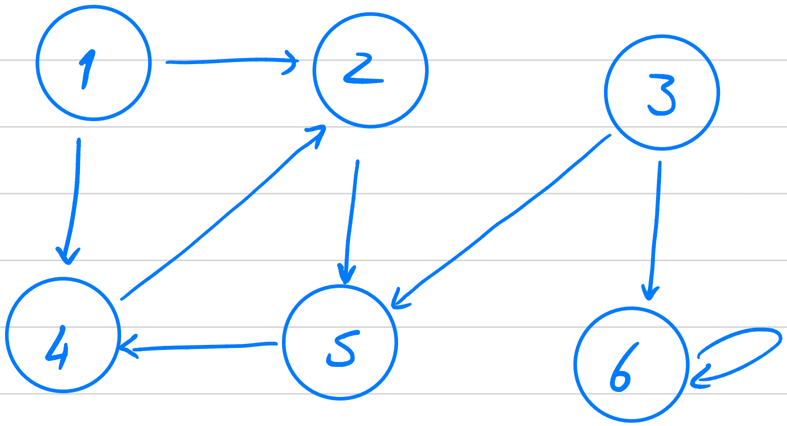
Aula 8

- Representação de Grafos
- DFS
  - Resultados Elementares
- Ordenação Topológica
- Componentes Fortemente Ligados

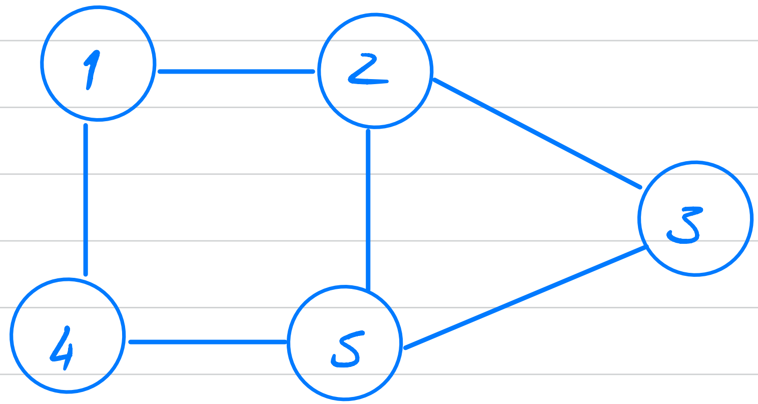
Grafos – Definições Elementares

Definição: Um grafo é um par $G = (V, E)$
onde:
- V – conjunto de vértices
- E – conjunto de arcos ($E \subseteq V \times V$)

- Um grafo diz-se:
  - Esparso: se $|E| \in O(|V|)$
  - Denso: se $|E| \in O(|V|^2)$

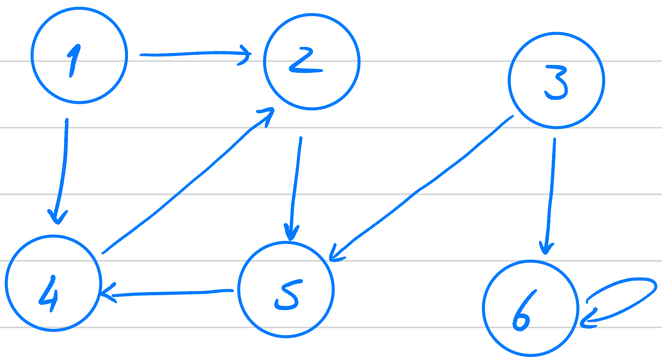


Dirigido: Os arcos têm sentido

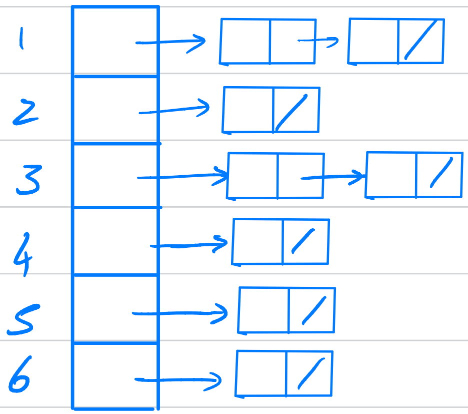


Não dirigido: Os arcos não têm sentido

Grafos - Representação



Lista de Adjacências



Espaço:

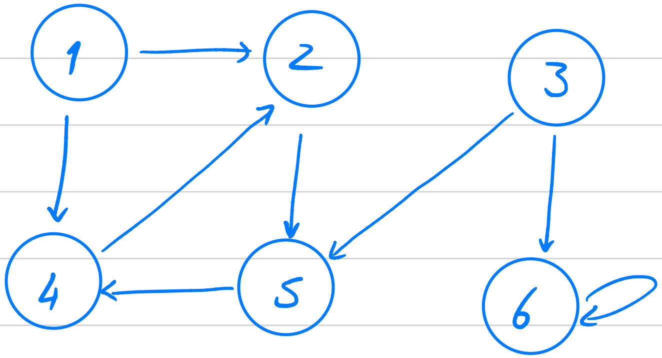
Grafos Esparsos:

Matriz de Adjacências

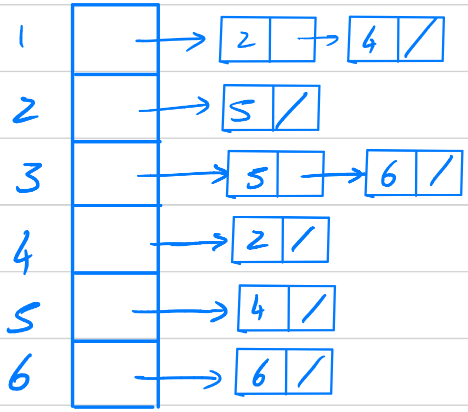
	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

Espaço:

Grafos - Representação



Lista de Adjacências



Espaço: $O(|V| + |E|)$

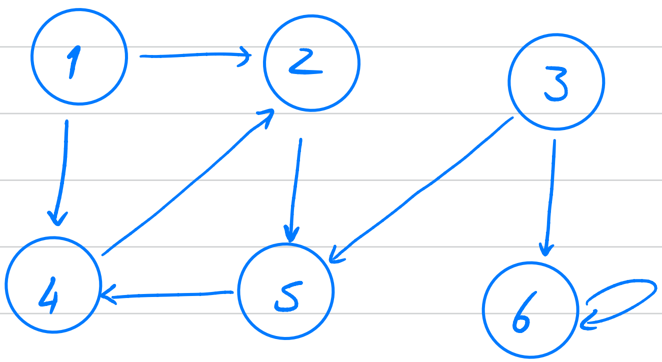
Grafos Esparsos: $O(|V|)$

Matriz de Adjacências

	1	2	3	4	5	6
1	0	1	0	1	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

Espaço: $O(|V|^2)$

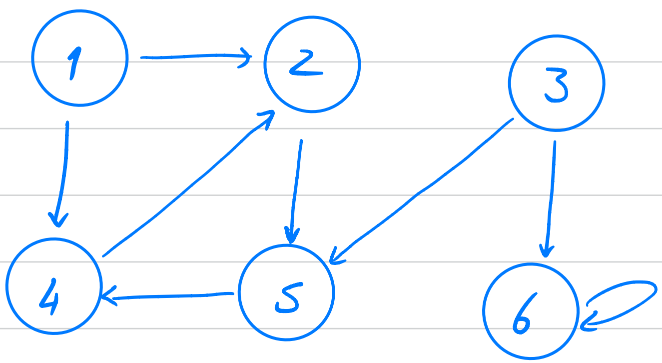
DFS (Depth-First-Search)



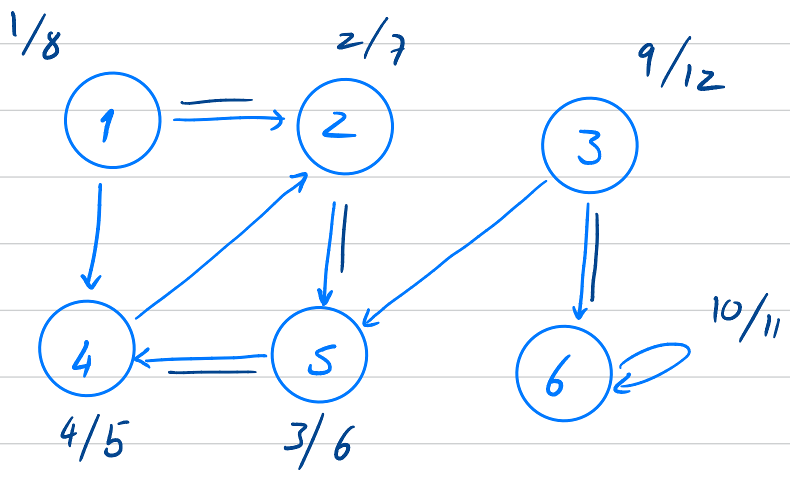
Floresta induzida por DFS

$G_\pi = (V, E_\pi)$

$E_\pi =$



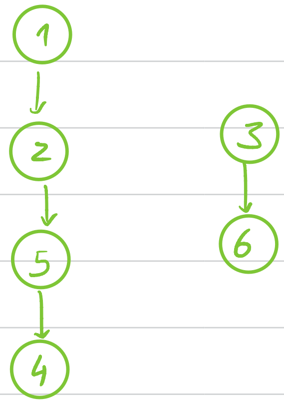
DFS (Depth-First-Search)

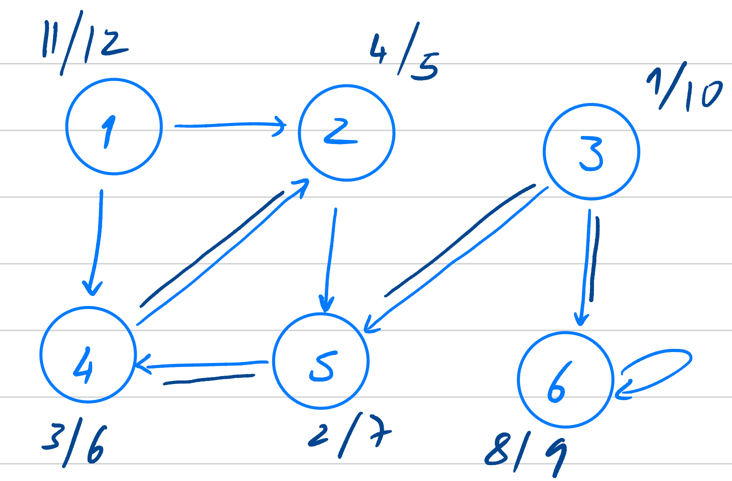


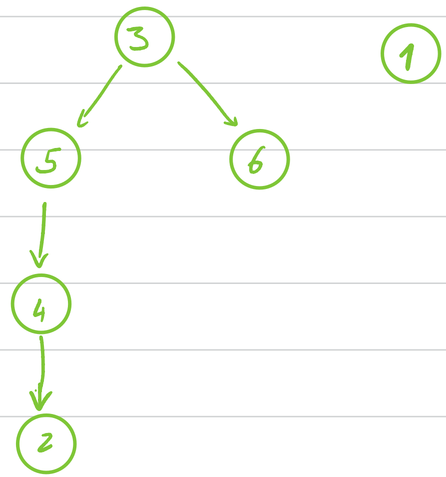
Floresta induzida por DFS

$$G_\pi = (V, E_\pi)$$

$$E_\pi = \{ (\pi[u], u) \mid \pi(u) \neq NIL \}$$







DFS (Depth-First-Search)

DFS(G)
for each $v \in G.V$ — Setup: cor e parent

time := 0;
for each $v \in G.V$ — Visitar cada nó
if

DFS-Visit(G, v, time)
assert(v.color == White)
Cor e tempo de descoberta

for each — Visitar os vizinhos de v

Cor e tempo de fim

return time

DFS (Depth-First-Search)

```
DFS(G)
  for each v ∈ G.V
      v.color := White ;  v.π = Nil
  time := 0;
  for each v ∈ G.V
      if v.color == White
          time := DFS-Visit(G, v, time)
```

Loop 1

Complexidade Naif:

Complexidade:

```
DFS-Visit(G, v, time)
   assert(v.color == White)
   v.color := Gray; time := time+1; v.d := time;
   for each u ∈ G.Adj[v]
       if u.color == White
           u.π := v
           time := DFS-Visit(G, u, time)
   v.color := Black; time := time+1; v.f := time;
   return time
```

Loop 2

DFS (Depth-First-Search)

```
DFS(G)
    for each v ∈ G.V
        v.color := White ;  v.π = Nil
    time := 0;
    for each v ∈ G.V                              ]
        if v.color == White                       ] Loop 1
            time := DFS-Visit(G, v, time)         ]

DFS-Visit(G, v, time)
    assert(v.color == White)
    v.color := Gray; time := time+1; v.d := time;
    for each u ∈ G.Adj[v]                         ]
        if u.color == White                       ]
            u.π := v                              ] Loop 2
            time := DFS-Visit(G, u, time)         ]
    v.color := Black; time := time+1; v.f := time;
    return time
```

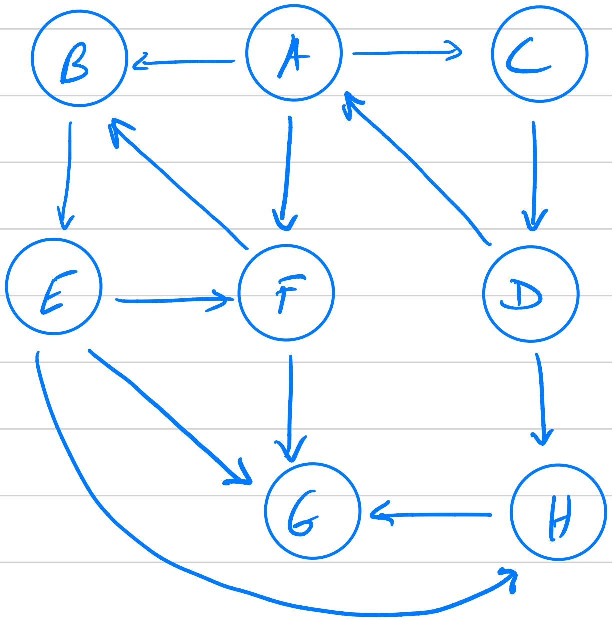
Complexidade Naif:

$O(|V| . |E|)$

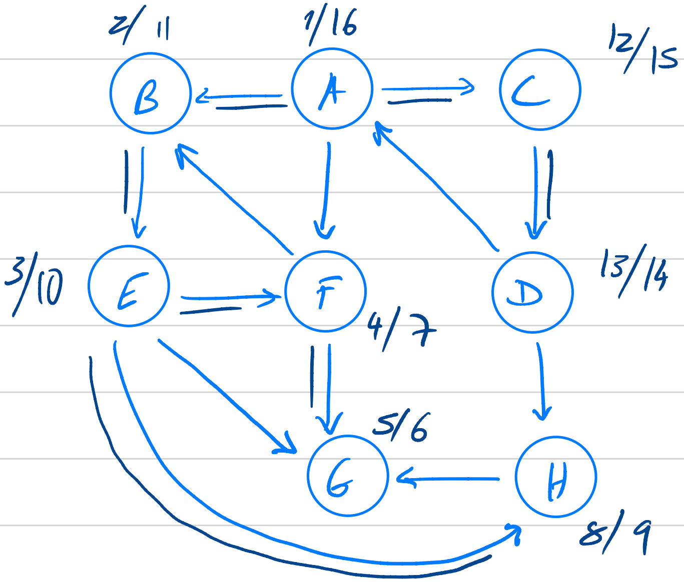
Complexidade:

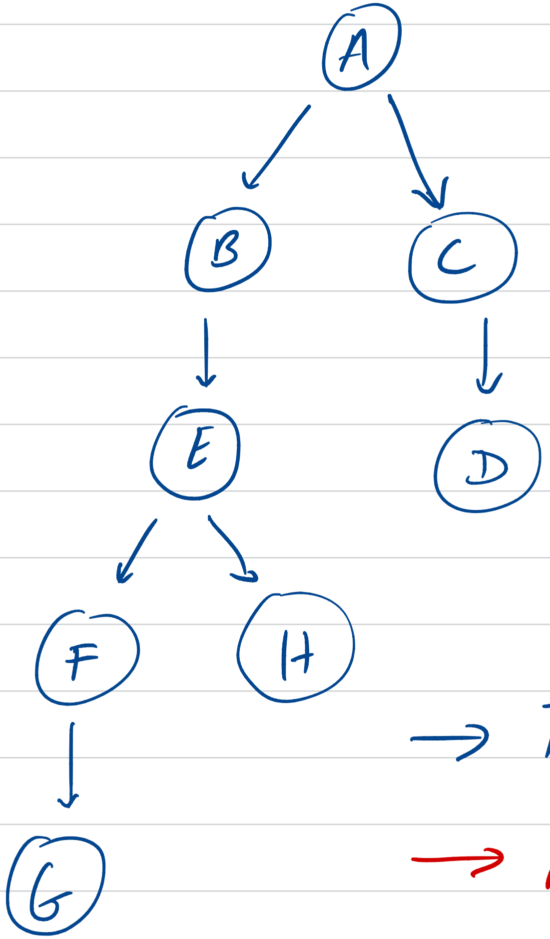
- DFS_Visit é invocado uma única vez por cada vértice do grafo
  $\Downarrow$
  O corpo do loop 2 é executado uma vez por cada arco do grafo
- $O(|E| + |V|)$

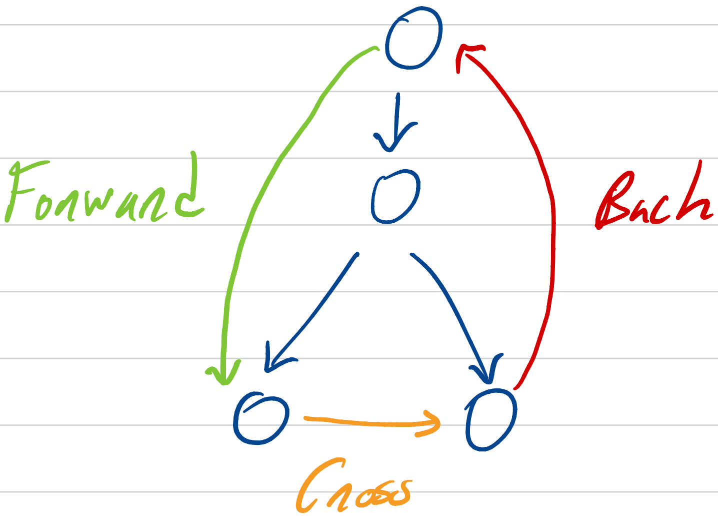
DFS - Classificação de Arcos



DFS - Classificação de Arcos







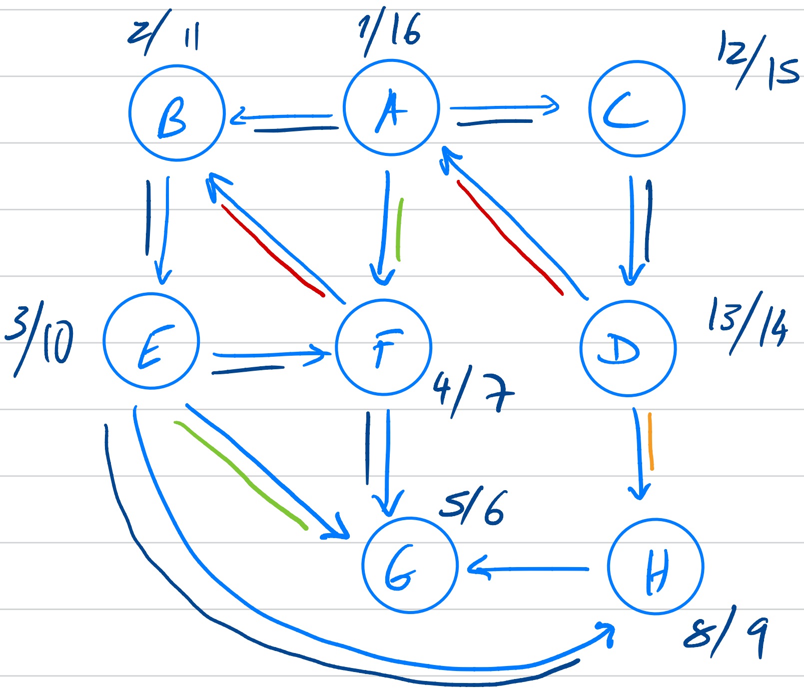
→ Tree edge

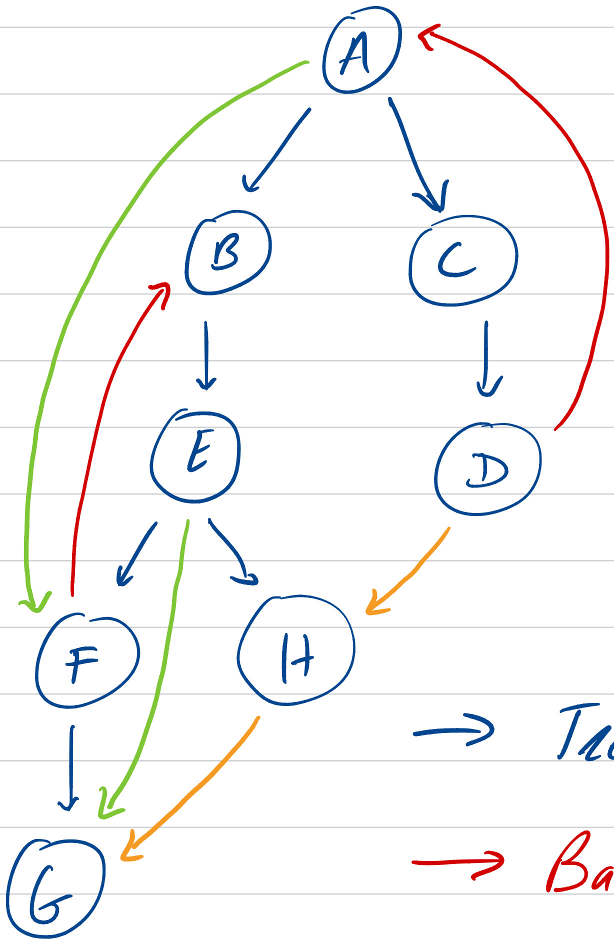
→ Back edge

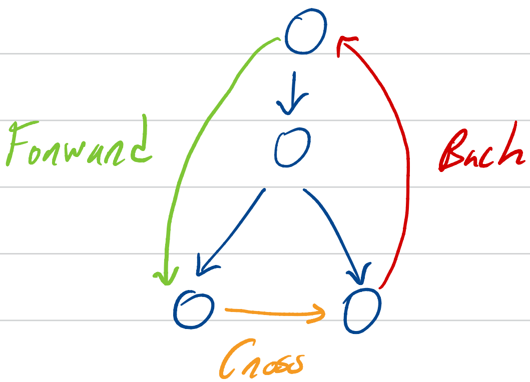
→ Cross edge

→ Forward edge

DFS - Classificação de Arcos







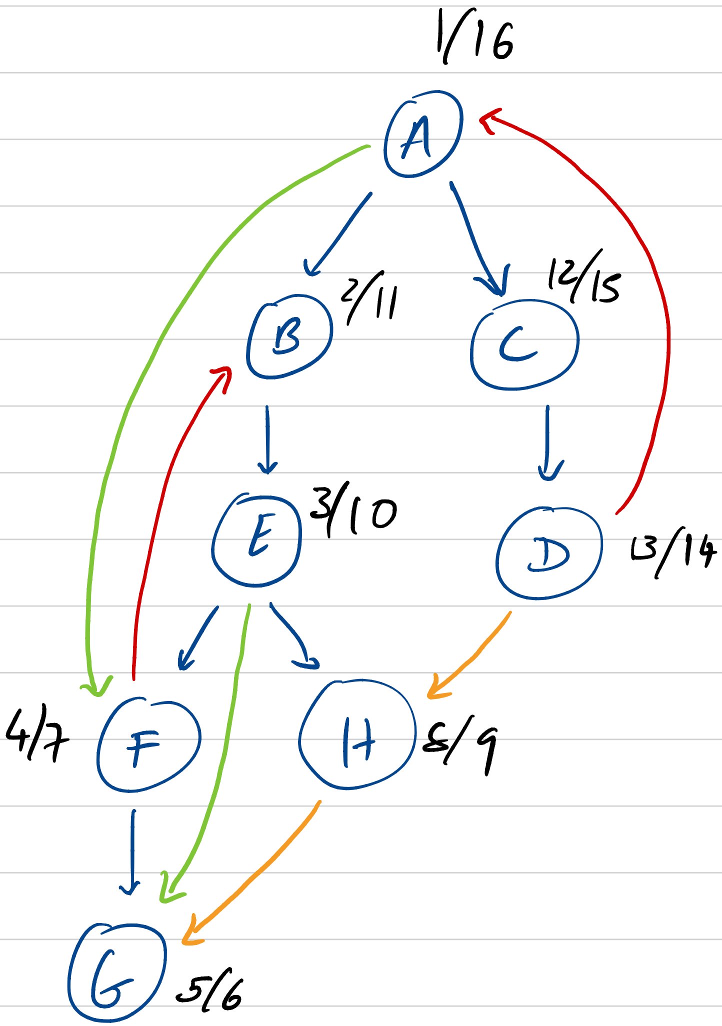
→ Tree edge

→ Back edge

→ Cross edge

→ Forward edge

DFS - Classificação de Arcos



→ Tree edge: (u, v)

$[_u\ [_v\ ]_v\ ]_u \iff d_u < d_v < f_v < f_u$

→ Back edge: (u, v)

$[_v\ [_u\ ]_u\ ]_v \iff d_v < d_u < f_u < f_v$

→ Cross edge: (u, v)

$[_v\ ]_v\ [_u\ ]_u \iff d_v < f_v < d_u < f_u$

→ Forward edge: (u, v)

$[_u\ [_v\ ]_v\ ]_u \iff d_u < d_v < f_v < f_u$

DFS - Propriedades

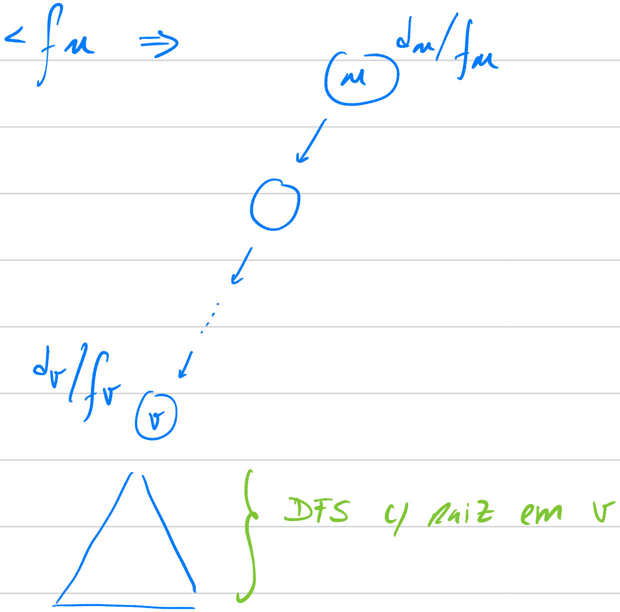
- Após uma DFS <u>nunca</u> temos: $[_u \; [_v \;]_u \;]_v$

* Suponhamos que $d_u < d_v$

  (I) $f_u < d_v \Rightarrow$ Os intervalos não se intersectam

  $[_u \;]_u \; [_v \;]_v$

  (II) $d_v < f_u \Rightarrow$



$[_u \; [_v \;]_v \;]_u$

DFS - Propriedades

Teorema do Caminho Branco

v é descendente de u na floresta DFS
sse no momento em que u é descoberto
existe um caminho branco a ligar u a v.

$\boxed{\Rightarrow}$ v é descendente de u

$\Rightarrow$ Existe um caminho branco entre u e v em d_u

$\boxed{\Leftarrow}$ Existe um caminho branco entre u e v

$\Rightarrow$ v é descendente de u na floresta DFS

· Suponhamos q v não é descendente de u

· Admitimos s/ perda de generalidade
q v é o primeiro vértice no caminho
branco q não é descendente de u

DFS - Propriedades

Teorema do Caminho Branco

v é descendente de u na floresta DFS
sse no momento em que u é descoberto
existe um caminho branco a ligar u a v.

$\boxed{\Rightarrow}$ v é descendente de u

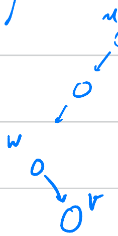
$\Rightarrow$ Existe um caminho branco entre u e v em d_u

$[_u \quad [_v \quad]_v \quad]_u$

$\boxed{\Leftarrow}$ Existe um caminho branco entre u e v

$\Rightarrow$ v é descendente de u na floresta DFS

· Suponhamos q v não é descendente de u



· Admitimos s/ perda de generalidade q v é o primeiro vértice no caminho branco q não é descendente de u

$[_u \quad [_v \quad]_v \quad [_w \quad]_w \quad]_v \quad]_u$

em ambos os casos v é descendente de u

DFS - Propriedades

Teorema do Caminho Branco

v é descendente de u na floresta DFS
sse no momento em que u é descoberto
existe um caminho branco a ligar u a v.

$\boxed{\Rightarrow}$ v é descendente de u

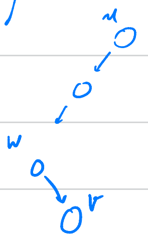
$\Rightarrow$ Existe um caminho branco entre u e v em d_u

$[_u \; [_v \;]_v \;]_u$

$\boxed{\Leftarrow}$ Existe um caminho branco entre u e v

$\Rightarrow$ v é descendente de u na floresta DFS

· Suponhamos q v não é descendente de u



· Admitimos s/ perda de generalidade
q v é o primeiro vértice no caminho
branco q não é descendente de u

$[_u \; [_v \;]_v \quad [_w \;]_w \;]_v \;]_u$

em ambos os casos v é
descendente de u

DFS - Propriedades

- Um grafo tem um caminho circular sse a DFS revela um arco para trás.

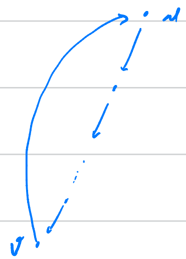
$\boxed{\Leftarrow}$ Back-edge $\Rightarrow$ Caminho circular

$\boxed{\Rightarrow}$ Caminho circular $\Rightarrow$ Back-edge

DFS - Propriedades

• Um grafo tem um caminho circular sse a DFS revela um arco para trás.

$\boxed{\Leftarrow}$ Back-edge $\Rightarrow$ Caminho circular



$\boxed{\Rightarrow}$ Caminho circular $\Rightarrow$ Back-edge

$\langle v_1, \ldots, v_n \rangle$ caminho circular

$v_1 = v_n$

$\Downarrow$ re-ordenamos os vértices do caminho circular para começar no vértice com menor tempo de descoberta

$\langle v_1', \ldots, v_n' \rangle$

$v_1' = v_n'$

• Qd v_1' é descoberto existe um caminho branco entre v_1' e v_{n-1}' $\Rightarrow$ Logo, v_{n-1}' é descendente de v_1' na floresta DFS $\Rightarrow$ (v_{n-1}', v_n') é arco para trás

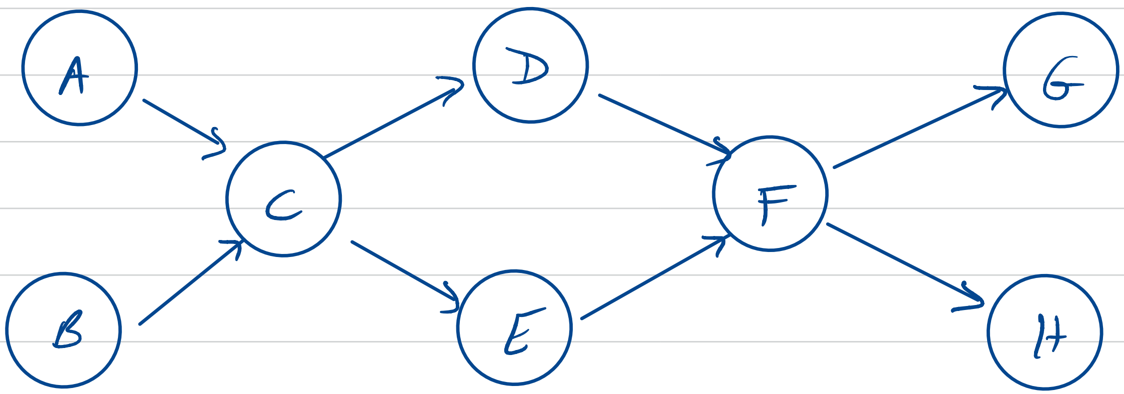
Definição [Grafo Dirigido Acíclico (Directed Acyclic Graph)]

Um grafo dirigido diz-se acíclico se não contém caminhos circulares (ciclos).

Um caminho circular num grafo $G = (V, E)$ é uma sequência de vértices $\langle v_1, \ldots, v_n \rangle$ tal que:

- $v_1 = v_n$
- $n > 1$
- $\forall_{1 \leq i \leq n-1}.\ (v_i, v_{i+1}) \in E$

Exemplo:



DAGs - Propriedades

Se $G = (V, E)$ é um DAG e $(u, v) \in E$, então $f_v < f_u$.

Prova

DAGs - Propriedades

Se $G = (V, E)$ é um DAG e $(u, v) \in E$, então $f_v < f_u$.

Prova

- Relações possíveis entre f_v e f_u
  - $[_u \ [_v \ ]_v \ ]_u \Rightarrow v$ é descendente de u
  - $[_v \ [_u \ ]_u \ ]_v$
    ↳ (u, v) é um arco para trás → Contradiz a hipótese de o grafo ser acíclico
  - $[_v \ ]_v \ [_u \ ]_u \Rightarrow (u, v)$ é arco de cruzamento
  - $[_u \ ]_u \ [_v \ ]_v$ ✗
    ↳ Não posso fechar u sem visitar todos os seus vizinhos brancos

Sources and Sinks

- Source: vértice que NÃO contém arcos de chegada (incoming edges)

- Sink: vértice que NÃO contém arcos de partida (outgoing edges)

Não esquecer:
Num DAG: $(u, v) \in E \Rightarrow f_u > f_v$

Proposição: Um DAG contém pelo menos um sink e uma source.

Prova:

- Sink:

- Source:

Sources and Sinks

- Source: vértice que NÃO contém arcos de chegada (incoming edges)

- Sink: vértice que NÃO contém arcos de partida (outgoing edges)

Não esquecer:
Num DAG: $(u,v) \in E \Rightarrow f_u > f_v$

Proposição: Um DAG contém pelo menos um sink e uma source.

Prova:

- Sink: Vértice com <u>menor</u> tempo de fim.

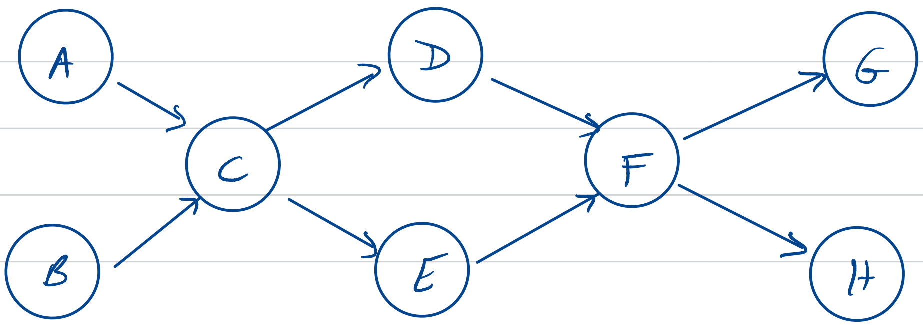
- Source: vértice com <u>maior</u> tempo de fim.
  - ↳ Seja u o vértice com maior tempo de fim. Suponhamos, por contradição, que existe w tal que:

$$w \rightarrow u \Rightarrow f_w > f_u$$

Definição [Ordenação Topológica]

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

Exemplo:



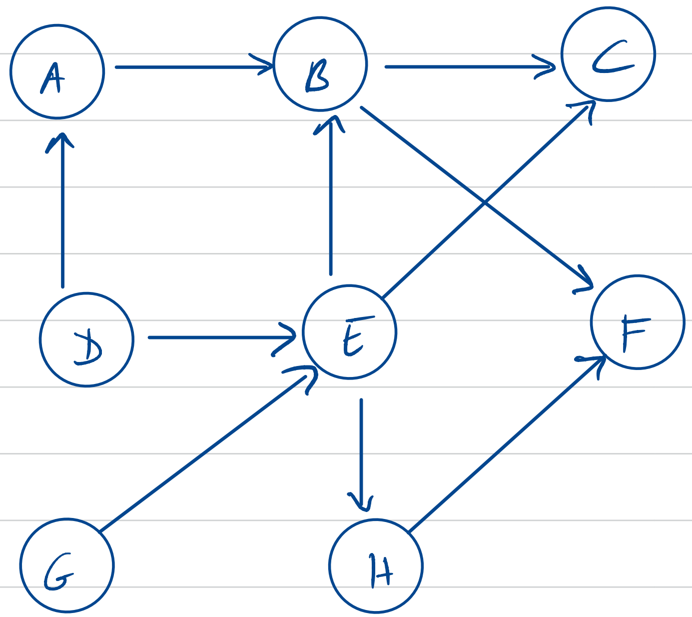
- Quantas ordenações topológicas admite o grafo?

$2 \times 2 \times 2 = 8$

$\langle A, B, C, D, E, F, G, H \rangle$

Ordenação Topológica - Algoritmo

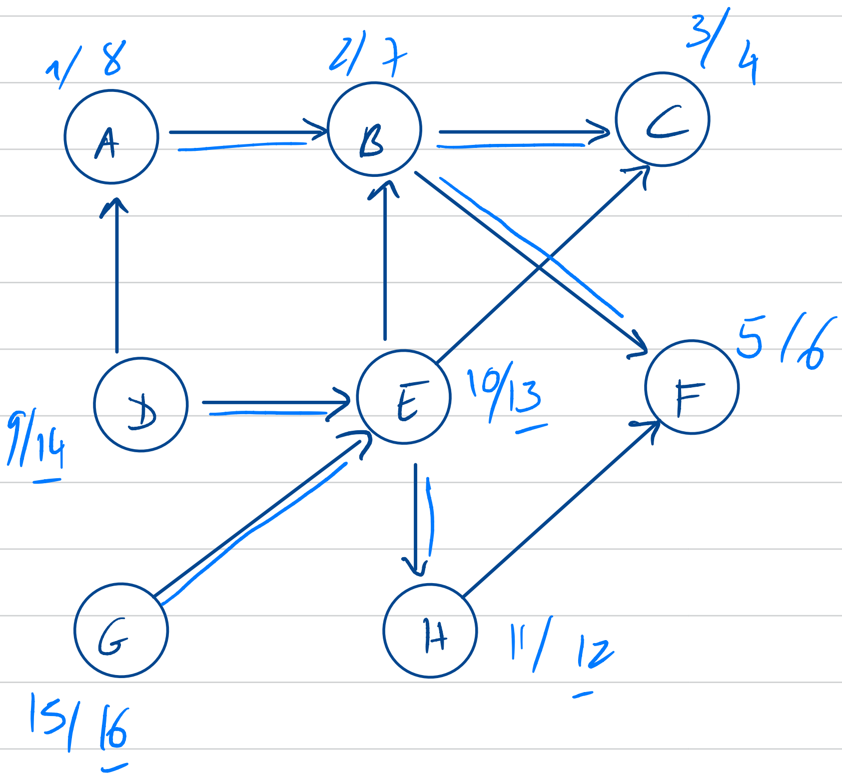
TopologicalSort(G)
- Compute DFS(G)
  - Quando um vértice é terminado, inserimo-lo no início de uma lista L
  - Retornar a lista L



Ordenação Topológica - Algoritmo

TopologicalSort(G)

- Compute DFS(G)
  - Quando um vértice é terminado, inserimo-lo no início de uma lista L
  - Retornar a lista L



G
D
E
H
A
B
F
C