

- Propiedades Elementales de Grafos (continuación)
- Grafos Dirigidos Acíclicos
- Componentes Fortemente Ligados

Aula 5

## DFS - Propriedades

### Teorema do Caminho Branco

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

$\Rightarrow v$  é descendente de  $u$

$\Leftarrow$  Existe um caminho branco entre  $u$  e  $v$   
 $\Rightarrow v$  é descendente de  $u$  na floresta DFS

## DFS - Propriedades

### Teorema do Caminho Branco

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

Se no momento em que  $u$  é descoberto

existe um caminho branco a ligar  $u$  a  $v$ .

$\Rightarrow v$  é descendente de  $u$

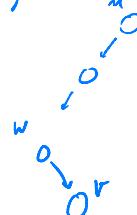
$\downarrow \Rightarrow$  Existe um caminho branco entre  $u$  e  $v$  em  $\alpha_u$

$\left[ u \ [v]_v \right]_u$

$\Leftarrow$  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  $u$

$\left[ u \ \left[ v \ [w]_w \right]_u \right]_v$

em ambos os casos  $v$  é  
descendente de  $u$

## DFS - Propriedades

- Um grafo tem um caminho circular se e só se a DFS revela um novo para trás.

$\Leftarrow$  Back-edge  $\Rightarrow$

$\Rightarrow$  Caminho circular  $\Rightarrow$  Back-edge

## DFS - Propriedades

- Um grafo tem um caminho circular se e só se a DFS revela um nó para trás.

$\Leftarrow$  Back-edge  $\Rightarrow$  Caminho circular



$\Rightarrow$  Caminho circular  $\Rightarrow$  Back-edge

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

$$v_i = v_n$$

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

$\langle v'_1, \dots, v'_n \rangle$

$$v'_1 = v_n$$

• 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 árvore DFS  $\Rightarrow (v_{n-1}, v'_1)$  é back-edge para trás

## Definição [ Grafo Dirigido Acíclico (Directed Acyclic Graph) ]

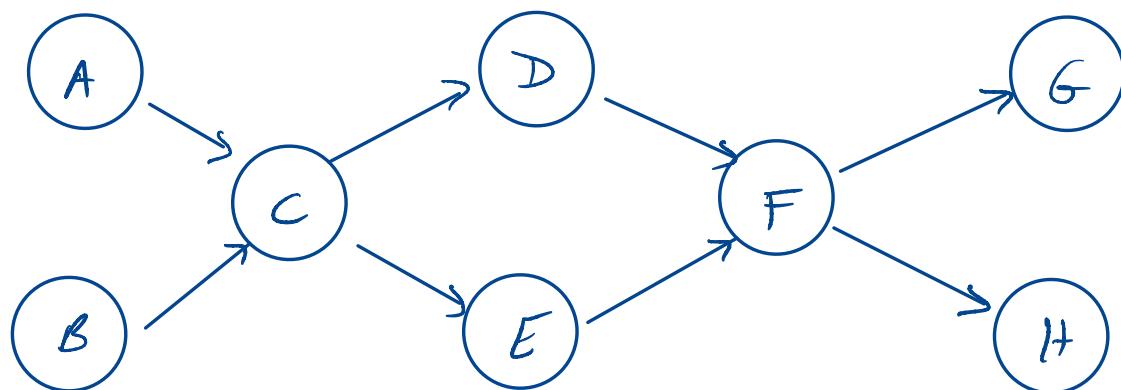
Um grafo dirigido diz-se acíclico se não contém caminhos循环 (ciclos).

Um caminho循环 num grafo  $G = (V, E)$  é uma sequência de vértices

$\langle v_1, \dots, v_n \rangle$  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  $G = (V, \bar{E})$  é um DAG e  $(u, v) \in E$ , então  $f_v < f_u$ .

Prova

## DAGs - Propriedades

Se  $G = (V, \bar{E})$  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  $u$

-  $[v]_u [u]_v \hookrightarrow (u, v)$  é um anel branco  $\nearrow$  contradiz a hipótese de o grafo ser acíclico

-  $[v]_v [u]_u \Rightarrow (u, v)$  é anel de cruzamento

-  $[u]_u [v]_v \times$   
 $\hookrightarrow$  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 saída (outgoing edges)

Não esquecer:

$$\text{Num DAG: } (u, v) \in E \Rightarrow f_u > f_v$$

**Propriedade:** 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 saída (outgoing edges)

Não esquecer:

$$\text{Num DAG: } (u, v) \in E \Rightarrow f_u > f_v$$

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

Prova:

- Sink: vértice com menor tempo de fim.
- Source: vértice com maior tempo de fim.

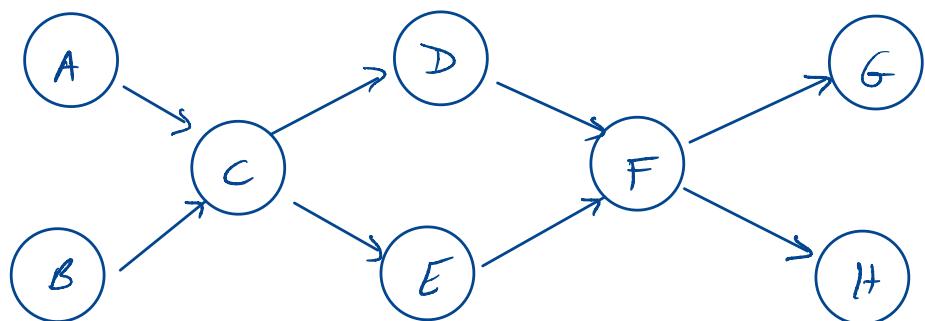
↳ Seja  $m$  o vértice com maior tempo de fim. Suponhamos, por contradição, que existe  $w$  tal que:

$$w \rightarrow m \Rightarrow f_w > f_m \quad \text{?}$$

## 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:



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

- Quantas ordenações topológicas admite o grafo?

$$2 \times 2 \times 2 = 8$$

## Ordenação Topológica - Algoritmo

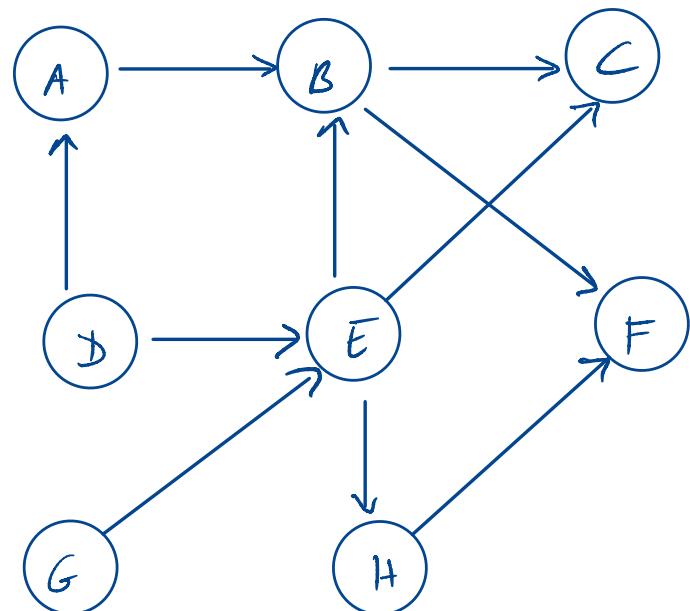
TopologicalSort( $G$ )

1. Compute DFS( $G$ )

- Quando um vértice é determinado, inserimo-lo

no início de uma lista  $L$

- Retornar a lista  $L$

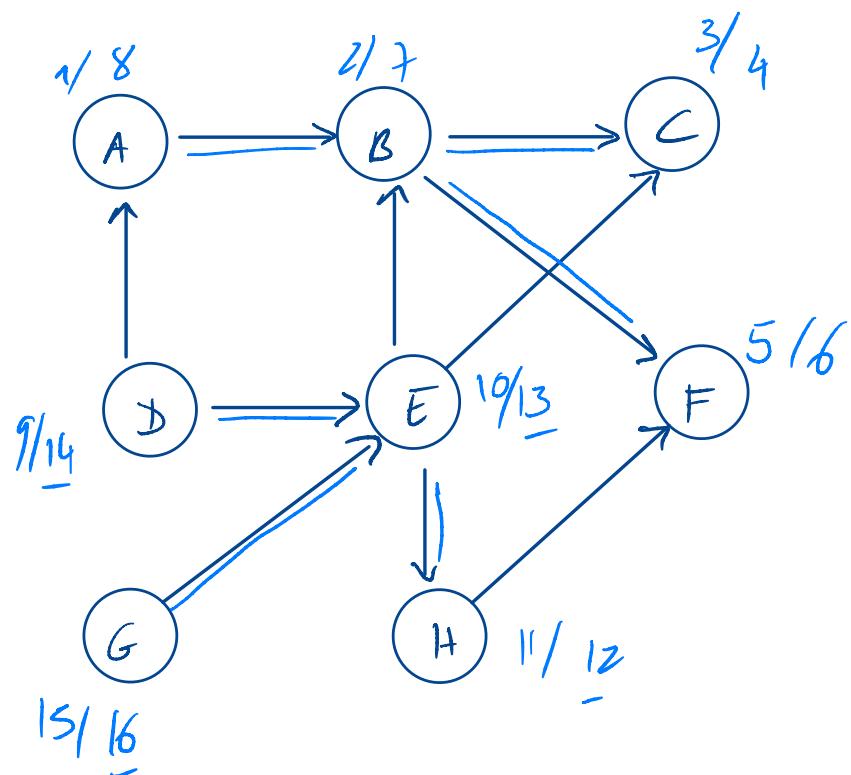


## Ordenação Topológica - Algoritmo

TopologicalSort( $G$ )

1. Compute DFS( $G$ )

- Quando um vértice é terminado, inserimo-lo no início de uma lista  $L$
- Retornare a lista  $L$



G  
D  
E  
H  
A  
B  
F  
C

## Componentes Fortemente Ligados

Definição [ Componente Fortemente Ligado (Strongly Connected Component - scc) ]

Dado um grafo  $G = (V, E)$ , um componente fortemente ligado de  $G$  é um conjunto de vértices  $C \subseteq V$  tal que:

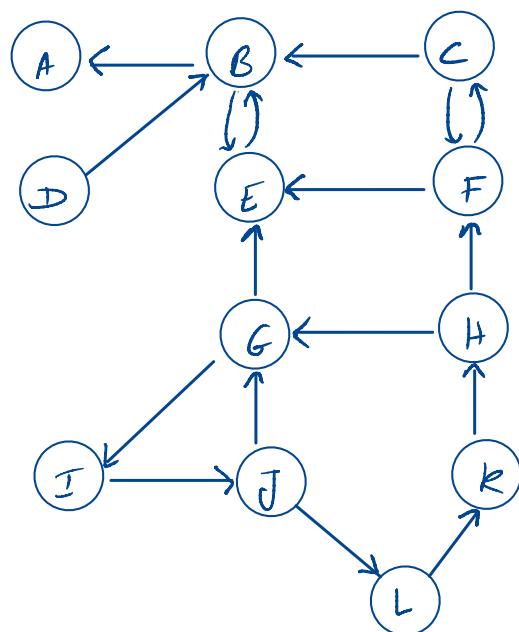
(I)  $\forall u, v \in C \quad u \rightarrow v \wedge v \rightarrow u$

$\hookrightarrow u$  é atingível a partir de  $v$

(II)  $C$  é maximal

(Não existe  $C'$  tal que  $C \subset C'$  e  $C'$  satisfaz (I))

Exemplo:



## Componentes Fortemente Ligados

Definição [ Componente Fortemente Ligado (Strongly Connected Component - scc) ]

Dado um grafo  $G = (V, E)$ , um componente fortemente ligado de  $G$  é um conjunto de vértices  $C \subseteq V$  tal que:

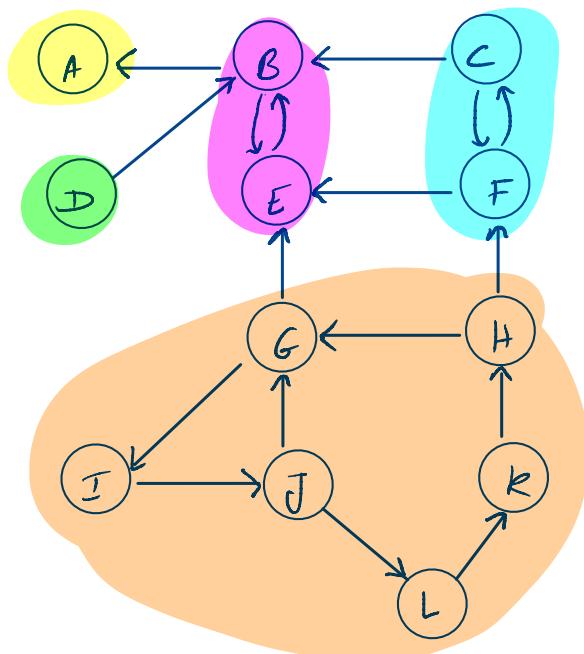
(I)  $\forall u, v \in C \quad u \rightarrow v \wedge v \rightarrow u$

$\hookrightarrow u$  é atingível a partir de  $v$

(II)  $C$  é maximal

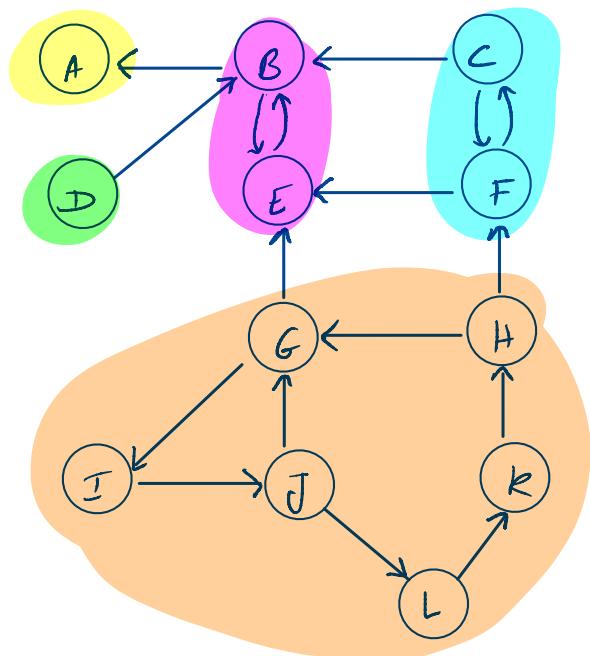
( Não existe  $C'$  tal que  $C \subset C'$  e  $C'$  satisfaz (I) )

Exemplo:



Componentes Fortemente Ligados

Exemplo:



Gráfios das SCCs:

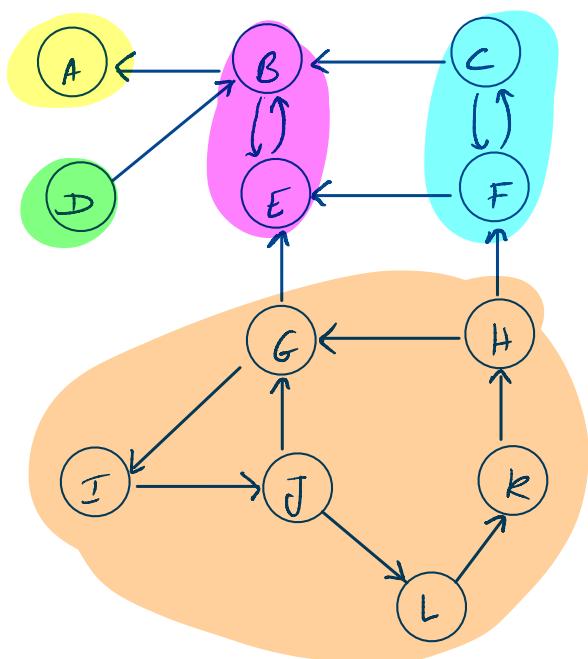
$$G_{SCC} = (V_{SCC}, E_{SCC})$$

$$V_{SCC} =$$

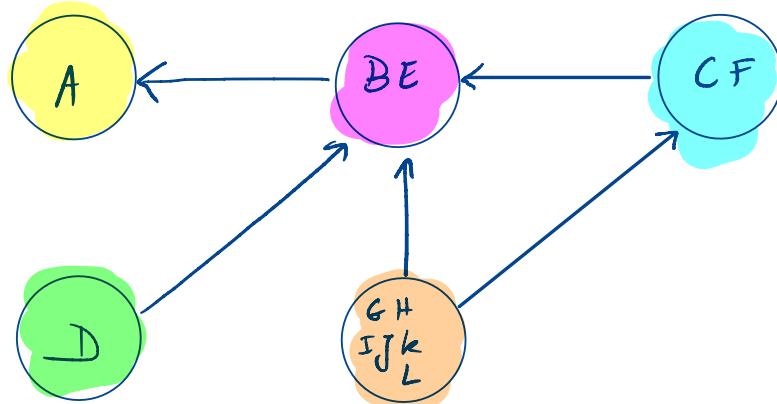
$$E_{SCC} :$$

## Componentes Fortemente Ligados

Exemplo:



Gráfios dos SCCs:



$$G_{\text{SCC}} = (V_{\text{SCC}}, E_{\text{SCC}})$$

$$V_{\text{SCC}} = \{C \mid C \text{ é um SCC de } G\}$$

$$E_{\text{SCC}}: (C_1, C_2) \in E_{\text{SCC}}$$

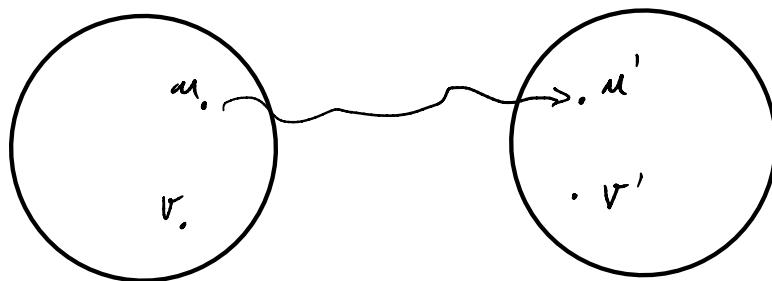
sse

$$\exists u, v \in V \text{ s.t. } u \in C_1 \wedge v \in C_2 \wedge (u, v) \in E$$

## Componentes Fortemente Ligados - Propriedades

Propriedade 1: Sejam  $C$  e  $C'$  dois SCCs de um grafo  $G = (V, E)$ .

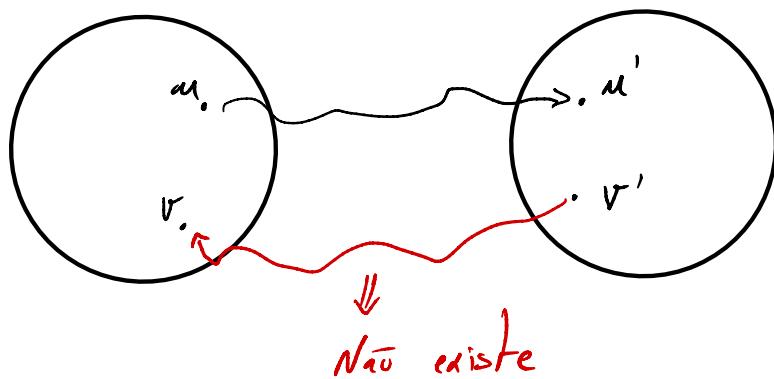
Dados dois vértices  $u, v \in C$  e dois vértices  $u', v' \in C'$ , se  $u \sim u'$   
então  $v' \not\sim v$ .



## Componentes Fortemente Ligados - Propriedades

Propriedade 1: Sejam  $C$  e  $C'$  dois SCCs de um grafo  $G = (V, E)$ .

Dados dois vértices  $u, v \in C$  e dois vértices  $u', v' \in C'$ , se  $u \sim u'$   
então  $v' \not\sim v$ .



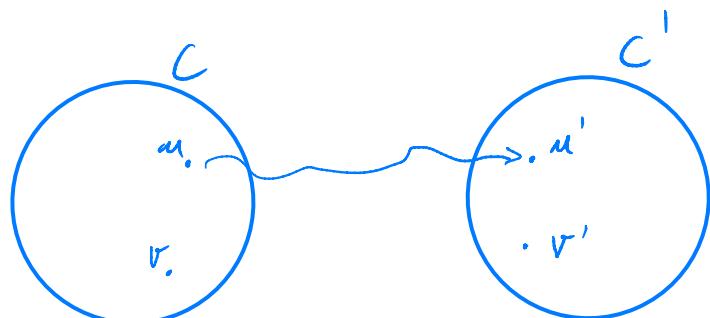
## Componentes Fortemente Ligados - Propriedades

Propriedade 2: Sejam  $C$  e  $C'$  dois SCCs num grafo dirigido  $G = (V, E)$ ; se existir um arco  $(u, u')$  de  $C$  para  $C'$ , então:

$$f(C) > f(C')$$

- $d(C) = \min \{ d(u) \mid u \in C \}$
- $f(C) = \max \{ f(u) \mid u \in C \}$

Prova:



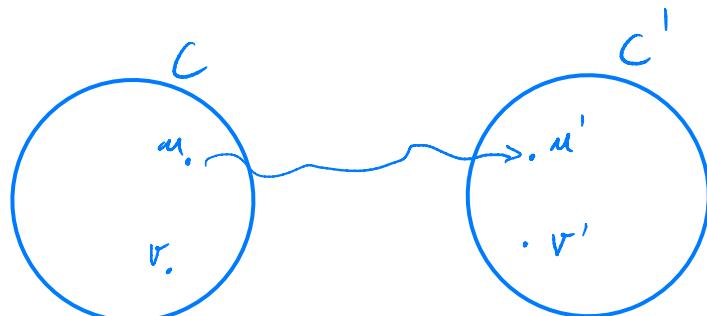
## Componentes Fortemente Ligados - Propriedades

Propriedade 2: Sejam  $C$  e  $C'$  dois SCCs num grafo dirigido  $G = (V, E)$ ; se existir um arco  $(u, u')$  de  $C$  para  $C'$ , então:

$$f(C) > f(C')$$

- $d(C) = \min \{ d(u) \mid u \in C \}$
- $f(C) = \max \{ f(u) \mid u \in C \}$

Prova:



①  $d_C < d_{C'}$

- Há um caminho branco q liga \$u\$ a todos os vértices de \$C'\$.  
Todos os vértices de \$C'\$ são descendentes de \$u\$.

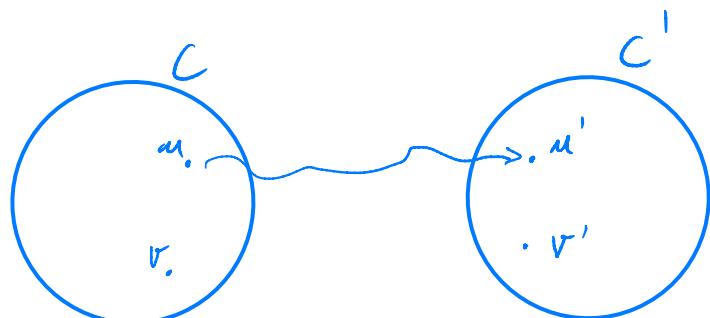
-  $f(C) < f(C')$

## Componentes Fortemente Ligados - Propriedades

Propriedade 2: Sejam  $C$  e  $C'$  dois SCCs num grafo dirigido  $G = (V, E)$ ; se existir um anco  $(u, u')$  de  $C$  para  $C'$ , então:

$$f(C) > f(C')$$

Prova:



II)  $d(C') < d(C)$

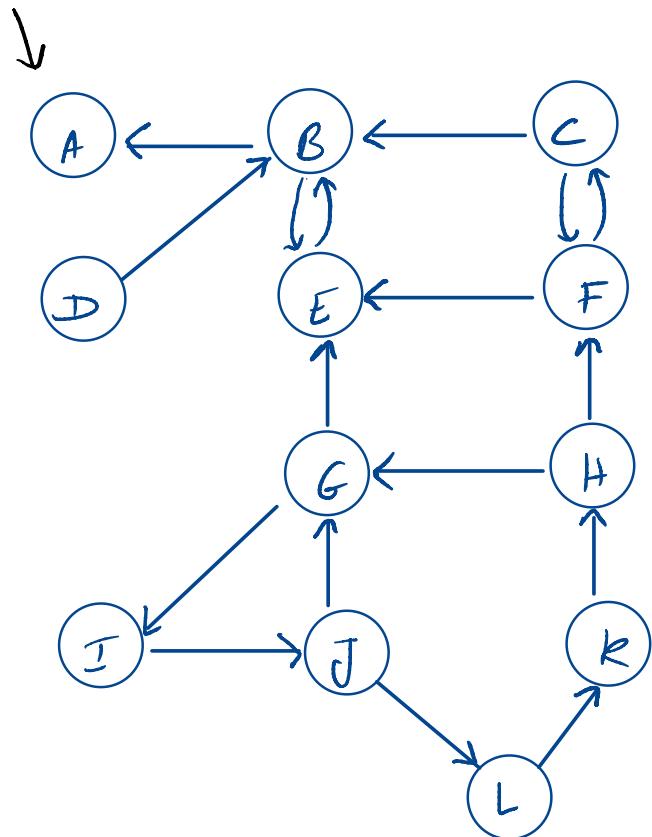
- Não existe nenhu-anco de  $C'$  p/  $C$
- Não existe um caminho a liga os vértices de  $C'$  aos vértices de  $C$ .

- $d(C) = \min \{d(u) \mid u \in C\}$
- $f(C) = \max \{f(u) \mid u \in C\}$

- Todos os vértices de  $C'$  são fechados antes de os vértices de  $C$  começarem a ser explorados.

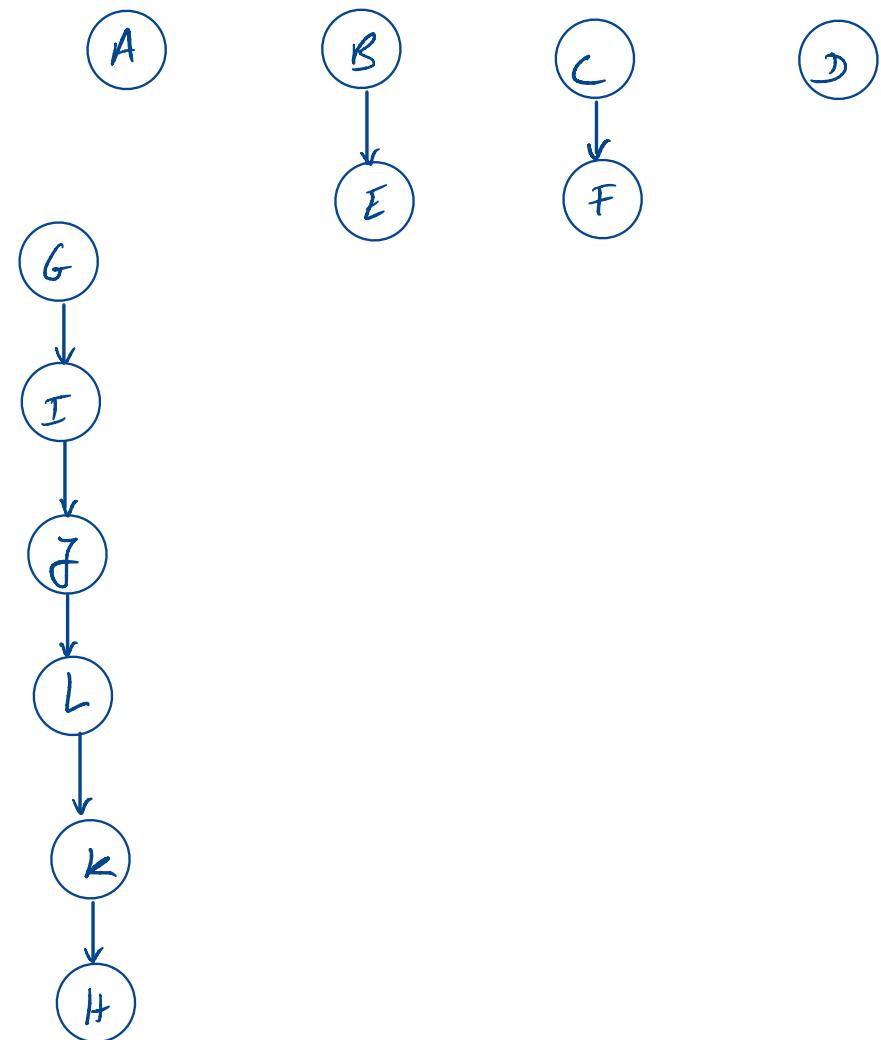
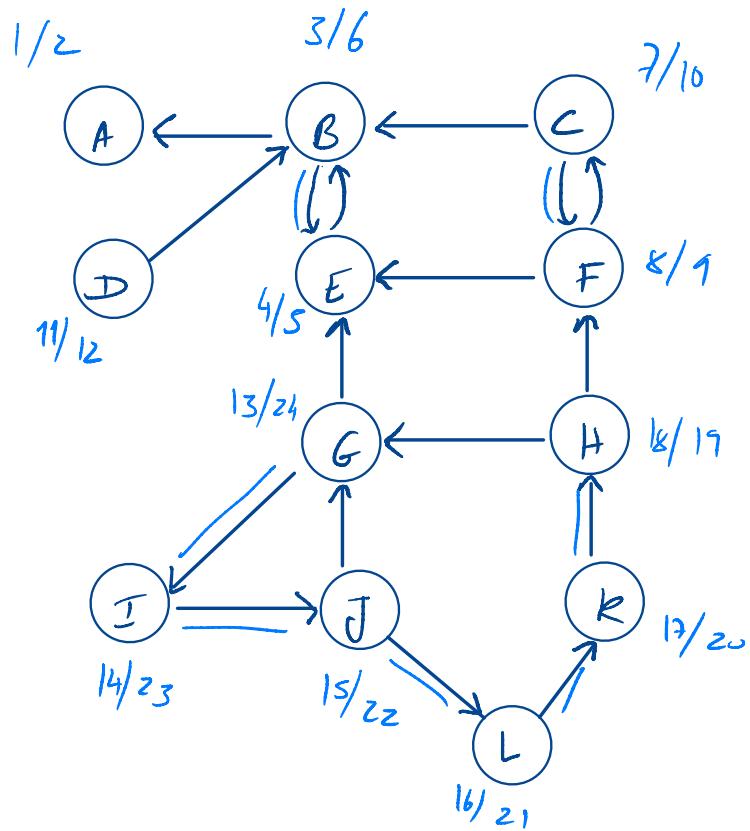
$$f(C') < d(C) < f(C)$$

## Componentes Fortemente Ligados - Algoritmo de Kosaraju - Sharir

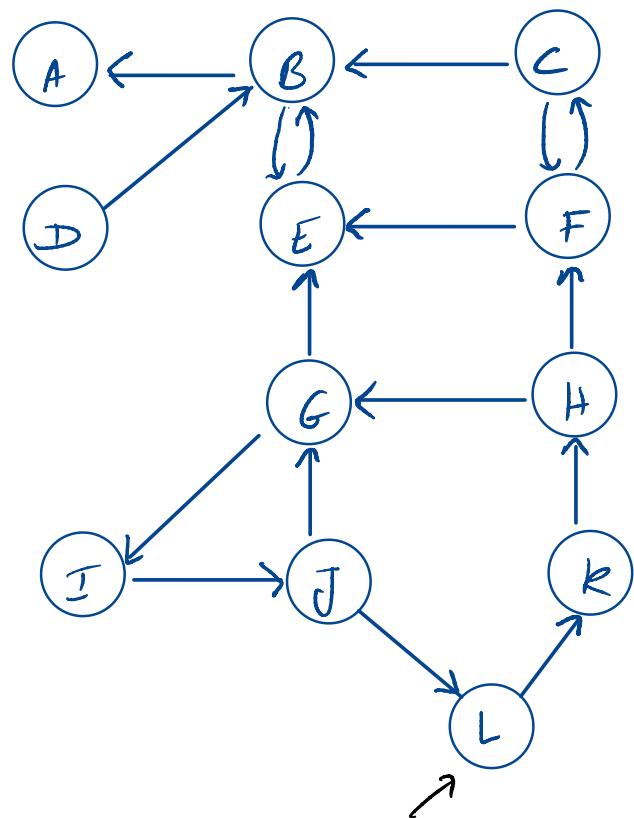


## Componentes Fortemente Ligados - Algoritmo de Kosaraju-Sharir

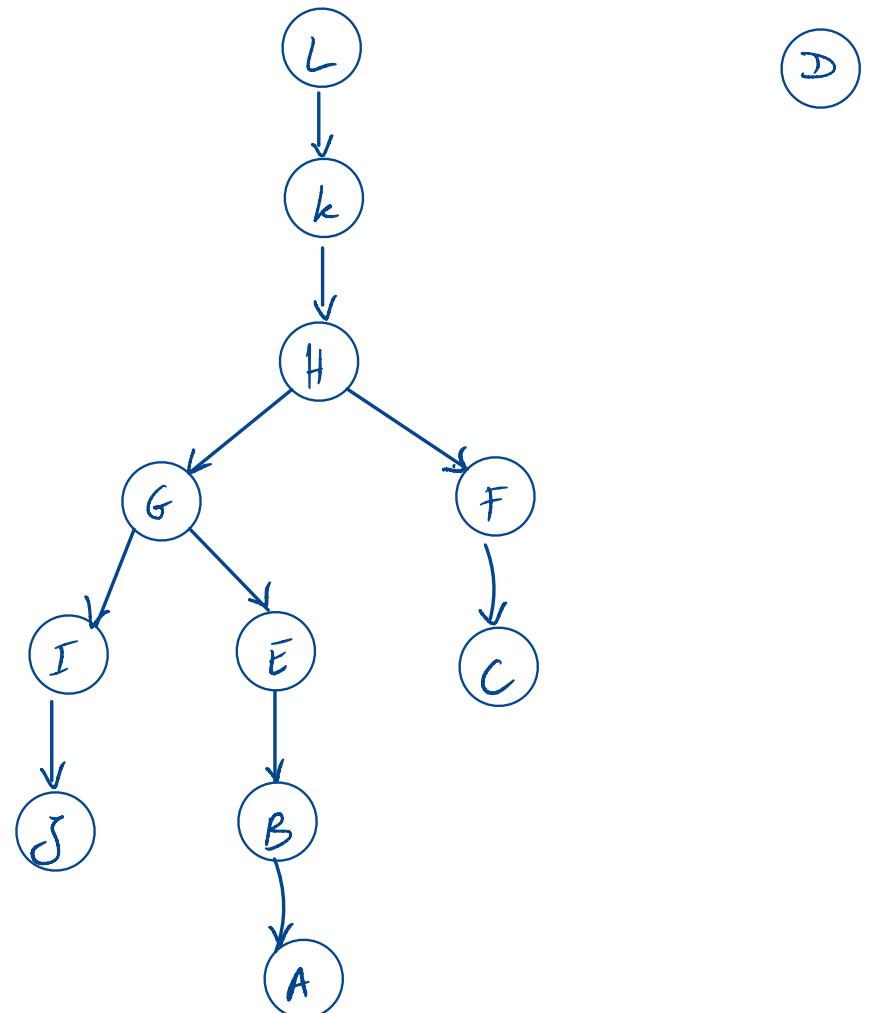
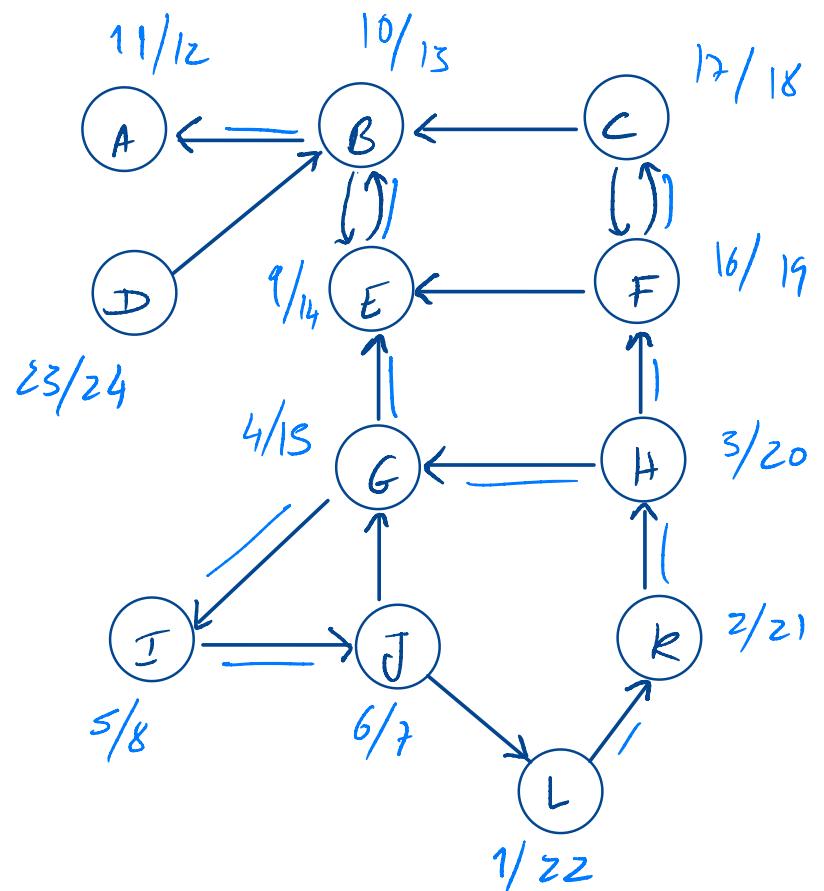
Floripa DFS



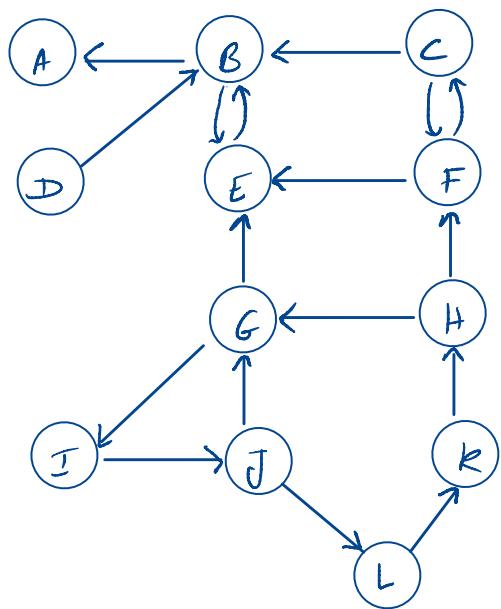
## Componentes Fortemente Ligados - Algoritmo de Kosaraju - Sharir



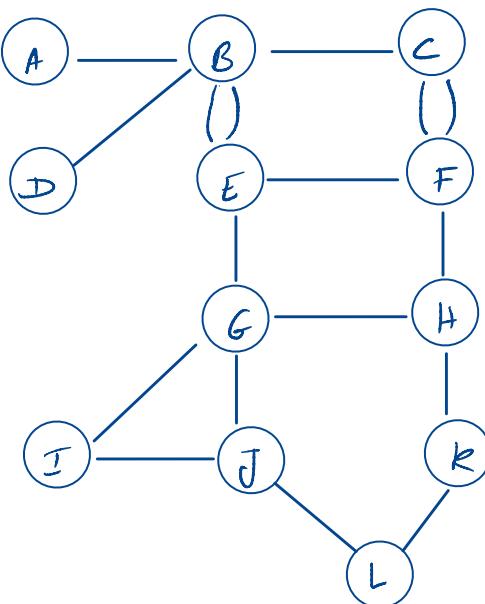
## Componentes Fortemente Ligados - Observações



## Componentes Fortemente Ligados - Grafos Transpostos



Grafo Original



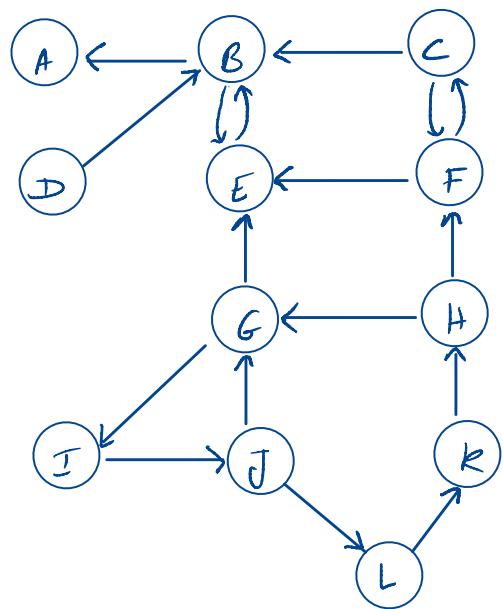
Grafo Transposto

Grafo Transposto

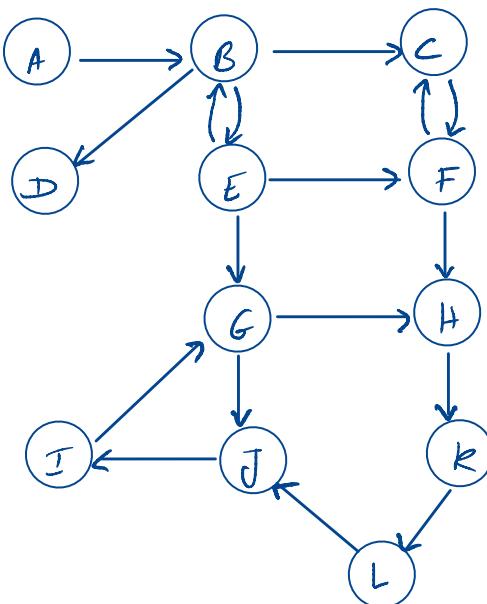
$$G^T = (V, E^T)$$

$$E^T = \{ (u, v) \mid (v, u) \in E \}$$

## Componentes Fortemente Ligados - Grafos Transpostos

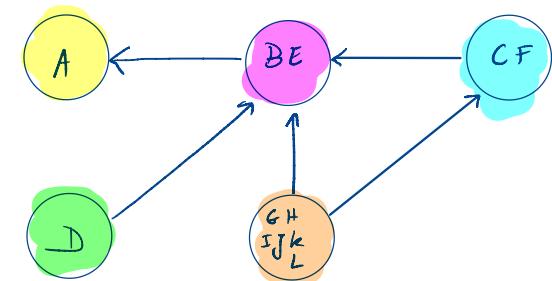


Grafo Original



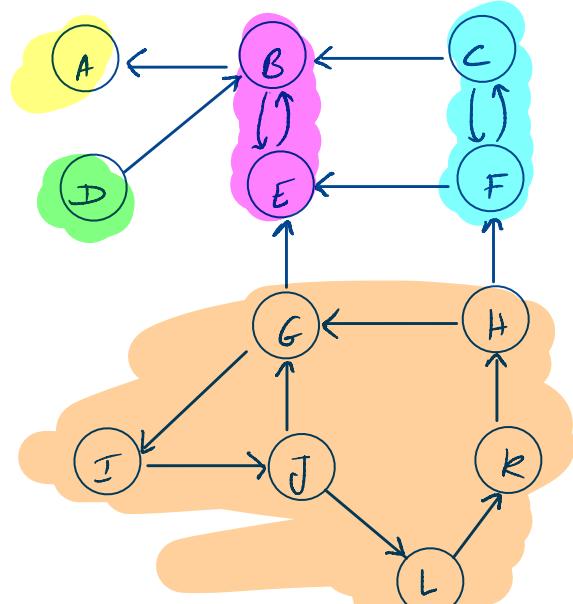
Grafo Transposto

Grafo Original

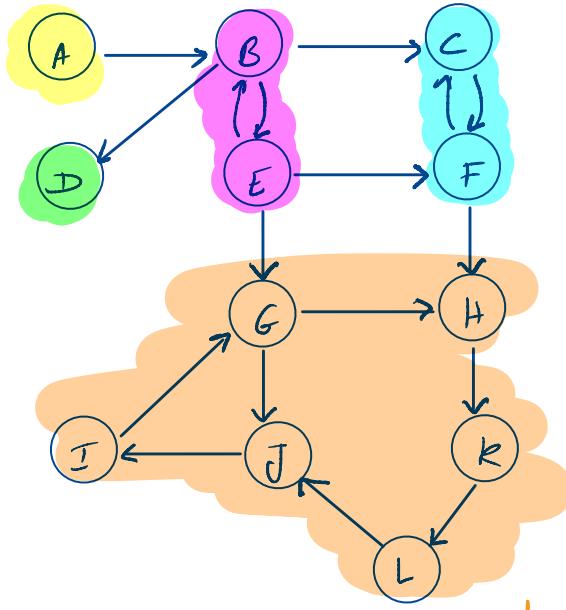


Grafo Transposto

## Componentes Fortemente Ligados - Grafos Transpostos

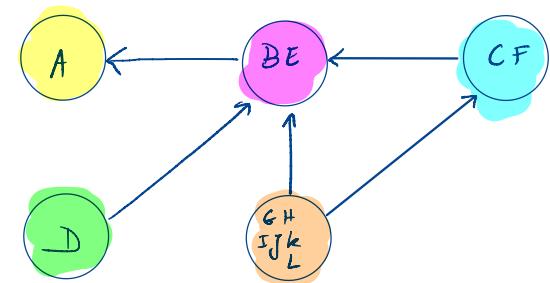


Grafo Original

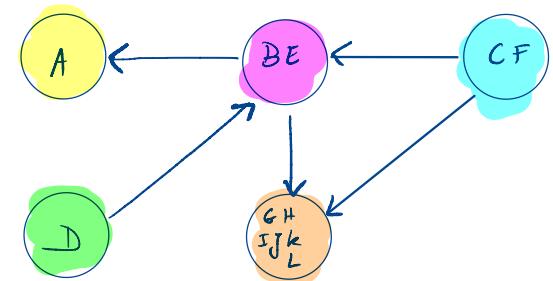


Grafo Transposto

Grafo Original



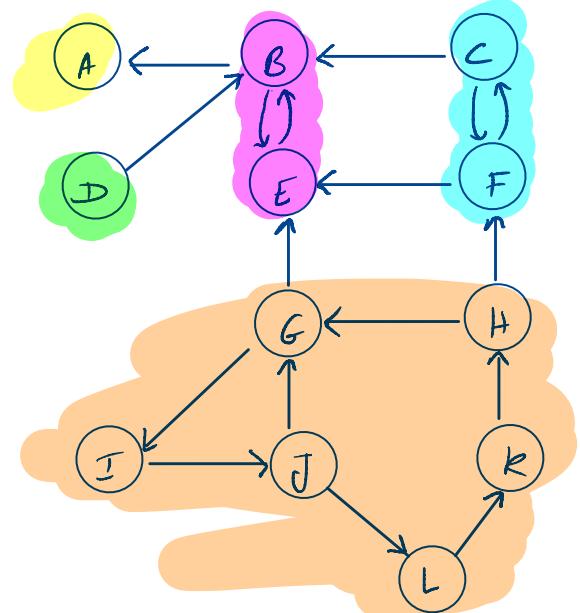
Grafo Transposto



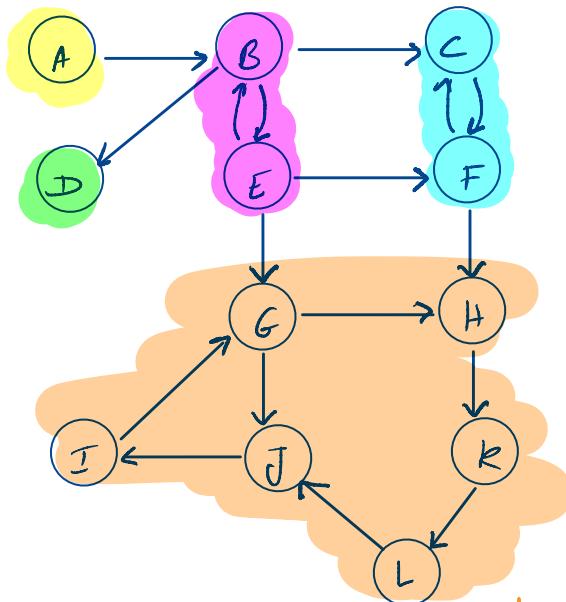
Observação 1:

Observação 2:

## Componentes Fortemente Ligados - Grafos Transpostos



Grafo Original



Grafo Transposto

Observação: Os SCCs do grafo original coincidem com os SCCs do grafo transposto

## Componentes Fortemente Ligados - Algoritmo de Kosaraju - Sharir

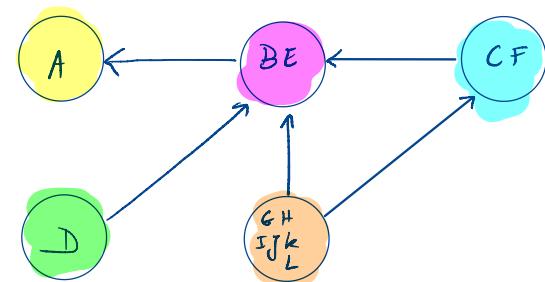
Observação 1: O SCC com maior tempo de fim no grafo  $G$  é um SCC source no grafo dos SCCs.

Observação 2: O vértice com maior tempo de fim pertence ao SCC com maior tempo de fim.

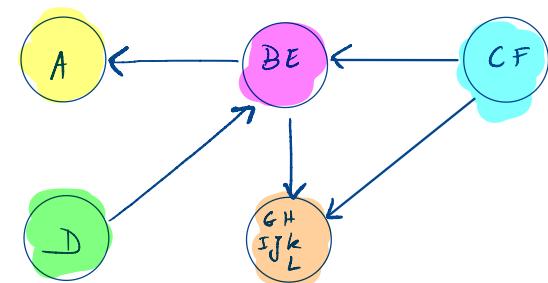
Observação 3: O SCC com maior tempo de fim no grafo  $G$  é um SCC sink no grafo dos SCCs do grafo transposto.

Observação Final: O vértice com maior tempo de fim pertence a um SCC sink no grafo dos SCCs do grafo transposto.

Grafo Original



Grafo Transposto

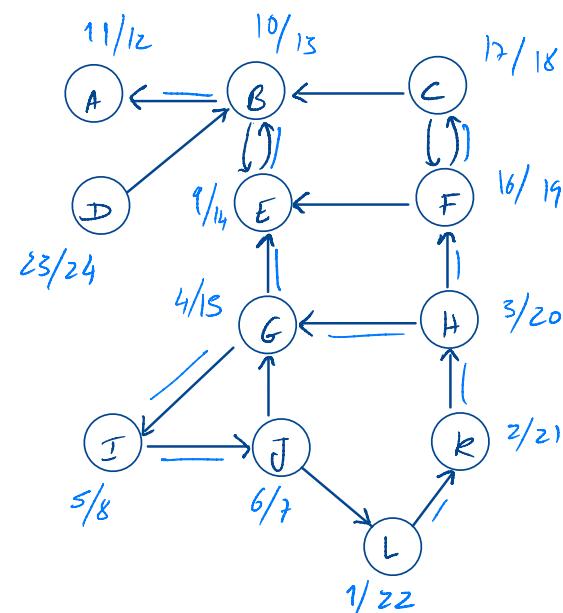


## Componentes Fortemente Ligados - Algoritmo de Kosaraju-Sharir

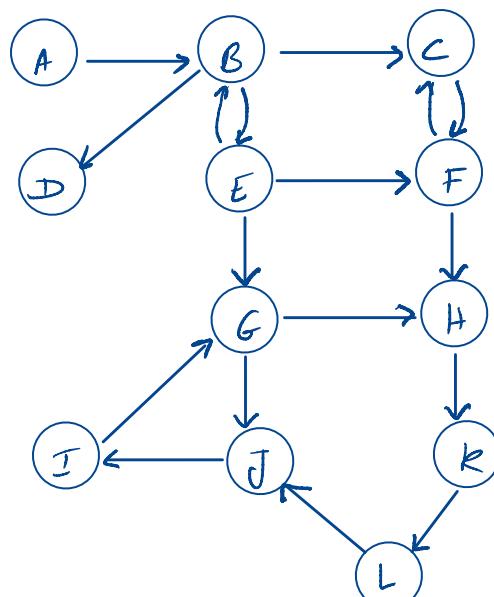
SCCs ( $G$ )

- Executar DFS ( $G$ ) para cálculo do tempo de fim
- Calcular  $G^T$
- Executar DFS ( $G^T$ ) escolhendo os nós por ordem inversa de tempo de fim na 1<sup>a</sup> DFS
  - O conjunto dos vértices de cada DFS corresponde a um SCC

1<sup>a</sup> DFS:



2<sup>a</sup> DFS:

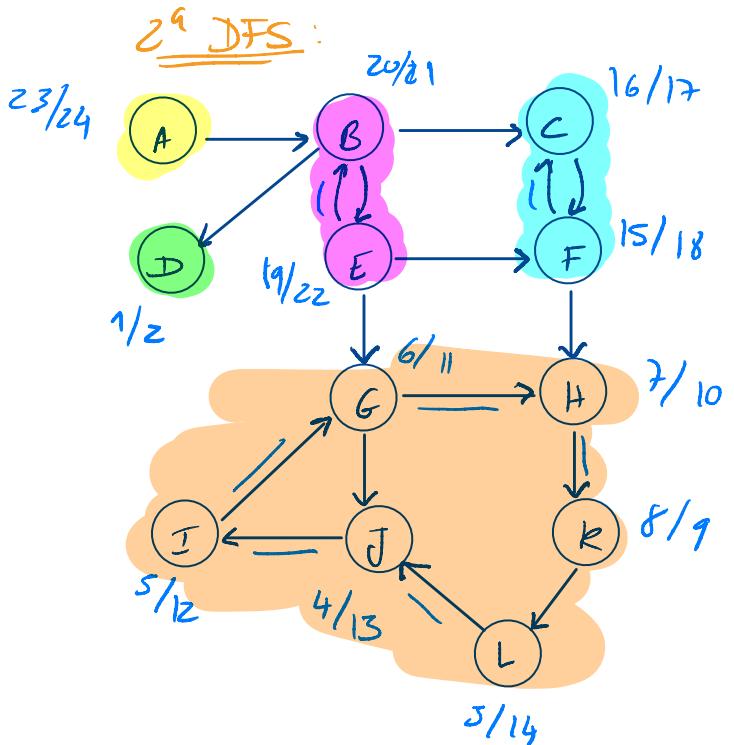
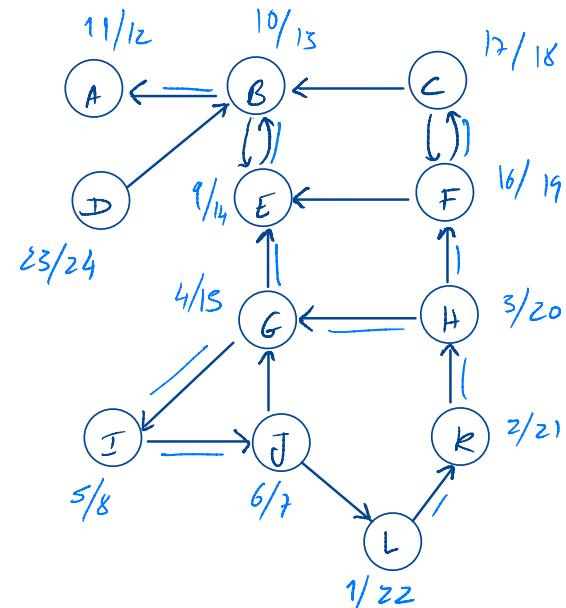


## Componentes Fortemente Ligados - Algoritmo de Kosaraju-Sharir

$\text{SCCs}(G)$

- Executar  $\text{DFS}(G)$  para cálculo do tempo de fim
- Calcular  $G^T$
- Executar  $\text{DFS}(G^T)$  escolhendo os nós por ordem inversa de tempo de fim na 1<sup>a</sup> DFS
  - O conjunto dos vértices de cada  $\text{DFS}$  corresponde a um SCC

1<sup>a</sup> DFS:



## Componentes Fortemente Ligados - Algoritmo de Kosaraju-Sharir

Consequência: Admitimos que as  $k$  primeiras árvores encontradas são SCCs e q a  $(k+1)$ -ésima árvore tb é um SCC.

- Seja  $x$  o primeiro vértice encontrado da  $(k+1)$ -ésima. Há dois factos a provar:

① Todos os vértices do SCC  $\bar{g}$  contém  $x$ ,  $C_x$ , são descendentes de  $x$ .

② Todos os descendentes de  $x$  pertencem a  $C_x$

## Componentes Fortemente Ligados - Algoritmo de Kosaraju-Sharir

Consequência: Admitimos que as  $k$  primeiras árvores encontradas são SCCs e q a  $(k+1)$ -ésima árvore tb é um SCC.

- Seja  $x$  o primeiro vértice encontrado da  $(k+1)$ -ésima. Há dois factos a provar:

- I Todos os vértices do SCC  $\bar{g}$  contêm  $x$ ,  $C_x$ , são descendentes de  $x$ .

$$y \in C_x \Rightarrow y \text{ é descendente de } x$$

- II Todos os descendentes de  $x$  pertencem a  $C_x$

$$y \text{ é descendente de } x \Rightarrow y \in C_x$$

Nota: Fim dos slides  
da Aula 6