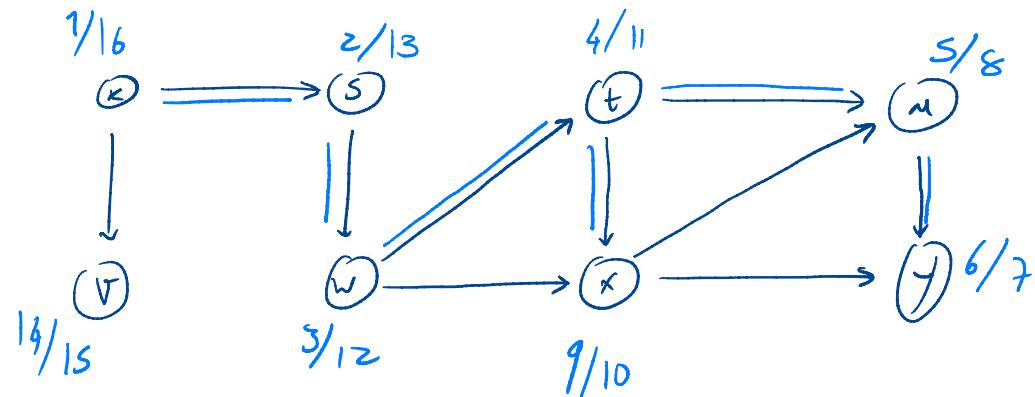
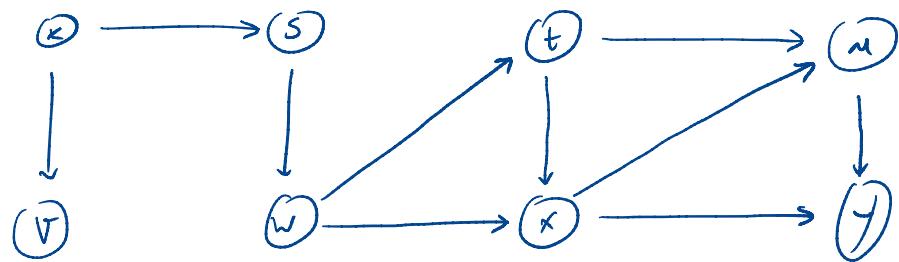


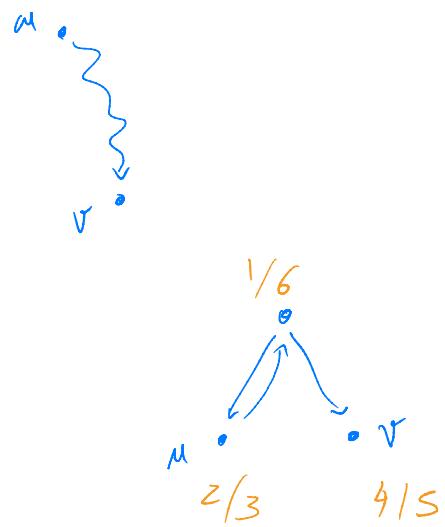
Fig 22.3



Ex 22.3-8

→ Provide a counter example for:

$\alpha \rightsquigarrow r \wedge d_\alpha < d_r \Rightarrow r$ é descendente de α na floresta DFS



$$\begin{array}{c} \text{[m] } \text{[v]} \\ \swarrow \quad \searrow \end{array} \quad \begin{array}{c} \text{[m] } \text{[v]} \\ \text{[v] } \text{[m]} \end{array} \Rightarrow v \text{ não é descendente de } m$$

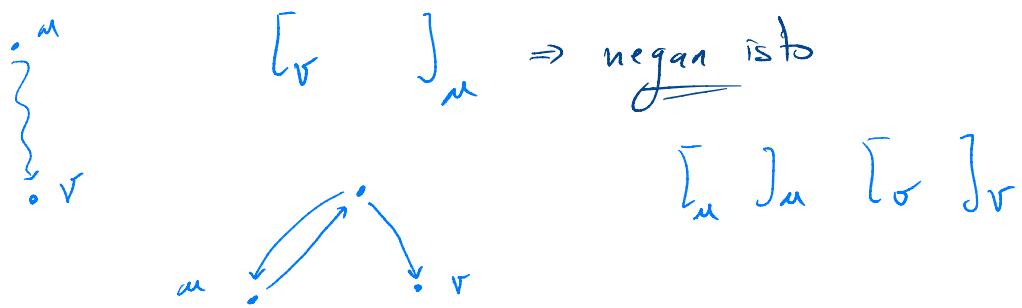
$$\begin{array}{c} \text{[m] } \text{[v]} \\ \swarrow \quad \searrow \end{array} \quad \begin{array}{c} \text{[m] } \text{[v]} \\ \text{[v] } \text{[m]} \end{array} \Rightarrow v \text{ é descendente de } m$$

- Queremos provar q é possível
termos: $f_M < d_q$

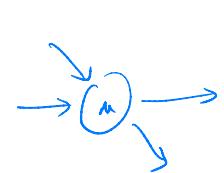
$$\begin{bmatrix} u \\ v \end{bmatrix}_M \begin{bmatrix} r \\ s \end{bmatrix}_V$$

Ex. 22.3 - 9

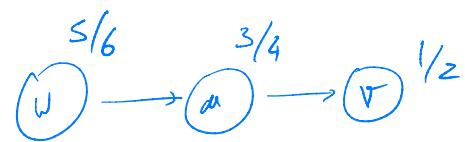
Se um grafo dirigido G tem um caminho entre u e v ,
então qualquer DFS resulta em $d_v < f_u$



Ex 22.3-11



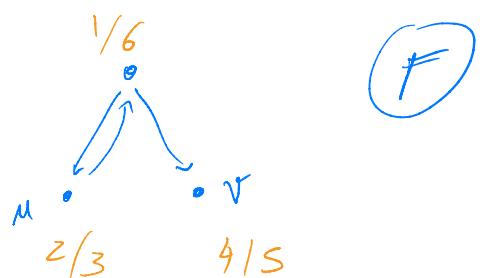
• how can m end up in
a DFS tree by itself?



- Para qualquer DFS existe sempre um vértice v tempo de fim igual a $\text{zf}(v)$

(T)

- Seja $u \in V$ um vértice atingível a partir de todos os vértices do grafo. u é necessariamente o primeiro vértice a ser fechado.



- Se $dv = du + 1$, então (u, v) é um anel de árvore

(T)

- Se $(u, v) \in E$ entao necessariamente que $du < dv$

(F)

- Se $fv < du \wedge (u, v) \in E$, então (u, v) é um anel de cruzamento

$[v]_r \sqsubset [u]_m$

• Cross edge

(T)

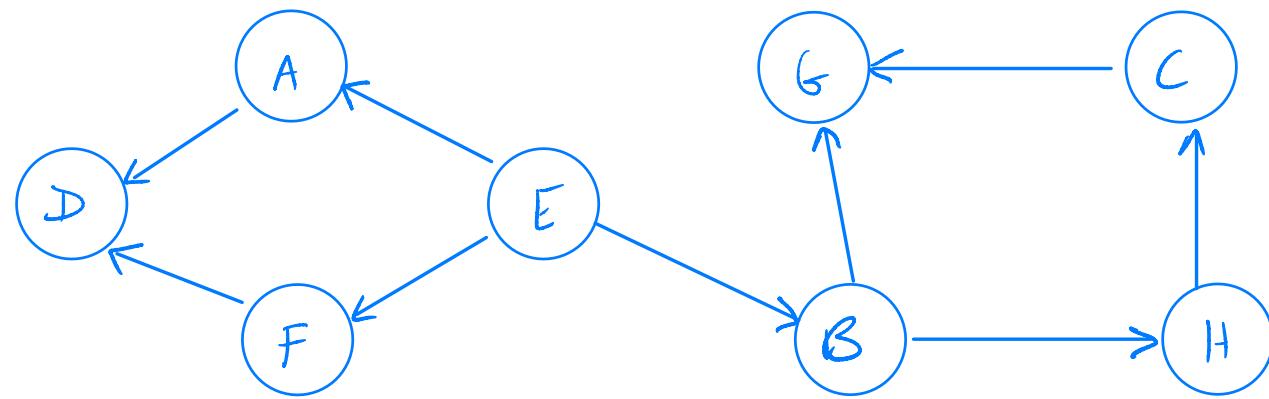
- Se $dv < du \wedge (u, v) \in E$, então (u, v) é um back edge

$[v]_r \sqsubset [u]_m$

(F)

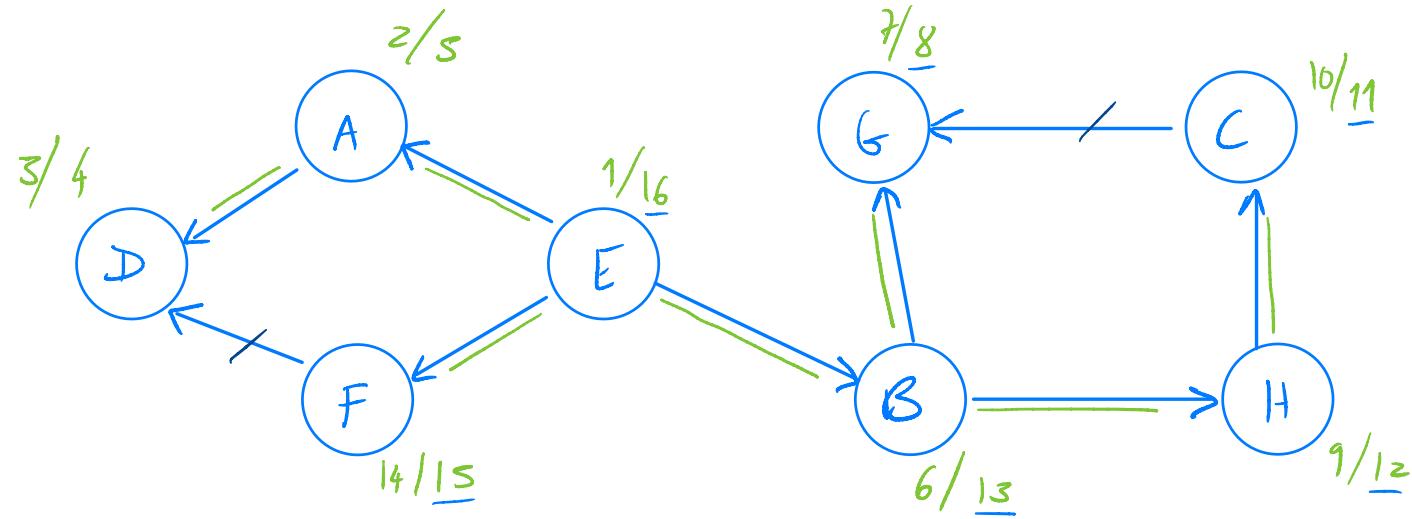
$v \xleftarrow{\text{ }} u$ cross edge
 $1/2 \quad 3/4$

TI - 12-13 - I.b



- Começar no vértice E e visitar os vértices por ordem lexicográfica

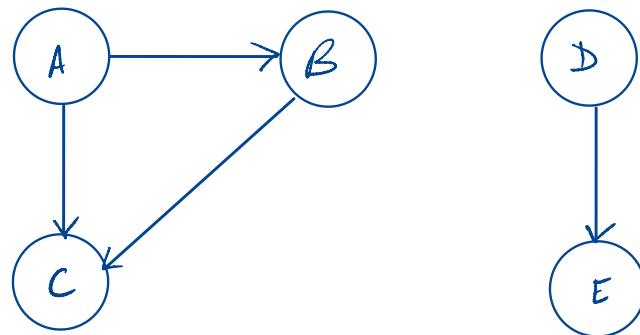
T1 - 12-13 - I.b



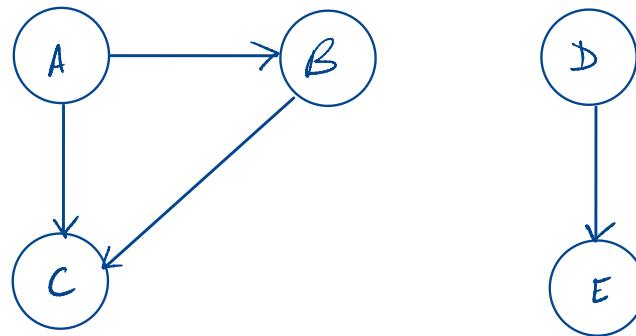
- E, F, B, H, C, G, A, D

T1 06/07 I.2

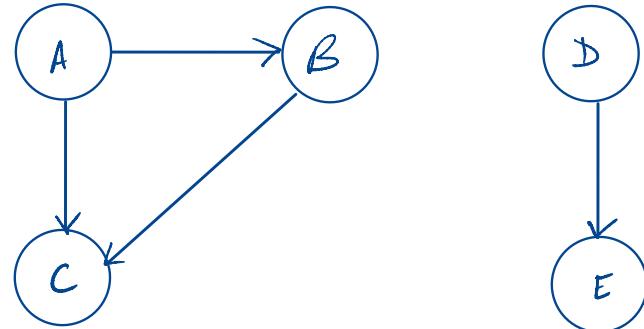
(I)



(II)



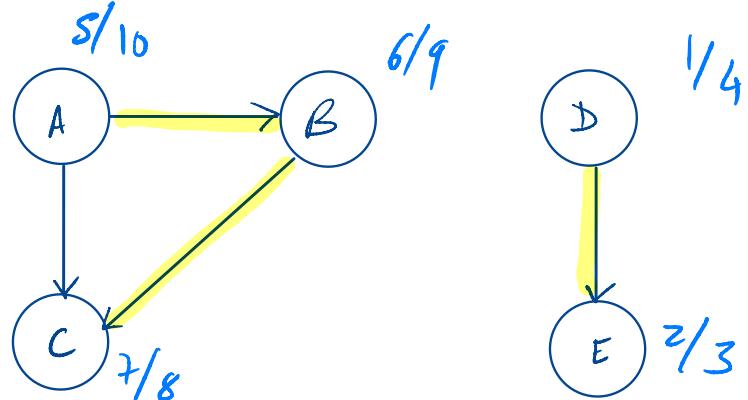
(III)



- 3 ordenações topológicas
e respectivas DFS

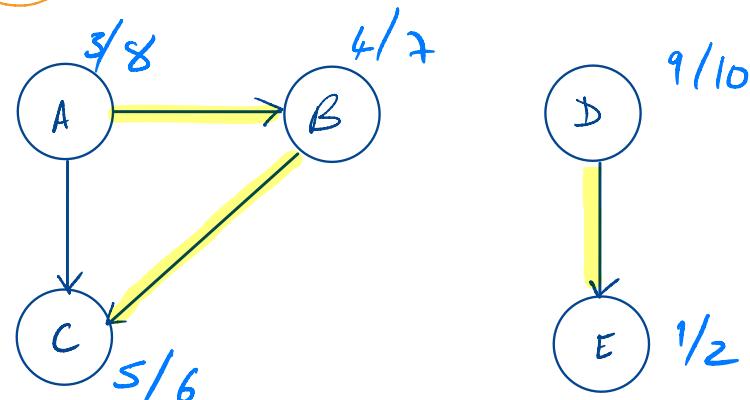
T1 06/07 I.2

I



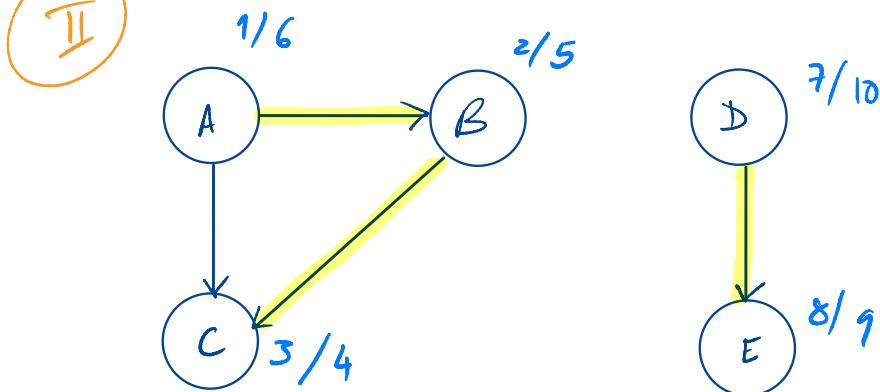
$\langle A, B, C, D, E \rangle$

III



$\langle D, A, B, C, E \rangle$

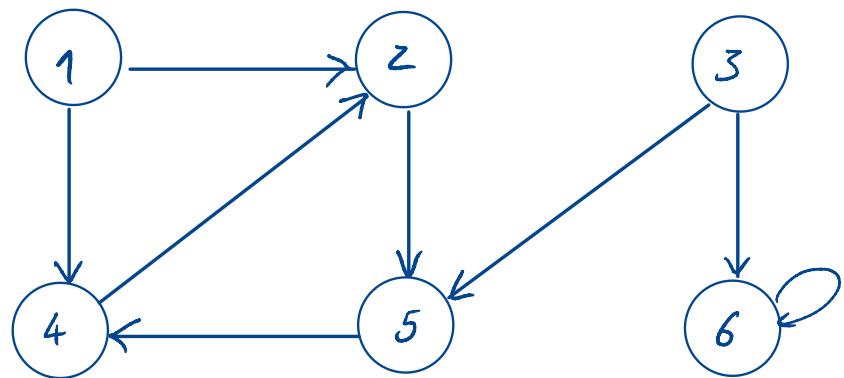
II



$\langle D, E, A, B, C \rangle$

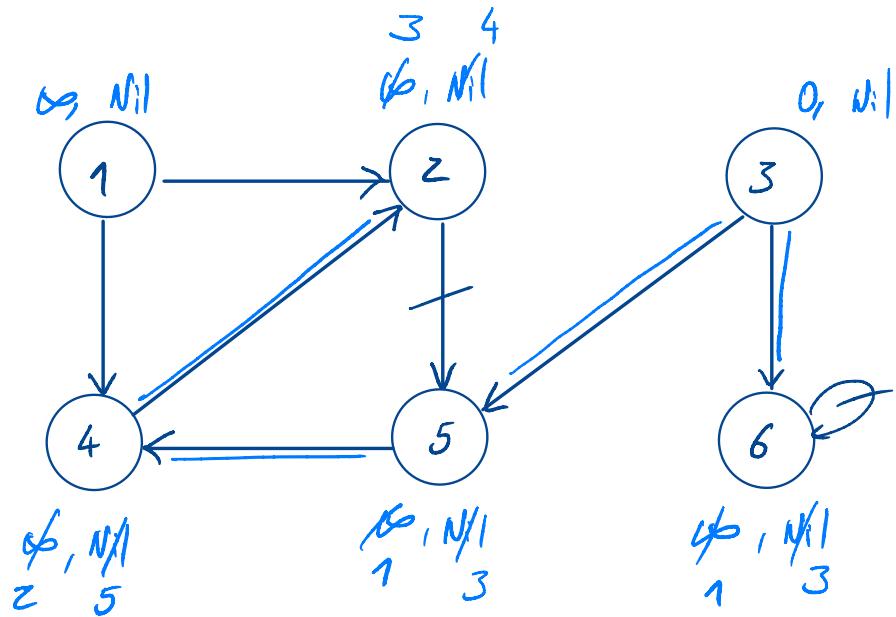
- 3 ordenações topológicas
e respectivas DFS

Ex 22.2-1



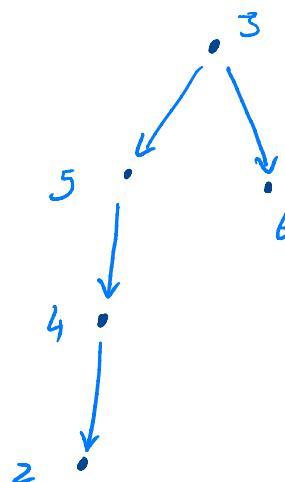
- Começar em 3 e utilizar ordem numérica ...

Ex 22.2 - 1



- Começar em 3 e utilizar ordem numérica ...

$Q : 3, 5, 6, 4, 2$



22.5.3

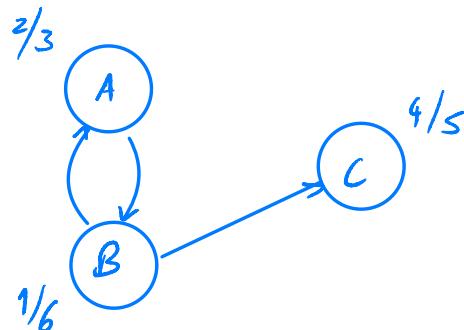
- Algoritmo p/ encontrar SCCs

- DFS(G)

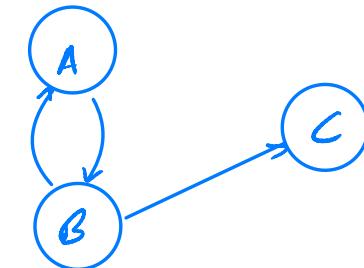
- DFS(G) \Rightarrow percorrendo os vértices por ordem crescente de tempo de fim

- Observação: O SCC com menor tempo de fim é um SCC sink

- Falha: o vértice com menor tempo de fim não pertence necessariamente ao SCC c/ menor tempo de fim.



- A 2nd DFS começa pelo vértice A e encontra todo o grafo.

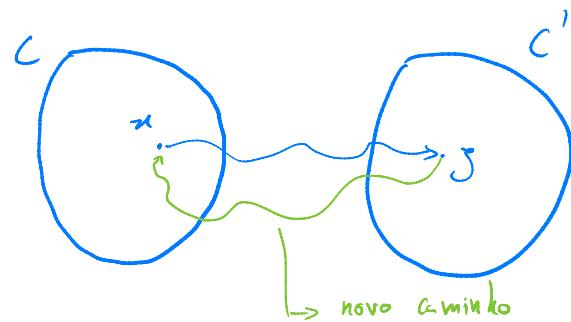


22.5.1

- O que acontece ao nº de SCCs de um dada grafo se adicionarmos um caminho entre dois nós

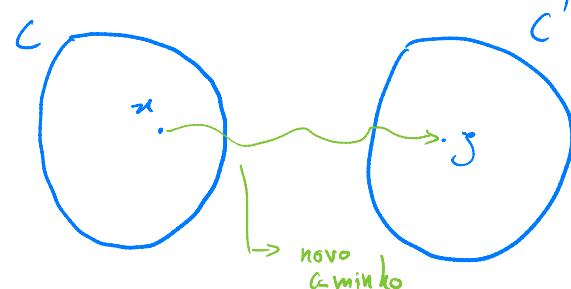
→ Diminui ou mantém-se

- Diminui

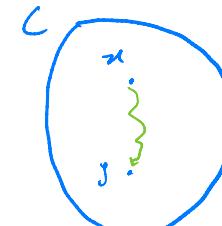


- Mantém-se

(I) Ligamos dois componentes desligados



(II) Nova ligação entre dois vértices dentro do mesmo componente

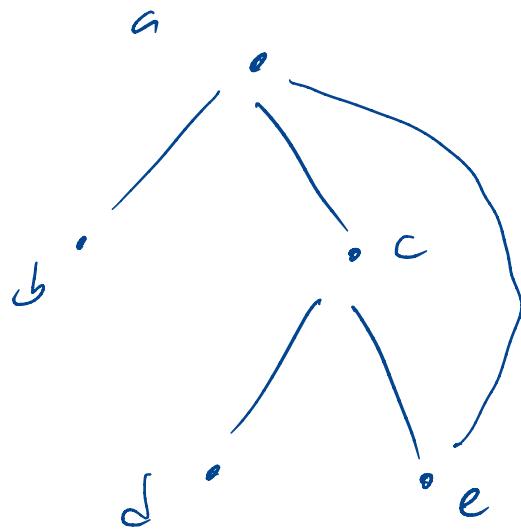


22.4-3

Determinar se um grafo não dirigido contém um ciclo simples em tempo $O(V)$.

- DFS modificada

Exemplo



22.4 - 3

Determinar se um grafo não dirigido contém um ciclo simples em tempo $O(V)$.

- DFS modificada

DFS(G)

for $v \in G.V$

$v.\text{visited} := \text{false}$; $v.\pi := \text{nil}$

for $v \in G.V$

 if ($\neg v.\text{visited}$ $\&$ DFS-visit(G, v))

 return true

return false

DFS-Visit(G, v)

$v.\text{visited} := \text{true}$

 for each $u \in G.\text{Adj}[v]$

 if ($\neg u.\text{visited}$) {

$u.\pi := v$;

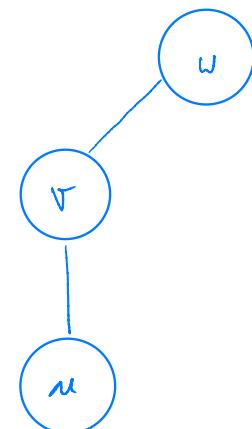
 return DFS-Visit(G, u)

 else if ($v.\pi \neq u$) {

 return true

}

{
return false



22.4 - 3

Determinar se um grafo não dirigido contém um ciclo simples
em tempo $O(V)$.

- DFS modificada

DFS(G)

for $v \in G.V$

$v.visited := \text{false}$; $v.PI := \text{nil}$

for $v \in G.V$

 if ($\neg v.visited \& \text{DFS-visit}(G, v)$)

 return true

return false

DFS-Visit(G, v)

$v.visited := \text{true}$

for each $u \in G.\text{Adj}[v]$

 if ($\neg u.visited$) {

$u.PI := v$;

 return DFS-Visit(G, u)

 else if ($v.PI \neq u$) {

 return true

}

return false

22.4 - 3

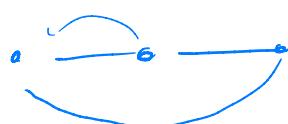
Determinar se um grafo não dirigido contém um ciclo simples em tempo $O(V)$.

Loop 1

- É executado no máximo $|V|$ vezes
(Vertices visitados não são re-visitados)

Loop 2

- O loop 2 é executado no máximo $|E|$ vezes
mas um grafo não dirigido acíclico tem no
máximo $|V|-1$ arcos.
- Assim o loop 2 é executado, no pior caso, $2|V|-1$ vezes.
O arco $2|V|-1$ é necessariamente arco para trás
porque num DFS num grafo não dirigido só havia
arcos p/ trás e arcos da árvore.



DFS(6)

for $v \in G.V$

$v.visited := false; v.PI := nil$

Loop 1

for $v \in G.V$

if ($v.visited \&& DB-visit(G, v)$)

return true

return false

DFS-Visit(G, v)

$v.visited := true$

for each $u \in G.Adj[v]$

if ($u.visited$) {

$u.PI := v;$

return DFS-Visit(G, u)

else if ($v.PI \neq u$) {

return true

Loop 2

}

}

}

return false

22.5 - 5

- Depois de calcular os SCCs calcular o grafo dos SCCs em tempo linear.

- Associam a cada SCC um id único e anotam cada vértice do grafo original com o id do seu SCC. Escreveremos al.scc para denotar o id do SCC a que u pertence.

Compute $\text{ESCC}(G, \text{GSCC})$

$k := |\text{GSCC.V}|$ // number of SCCs

let $A[1..k]$ be a new array whose elements are initially 0

for each SCC index i in GSCC.V

 let C be the vertices of G in SCC i

 for each $u \in C$

 for each $v \in G.\text{Adj}[u]$

 if $((u.\text{scc} \neq v.\text{scc}) \text{ or } (A[v.\text{scc}] == 0))$

$\text{GSCC.addEdge}(u.\text{scc}, v.\text{scc})$

$A[v.\text{scc}] := 1$

 for each $u \in C$

 for each $v \in G.\text{Adj}[u]$

$A[v.\text{scc}] := 0$

$$\text{GSCC} = (\text{V}_{\text{SCC}}, \text{E}_{\text{SCC}})$$

- $\text{V}_{\text{SCC}} \Rightarrow$ ids dos SCCs de G

- E_{SCC}

Observação: Visitar as adjacências dos vértices de cada SCC, num SCC de cada vez, mantendo um array A de 0s e 1's de tamanho igual ao n° de SCCs.

$A[j] = 1$ se já foi encontrado um arco a ligar o SCC que a ser visitado ao SCC j .

Complexidade:

$$\sum_{v \in \text{ESCC}(G)} \left(\sum_{u \in V} \left(O(1) + \sum_{v \in \text{Adj}[u]} O(1) \right) \right) = O(|V| + |E|)$$