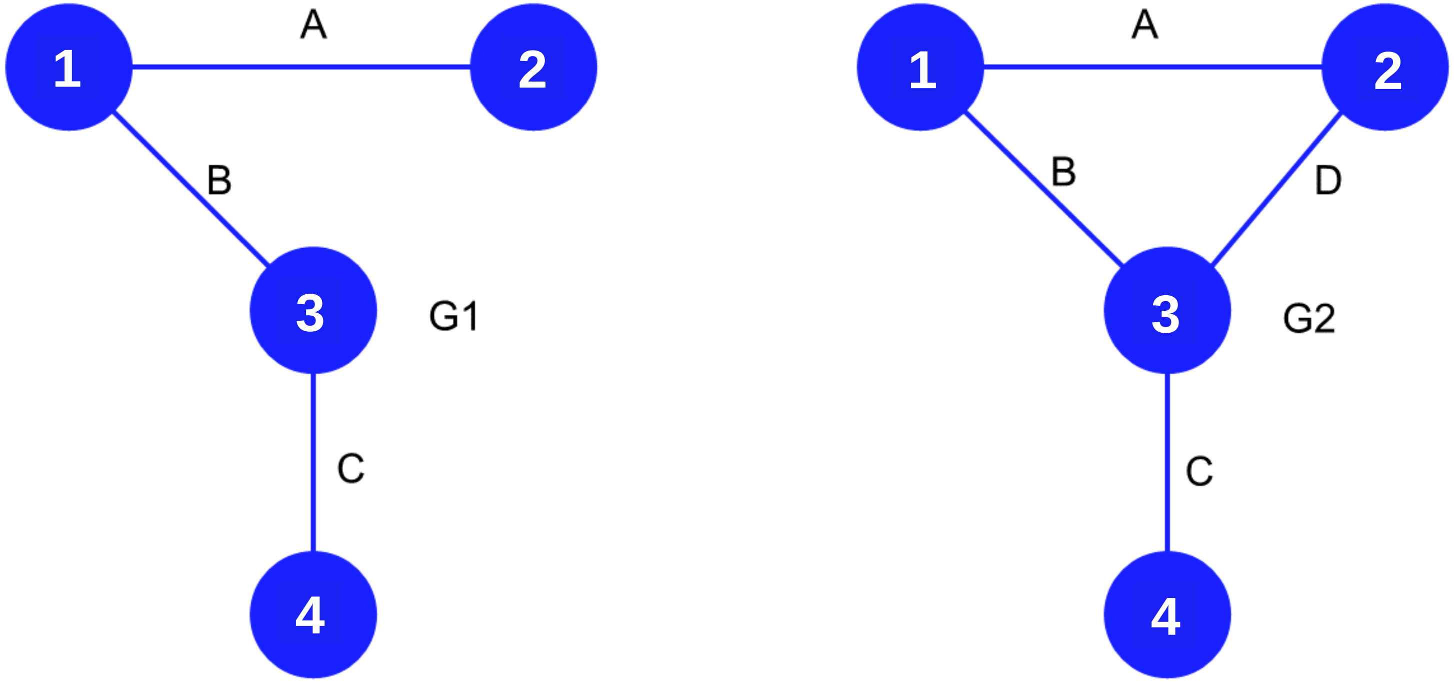




Fonte: Autoria própria.

Terminologia da Teoria dos Grafos

- Um **caminho** do nó n0 para o nó nK é uma sequência n0, a0, n1, a1, ..., nk-1, ak-1, nk.
- O **comprimento** de um caminho é o número de arcos que ele contém.
- Um grafo é **conexo** se existe um caminho de qualquer nó para qualquer outro.
- Um **ciclo** em um grafo é um caminho de algum nó n0 para ele mesmo, tal que nenhum arco aparece mais de uma vez, n0 é o único nó que aparece mais de uma vez e n0 aparece apenas nas extremidades.
- Um grafo sem ciclos é dito **acíclico**.
- Na figura, G1 é acíclico e G2 apresenta ciclo.



Fonte: Autoria própria.

Interatividade

Considere as seguintes asserções:

I. Em muitas áreas da computação, é conveniente modelar um algoritmo ou um programa usando um grafo.

II. Instalações de fornecimento de luz, água e esgoto podem ser representadas por um grafo.

III. Uma rede de radares instalada sobre uma determinada rede de ruas, avenidas pode ser representada por um grafo.

Está correto o que se afirma em:

a) I, apenas.
b) II, apenas.
c) III, apenas.
d) I e III, apenas.
e) I, II e III.

Resposta

Considere as seguintes asserções:

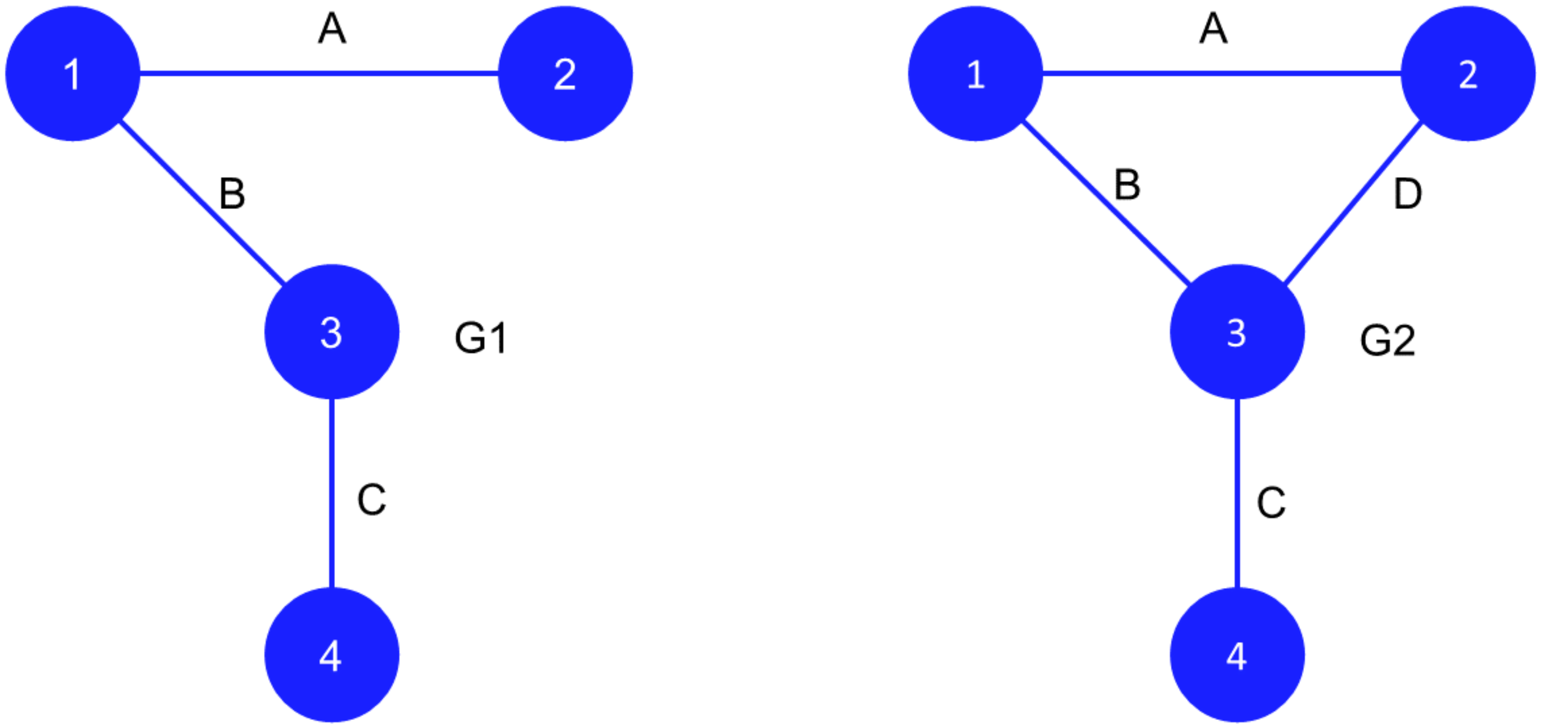
I. Em muitas áreas da computação, é conveniente modelar um algoritmo ou um programa usando um grafo.
II. Instalações de fornecimento de luz, água e esgoto podem ser representadas por um grafo.
III. Uma rede de radares instalada sobre uma determinada rede de ruas, avenidas pode ser representada por um grafo.

Está correto o que se afirma em:

a) I, apenas.
b) II, apenas.
c) III, apenas.
d) I e III, apenas.
e) I, II e III.

Terminologia da Teoria dos Grafos

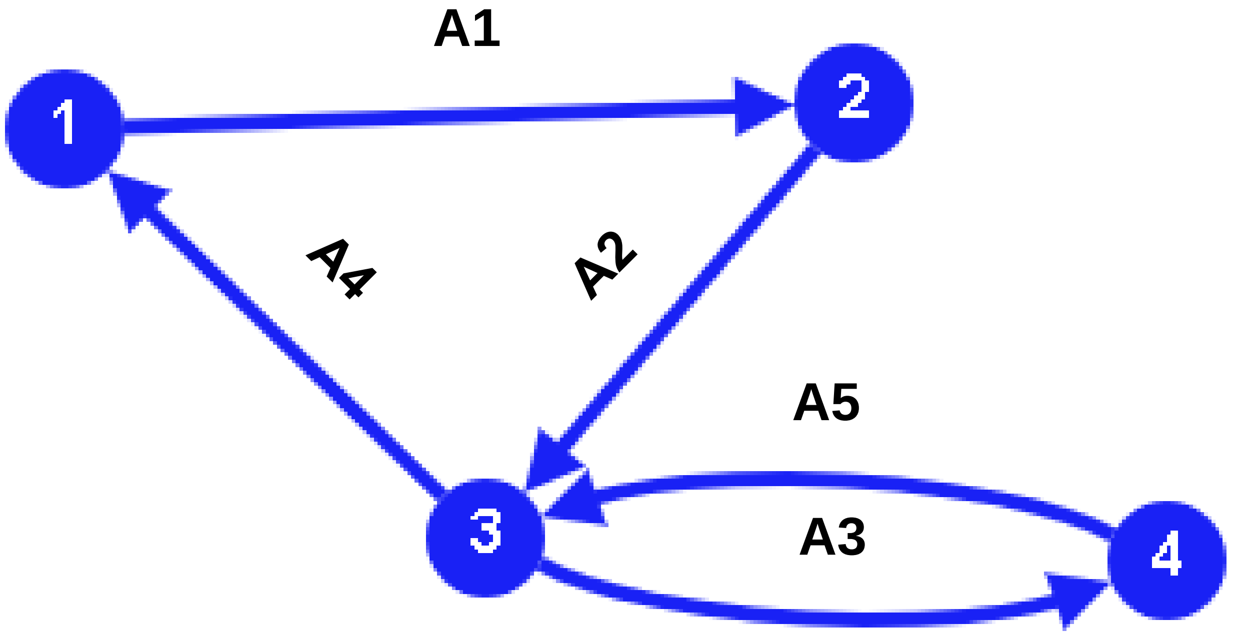
- G1 e G2 são conexos.
- Para ambos grafos, têm-se como exemplos os caminhos de comprimento 3, a saber:
  - 2A1B3C4 em G1 e G2;
  - 4C3D2A1 em G2.
- Ambos são grafos conexos:
  - Ciclo em G2: 3B1A2D3;
  - Figura: G1: acíclico;
  - G2: grafo com ciclo.



Fonte: Autoria própria.

Terminologia da Teoria dos Grafos

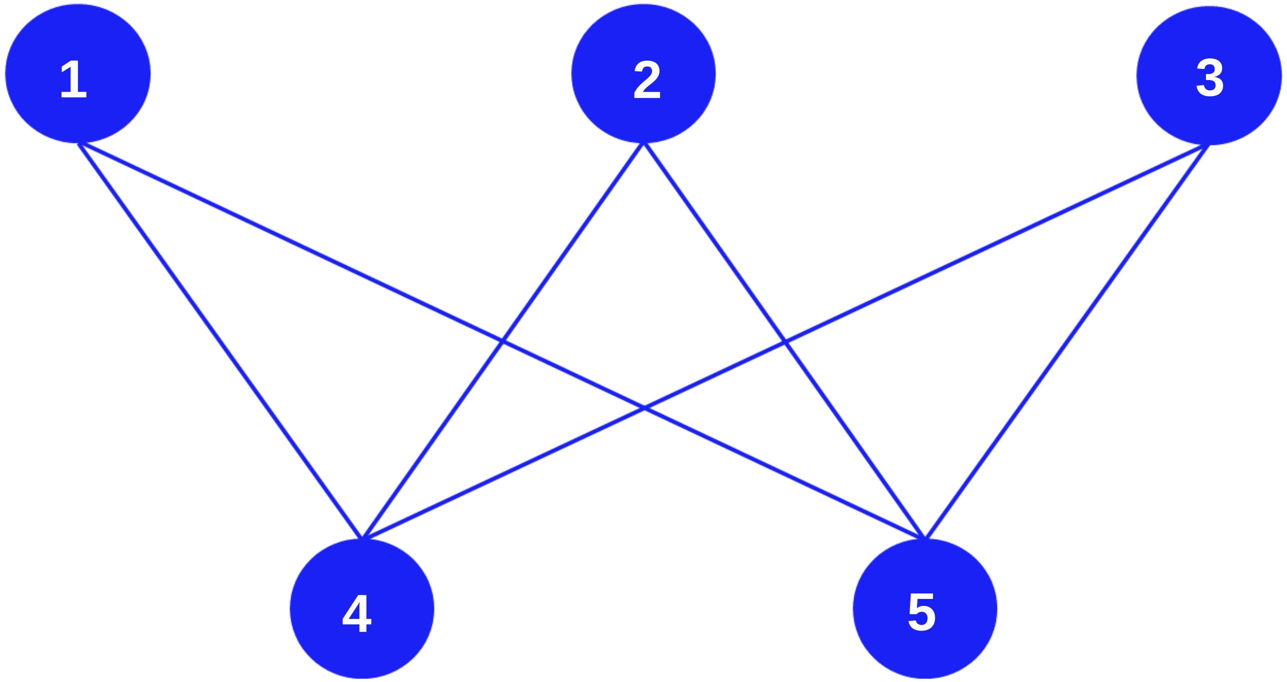
- Um caminho de um nó n0 para um nó nk em um grafo direcionado é uma sequência, em que para cada ni, ni é o ponto inicial e ni+1, o ponto final do arco ai. Se existe um caminho de n0 para nk é acessível de n0. A definição de ciclo também pode ser estendida para grafos direcionados.
- Dígrafo – exemplo.
- Exemplo de caminho: 1 A1 2.
- Exemplo de ciclo: 3 A3 4 A5 3.



Fonte: Autoria própria.

Terminologia da Teoria dos Grafos

- **Definição: Grafo Bipartido Completo.** Um grafo é um grafo bipartido completo se seus nós podem ser divididos em dois conjuntos disjuntos não vazios N1 e N2, tais que dois nós são adjacentes se, e somente se, um deles pertence a N1 e o outro pertence a N2.
- Se |N1| = m e |N2| = n, um tal grafo é denotado por Km,n.
- Na figura: N1= {1, 2, 3} e, portanto, |N1| = 3.
- N2 ={4,5} e, portanto, |N2| = 2.
- N1 e N2 são disjuntos.
- Figura: Grafo K3,2.

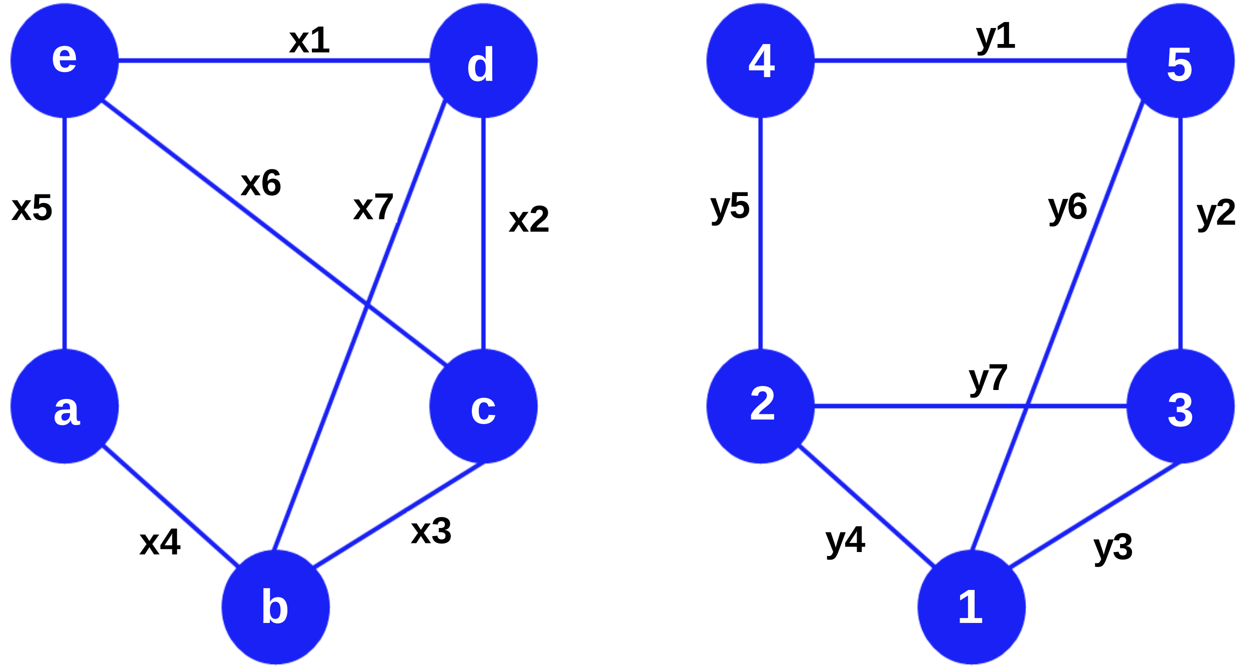


Fonte: Autoria própria.

Grafos isomorfos

- Dois grafos (N1, A1, g1) e (N2, A2, g2) são isomorfos se existem bijeções f1: N1→N2 e f2: A1→A2, tais que para cada arco a ∈ A1, g1(a) = x-y se e somente se g2[f2(a)]=f1(x) – f1(y).

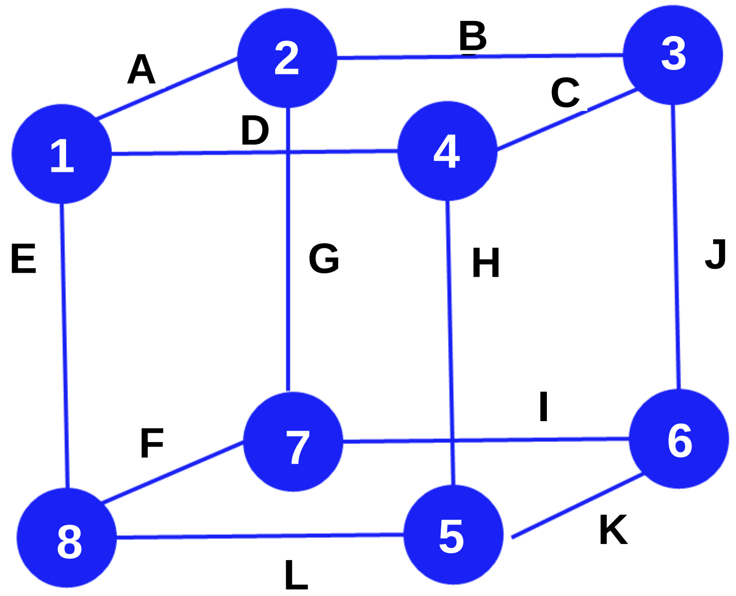
- f1: a → 4   f2: x4 → y1
  - b → 5   f2: x3 → y7
  - c → 3   f2: x2 → y3
  - d → 1   f2: x1 → y4   f2: x6 → y2
  - e → 2   f2: x5 → y5   f2: x7 → y6



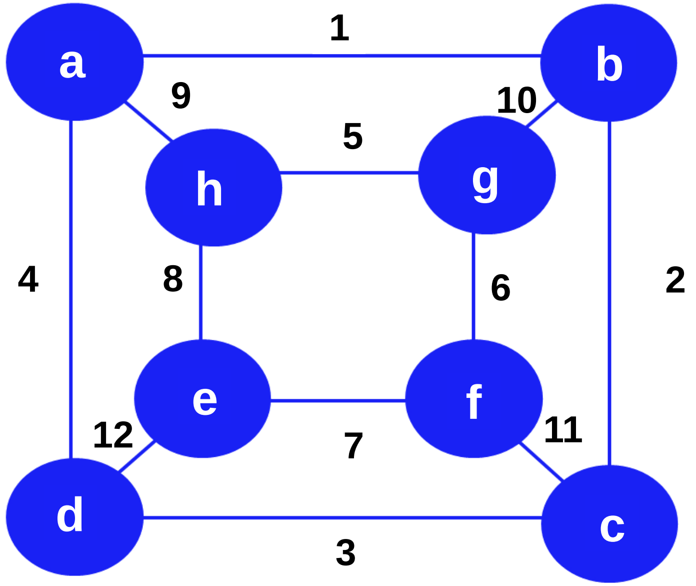
Fonte: Autoria própria.

Grafos isomorfos – Outro exemplo

- Os grafos ao lado são isomorfos.
- Bijeção entre os nós:
- f1: 1 → a; 2 → b; 3 → c; 4 → d; 5 → e;
  6 → f; g → 7; h → 8.
- Bijeção entre as arestas:
- f2: A → 1; B → 2; C → 3; D → 4; E → 9;
  F → 5; G → 10; H → 12; I → 6; J → 11;
  K → 7; L → 8.

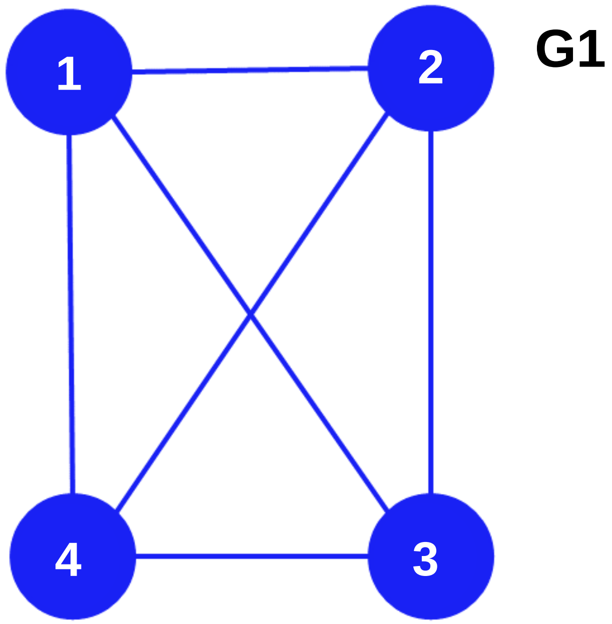


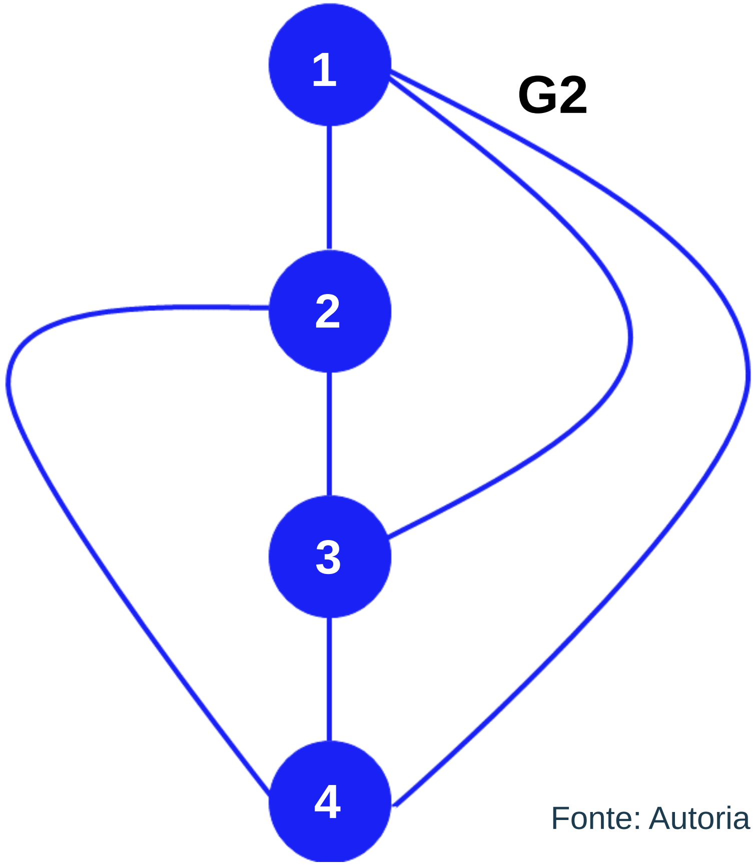
Fonte: Autoria própria.



Grafos planares

- Um **<u>grafo planar</u>** é um grafo que pode ser representado de modo que seus arcos se intersectam apenas em nós.
- G1 e G2 são isomorfos e planares.

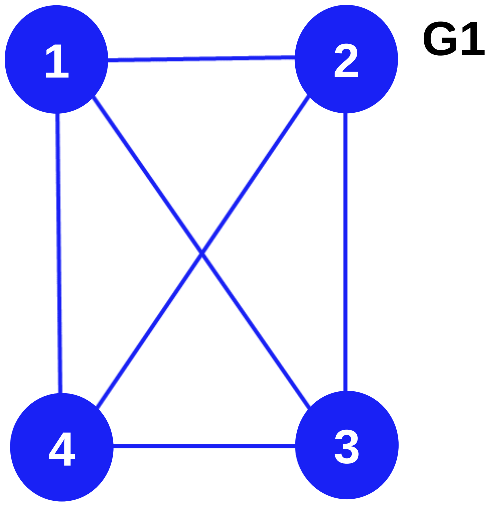


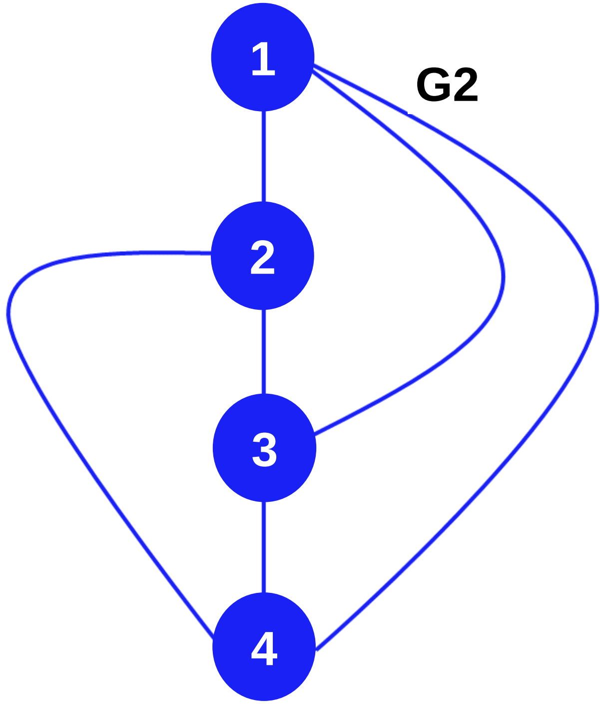


Fonte: Autoria própria.

Teorema sobre o número de nós e arcos

- Para um grafo planar simples e conexo com n nós e a arcos:
- Se a representação planar divide o plano em r regiões, então: ***n-a+r=2***.
- Se n ≥ 3, então: ***a ≤ 3n – 6.***
- Se n ≥ 3 e se não existem ciclos de comprimento 3, então: ***a ≤ 2n – 4***.
- Para o grafo K4, n = 4 e a = 6.
- Tem-se que: r = 4.
- Tem-se que a ≤ 3n – 6, ou seja, 6 ≤ 6.
- Apresenta ciclo de comprimento 3.

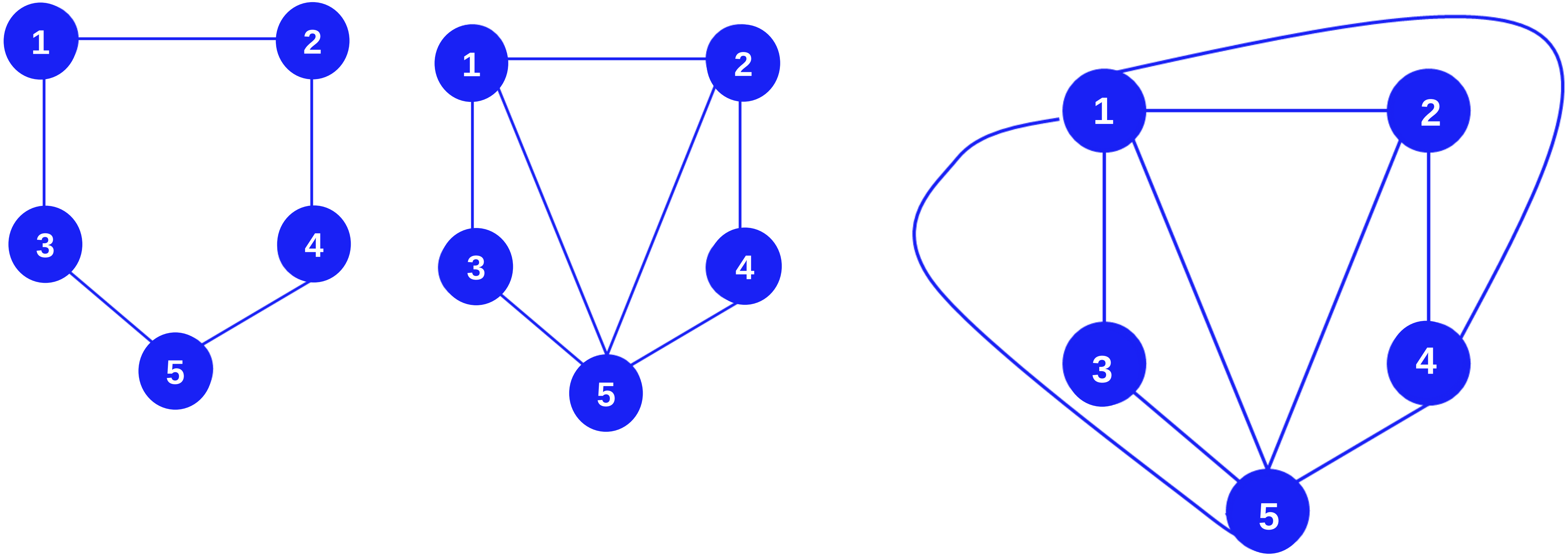




Fonte: Autoria própria.

K5 – Não planar

O grafo K5 é não planar:

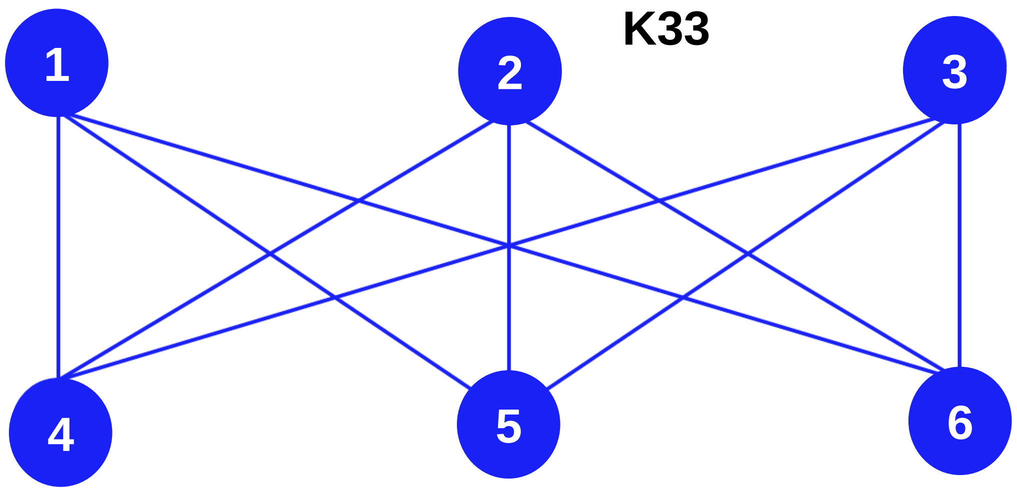


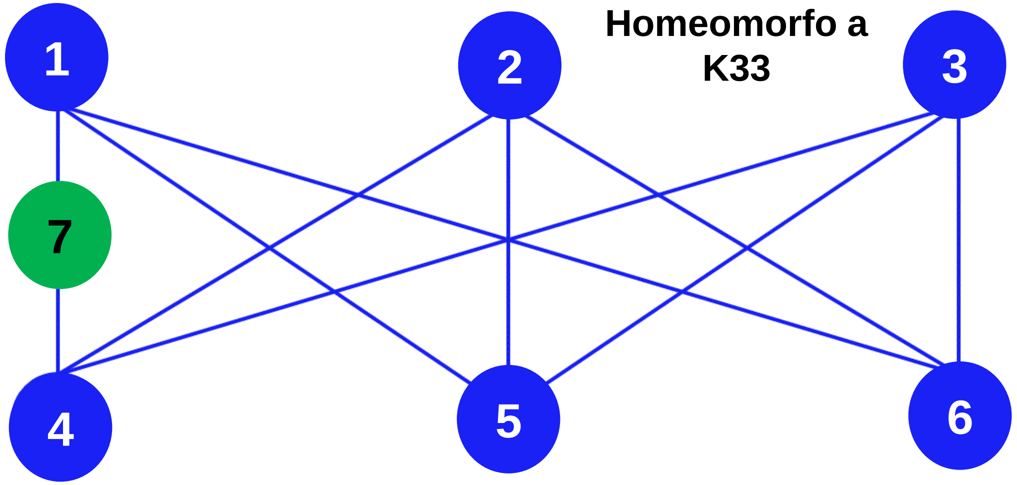
Fonte: Autoria própria.

Grafos homeomorfos

- Dois grafos são ditos **homeomorfos** se ambos podem ser obtidos do mesmo grafo por uma sequência de subdivisões elementares, nas quais um único arco x-y é substituído por dois novos arcos x-v e v-y.

- K33 e Grafo Homeomorfo a K33.

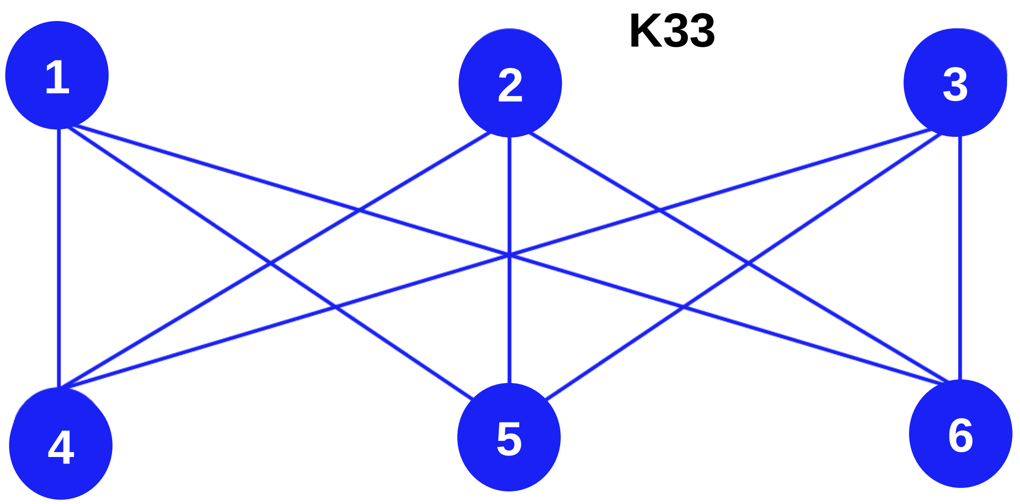


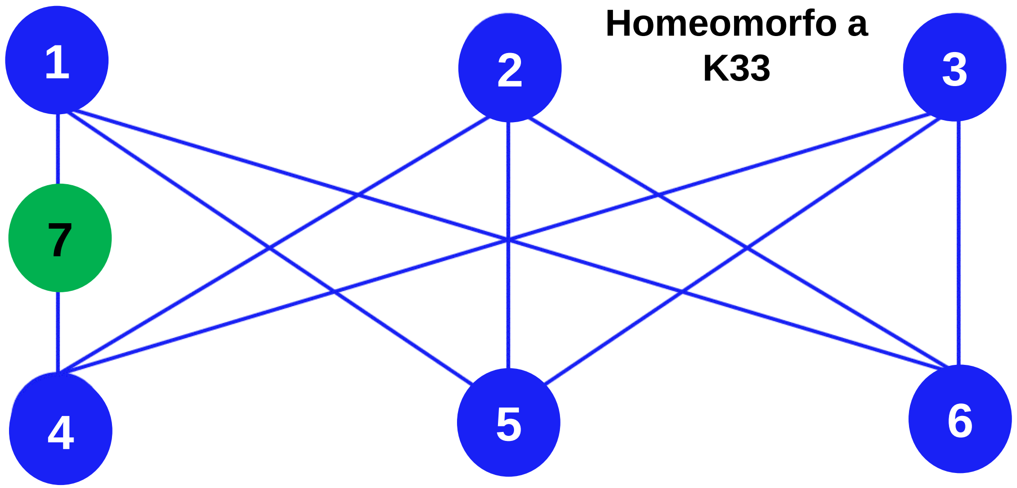


Fonte: Autoria própria.

Teorema de Kuratowsky

- Um grafo é não planar se, e somente se, ele contém um subgrafo que é homeomorfo a K5 ou a K33.
- Grafos não planares:

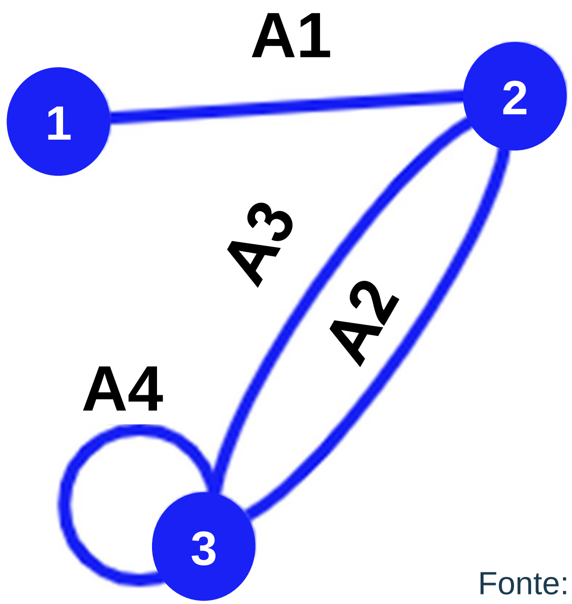




Fonte: Autoria própria.

Representação de grafos no computador – Matriz de adjacência

- Suponha que um grafo tem n nós numerados n1, n2, ..., nn, para fins de identificação dos mesmos. Por outro lado, uma vez que estes sejam ordenados, pode-se formar uma matriz quadrada n x n, em que o elemento i,j é o número de arcos entre os nós ni e nj. Essa matriz é chamada de matriz de adjacência.
- A matriz de um grafo não direcionada, é simétrica.
- M = $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 2 \\ 0 & 2 & 1 \end{bmatrix}$ é a matriz de adjacência para a figura seguinte:

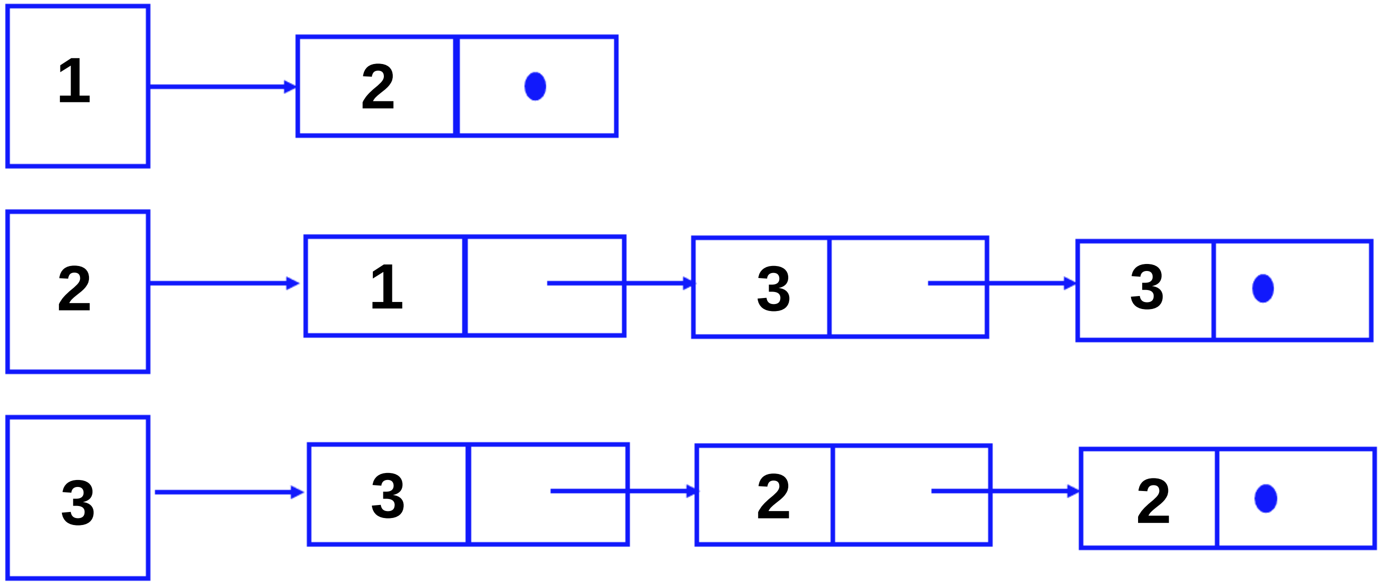


Fonte: Autoria própria.

Representação de grafos no computador – Lista de adjacência

- Um grafo com relativamente poucos arcos pode ser representado de modo mais eficiente armazenando-se apenas os elementos não nulos da matriz de adjacência, ou seja, através de uma lista de adjacência. A lista de adjacência contém um arranjo de ponteiros, um para cada nó. Cada ponteiro do arranjo aponta para uma lista encadeada de nós adjacentes.

- M = $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 2 \\ 0 & 2 & 1 \end{bmatrix}$ é a matriz de adjacência para a seguinte lista de adjacência:



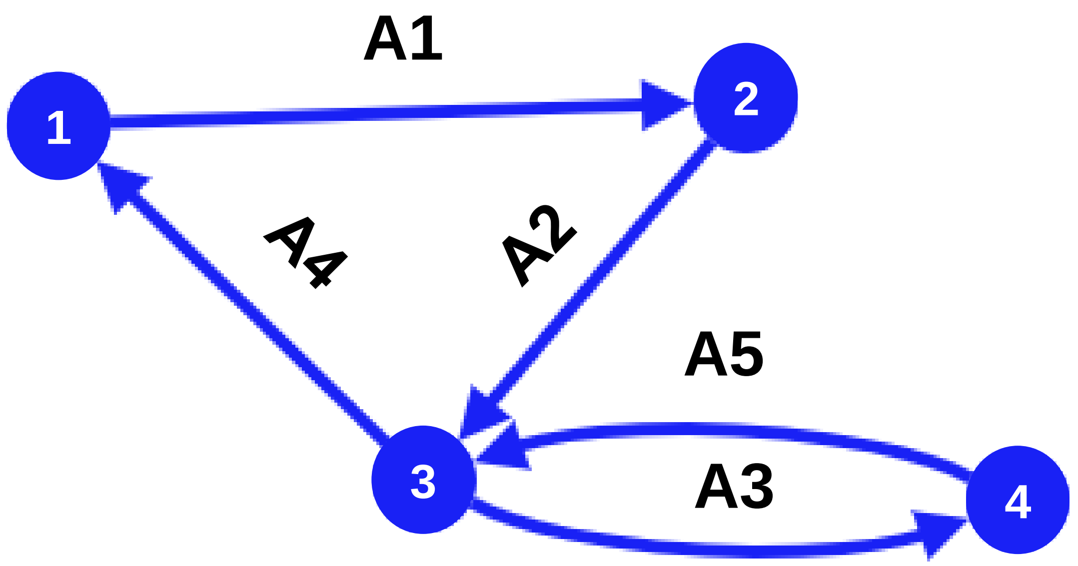
Fonte: Autoria própria.

Representação de grafos no computador para dígrafos

- O conceito de matriz de adjacência também se aplica a dígrafos. Naturalmente, neste caso, a matriz de adjacência não é simétrica.

Como exemplo, para a figura que se segue, a matriz de adjacência é:

- M = $\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$



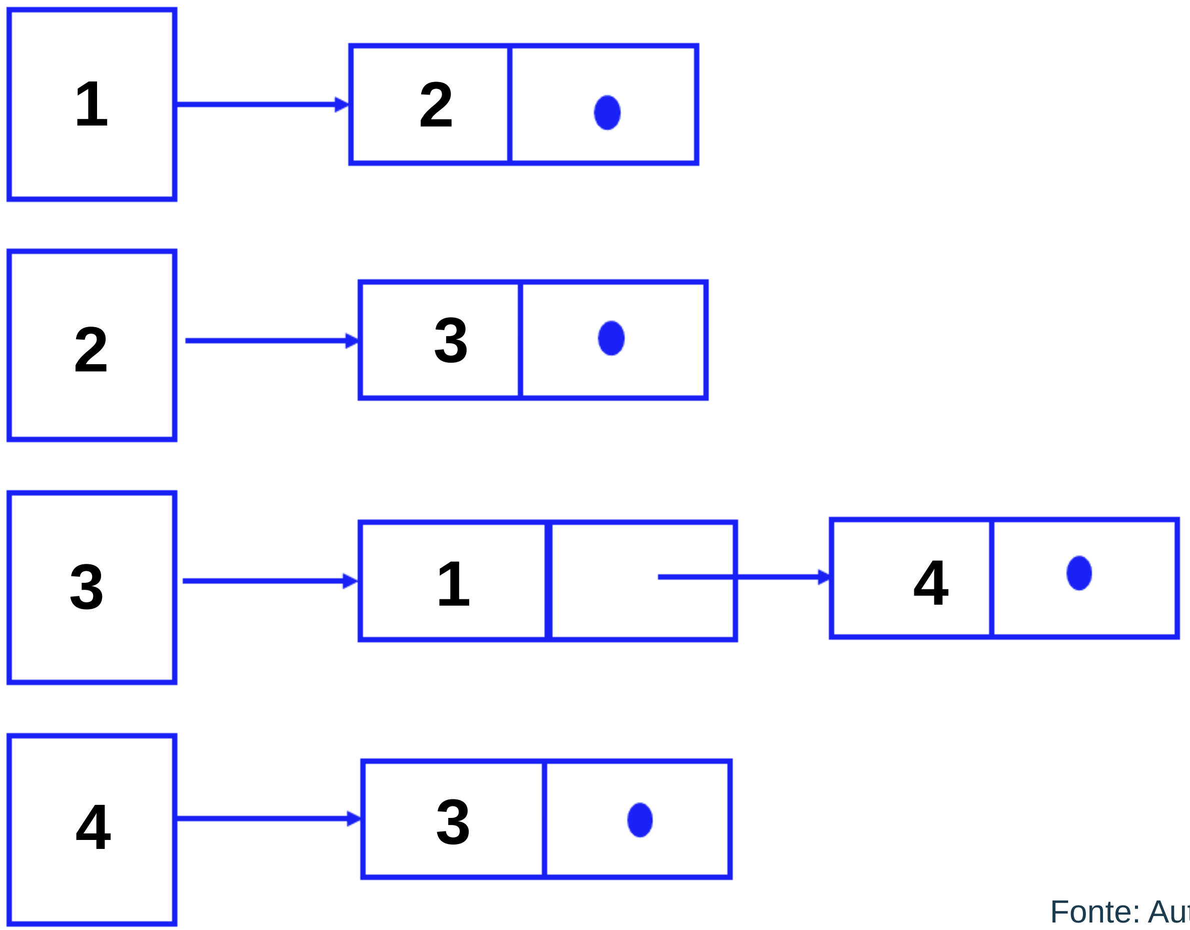
Fonte: Autoria própria.

Representação de grafos no computador para dígrafos

- O conceito de lista de adjacência também se aplica a dígrafos.
- Para a matriz de adjacência do exemplo anterior, tem-se a seguinte lista de adjacência.

- M = $\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

- Lista de adjacência



Fonte: Autoria própria.

Interatividade

A respeito do grafo K3,2, pode-se afirmar que:

I. Trata-se de um grafo planar.
II. Trata-se de um grafo simples.
III. Trata-se de um grafo conexo.

Está correto o que se afirma em:

a) I, apenas.
b) I e II, apenas.
c) I, II e III.
d) II, apenas.
e) III, apenas.

Resposta

A respeito do grafo K3,2, pode-se afirmar que:

I. Trata-se de um grafo planar.

II. Trata-se de um grafo simples.

III. Trata-se de um grafo conexo.

Está correto o que se afirma em:

a) I, apenas.

b) I e II, apenas.

c) I, II e III.

d) II, apenas.

e) III, apenas.

Árvores

- Uma **árvore** é um grafo conexo acíclico com um nó especial, denominado raiz da árvore.

- A **profundidade** de um nó em uma árvore é o comprimento do caminho da raiz ao nó; a raiz tem profundidade 0.

- **A profundidade (altura) de uma árvore** é a maior profundidade dos nós na árvore.

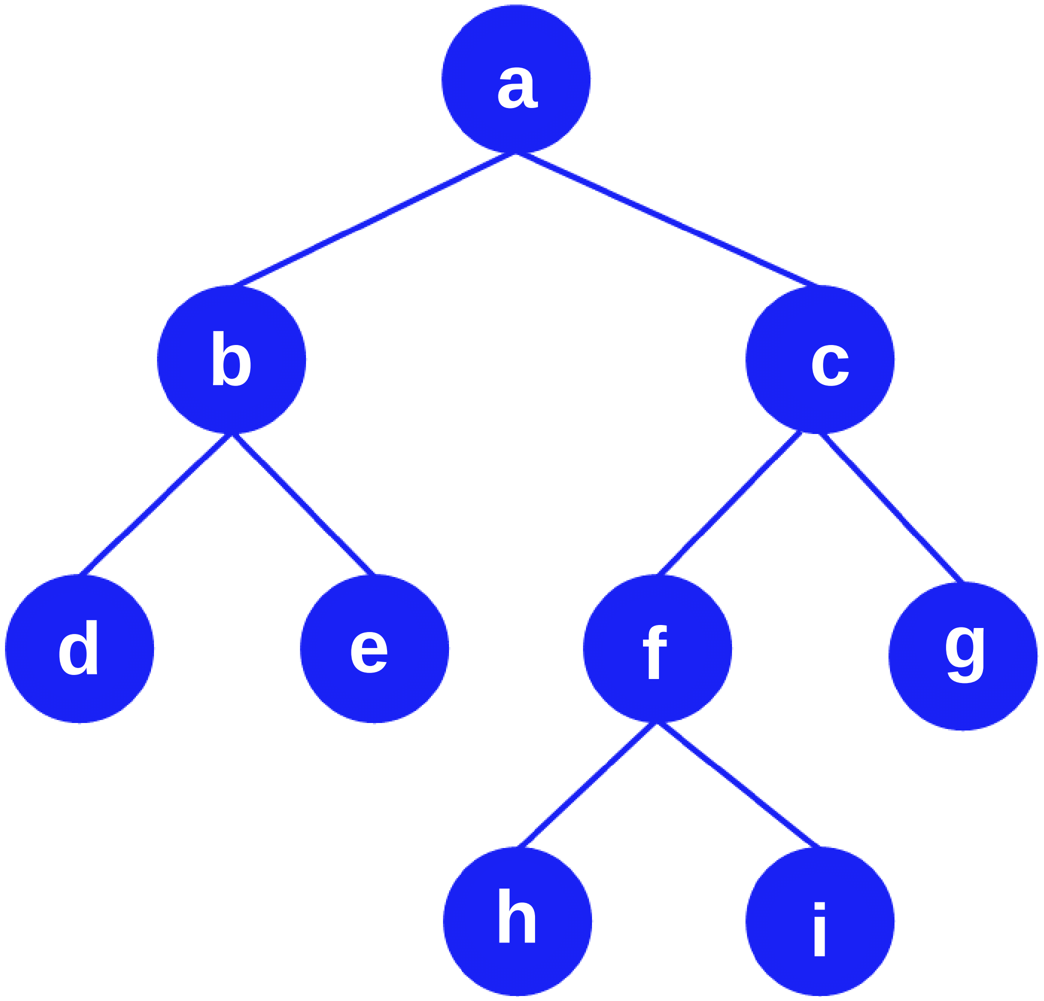
- Um nó sem filhos é chamado de **folha** da árvore: todos os nós que não são folhas são **nós internos.**

- Uma **floresta** é um grafo acíclico (não necessariamente conexo); logo, uma floresta é uma coleção de árvores disjuntas.

Árvores

Para a árvore, tem-se:

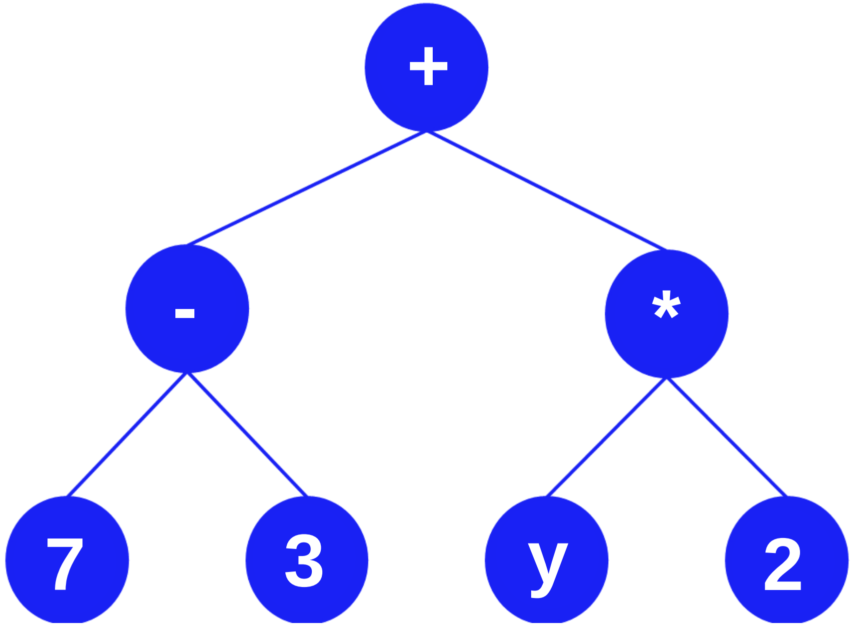
- Altura = 3
- Profundidade do nó h = 3

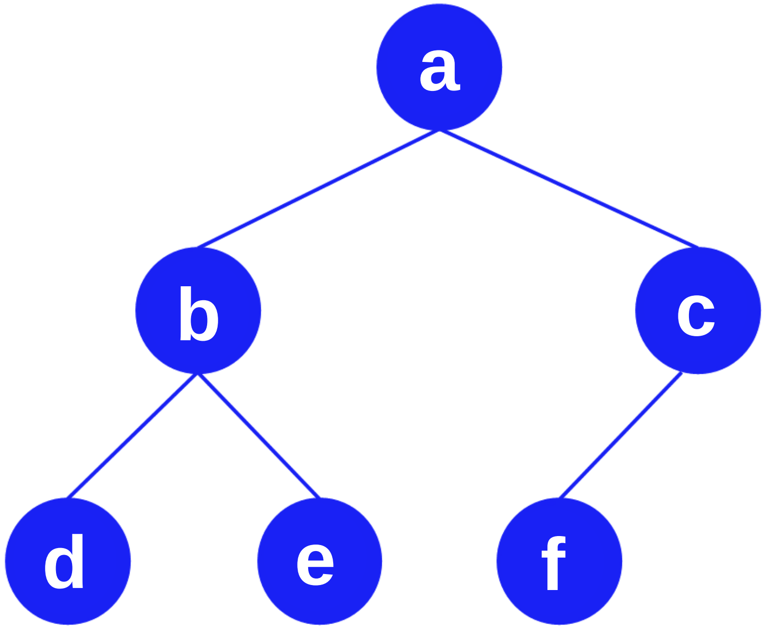


Fonte: Autoria própria.

Árvores

- As árvores denominadas **árvores binárias** apresentam para cada nó, no máximo, dois filhos.
- **Uma árvore binária cheia** é uma árvore que tem todos os nós internos com dois filhos e onde todas as folhas estão na mesma profundidade.
- Uma **árvore binária completa** é uma árvore binária que é quase cheia; o nível mais baixo da árvore vai se completando da esquerda para a direita, mas pode ter folhas faltando.
- Ao lado, árvore binária cheia e árvore completa, respectivamente.





Fonte: Autoria própria.

Representação de árvores binárias

- Exemplo de representação de árvore em tabela binária.

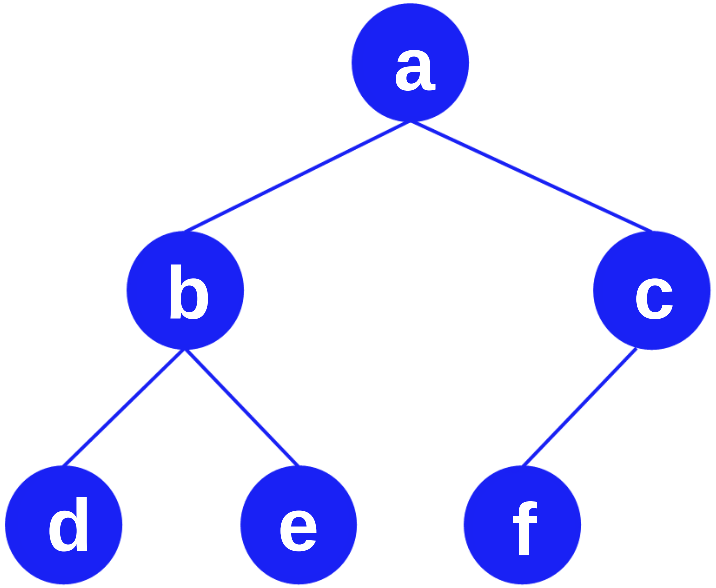


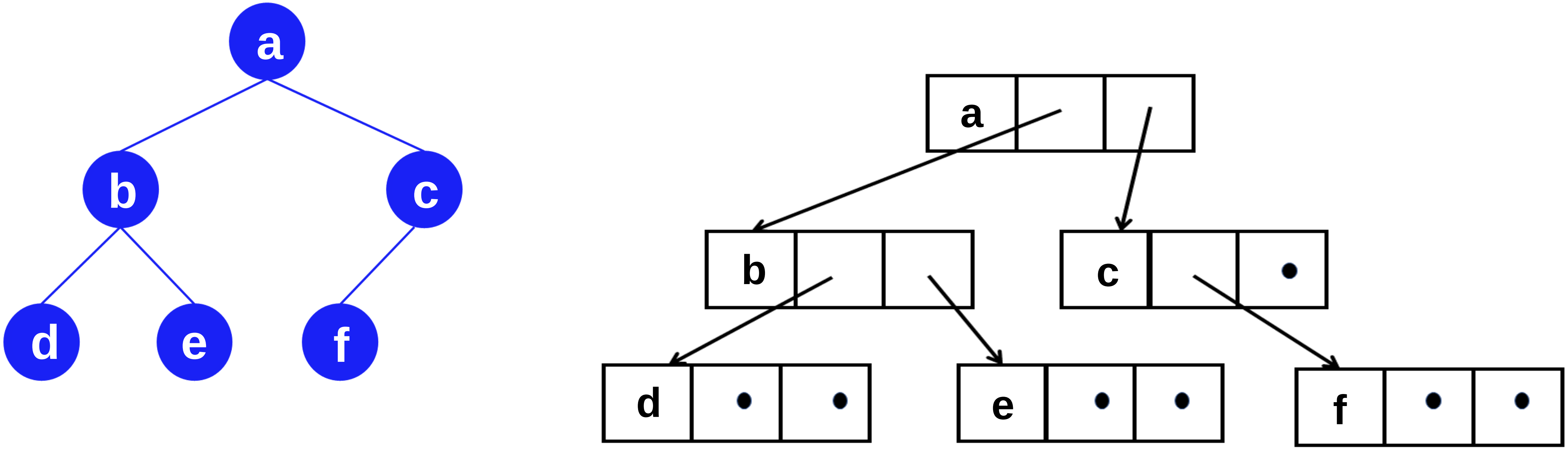
Tabela: Árvore Binária

	Esq.	Dir.
a	b	c
b	d	e
c	f	0
d	0	0
e	0	0
f	0	0

Fonte: Autoria própria.

Representação de árvores binárias

- Exemplo de representação de árvore binária com ponteiros.



Fonte: Autoria própria.

Algoritmos de percurso em árvores

- Para localizar um elemento em uma árvore, há que se percorrer a árvore, ou seja, visitar todos os seus nós. Os três algoritmos mais comuns de percurso em árvores são os percursos em **pré-ordem**, em ordem **simétrica** e em **pós-ordem.**

- No percurso em **pré-ordem**, a raiz é visitada primeiro e depois processam-se as subárvores, da esquerda para a direita.

- No percurso em **ordem simétrica**, a subárvore da esquerda é percorrida em **ordem simétrica**, depois a raiz é visitada e seguidamente as outras subárvores são visitadas da esquerda para a direita, sempre em ordem simétrica.

- No percurso em **pós-ordem**, a raiz é a última a ser visitada após o percurso, em **pós-ordem**, de todas as subárvores da esquerda para a direita.

Algoritmo Pré-Ordem (Gersting, J. L.)

Pré-Ordem(árvore T)

//Escreve os nós de uma árvore com raiz r em pré-ordem

a, b, d, e, c, f, h, i, g

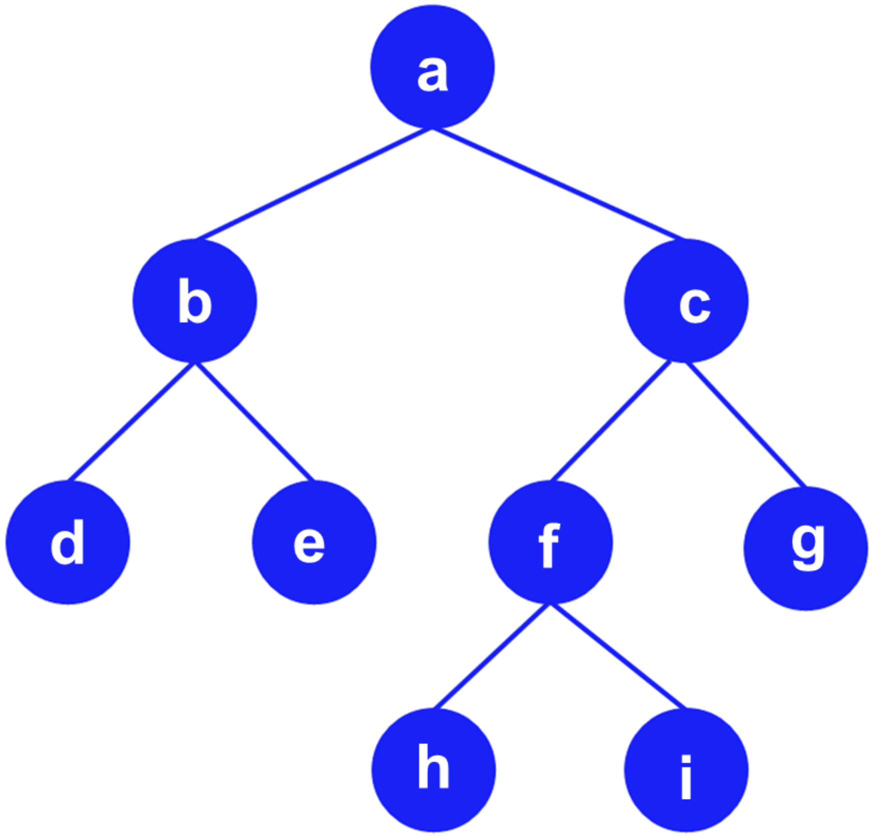
Escreva(r)

Para i = 1 até t faça

Pré-Ordem(Ti)

Fim do Para

Fim Pré-Ordem



Fonte: Autoria própria.

Algoritmo Ordem_Simétrica (Gersting, J.L.)

Ordem_Simetrica(árvore T)

//Escreve os nós de uma árvore com raiz r em simétrica

Ordem_Simétrica(Ti)

Escreva(r)

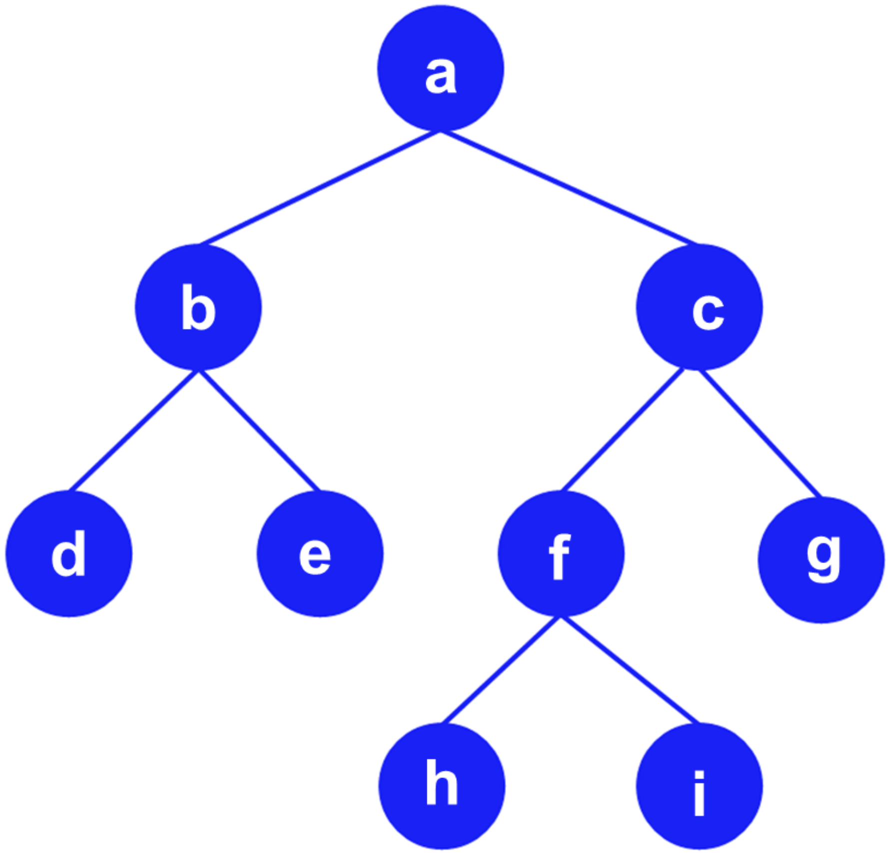
Para i = 2 até t faça

Ordem_Simetrica(Ti)

Fim do Para

Fim Ordem_Simetrica

d, b, e, a, h, f, i, c, g



Fonte: Autoria própria.

Algoritmo Pós-Ordem (Gersting, J. L.)

Pós-Ordem(árvore T)

//Escreve os nós de uma árvore com raiz r em pós-ordem

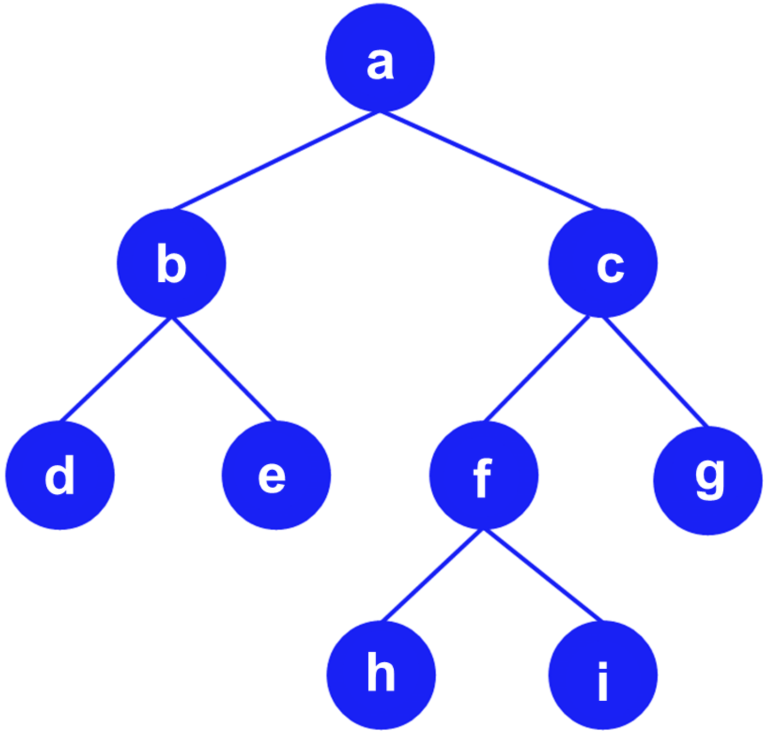
Para i = 1 até t faça

Pós-Ordem(Ti)

Fim do Para

Escreva(r).

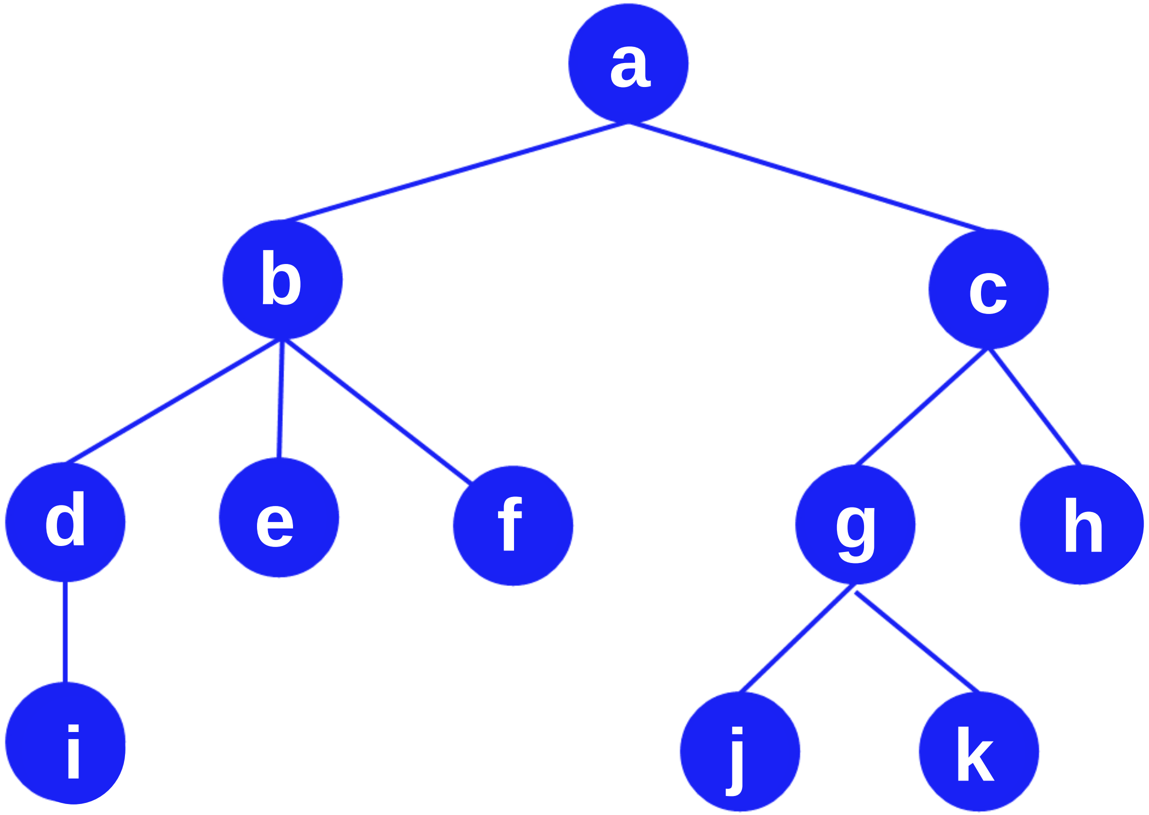
Fim Pós-Ordem d, e, b, h, i, f, g, c, a



Fonte: Autoria própria.

Percurso em árvore – Outro exemplo (Gersting, J. L)

- Considere-se a árvore na figura ao lado.
- Pré-Ordem: a, b, d, i, e, f, c, g, j, k, h.
- Ordem Simétrica; i, d, b, e, f, a, j, g, k, c, h.
- Pós-Ordem: i, d, e, f, b, j, k, g, h, c, a.



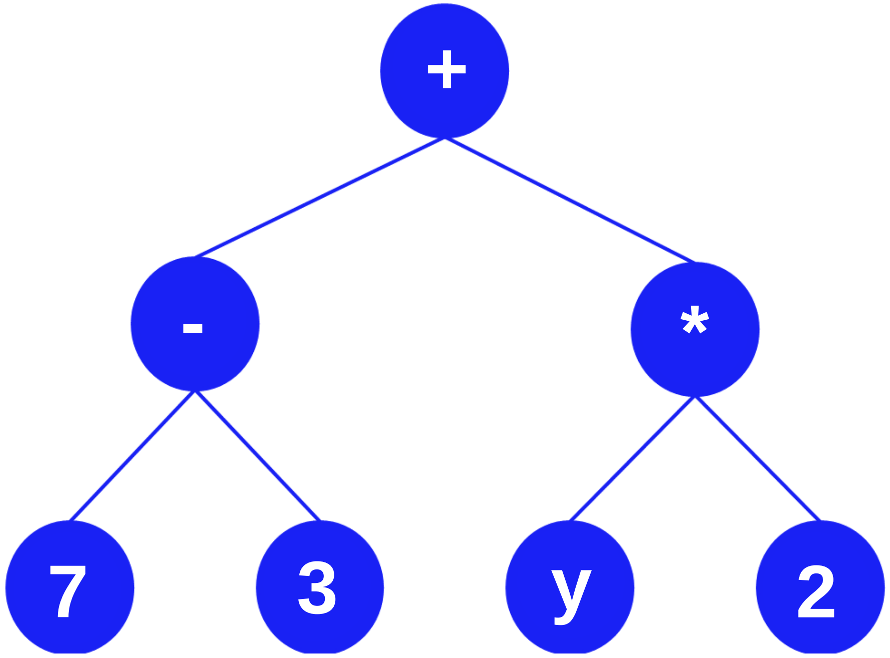
Fonte: Autoria própria.

Representação de expressões algébricas

- Expressões algébricas envolvendo operações binárias podem ser representadas por árvores binárias.

Tem-se que:

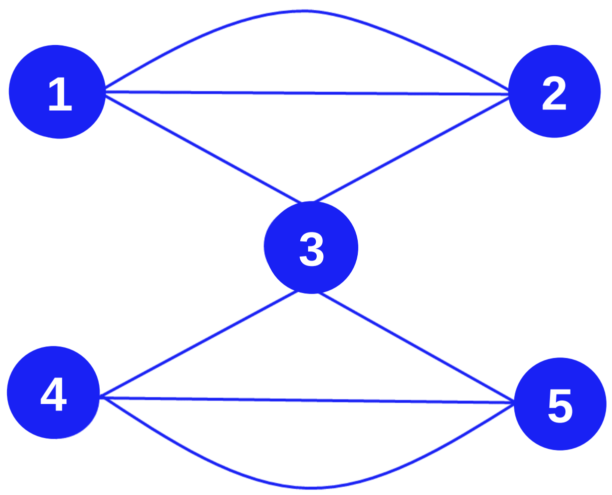
- O percurso pré-ordem:+-73*y2 resulta na notação prefixa ou notação polonesa.
- Percurso ordem simétrica:7-3+y*2 resulta na notação infixa.
- Percurso pós-ordem: 73-y2*+ resulta na notação pós-fixa ou notação polonesa reversa.



Fonte: Autoria própria.

Caminho de Euler

- **<u>Definição: caminho de Euler</u>:** um caminho de Euler, em um grafo G, é um caminho que usa cada arco em G, exatamente, uma vez.
- **<u>Teorema sobre os nós ímpares em um grafo</u>:** o número de nós ímpares em qualquer grafo é par.
- **<u>Teorema sobre os caminhos de Euler</u>:** existe um caminho de Euler em um grafo conexo se, e somente se, não existirem nós impares ou existirem, exatamente, dois nós ímpares. No caso em que não existirem nós ímpares, o caminho pode começar em qualquer nó; no caso de dois nós impares, o caminho precisa começar em um deles e terminar no outro.
- Para o grafo G, o número de **<u>nós ímpares = 4. Não há o caminho de Euler</u>.**

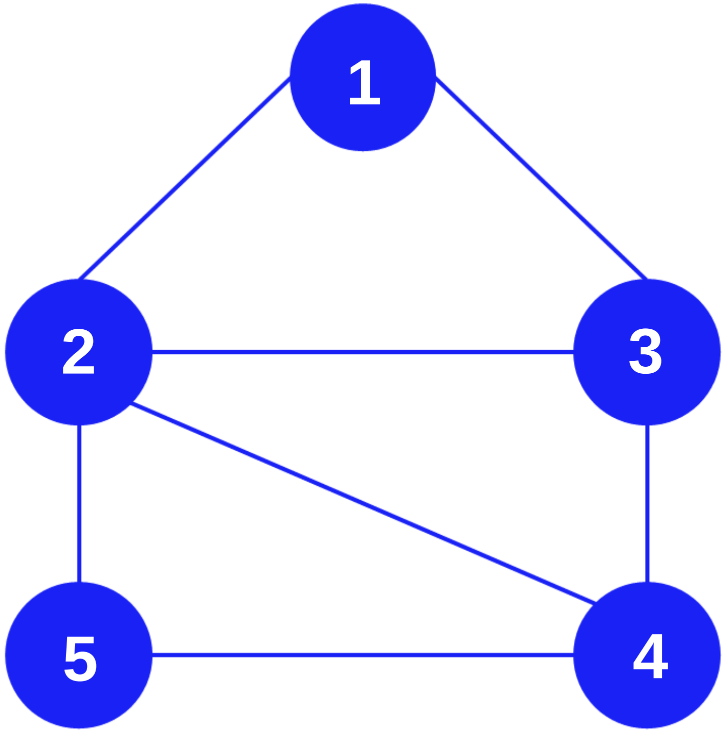


Fonte: Gersting (2014).

Caminho de Euler

Considere o seguinte grafo:

- Grau (1) = grau (5) = 2;
- Grau (2) = 4;
- Grau (3) = grau (4) = 3;
- Existem 2 nós impares; logo, existe um caminho de Euler: 31254234.



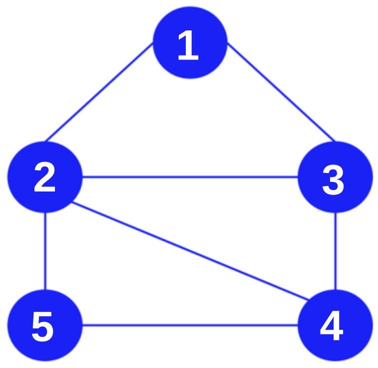
Fonte: Autoria própria.

Caminho de Euler

Observe o grafo apresentado:

Apresenta a seguinte matriz de adjacência e, a partir da inspeção, é possível criar o vetor grau, como se segue:

- $M=\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \Rightarrow \quad \text{grau} = \begin{bmatrix} 2 \\ 4 \\ 3 \\ 3 \\ 2 \end{bmatrix}.$



Fonte: Autoria própria.

- Inspecionando o vetor grau, conclui-se que há 2 nós ímpares: 3 e 4, e que, portanto, existe o caminho de Euler.

Algoritmo caminho de Euler (GERSTING, 2014)

CaminhoDeEuler (matriz n x n A):

- // Determina se existe um caminho de Euler em um grafo conexo com a matriz // de adjacência A.

Variáveis locais:

- Inteiro total; // número de nós ímpares encontrados, até agora;
- Inteiro grau; // o grau de um nó;
- Inteiros: *i, j*; // índice dos arranjos.
- Total = 0.

Algoritmo caminho de Euler (GERSTING, 2014)

Enquanto total < = 2 e i < 0, faça:

Grau = 0

Para *j* = 1, até *n*, **faça**:

Grau = grau + A[i, j] // encontra o grau nó *i*.

Fim do para:

Se ímpar (grau), **então**:

Total = Total + 1 // outro nó de grau ímpar encontrado.

Fim do se.

Algoritmo caminho de Euler (GERSTING, 2014)

```
        I = i + 1
        Fim do enquanto:
        Se total > 2, então:
                Escreva (“Não existe um caminho de Euler”).
       Se não:
                Escreva (“Existe um caminho de Euler”).
        Fim do se.
Fim do CaminhoDeEuler.
```

Interatividade

Considere as seguintes afirmações:

I. Um grafo com quatro nós ímpares ainda pode ser conexo.
II. Existe um caminho de Euler em qualquer grafo com um número par de nós ímpares.
III. Existe um algoritmo com desempenho polinomial quadrático que testa a existência de um caminho de Euler em um grafo com n nós.

Está correto o que se afirma em:

a) I, apenas.
b) II, apenas.
c) III, apenas.
d) I, II e III.
e) I e III, apenas.

Resposta

Considere as seguintes afirmações:

I. Um grafo com quatro nós ímpares ainda pode ser conexo.
II. Existe um caminho de Euler em qualquer grafo com um número par de nós ímpares.
III. Existe um algoritmo com desempenho polinomial quadrático que testa a existência de um caminho de Euler em um grafo com n nós.

Está correto o que se afirma em:

a) I, apenas.
b) II, apenas.
c) III, apenas.
d) I, II e III.
e) I e III, apenas.

Caminho hamiltoniano

- O problema do circuito hamiltoniano corresponde a verificar se existe um circuito que percorre todos os nós de um grafo uma única vez.

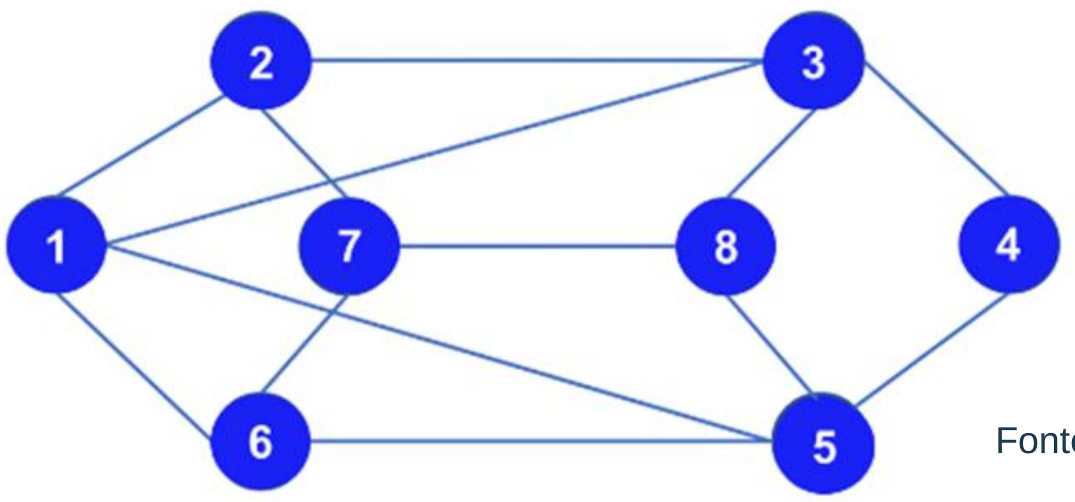
- Ao contrário do caminho de Euler, não existe um critério simples para determinar a existência de um circuito hamiltoniano. Trata-se de um problema cuja solução conhecida é de ordem combinatória, ou seja, em uma situação de pior caso, para um grafo com n nós, deverão ser considerados n! permutações dos nós.

- Um caminho hamiltoniano é um caminho que passa por todos os vértices do grafo sem repeti-los. É importante destacar que nem todo grafo possui um caminho hamiltoniano.

- Encontrar um caminho hamiltoniano em um grafo é um problema NP-completo, o que significa que não há um algoritmo conhecido que possa resolver o problema de forma eficiente para todos os casos.

Algoritmos de percurso – Percurso em nível

- O percurso em dado grafo G simples e conexo consiste em escrever todos os nós em determinada ordem.
- A busca em nível começa em um nó arbitrário, anota todos os seus nós adjacentes e seguidamente os nós adjacentes àqueles anteriormente encontrados e assim por diante.



Fonte: Autoria própria.

O percurso em nível a partir do nó 1 resulta na seguinte lista de nós:

- 1, 2, 3, 6, 5, 7, 4, 8.

Busca em nível

EmNível (grafo G; nó a): // Escreve os nós do grafo G em ordem de nível a partir do nó a.

```
Variáveis locais: fila de nós F:
Inicialize F como sendo vazio;
Marque a como tendo sido visitado;
Escreva (a);
Inserir (a, F).
Enquanto F não é vazio, faça:
    Para cada nó n adjacente à frente (F), faça:
        Se n não foi visitado, então:
            Marque n como tendo sido visitado;
            Escreva n;
            Inserir (n, F).
        Fim_se.
```

Busca em nível

```
                Fim_para.
                Retirar (F).
        Fim_enquanto.
Fim EmNivel.
```

Busca em largura

- A busca em largura (BFS - *Breadth-First Search*) é um algoritmo para percorrer um grafo de forma sistemática e explorar todos os vértices e arestas do grafo de maneira ordenada.
  A ideia básica é visitar todos os vértices de um mesmo nível antes de avançar para os vértices de níveis mais profundos. A seguir, segue uma descrição geral do algoritmo de busca em largura.

1. Comece a busca em um vértice inicial do grafo, marcando-o como visitado.
2. Adicione o vértice inicial em uma fila.
3. Enquanto a fila não estiver vazia:
   a. Remova um vértice da fila.
   b. Visite todos os vizinhos desse vértice que ainda não foram visitados e marque-os como visitados.
   c. Adicione todos os vizinhos visitados na fila.

Busca em largura (continuação)

4. Quando não houver mais vértices na fila, a busca em largura estará concluída.

- O processo anterior é repetido até que todos os vértices do grafo sejam visitados. A fila é usada para manter o controle dos vértices que ainda precisam ser visitados. Quando um vértice é visitado, todos os seus vizinhos que ainda não foram visitados são adicionados à fila, para que eles possam ser visitados posteriormente.

Busca em profundidade

- A busca em profundidade (DFS - *Depth-First Search*) é um algoritmo para percorrer um grafo de forma sistemática e explorar todos os vértices e arestas do grafo de maneira ordenada. A ideia básica é visitar todos os vértices de um ramo do grafo antes de voltar para a raiz do ramo e explorar outro ramo. A seguir, segue uma descrição geral do algoritmo de busca em profundidade.

1. Comece a busca em um vértice inicial do grafo, marcando-o como visitado.
2. Para cada vizinho do vértice inicial que ainda não foi visitado, repita os passos 3 a 5:
3. Visite o vizinho e marque-o como visitado.
4. Recursivamente, visite todos os vizinhos desse vértice que ainda não foram visitados e marque-os como visitados.
5. Retorne ao vértice atual.
6. Quando não houver mais vértices a serem visitados, a busca em profundidade estará concluída.

Ordenação topológica

- A ordenação topológica é um conceito importante em teoria dos grafos que se aplica a grafos direcionados acíclicos (DAGs - *Directed Acyclic Graphs*). Uma ordenação topológica é uma ordenação linear dos vértices do grafo que respeita a direção das arestas. Em outras palavras, se existe uma aresta direcionada do vértice u para o vértice v, então u aparece antes de v na ordenação.

- Existem diferentes algoritmos para encontrar uma ordenação topológica em um DAG.

Um dos algoritmos mais simples é baseado em uma busca em profundidade. O algoritmo funciona da seguinte maneira:

Ordenação topológica (continuação)

1. Inicialmente, marque todos os vértices do grafo como não visitados.
2. Escolha um vértice não visitado e execute uma busca em profundidade a partir dele.
3. À medida que os vértices são visitados durante a busca em profundidade, adicione-os a uma lista.
4. Retorne à etapa 2 e escolha outro vértice não visitado. Repita até que todos os vértices tenham sido visitados.

- Ao final do algoritmo, a lista criada contém uma ordenação topológica do grafo. Esse algoritmo possui uma complexidade de tempo de O(V+E), em que V é o número de vértices e E é o número de arestas do grafo.

Componentes fortemente conexas

- Uma das aplicações de busca em profundidade é a decomposição em suas componentes fortemente conexas.

- Um grafo dirigido é fortemente conexo se cada vértice pode ser alcançado de qualquer outro vértice.

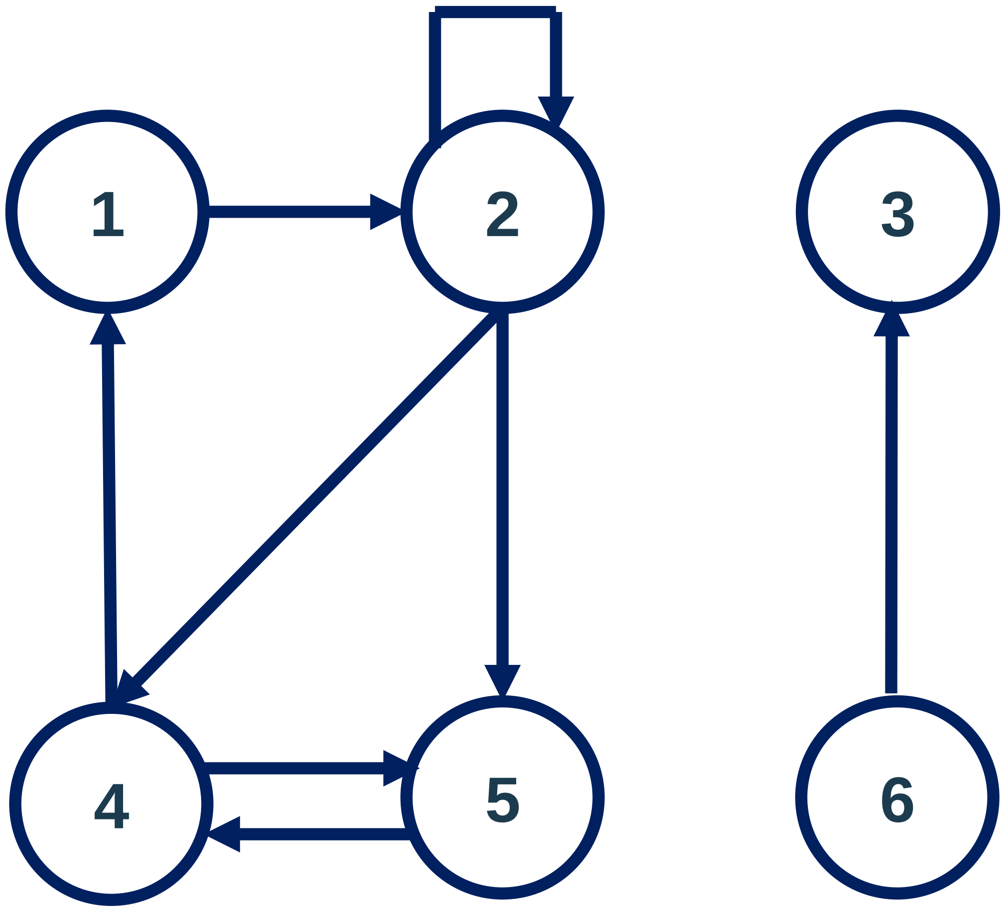
- As componentes fortemente conexas de um grafo dirigido são as classes de equivalência de vértices sob a relação “são mutuamente acessíveis”.

- Um grafo dirigido é fortemente conexo se tem somente uma componente fortemente conexa.

- Exemplos de aplicações: verificar se uma rede está desconectada, para fins de clusterização.

Componentes fortemente conexas

- O grafo da figura que se segue tem três componentes fortemente conexas: {1, 2, 4, 5}, {3} e {6}.
- Todos os pares de vértices em {1, 2, 4, 5} são mutuamente acessíveis. Os vértices {3, 6} não formam uma componente fortemente conexa, visto que o vértice 6 não pode ser alcançado do vértice 3.



Fonte: Adaptado de: Cormen (2012).

Interatividade

Considere as seguintes afirmações e assinale a alternativa correta.

I. A busca em largura é geralmente implementada utilizando uma estrutura de dados fila, que armazena os vértices que ainda não foram visitados em ordem de descoberta. Quando um vértice é descoberto, ele é adicionado à fila e, quando é visitado, é removido da fila.

II. A busca em profundidade utiliza uma abordagem recursiva para explorar todos os vértices do grafo.

III. Uma ordenação topológica é uma ordenação linear dos vértices do grafo que respeita a direção das arestas. Em outras palavras, se existe uma aresta direcionada do vértice u para o vértice v, então u aparece antes de v na ordenação.

Está correto o que se afirma em:

a) I, apenas.
b) I e II, apenas.
c) I e III, apenas.
d) I, II e III.
e) II, apenas.

Resposta

Considere as seguintes afirmações e assinale a alternativa correta.

I. A busca em largura é geralmente implementada utilizando uma estrutura de dados fila, que armazena os vértices que ainda não foram visitados em ordem de descoberta. Quando um vértice é descoberto, ele é adicionado à fila e, quando é visitado, é removido da fila.

II. A busca em profundidade utiliza uma abordagem recursiva para explorar todos os vértices do grafo.

III. Uma ordenação topológica é uma ordenação linear dos vértices do grafo que respeita a direção das arestas. Em outras palavras, se existe uma aresta direcionada do vértice u para o vértice v, então u aparece antes de v na ordenação.

Está correto o que se afirma em:

a) I, apenas.

b) I e II, apenas.

c) I e III, apenas.

d) I, II e III.

e) II, apenas.

Referências

- BOAVENTURA NETTO, P. O.; JURKIEWICZ, S. *Grafos*: introdução e prática. São Paulo: Blucher, 2009.
- CALAÇA, O. Uma introdução às redes neurais para grafos (GNN). *Medium*, 2020. Disponível em: https://bit.ly/3oxL8Od. Acesso em: 24 abr. 2023.
- CORMEN, T. H. *et al*. *Algoritmos*: teoria e prática. Rio de Janeiro: Elsevier, 2012.
- GERSTING, J. L. *Fundamentos matemáticos para a ciência da computação*: matemática discreta e suas aplicações. 7. ed. Rio de Janeiro: LTC, 2014.
- GOLDBARG, M.; GOLDBARG, E. *Grafos*: conceitos, algoritmos e aplicações. Rio de Janeiro: Elsevier, 2012.
- GÖTTLICH, S.; TOTZECK, C. Parameter calibration with stochastic gradient descent for interacting particle systems driven by neural networks. *Mathematics of Control*, Signals, and Systems, 2021.
- HAYKIN, S. *Redes neurais*: princípios e prática. Porto Alegre: Bookman, 2007.

Referências

- LESKOVEC, J. *CS224W*: Machine learning with graphs. Califórnia: Universidade de Stanford, 2022.
- LIPSCHUTZ, S.; LIPSON, M. *Matemática discreta*. Coleção Schaum. 3. ed. Porto Alegre: Bookman, 2013.
- NICOLETTI, M.; HRUSCHKA JÚNIOR, E. R. *Fundamentos da teoria dos grafos para computação*. 3. ed. Rio de Janeiro: LTC, 2018.
- NORVIG, P. *Inteligência artificial*. 3. ed. Rio de Janeiro: LTC, 2013.
- PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de* software: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.
- SZWARCFITER, J. L.; MARKENZON, L. *Estruturas de dados e seus algoritmos*. 3. ed. Rio de Janeiro: LTC, 2010.
- SZWARCFITER, J. L.; PINTO, P. E. D. *Teoria computacional de grafos*: os algoritmos com programas Python. Rio de Janeiro: Elsevier, 2018.

ATÉ A PRÓXIMA!