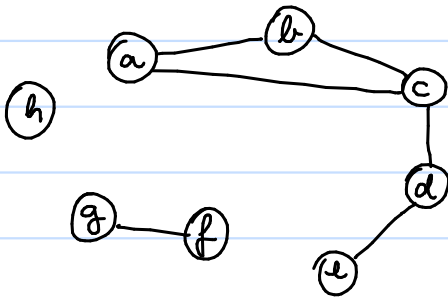


Grafos

DEFINIÇÃO: Um **grafo** G é um par de conjuntos (V, E) onde V é um conj. de elementos chamados **vértices** e E é um conj. de elem. chamados de **arestas**, sendo que cada aresta é um par de vértices

→ Exemplo: $H = (V, E)$ com $V = \{a, b, c, d, e, f, g, h\}$
e $E = \{\{a, b\}, \{a, c\}, \{b, c\}, \{c, d\}, \{d, e\}, \{f, g\}\}$



Obs: essa definição é de **grafos simples**

Obs 2: $E = \{ab, ac, bc, cd, de, fg\}$.

Obs 3: Usaremos $V(\cdot)$ e $E(\cdot)$

Terminologias básicas em grafos

→ Se $e = uv$ é uma aresta de um grafo G , então:

↳ u e v são **vizinhos / adjacentes**

↳ u e v são **extremos** de e

↳ e **incide** em u e em v

→ Dado $u \in V(G)$:

↳ o **grau** de u é o nº de arestas incidentes a u ($d_G(u)$)

↳ a **vizinhança** de u é o conj. de vizinhos de u ($N_G(u)$)

→ Dado um grafo G :

↳ o **grau mínimo** é $\delta(G) = \min \{d(v) : v \in V(G)\}$

↳ o **grau máximo** é $\Delta(G) = \max \{d(v) : v \in V(G)\}$

1) resultados mais básicos

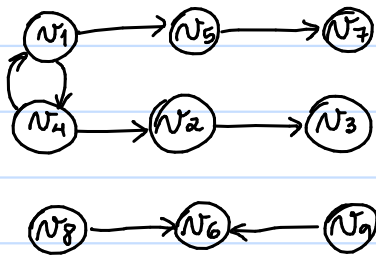
TEOREMA DO APERTO DE MÃOS: Para todo grafo G vale que

$$\sum_{v \in V(G)} d(v) = 2|E(G)|.$$

Digrafos

DEFINIÇÃO: Um **digrafo** D é um par de conjuntos (V, E) onde V é um conj. de elementos chamados **vértices** e E é um conj. de elem. chamados de **arcos**, sendo que cada arco é um par ordenado de vértices.

→ Exemplo: $H = (V, E)$ com $V = \{v_1, v_2, \dots, v_9\}$ e $E = \{(v_1, v_5), (v_2, v_3), (v_1, v_4), (v_4, v_2), (v_4, v_1), (v_5, v_7), (v_8, v_6), (v_9, v_6)\}$.



Obs: essa definição é de **digrafo simples**

Obs 2: $(x, y) \neq (y, x)$. $E = \{v_1, v_5, v_2, v_3, \dots\}$

Obs 3: Usaremos $V(\cdot)$ e $E(\cdot)$

→ Todo grafo é um digrafo (pensamos em $\{x, y\}$ como (x, y) e (y, x)).

Terminologias básicas em digrafos

→ Se $e = uv$ é um arco de um digrafo D , então:

↳ u é **cabeca** e v é **cauda**

↳ e **sai** de u e **entra** em v

→ Dado $u \in V(D)$:

↳ o **grau de entrada** de u é o nº de arcos que entram em u ($d_D^-(u)$) e o **grau de saída** é o nº de arcos que saem de u ($d_D^+(u)$)

↳ a **vizinhança de entrada** de u é o conj. de vértices extremos dos arcos que entram em u ($N_D^-(u)$) e a **vizinhança de saída** é o conj. de vértices extremos dos arcos que saem de u , exceto por u ($N_D^+(u)$).

TEOREMA DO APERTO DE MÃOS: Para todo digrafo D vale que

$$\sum_{v \in V(G)} d^+(v) = \sum_{v \in V(G)} d^-(v) = |E(G)|.$$

Grafos e digrafos ponderados

→ Podemos associar valores (pesos) a vértices e/ou arestas dos (di)grafos.

$$\hookrightarrow w: V(G) \rightarrow \mathcal{N}$$

$$\hookrightarrow w: E(G) \rightarrow \mathcal{N}$$

Subgrafos

→ Dados grafos H e G , H é **subgrafo** de G , denotado $H \subseteq G$, se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$.

Parseios

→ Uma sequência de vértices $X = (v_0, v_1, \dots, v_k)$ é um **parseio** de um grafo G se $v_i \in V(G)$ para todo $0 \leq i \leq k$ e $v_i v_{i+1} \in E(G)$ para todo $0 \leq i \leq k-1$.

→ **comprimento** = n.º de arestas

→ **aberto**: se $v_0 \neq v_k$

fechado: se $v_0 = v_k$.

v_0 e v_k são **extremos** e v_1, \dots, v_{k-1} são **vértices internos**.

→ **caminho**: parseio que não repete vértices.

ciclo: parseio fechado que não repete vértices internos.

→ **uv -caminho**: caminho de u até v

Classes de grafos

→ **completos**: grafos K_m com $V(K_m) = \{v_1, \dots, v_m\}$

$$\text{e } E(K_m) = \{xy : x, y \in V(K_m)\}$$

$$\hookrightarrow |E(K_m)| = \binom{m}{2} = \frac{m(m-1)}{2}$$

→ Fixado número n de vértices, todo grafo com n vértices é subgrafo de um grafo completo

$$\text{Para todo } G, \quad |E(G)| \leq \frac{n(n-1)}{2}.$$

Conexidade em grafos

→ Um grafo G é **conexo** se existe uv -caminho para todo $u, v \in V(G)$.
Caso contrário, G é **desconexo** e consiste de vários **componentes conexos**, que são subgrafos conexos maximais.

Conexidade em digrafos

→ Um digrafo D é **fortemente conexo** se existe uv -caminho para todo $u, v \in V(G)$.
Caso contrário, D consiste de vários **componentes fortemente conexos**, que são subdigrafos fortemente conexos maximais.

Distância em grafos / digrafos

→ Dados um (di) grafo G e dois vértices $u, v \in V(G)$:

$$\text{dist}_G(u, v) = \begin{cases} \text{comprimento de um } uv\text{-caminho de menor compr.} & \text{se há } uv\text{-cam.} \\ \infty & \text{c.c.} \end{cases}$$

Distância em grafos / digrafos ponderados

→ Dados um (di) grafo G , uma função $w: E(G) \rightarrow \mathbb{R}$ e dois vértices $u, v \in V(G)$:

$$\text{dist}_G^w(u, v) = \begin{cases} \text{peso de um } uv\text{-caminho de menor peso} & \text{se há } uv\text{-cam.} \\ \infty & \text{c.c.} \end{cases}$$

Pseudocódigos e análise de algoritmos em grafos

→ Vamos resolver o problema de calcular o grau de um vértice.

GRAU(G, v)
Devolve $d_G(v)$

ou

GRAU(G, v)
grau = 0
Para cada $u \in N_G(v)$
 grau = grau + 1
Devolve grau

"Realidade":

GRAU(M, m, v)
grau = 0
Para $u = 1$ até $|V(G)|$
 Se $M[v][u] = 1$
 grau = grau + 1
Devolve grau

$\Theta(m)$

GRAU(L, m, v)
grau = 0
atual = $L[v]$
Enquanto atual \neq NULL
 grau = grau + 1
 atual = atual.prox
Devolve grau

$\Theta(d(v)), \Theta(m)$

→ Vamos agora encontrar o grau máximo de um grafo ($\Delta(G)$)

GRAU_MAXIMO(G)
max = 0
Para cada $v \in V(G)$
 Se $d_G(v) > \text{max}$
 max = $d_G(v)$
Devolve max

→ $\Theta(m)$

→ $\Theta(m)$ ou $\Theta(d(v))$

$$\sum_{v \in V(G)} \Theta(m) = \Theta(m^2)$$

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

MATRIZ: $\Theta(m^2)$

LISTAS: $\Theta(m + m)$

Árvores

→ Se há n componentes conexas com um vértice cada, qual o menor número de arestas a serem acrescentadas para ficar com uma única componente conexa?

↳ $n - 1$

→ **Árvore**: grafo conexo e sem ciclos.

↳ **folha**: vértices de grau 1.

→ **floresta**: grafo que não contém ciclos.

→ Podemos enraizar uma árvore em qualquer um de seus vértices pois não há vértices especiais

TEOREMA: Seja G um grafo. As seguintes afirmações são equivalentes:

1) G é uma árvore

2) Existe um único caminho entre quaisquer dois vértices de G .

3) G é conexo e para toda $e \in E(G)$, $G - e$ é desconexo.

4) G é conexo e $|E(G)| = |V(G)| - 1$.

5) G não tem ciclos e $|E(G)| = |V(G)| - 1$.

6) G não tem ciclos e para todo par $x, y \in V(G)$ com $xy \notin E(G)$, $G + xy$ tem exatamente um ciclo.

Consequências do teorema

→ Se T é árvore e $xy \in E(T)$ com $x, y \in V(T)$, então $T+xy$ tem exatamente um ciclo.

Se $f \in E(T)$ pertence ao ciclo, então $T+xy-f$ é uma árvore.

→ Se $T \subseteq G$ é árvore e $V(T) = V(G)$, então G é conexo.

→ Se $T \subseteq G$ é árvore, $V(T) \neq V(G)$ e existe $xy \in E(G)$ com $x \in V(T)$ e $y \in V(G) \setminus V(T)$, então $T+xy$ é árvore.

Árvore Geradora

→ $T \subseteq G$ é árvore geradora se T é árvore e $V(T) = V(G)$.

Algoritmo genérico de busca (árvore implícita)

BUSCA(G, s)

```
1  $s$ .visitado = 1
2 enquanto houver aresta com um extremo visitado e outro não faça
3   Seja  $xy$  uma aresta com  $x$ .visitado == 1 e  $y$ .visitado == 0
4    $y$ .visitado = 1
5    $y$ .pred =  $x$ 
```

CHAMABUSCA(G)

```
1 para todo vértice  $v \in V(G)$  faça
2    $v$ .visitado = 0
3    $v$ .pred = null
4 seja  $s \in V(G)$  qualquer
5 BUSCA( $G, s$ )
```

Invariante: no início da t -ésima iteração, v .visitado = 1 sse existe sv -caminho.

Consequências do algoritmo de busca

Após a execução de Busca(G, s):

- v .visitado = 1 se e somente se há sv -caminho em G
- v .visitado = 1 se e somente se v e s estão na mesma componente conexa
- podemos construir um sv -caminho:
(s, \dots, v .predcessor, v)
(s, \dots, v .predcessor.predcessor, v .predcessor, v)
(outros sv -caminhos podem existir).

Construindo um s-v-caminho

CONSTROICAMINHO(G, s, v)

```
1 seja  $L$  uma lista vazia
2 se  $v.visitado == 0$  então
3   | devolve  $L$ 
4  $atual = v$ 
5 enquanto  $atual \neq s$  faça
6   | INSERENOINICIOLISTA( $L, atual$ )
7   |  $atual = atual.pred$ 
8 INSERENOINICIOLISTA( $L, s$ )
9 devolve  $L$ 
```

Como responder a árvore?

BUSCA(G, s)

```
1  $s.visitado = 1$ 
2 enquanto houver aresta com um extremo visitado e outro não faça
3   | Seja  $xy$  uma aresta com  $x.visitado == 1$  e  $y.visitado == 0$ 
4   |  $y.visitado = 1$ 
5   |  $y.pred = x$ 
```


Algoritmo - BFS

BUSCALARGURADISTANCIA(G, s)

1 s .visitado = 1

2 s .dist = 0

3 cria fila vazia F

4 ENFILEIRA(F, s)

5 enquanto F .tamanho > 0 faça

6 $u =$ DESENFILEIRA(F)

7 para todo vértice $v \in N(u)$ faça

8 se v .visitado == 0 então

9 v .visitado = 1

10 v .dist = u .dist + 1

11 v .pred = u

12 ENFILEIRA(F, v)

$m_s = m^e$ vértices na comp. de s

$m_s = m^e$ arestas na comp. de s

$m = |V(G)|$ $m = |E(G)|$

$V_s(G) = \text{conj. vértices na comp. de } s$

m_{s+1} } pois uma vez visitado, não
 m_s } "desvisitamos" \therefore cada v passa só
uma vez pela linha 7

MATRIZ = $\theta(m)$ p/ cada u }
LISTA = $\theta(d(u))$ p/ cada u } *

$\sum_{u \in V_s(G)} * = \theta(m m_s)$
 $\theta(m m_s)$

TOTAL MATRIZ: $\theta(m_s) + \theta(m \cdot m_s) = \theta(m \cdot m_s) = O(m^2)$ pois $m \geq m_s$.

TOTAL LISTAS: $\theta(m_s) + \theta(m_s) = \theta(m_s + m_s) = O(n + m)$ pois $m \geq m_s$,
 $m \geq m_s$.

Por fim, note que a árvore T tal que

$$V(T) = \{v \in V(G) : v.\text{predecessor} \neq \text{null}\} \cup \{s\}$$

$$E(T) = \{\{v.\text{predecessor}, v\} : v \in V(T) \setminus \{s\}\}$$

é uma árvore geradora de G , contém um único sv -caminho para qualquer $v \in V(T)$ e é chamada de *árvore de busca em largura*.

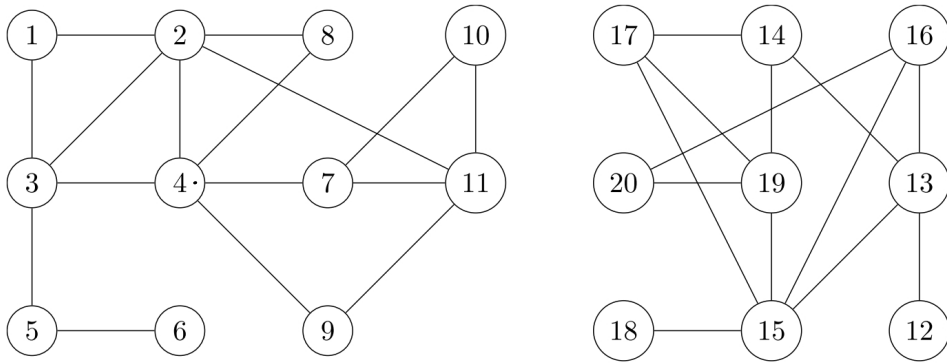
LEMA: Sejam G um grafo e $s \in V(G)$. Na execução de BUSCALARGURADISTANCIA(G, s), se u e v são dois vértices que estão na fila e u entrou antes de v na fila, então $u.\text{dist} \leq v.\text{dist} \leq u.\text{dist} + 1$

TEOREMA: Sejam G um grafo e $s \in V(G)$. Ao fim de BUSCALARGURADISTANCIA(G, s), para todo $v \in V(G)$ vale que $v.\text{dist} = \text{dist}_G(s, v)$.

Busca em profundidade - DFS (Depth-first search)

→ Explora a vizinhança dos vértices já visitados na ordem "último a entrar, primeiro a sair".

Exemplo de execução da DFS ($s = 7$)



PILHA : ()

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
visitado	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
predecessor																				

BUSCAPROFRECURSIVA(G, s)

1 $s.visitado = 1$

2 para todo vértice $v \in N(s)$ faça

3 se $v.visitado == 0$ então

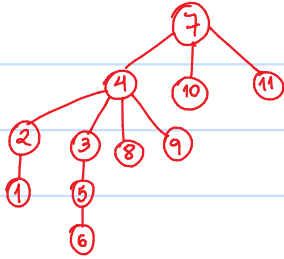
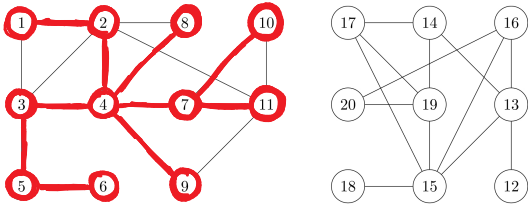
4 $v.pred = s$

5 BUSCAPROFRECURSIVA(G, v)

→ $\Theta(m)$ ou $\Theta(d(s))$

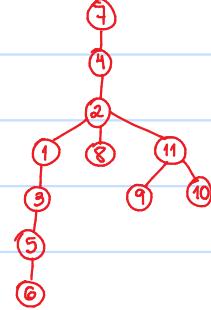
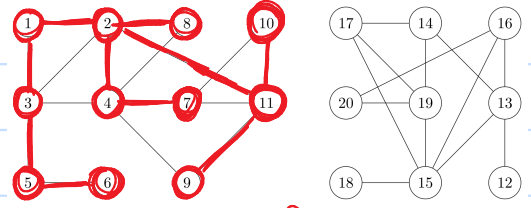
→ uma para cada vértice visitado (m_s)

Resultado de BuscaLargura (G,7):



- caminhos mínimos
- grafos bipartidos

Resultado de BuscaProfundidade (G,7):



- ciclos
- caminhos
- componentes

- arestas de corte
- labirintos

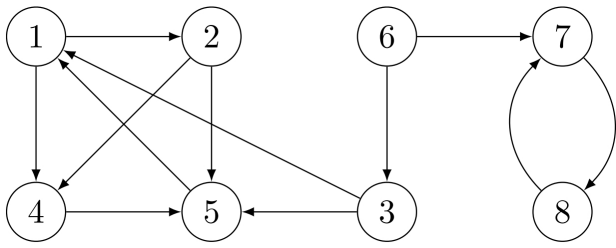
Componentes conexas

BUSCACOMPONENTES(G)

```
1 para todo vértice  $v \in V(G)$  faça
2    $v.visitado = 0$ 
3    $v.pred = null$ 
4  $qtdComponentes = 0$ 
5 para todo vértice  $s \in V(G)$  faça
6   se  $s.visitado == 0$  então
7      $s.visitado = 1$ 
8      $s.componente = s$ 
9      $qtdComponentes = qtdComponentes + 1$ 
10    BUSCA( $G, s$ )
11 devolve  $qtdComponentes$ 
```

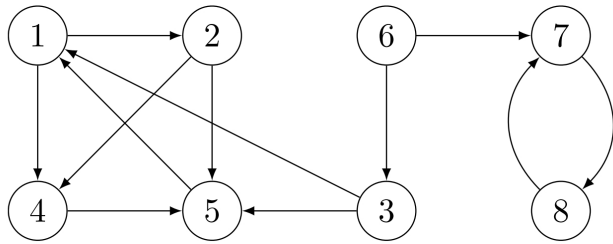
- Atualizar nos buscas:
 $nr.componente = u.componente$
se v for visitado por
cousa de u

Exemplo de execução - BFS e DFS em digrafos



BFS (D, 3)

FILA: ()



DFS (D, 3)

PILHA: ()

"Problemas" dos buscas em digrafos

→ Não obtemos sempre uma arborescência geradora, mesmo que uma exista.

→ Os vértices visitados não fazem parte de uma componente conexa do grafo subjacente e nem de uma componente fortemente conexa.

Boas notícias

→ A busca em largura ainda encontra distâncias

→ A busca em profundidade, com poucas alterações, encontra componentes fortemente conexos.