

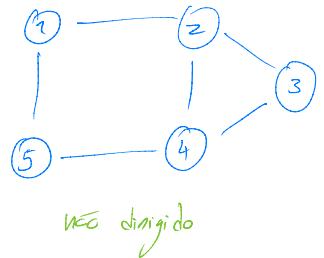
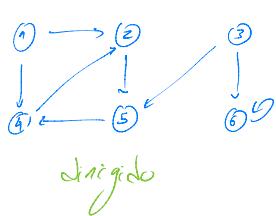
## Aula 4

- Grafos
  - Representação
  - Busca em Profundidade Primino (DFS)
    - Classificação das arcos
    - Grafos Aciclicos - Propriedades
  - Ordenação Topológica

(7)

### Definição 1 [Grafo]

Um grafo  $G$  é um par  $(V, E)$  constituído por um conjunto de vértices  $V$  e um conjunto de arestas/arcos  $E$ , onde  $E \subseteq V \times V$ .



- Dizemos que um grafo é dirigido se as arestas têm sentido  $(u, v) \neq (v, u)$

Dizemos que um grafo é não dirigido no caso contrário:  $(u, v) \sim (v, u)$

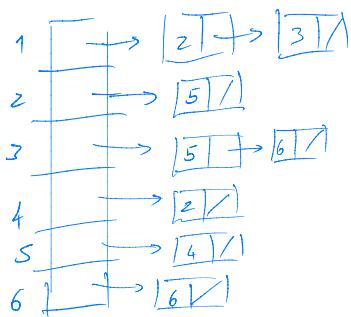
- Dizemos q um grafo é espesso quando o n° de arestas do grafo é "muito inferior" a  $|V| \times |V|$ . De contrário, o grafo diz-se denso.

Grafo espesso -  $|E| = O(|V|)$

Grafo denso -  $|E| = \Theta(|V|^2)$

- Representações de grafos

- Lista de Adjacências



$|E| + |V|$  - grafos dirigidos

$\geq |E| + |V|$  - grafos não dirigidos  
 $O(|E| + |V|)$

Se o grafo for espesso:  $O(|V|)$

Perguntas:

- \* Algoritmo para verificar se  $v \in N(u)$

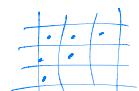
- Matriz de Adjacências

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Grafo dirigido:  $|V|^2$

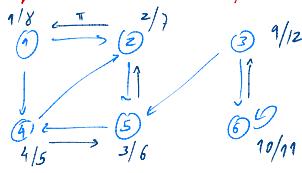
Grafo não dirigido:  $\frac{|V| \times (|V| + 1)}{2}$

$\Theta(|V|^2)$



$$\sum_{i=1}^n i = \frac{n(n+1)}{2} = \frac{3 \times 3}{2}$$

### Exemplo 1 [Procurando Profundidade Primeiro (Depth-first search, DFS)]



$$G_{\text{PI}} = (V, E_{\text{PI}}) \quad \text{Fluxo iniciado pelo DFS}$$

$$E_{\text{PI}} = \{(v, \pi, v) \mid v \in V \wedge v, \pi \neq N(v)\}$$

## Algoritmo 1 (DFS)

$\text{DFS}(G)$

```

for each  $v \in V[G]$ 
    v.color = white; v.PI = nil
    time = 0
loop1 |   for each  $v \in V[G]$ 
        if v.color == White
            time = DFS-visit(G, v, time)

```

DFS-Visit ( $G, v, \text{time}$ )

if ( $v \text{ color} \neq \text{White}$ ) error "..."

time = time + 1; v.d = time

v.color = Gray

for each  $m \in \text{Adj}(G, v)$

if ( $m \text{ color} == \text{white}$ )

$m.PI = v$

time = DFS-Visit( $G, m, \text{time}$ )

v.color = Black; time = time + 1; v.f = time

return time

Complexidade [Naive]

$O(|V| \times |E|)$

Complexidade [Tight Bound]

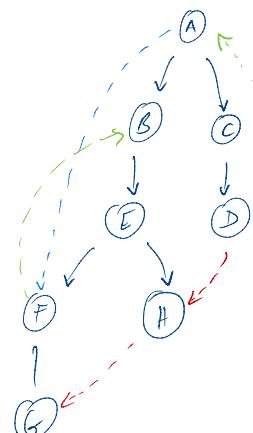
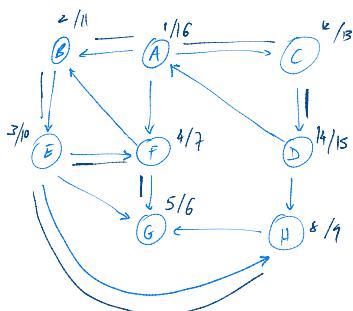
- DFS Visit é invocado uma vez por cada vértice do grafo

Assim sendo, o número de iterações da loop 2 é dado pelas expressões:

$$\sum_{v \in V} \text{Adj}(G, v) = O(|E|)$$

$$O(|V| + |E|)$$

## Exemplo 2 [DFS - classification of edges]



- Tree edges
- Forward edges
- Back edges
- Cross edges

### - tree edges (arcos da árvore)

Nós que são parte da árvore DFS

- Se  $(u, v)$  é tree edge então:

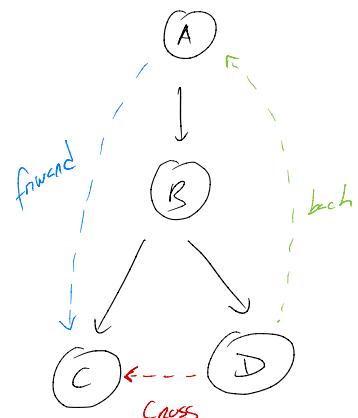
$$l_u \leq l_v \leq j_u \Leftrightarrow d_u < d_v < f_u < f_v$$

### - forward edge (arcos para a frente)

Nós que ligam um nó a um descendente nô filho na árvore DFS

- Se  $(u, v)$  é forward edge então:

$$l_u \leq l_v \leq j_u \Leftrightarrow d_u < d_v < f_u < f_v$$



### - back edge (arcos para trás)

Nós que ligam um nó a um nó não antecessor ou descendente na árvore DFS

- Se  $(u, v)$  é back edge então:

$$l_u \leq l_v \leq j_u \Leftrightarrow d_v < d_u < f_u < f_v$$

### - cross edge (arcos de cruzamento)

Nós que ligam um nó a um nó não antecessor ou descendente na árvore DFS

- Se  $(u, v)$  é cross edge:

$$l_u \leq l_v \leq j_u \Leftrightarrow d_v < f_u < d_u < f_v$$

### Teorema 1 [Parentesis]

Dada uma DFS num grafo  $G = (V, E)$ , para quaisquer dois vértices  $u \neq v$  em  $V$ , uma e apenas uma das seguintes proposições é verdadeira:

- $[f_u \ f_v]_u \wedge v$  é descendente de  $u$  na árvore DFS
- $[f_v \ f_u]_v \wedge u$  é descendente de  $v$  na árvore DFS
- $[f_u \ f_v]_u \wedge [f_v \ f_u]_v \quad | \quad u \neq v$  não estão relacionados
- $[f_v \ f_u]_v \wedge [f_u \ f_v]_u \quad | \quad u \neq v$  na árvore DFS

Portanto não existem casos do tipo

$$[f_u \ f_v]_u \wedge [f_v \ f_u]_v$$

#### Prova.

- Suponhamos  $f_u < f_v$ , há dois casos a considerar:

- $f_u < f_v \Rightarrow$  os intervalos não se interseccionam
- $f_v < f_u \Rightarrow v$  começou a ser explorado enquanto  $u$  ainda estava a ser explorado, pelo que  $u$  só vai acabar de ser explorado depois de  $v$  acabar de ser explorado.

Concluímos que:  $f_v < f_u$ .

- O caso em  $f_v < f_u$  é simétrico.

### Teorema 2 [Caminho Branca]

Dada uma DFS num grafo  $G = (V, E)$ , um vértice  $u$  é descendente de um vértice  $v$  na árvore DFS, se e apenas se no momento da em  $\tilde{g}$   $u$  é descoberto existe um caminho de vértices brancos a ligar  $u$  a  $v$ .

#### Prova.

$\boxed{\Rightarrow}$   $v$  é descendente de  $u$  na árvore DFS

- $d_u < d_v$
- Quando  $u$  é descoberto existe um caminho de vértices brancos a ligar  $u$  a  $v$

$\boxed{\Leftarrow}$  • Quando  $u$  é descoberto existe um caminho de vértices brancos a ligar  $u$  a  $v$ .

- Assumimos sem perda de generalidade  $\tilde{g}$  todos os vértices no caminho entre  $u$  e  $v$  são descendentes de  $u$  na árvore DFS excepto  $v$
- Seja  $w$  o predecessor de  $v$
- Como  $w$  é um descendente de  $u$  na árvore DFS, concluímos a partir do Teorema 1  $\tilde{g}$ :

$$d_u < d_v < f_w < f_u$$

- $w$  é um descendente de  $u$  na árvore DFS, mas  $v$  não é. A única forma de tal acontecer é termos:  $d_v < d_w$

$$d_u < d_v < d_w < f_w < f_u$$

- O Teorema dos parentesis implica  $\tilde{g} [f_v \ f_w]$  está intimamente contido em  $[f_u \ f_w]$  e  $\tilde{g} w$  é descendente de  $u$  na árvore DFS.

(4)

## Definição [ DAG - Directed Acyclic Graph - Grafo Dirigido Aciclico ]

Vm grafo dirigido  $G = (V, E)$  diz-se aciclico se n̄o contém caminhos circulares.

Um caminho circular p̄ um grafo  $G = (V, E)$  é uma sequência de vértices  $\langle v_0, \dots, v_n \rangle$  tal que:

- $\forall i < n. (v_i, v_{i+1}) \in E$
- $v_0 = v_n$

Lema 1: Um grafo dirigido tem um caminho circular se e só se num DFS no grafo nevera um arco para trás.

Prova:

• Bach edge  $\Rightarrow$  Caminho circular

- 
- Existem tree edges  $\{(v_i, v_{i+1}) | 0 \leq i < n\}$  q ligam  $v_0$  a  $v_n$
- Existe um arco para trás  $(v_n, v_0)$
- Concluímos q o grafo contém o caminho circular  $\langle v_0, \dots, v_n, v_0 \rangle$ .

• Caminho circular  $\Rightarrow$  Bach edge

- Seja  $\langle v_0, \dots, v_n, v_0 \rangle$  o caminho circular q começo no vértice  $v_0$  menor valor do tempo de descoberta
- Quando  $v_0$  é descoberto existe um caminho de vértices brancos q liga  $v_0$  a todos os vértices  $\{v_i\}_{i=1}^n$ , em particular ao vértice  $v_n$ .
- O Teorema do Comitite Branca garante q  $v_n$  é descendente de  $v_0$  na árvore DFS, donde se conclui q o arco  $(v_n, v_0)$  é m̄m bach edge.

Lema 2: Se  $G = (V, E)$  é um dg,  $\exists (u, v) \in E$ , ento  $v_f < u_f$ .

Prova: Um dg só tem 3 tipos de arco. Os 3 tipos de arco verificam esta propriedade.

Lema 3: Se  $G = (V, E)$  é um dg, ento  $G$  contém pelo menos um vértice fonte (source) e um vértice alvo (sink), ou seja um vértice dize-se source se n̄o tem arcos de chegada e dize-se alvo se n̄o tem arcos de saída (respectivamente incoming e outgoing).

Prova:

- Seja  $s$  o n̄o com maior tempo de fim.

$$\forall v \in V. v_f \leq s.f$$

$$\forall v \in V. [v]_s \sqsubseteq [v]_s \text{ ou } [s]_s \sqsubseteq [v]_s$$

Mais perto fonte um alvo de $v$ quem é prego sobre	Mais perto alvo um arco de $v$ / $s$ quem seria um back edge
--	---

$$[v]_s \sqsubseteq [s]_s \text{ ou } [v]_s \sqsubseteq [v]_s$$

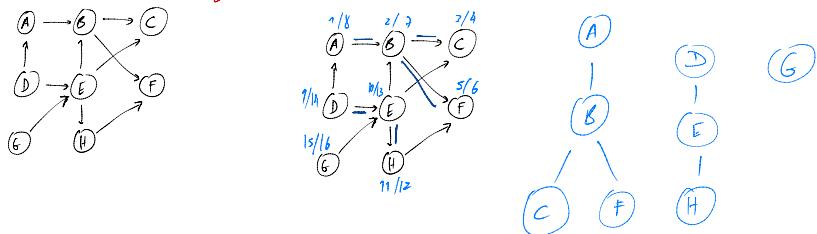
- Seja  $t$  o n̄o com menor tempo de fim

$$\forall v \in V. t_f \leq v_f$$

$$[v]_t \sqsubseteq [t]_t \text{ ou } [t]_t \sqsubseteq [v]_t$$

≤ mais perto fonte  
 alvo de  $t$  quem é back edge

## Exemplo 3 [Ordenação Topológica]



Ordenação Topológica: G, D, E, H, A, B, F, C

## Definição 2 [Ordenação Topológica]

Dado um dag  $G = (V, E)$ , uma ordenação topológica de  $G$  é uma sequência contendo todos os vértices de  $V$  tal que se  $(u, v) \in E$ ,  $u$  aparece antes de  $v$  na ordenação topológica de  $G$ .

## Algoritmo 2 [Topological Sort]

TopologicalSort( $G$ )

- Imprime DFS( $G$ )
  - Utiliza a DFS( $G$ )
  - Quando um vértice é terminado insere-o no final da lista ligada
  - Reforma a lista de vértices

## [Correção da Ordenação Topológica]

Basta juntarmos só o algoritmo é correcto basta juntarmos que se  $(u, v) \in E$  entra  $u.f > v.f$

- $(u, v)$  pode ser:
  - tree edge:  $u.f > v.f$
  - forward edge:  $u.f > v.f$
  - cross edge:  $u.f > v.f$