① 13ª Aula

Algoritmos básicos em grafos

Definição: Um grafo consiste de um conjunto V de vértices (finito) e um conjunto E de arestas, $E \subseteq V \times V$.

$G = (V, E)$

- Problema famoso: "Sete pontes de Königsberg"

②

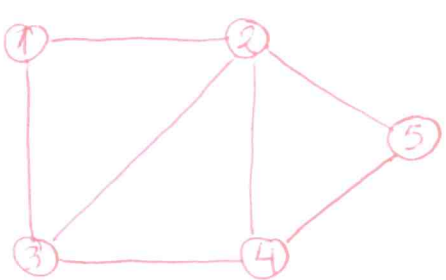
→ Um grafo pode ser representado graficamente

Exemplo:

$V = \{1, 2, 3, 4, 5\}$

$E = \{12, 13, 23, 24, 25, 34, 45\}$

G_1



③

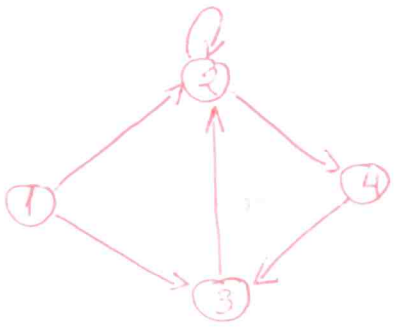
→ Um grafo pode ser orientado

Exemplo:

$V = \{1, 2, 3, 4\}$

$E = \{\vec{12}, \vec{13}, \vec{32}, \vec{24}, \vec{43}, \vec{22}\}$

G_2



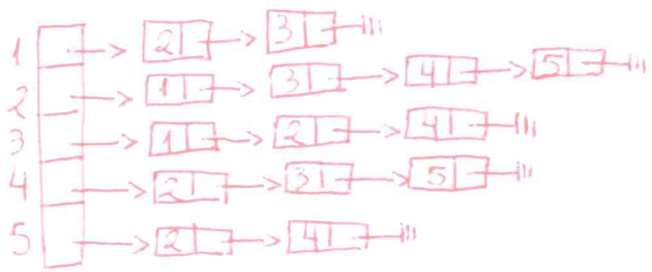
Em muitas aplicações, pode-se ter pesos nos vértices ou nos arestas.

④ Representações de grafos

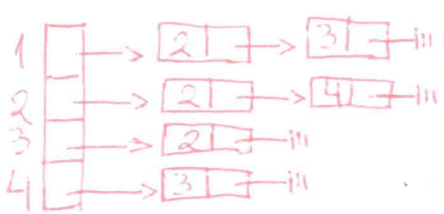
→ Lista de adjacência: "para cada vértice existe uma lista ligada dos vértices adjacentes"

Exemplos:

G_1



G_2



Espaço:

$\underbrace{n + 2m}$

$\Theta(n+m)$

$n = |V|$ e $m = |E|$

$\underbrace{n + n}$

$\Theta(n+m)$

(5)

→ Matriz de adjacência: "uma matriz indexada por $V \times V$ que tem 1 na posição uv se e somente se u e v são adjacentes, e 0 caso contrário"

Exemplos:

G_1:

i \ j	1	2	3	4	5
1	0	1	1	0	0
2	1	0	1	1	1
3	1	1	0	1	0
4	0	1	1	0	1
5	0	1	0	1	0

Matriz simétrica

G_2:

	1	2	3	4
1	0	1	1	0
2	0	1	0	1
3	0	1	0	0
4	0	0	1	0

Espaço: $\Theta(n^2)$

(6) Busca em largura

→ s é um vértice de $G = (V, E)$

→ A busca em largura percorre as arestas de G, descobrindo os vértices que são alcançáveis a partir de s

→ Distância = Nº de arestas

→ Determina um caminho mínimo de s até um vértice t de G

(7)

→ Antes de encontrar um vértice de distância K+1 de s, todos os vértices de distância K são encontrados

→ Algoritmo utiliza uma "fila" para os vértices cinza

→ Cores dos vértices
* Branco: "não descobertos"
* Cinza: "fronteira dos descobertos"
* Preto: "descobertos"

(8) BFS(G, s)

O(n) { para cada vértice u em V(G)\{s} faça
    cor(u) ← branco
    d(u) ← ∞
    π(u) ← null

O(1) { cor(s) ← cinza
    d(s) ← 0
    π(s) ← null
    Q ← {s}

    enquanto Q ≠ ∅ faça
        u ← primeiro(Q)
O(m) → para cada v em Adj(u) faça
nototal   se cor(v) = branco
                [insere(v, Q)
                [d(v) ← d(v)+1
                [π(v) ← u

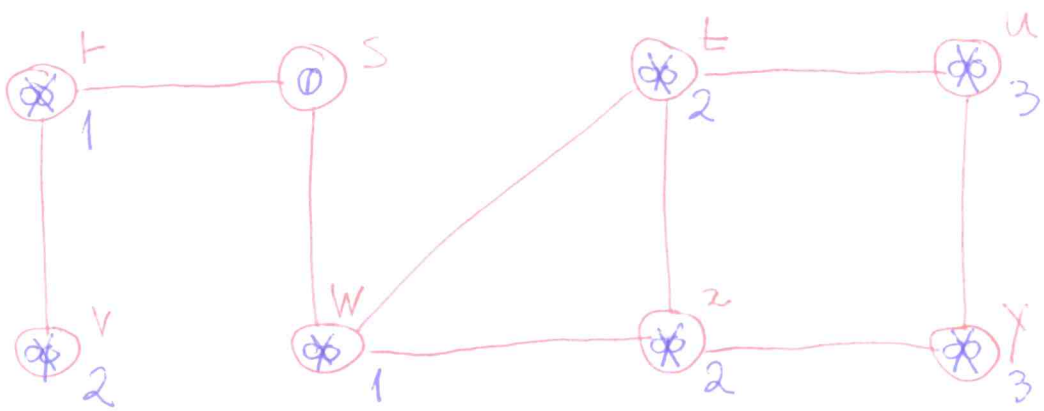
(9)

```
        remove-primeiro(Q)
        cor(u) ← preto
    }
}
```

→ Note que cor, d e π são atributos dos vértices

→ Complexidade do BFS: $O(n+m)$

(10) Exemplo



Q	~~r~~	~~s~~	~~v~~	~~w~~	~~t~~	~~x~~	~~u~~	~~y~~
d	0	1	1	2	2	2	3	3
π								

(11) # Busca em profundidade

→ Ideia: "prossegue sempre a partir do vértice descoberto mais recente, até que este não tenha mais vizinhos descobertos"

→ Vértices recebem dois rótulos (timestamp)
  * cinza: "quando é descoberto"
  * preto: "depois que todos os vizinhos são visitados"

→ Vetores d( ) e f( ) armazenam os rótulos

(12)

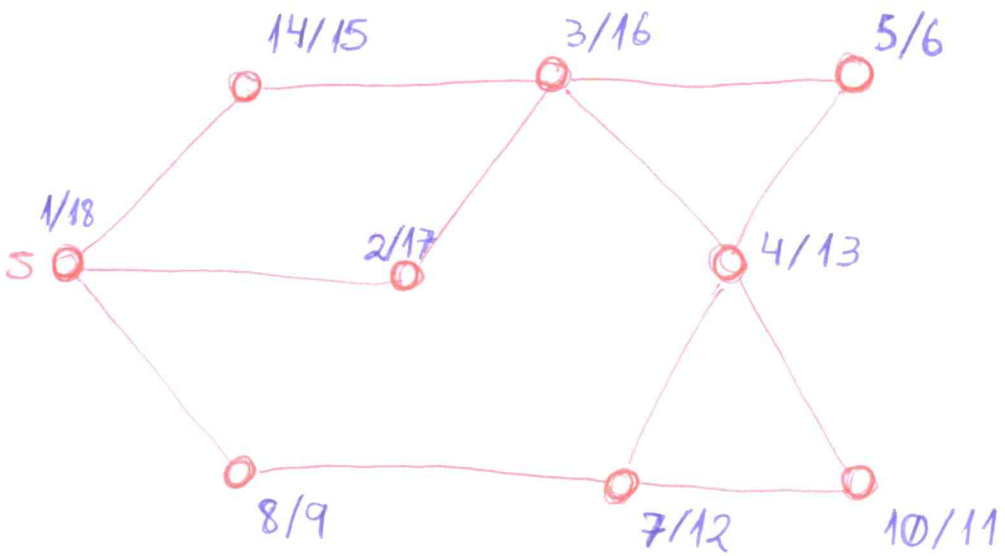
DFS(G){
O(n) → para cada vértice u em V(G) faça
  cor(u) ← branco
  π(u) ← null
tempo ← 0
O(n) → para cada vértice u em V(G) faça
  se cor(u) = branco
    ? ——→ DFS-Visit(u)
O(n) }

(13)

```
DFS-Visit(u){
cor(u) ← cinza // acabou de ser descoberto
tempo ← tempo + 1
d(u) ← tempo
O(m) → para cada v em Adj(u) faça
        se cor(v) = branco
            π(v) ← u
            DFS-Visit(v)
cor(u) ← preto // todos os vizinhos foram visitados
tempo ← tempo + 1
f(u) ← tempo
}
```

Complexidade: $O(n+m)$

(14) Exemplo



tempo: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18