

# Grafos: noções básicas

Prof. Jefferson T. Oliva

Algoritmos e Estrutura de Dados II (AE23CP)  
Engenharia de Computação  
Departamento Acadêmico de Informática (Dainf)  
Universidade Tecnológica Federal do Paraná (UTFPR)  
Campus Pato Branco

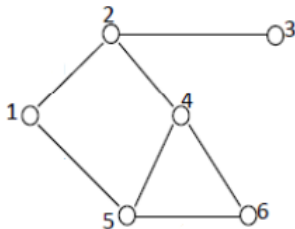
- Terminologia
- Algoritmos em Grafos
- Representação de Grafos
  - Matriz de adjacência
  - Lista de adjacência

- Muitas aplicações necessitam considerar conexões entre pares objetos
  - Redes sociais
  - Páginas web
  - Empresas ou organizações
  - Mapas
- Representação dessas conexões: grafos

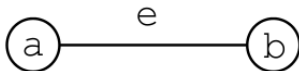
## Terminologia

# Terminologia

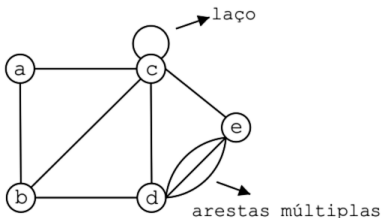
- Um **grafo** é um par  $G = (V, E)$ , onde
  - $V$  é o conjunto de vértices
  - $E$  é o conjunto de arestas
- Exemplo na figura abaixo:
  - $V = \{1, 2, 3, 4, 5, 6\}$
  - $E = \{(1, 2), (1, 5), (2, 3), (2, 4), (4, 5), (4, 6), (5, 6)\}$



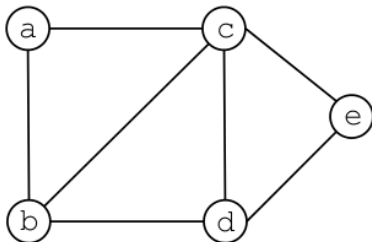
- Dada uma aresta  $e = (a, b)$ , dizemos que os vértices  $a$  e  $b$  são os **extremos** da aresta  $e$  se  $a$  e  $b$  são **adjacentes**
- Seguindo a notação anterior, aresta  $e$  é **incidente** aos vértices  $a$  e  $b$



- Dizemos que um grafo é **simples** quando não possui laços ou arestas múltiplas
- Um **laço** é uma aresta com extremos idêntico
- **Arestas múltiplas** são duas ou mais arestas com o mesmo par de vértices como extremos



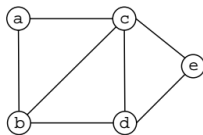
- Denotamos por  $|V|$  e  $|E|$  a cardinalidade dos conjuntos de vértices e arestas de um grafo  $G$ , respectivamente
- No exemplo abaixo temos  $|V| = 5$  e  $|E| = 7$



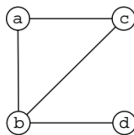
- O **tamanho** do grafo  $G$  é dado por  $|V| + |E|$



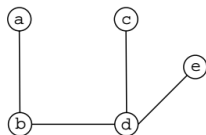
- Um **subgrafo**  $H = (V', E')$  de um grafo  $G = (V, E)$  é um grafo tal que  $V' \subseteq V$  e  $E' \subseteq E$
- Um *subgrafo gerador* de  $G$  é um subgrafo  $H$  com  $V' = V$



Grafo G

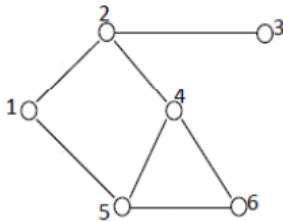


Subgrafo não gerador

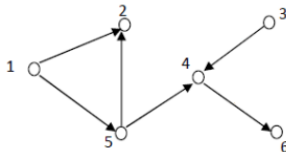


Subgrafo gerador

- Existem 2 tipos de grafos:
  - Não direcionado** (não orientado)

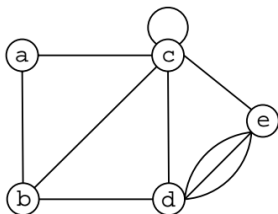


- Direcionado** (orientado)



- Grafo direcionado
  - Arestas são representadas por setas
  - Pode conter arestas de um vértice para si mesmo (*self-loop*)
- Grafo não-direcionado
  - Arestas  $(x, y)$  e  $(y, x)$  são consideradas como uma única aresta
- Adjacência
  - Grafo direcionado: se  $(a, b)$  é uma aresta, então  $b$  é adjacente a  $a$
  - Grafo não-direcionado: adjacência entre vértices é simétrica

- **Grau** de um vértice ( $d(v)$ ): determinado pela quantidade de arestas que incidem o vértice  $v$ 
  - Grafo direcionado: quantidade de arestas “saem” (*out-degree*) + quantidade de arestas “entram” (*in-degree*)



$$d(a) = 2$$

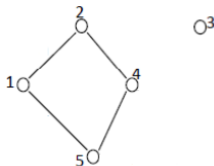
$$d(b) = 3$$

$$d(c) = 6$$

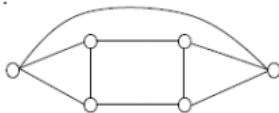
$$d(d) = 5$$

$$d(e) = 4$$

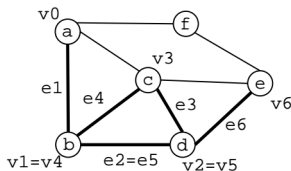
- **Vértice isolado:** não há conexões por arestas



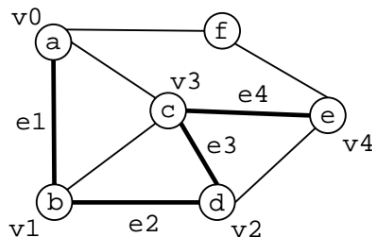
- **Grafo k-regular:** todos os vértices tem grau  $k$



- Um **passeio**  $P$  de  $v_0$  a  $v_n$  no grafo  $G$  é uma sequência finita e não vazia  $(v_1, e_1, v_2, \dots, e_n, v_n)$ , tal que, para todo  $1 \leq i \leq n$ ,  $v_{i-1}$  e  $v_i$  são extremos de  $e_i$
- O **comprimento** do passeio  $P$  é dado pelo seu número de arestas ( $n$ )

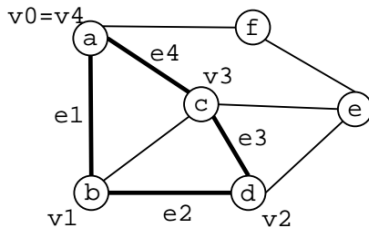


- **Caminho:** passeio em que todos os vértices são distintos
- **Trilha:** passeio em que todas as arestas são distintas
- **Passeio simples** (também denominado **caminho simples**): não há repetição de vértices e nem de arestas na sequência



Caminho Simples

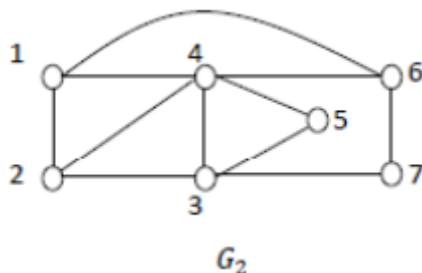
- Um **ciclo** ou **caminho fechado** é um caminho onde  $v_0 = v_n$
- Um ciclo é dito ser **simples** se  $v_0, \dots, v_{n-1}$  são distintos
- Um grafo que não possui ciclos é dito ser acíclico



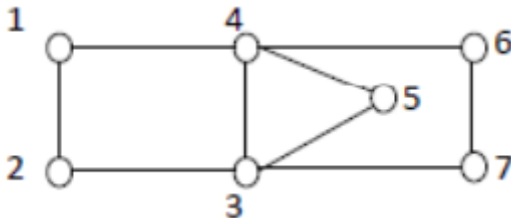
Ciclo



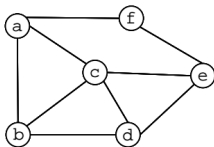
- Caminho Hamiltoniano: caminho passa por todos os vértices
- Ciclo Hamiltoniano: ciclo que passa por todos os vértices
- Grafo Hamiltoniano



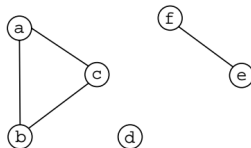
- Caminho Euleriano: trilha que passa por todas as arestas
- Ciclo Euleriano: ciclo que passa por todas as arestas
- Grafo Euleriano
  - Teorema: um grafo  $G$  é euleriano e somente se todos os vértices de  $G$  têm grau par



- **Grafo conexo:** cada par de vértices está conectado por uma aresta
- **Grafo fortemente conexo:** cada par de vértices são alcançáveis a partir de um outro
- **Grafo não conexo:** contém vértices não acessíveis por um determinado componente



Conexo

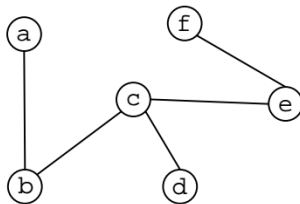


Não-conexo com  
3 componentes conexos

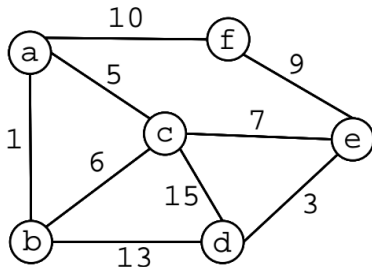
- **Grafo orientado fortemente conexo:** caso exista caminho orientado entre quaisquer par de vértices



- Um grafo  $G$  é uma **árvore** se é conexo e acíclico:
  - $G$  é conexo e possui exatamente  $|V| - 1$  arestas
  - A remoção de qualquer aresta desconecta o grafo (*minimal* conexo)
  - Para todo par de vértices  $u, v$  de  $G$ , existe um único caminho de  $u$  a  $v$  em  $G$



- **Grafo ponderado:** cada aresta está associado a um valor  $c(e)$ , o qual denominamos custo (ou peso) da aresta



## Algoritmos em Grafos

- **Caminho mínimo:** dado um conjunto de cidades, as distâncias entre elas e duas cidades  $A$  e  $B$ , determinar um caminho (trajeto) mais curto de  $A$  até  $B$
- **Árvore geradora de peso mínimo:** dado um conjunto de computadores, onde cada par de computadores pode ser ligado usando uma quantidade de fibra ótica, encontrar uma rede interconectando-os que use a menor quantidade de fibra ótica possível



- **Problema do caixeiro viajante:** dado um conjunto de cidades, encontrar um passeio que sai de uma cidade, passa por todas as cidades e volta para a cidade inicial tal que a distância total a ser percorrida seja menor possível
- **Problema chinês do correio:** dado o conjunto das ruas de um bairro, encontrar um passeio que passa por todas as ruas voltando ao ponto inicial tal que a distância total a ser percorrida seja menor possível

## Representação de Grafos

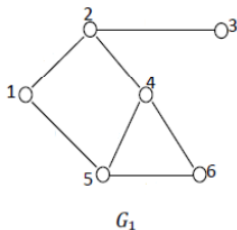
# Representação de Grafos

- A complexidade dos algoritmos para solução de problemas modelados por grafos depende da sua representação interna
- Principais abordagens de representação de grafos
  - Matriz de adjacência
  - Lista de adjacência
- O uso de uma ou outra para um determinado algoritmo depende da natureza das operações que ditam a complexidade do algoritmo

# Representação de Grafos

## Matriz de adjacência

- Seja  $A$ , uma matriz  $n \times n$  de adjacência, onde:
  - $|V| = n$
  - $a_{i,j} = 1$ , se  $(i,j) \in E$
  - $a_{i,j} = 0$ , se  $(i,j) \notin E$
- A matriz de adjacência é simétrica para grafos não orientados

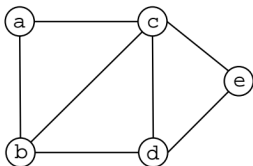


$$A_{G_1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

# Representação de Grafos

## Matriz de adjacência

- Outro exemplo de matriz de adjacência

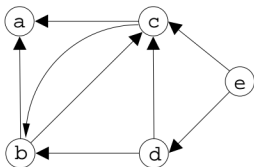


	a	b	c	d	e
a	0	1	1	0	0
b	1	0	1	1	0
c	1	1	0	1	1
d	0	1	1	0	1
e	0	0	1	1	0

# Representação de Grafos

## Matriz de adjacência

- Exemplo de um grafo orientado e a sua respectiva matriz de adjacência



	a	b	c	d	e
a	0	0	0	0	0
b	1	0	1	0	0
c	1	1	0	0	0
d	0	1	1	0	0
e	0	0	1	1	0

# Representação de Grafos

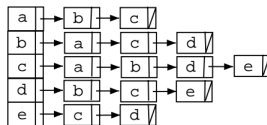
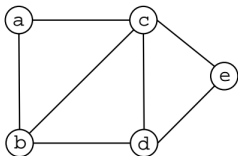
## Lista de adjacência

- Seja  $G = (V, E)$  um grafo simples (orientado ou não)
- A representação de  $G$  por uma lista de adjacência consiste em:
  - Para cada vértice  $v$  há uma lista encadeada ( $Adj[v]$ ) dos vértices adjacentes a  $v$
  - O vértice  $u$  aparece em  $Adj[v]$  se há aresta  $(v, u)$  no grafo  $G$
  - Os vértices podem estar em qualquer ordem em uma lista de adjacência

# Representação de Grafos

## Lista de adjacência

- Exemplo de um grafo não orientado representado por listas de adjacência

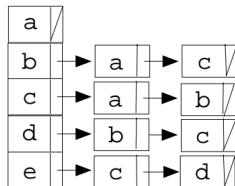
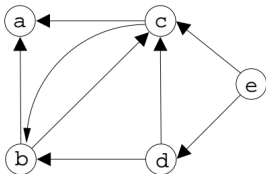




# Representação de Grafos

## Lista de adjacência

- Exemplo de um grafo orientado representado por listas de adjacência



- Matriz  $\times$  Lista de adjacência
  - Matriz de adjacência:
    - Fácil verificar se  $(i, j)$  é uma aresta de  $G$
    - Complexidade de espaço:  $\Theta(|V|^2)$
    - Adequada para grafos densos ( $|E| = \Theta(|V|^2)$ )
  - Lista de adjacência:
    - Fácil descobrir os vértices adjacentes a um dado vértice  $v$  (ou seja, listar  $Adj[v]$ )
    - Complexidade de espaço:  $\Theta(|V| + |E|)$
    - Adequada para grafos esparsos ( $|E| = \Theta(|V|)$ )



Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.  
*Introduction to Algorithms.*  
Third edition, The MIT Press, 2009.



Marin, L. O.  
Grafos: Noções Básicas e Representação. AE23CP –  
Algoritmos e Estrutura de Dados II.  
*Slides.* Engenharia de Computação. Dainf/UTFPR/Pato  
Branco, 2017.



Tenenbaum, A.; Langsam, Y.  
*Estruturas de Dados usando C.*  
Pearson, 1995.



Ziviani, N.  
*Projeto de Algoritmos - com implementações em Java e C++.*  
Thomson, 2007.