

# (VERÃO COM ALGORITMOS)

AULA 1 - 20/12/18

ALGORITMO: UNION-FIND (CONJUNTOS DISJUNTOOS)

UNE( $u, v$ ):

$u = \text{FIND}(u)$

$v = \text{FIND}(v)$

SE  $\text{TAM}[u] \geq \text{TAM}[v]$ :

$\text{CONS}[v] = u$

$\text{TAM}[u] = \text{TAM}[u] + \text{TAM}[v]$

SENÃO:

$\text{CONS}[u] = v$

$\text{TAM}[v] = \text{TAM}[v] + \text{TAM}[u]$

FIND( $v$ ):

SE  $v = \text{CONS}[v]$ , RETORNE  $v$

RETORNE  $\text{CONS}[v] = \text{FIND}(\text{CONS}[v])$ .

ORDENAÇÃO TOPOLOGICA; ORDEM  $v_1, v_2, \dots, v_m$   
TAL QUE SE  $v_i v_j \in E(G)$ ,  $i < j$  NA ORDENAÇÃO

ALGORITMO: ORDENAÇÃO TOPOLOGICA

ENTRADA: DIGRAFO  $G = (V, E)$

SAÍDA:  $v_1, v_2, \dots, v_m$  UMA O.T. DE  $G$

1 DETERMINE  $g[v]$   $\forall v \in V(G)$

2  $F = \emptyset$

3 PARA TODO  $v \in V(G)$ :

4 SE  $g[v] = 0$ ,  $F = F \cup \{v\}$

5 PARA  $j$  DE 1 ATÉ  $n$ :

6 SE EXISTE  $v \in F$ ,  $F = F - \{v\}$

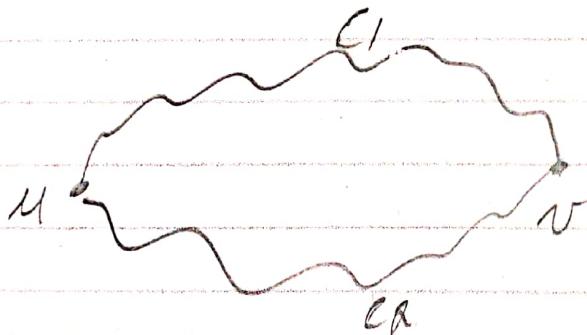
- 7  $v_i = N$   
 8 PARA TODO  $w \in \text{Adj}[v_i]$   
 9  $g[w] = g[v_i] -$   
 10 SE  $g[w] = 0$ ,  $F = F \cup \{w\}$

TEOREMA:  $G$  TEM O.T.  $\Leftrightarrow G$  NÃO TEM CICLO

ASSUMA  $c = v_1, v_2, \dots, v_k$  UM CICLO DE  $G$ .  
 QUE  $G$  TEM O.T., SEJA  $v_i$  O ÚLTIMO VERTICE NA O.T., COMO  $v_i \in c$ , SEMPRE HÁ VERA ARESTAS  $v_i v_{i+1}$ , ENTÃO  $v_i$  NÃO SERÁ O ÚLTIMO DA O.T. INDEPENDENTEMENTE DA ESCOLHA DE  $v_i$ . PORTANTO,  $G$  NÃO TEM ORDENAÇÃO TOPOLOGICA.

TEOREMA:  $G$  É ÁRVORE  $\Leftrightarrow$  HÁ 1 ÚNICO CAMINHO ENTRE CADA PAR DE VERTICES

( $\Rightarrow$ ) SUPONHA QUE HÁ 2 CAMINHOS DIFERENTES ENTRE  $u$  E  $v$  EM  $V(G)$ .



POSSO IR DE  $u$  ATÉ  $v$  POR  $c_1$  E RETORNAR A  $u$  POR  $c_2$ , FORMANDO ENTÃO UM CICLO  $c = c_1 c_2$ , ENTÃO  $G$  NÃO PODE SER ÁRVORE.

( $\Leftarrow$ ) SE HÁ CAMINHOS ENTRE TODOS  $u, v$ , ENTÃO  $G$  É CONEXO.

SE HÁ SOMENTE 1 CAMINHO ENTRE OS  $u, v$ , ENTÃO  $G$  NÃO TEM CICLOS (PROVA DE IDA).

CONEXO + NÃO TEM CICLOS = ÁRVORE

AULA 2 - 11/01/19

## PROPRIEDADES: ÁRVORES (CONTINUAÇÃO AULA 1)

- 1) TODA ÁRVORE COM  $n$  VERTICES POSSUI EXATAMENTE  $n-1$  ARESTAS
- 2) EM TODA ÁRVORE EXISTE UM ÚNICO CAMINHO ENTRE CADA PAR DE VERTICES
- 3) TODA ÁRVORE É UM GRAFO CONEXO COM  $n$  VERTICES E  $n-1$  ARESTAS
- 4) EM UMA ÁRVORE, PARA  $v, w$  NÃO ADJACENTES, A ADIÇÃO DA ARESTA  $vw$  PRODUZ EXATAMENTE UM CICLO.

ALGORITMO: GRAU DOS VERTICES

ENTRADA: GRAFO  $G = (V, E)$

SÍDA:  $g$ , VETOR COM GRAU DOS VERTICES

1 PARA TODO  $v \in V(G)$ :

2  $g[v] = 0$

3 PARA TODO  $w \in \text{Adj}[v]$ :

4  $g[v] = g[v] + 1$

CORRETUDE: DIRETAMENTE DO FATO DE QUE

$$g[v] = |\text{Adj}[v]|$$

COMPLEXIDADE (FAG 73 NOVO SAVME):

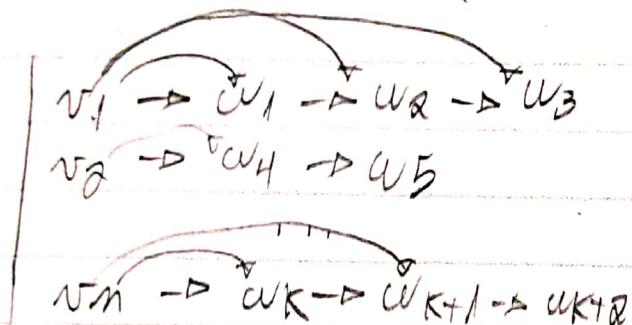
A DETERMINAÇÃO DO GRAU DE CADA VERTICE  $v$  (UNHA 1) REQUER  $O(1 + |\text{Adj}[v]|) = O(1 + g[v])$  OPERAÇÕES. LOGO, A COMPLEXIDADE DO ALGORITMO "GRAU DOS VERTICES" É  $O\left(\sum_{i=1}^n (1 + g[v_i])\right) = O\left(\sum_{i=1}^n 1 + \sum_{i=1}^n g[v_i]\right) = O(n + m)$

ALGORITMO  
ÓTIMO

## BUSCA EM GRAFOS

CARACTERÍSTICA: PASSAR POR TODAS AS ARESTAS EXATAMENTE 1 VEZ (GRAFOS DIRECIONADOS) OU EXATAMENTE Q VEZES (GRAFOS NÃO DIRECIONADOS)

EJS: VETOR CARACTERÍSTICO  $c$ : ACESSAR A INFORMAÇÃO SOBRE OS VERTÉCIS EM TEMPO CONSTANTE. CASO CONTRÁRIO, SERIA NECESSÁRIO PERCORRER A ESTRUTURA DE DADOS  $Q$ .



O ALGORITMO DE BUSCA GERAL AVANÇA O "PONTEIRO" DE CADA LISTA, IPOSIÇÃO POR VEZ:

ALGORITMO: BUSCA GERAL

ENTRADA: (DII) GRAFO CONEXO.  $G = (V, E)$

SAÍDA: SEQUÊNCIA DAS ARESTAS ORDENAÇÕES CONFORME SÃO VISITADAS.

- 1 PARA TODO  $v \in V(G)$ :
- 2  $c[v] = 0$ ,  $\pi[v] = -1$
- 3 SEJA  $m \in V(G)$ :
- 4  $Q = \{m\}$ ,  $\pi[m] = m$ ,  $c[m] = 1$
- 5 ENQUANTO  $Q \neq \emptyset$ :
- 6 SEJA  $v \in Q$
- 7 SE EXISTE  $w \in Adj[v]$  COM  $w$  NÃO VISITADA:
- 8 SE  $c[w] == 0$ :
- 9 VISITE  $I(vw)$ ,  $\pi[w] = v$ ,  $c[w] = 1$ ,  $Q = Q \cup \{w\}$
- 10 SENÃO VISITE  $R(vw)$
- 11 SENÃO  $c[v] = 2$ ,  $Q = Q \setminus \{v\}$

## CORRETUDE:

1) TODAS AS ARESTAS SÃO VISITADAS POIS SE  $v \in E(G)$  COM  $v \in A_d[x]$ , QUANDO  $x$  FOR CONSIDERADO NA LINHA 6, ENTÃO EM ALGUM MOMENTO  $x$  SERÁ VISITADA. COMO O GRAFO É CONEXO, TODO VERTICE  $v$  SERÁ INCLUÍDO EM  $Q$ .

COMO EXISTEM 2 MÉTODOS VISIT<sub>1</sub> E VISIT<sub>2</sub> QUE ATUALIZAM O PONTEIRO PARA A PRÓXIMA ARESTA NÃO VISITADA, UMA ÚNICA ARESTA EM  $A_d(x)$  NUNCA SERÁ VISITADA MAIS DO QUE UMA ÚNICA VEZ.

## COMPLEXIDADE:

ENTRADA: LISTA DE ADJACÊNCIAS:  $O(n+m)$

SÁIDA: ARESTAS NA SEQUÊNCIA EM QUE SÃO VISITADAS:  $O(m)$

EXECUÇÃO: 1-2  $\Rightarrow O(m)$

3-4  $\Rightarrow O(1)$

LAGO 5-11  $\Rightarrow O(n+m)$ , POIS TUDO DENTRO DO LAGO 5-11 É  $O(1)$ , E O LAGO EXTERNO É EXECUTADO  $\sum_{i=1}^m (deg(v_i)) = m + qm$  VEZES. PORTANTO,  $O(n+m)$

TOTAL:  $O(m) + O(1) + O(n+m) = O(n+m)$

<u>ESTRUTURA</u>	<u>BUSCA</u>	<u>DIGRAFO</u>	<u>GRAFO</u>
FILA	LARGURA (BFS)	TÍO, PRIMO, IRMÃO	
PILHA	PROFOUNDIDA DE (DFS)	RETORNO	AVANÇO, RETORNO, CRUZAMENTO

RES: ANÁLISE DA COMPLEXIDADE DA BUSCA GERAL SERVE TANTO PARA BFS COMO PARA DFS.

APLICAÇÕES:

BFS + DIGRAFO: DETECÇÃO DE CICLOS ÍMPARES,  
CALCULAR DISTÂNCIA

DFS + DIGRAFO: COMPONENTES BICONEXAS

DFS + GRAFO: ORDENAÇÃO TOPOLOGICA, COMPONENTES FORTEMENTE CONEXOS.

TEOREMA: A ESTRUTURA X FORNECE UMA ÁRVORE GERAADORA DE  $G$ .

X É CONSTRUÍDA A PARTIR DE ARESTAS QUE LIGAM UM VÉRTICE VISITADO  $x$  A UM VÉRTICE NÃO VISITADO  $y$ , SENDO QUE  $y$  PASSARÁ A SER VISITADO A PARTIR DESSE MOMENTO, LOGO EXISTEM NO MÁXIMO  $m-1$  ARESTAS.

PORÉM, COMO  $G$  É CONEXO, TODOS OS VÉRTICES SÃO VISITADOS, ENTÃO EXISTEM EXATAMENTE  $m-1$  ARESTAS.

$G$  É CONEXO POIS TODO VÉRTICE INCLUÍDO NA ESTRUTURA X É LIGADO A UM VÉRTICE QUE JÁ

PERTENCE À ESTRUTURA.

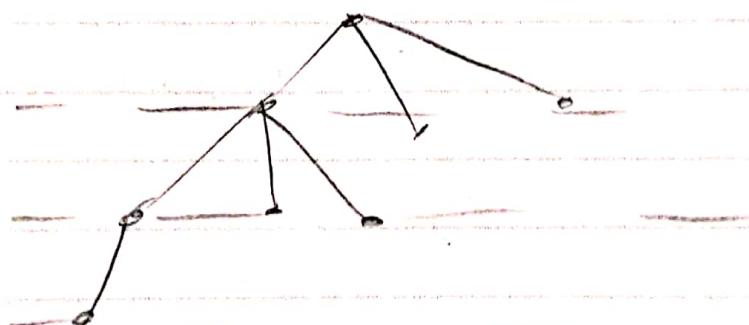
COMO TEM  $n-1$  ARESTAS E É CONEXA, A ESTRUTURA É UMA ÁRVORE

X) É CONSTRUÍDA DE MODO QUE NO FINAL DA EXECUÇÃO, TODO VERTICE  $v \in V(G)$  PERTENCE A  $T$ . ENTÃO,  $T$  É ÁRVORE GERADORA DE  $G$ .

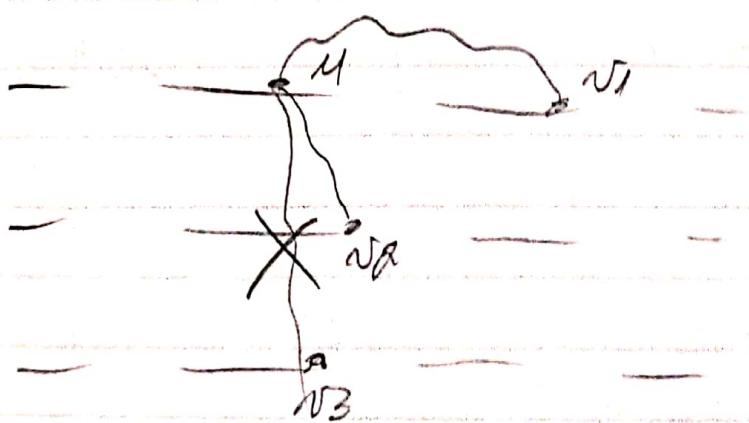
### PROPRIEDADES: BUSCA EM LARGURA (BFS)

II) SE  $uv \in E(G)$ , ENTÃO  $|Nível[u] - Nível[v]| \leq 1$  (OU ESTÃO NO MESMO NÍVEL, OU SÃO DESCENDENTES IMEDIATOS)

SUPONHA QUE  $L[u] < L[v]$



CONSEQUÊNCIA  
DIRETA DO FATO  
DE Q SER UMA  
FILA



2) DISTÂNCIA  $(u, v) = Nível[v]$ , COM A RAIZ DA BUSCA,  $\forall v \in V(G)$ .

## LISTA 2

① ALGORITMO: NÚMERO DE COMPONENTES CONEXAS  
ENTRADA: GRAFO  $G = (V, E)$  CONEXO  
SAÍDA: NCC, NÚMERO DE COMPONENTES CONEXAS

1  $NCC = 0$   
2 PARA TODO  $v \in V(G)$ :  
3  $C[v] = 0$   
4 ENQUANTO EXISTE  $v \in V(G)$  TAL QUE  $C[v] \neq \emptyset$ :  
5 SEJA  $h \in V(G)$  A RAIZ DA BUSCA  
6  $Q = \{h\}$ ,  $\pi[h] = h$ ,  $C[h] = 1$   
7 ENQUANTO  $Q \neq \emptyset$ :  
8 SEJA  $v \in Q$ .  
9 SE EXISTE  $w \in \text{ADJ}[v]$  COM  $w$  NÃO VISITADA:  
10 SE  $C[w] = 0$ ; VISITE  $l(w)$ ,  $\pi[w] = v$ ,  
 $C[w] = 1$ ,  $Q = Q \cup \{w\}$   
11 SENÃO VISITE  $g(w)$   
12 SENÃO  $C[v] = \emptyset$ ,  $Q = Q \setminus \{v\}$

CORRETUDE: IGUAL A DO ALGORITMO "BUSCA GERAL"

COMPLEXIDADE:

LAGO 4-12 :  $O(n_1 + m_1) + O(n_2 + m_2) + \dots + O(n_t + m_t) =$   
 $O(n + m) = O(n + m)$

② ALTERAR NA LINHA 7 DO ALGORITMO "BUSCA GERAL": ONDE ERA "SE EXISTE  $w$ " SE Torna "PARA TODO  $w$ ".

# AULA 3 - 15/01/2018

## BUSCA EM PROFUNDIDADE (EDFS)

CRITÉRIO DE ESCOLHA DAS ARESTAS: VISITE A PRIMEIRA ARESTA DESCOBERTA A CADA ITERAÇÃO

### ALGORITMO: DFS

ENTRADA: GRAFO  $G = (V, E)$

SÍLDA: ARESTAS  $e_1, e_2, \dots, e_n$  NA ORDEM EM QUE SÃO VISITADAS; ARV, CRU, RET, AV CONJUNTOS DE ARESTAS CLASSIFICADAS.

- 1 PARA TODO  $v \in V(G)$ ,  $c[v] = 0$ ,  $\pi[v] = -1$ ,  $pe[v] = -1$ ,  $ps[v] = -1$
- 2  $CRU = \{z\}$ ,  $ARV = \{z\}$ ,  $AV = \{z\}$ ,  $RET = \{z\}$
- 3 SEJA  $H$  A RAIZ DA BUSCA
- 4  $Q = \{H\}$ ,  $\pi[H] = n$ ,  $cpe = 1$ ,  $pe[H] = 1$ ,  $cps = 0$
- 5 ENQUANTO  $Q \neq \emptyset$ :
  - 6 SE EXISTE  $w \in Adj(H)$  COM  $w$  NÃO VISITADA:
    - 7 SE  $c[w] = 0$ :
      - 8  $ARV = ARV \cup \{vw\}$ ,  $c[w] = 1$ ,  $\pi[w] = v$ ,  
VISITAR  $w$ )
      - 9  $Q = Q \cup \{w\}$ ,  $cpe = cpe + 1$ ,  $pe[w] = cpe$
    - 10 SENÃO VISITA  $\varphi(w)$ , CLASSIFICAR  $(v, w)$
    - 11 SENÃO  $c[w] = 2$ ,  $Q = Q \setminus \{w\}$ ,  $cps = cps + 1$ ,  $ps[w] = cps$

### CLASSIFICAR ( $v, w$ ):

- 1 SE  $pe[w] < pe[v]$ ,  $AV = AV \cup \{vw\}$

- 2 SENÃO:

- 3 SE  $c[w] = 1$  E  $c[v] = 2$ ,  $CRU = CRU \cup \{vw\}$
- 4 SE  $c[w] = 1$  E  $c[v] = 1$ ,  $RET = RET \cup \{vw\}$

## ORDENAGÃO TOPOLOGICA COM DFS

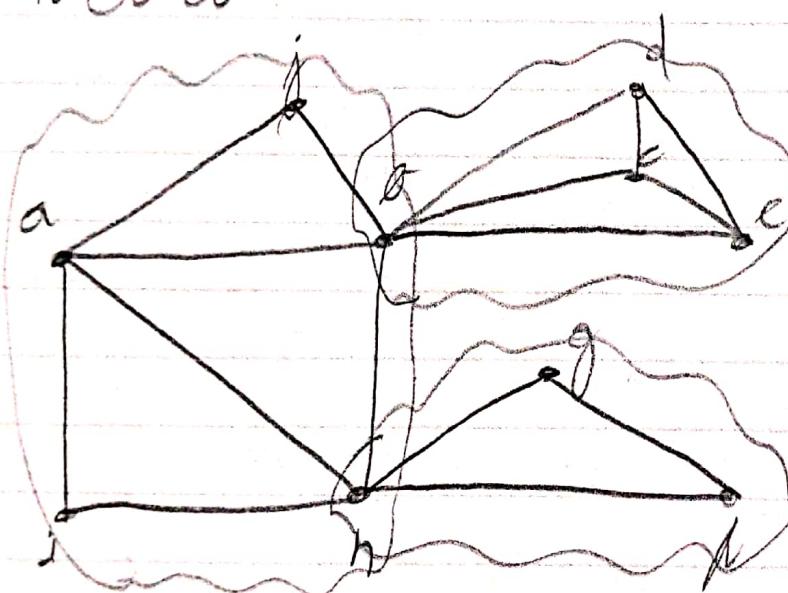
1) FAÇA UMA BUSCA EM PROFUNDIDADE, COMPUTANDO PE E PS

2) ORDENE OS VÉRTICES  $v_1, v_2, \dots, v_m$  DE FORMA QUE  $PS(v_1) \geq PS(v_2) \geq \dots \geq PS(v_m)$

## COMPONENTES BICONEXAS

obs:  $v$  É ARTICULAÇÃO SE:

- $v$  É RAIZ E POSSUI MAIS DE 1 FILHO
- $v$  NÃO É RAIZ E POSSUI FILHO  $w$  TAL QUE  $lowpt(w) = v$



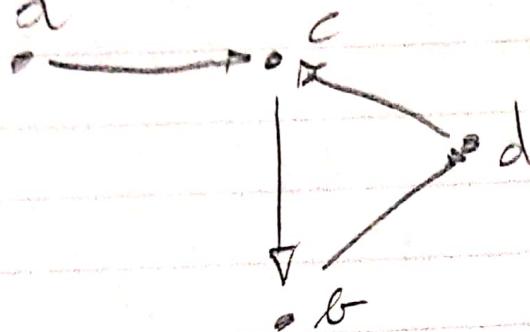
v	t	lowpt	g
a	-1		a
b	a	a	b
c	b	b	c
d	f	b	b
e	f	b	b
f	h	h	f
g	h	h	h
h	o	a	a
i	h	a	a
j	b	a	a

CORRETUDE (ALGORITMO CBICONX): OLHAR LEMA 4.1 NO LIVRO DO SAYME

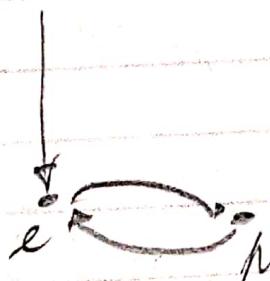
## COMPONENTES FORTEMENTE CONEXAS

obs:  $\text{low}(v) = z$ , ONDE  $z \in O VERTICE COM MENOR PE NO MESMO CFC QUE } v$

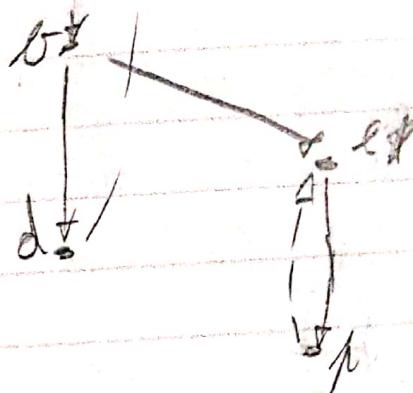
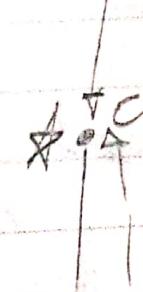
$a$



TARJAN



$a$



$v$	$\pi$	PE	low
a	-1	1	1
b	c	3	2
c	a	2	2
d	b	4	2
e	b	5	5
f	e	6	5

SE  $\text{low}[v] == \text{PE}[v]$

VERTICE FORTÉ

## COLORAÇÃO

FUNÇÃO  $c: V(G) \rightarrow \{1, 2, \dots, k\}$  TAL QUE PARA TODA  $u, v \in E(G)$ ,  $c(u) \neq c(v)$

NÚMERO CROMÁTICO DE UM GRAFO ( $\chi(G)$ ): MENOR NÚMERO DE CORES EM QUE EXISTE UMA COLORAÇÃO VÁLIDA

ALGORITMO: ZIKOV

ENTRADA: GRAFO  $G = (V, E)$

SAÍDA:  $\chi(G)$ ,  $c: V \rightarrow \{1, 2, \dots, \chi(G)\}$  COLORAÇÃO DE  $G$

1 SE  $G = K_p$ ,  $\chi(G) = p$

2 SE NÃO, SEJA M, N  $\in V(G)$  TAL QUE  $M \notin V(G)$

3  $\chi(G) = \min\{\chi(G+MN), \chi(G-MN)\}$

CORRETUDE:

TEOREMA 5.1: SEJA G UM GRAFO NÃO COMPLETO E  $u, v$  UM PAR DE VERTÉICES NÃO ADJACENTES. ENTÃO, O NÚMERO CROMÁTICO SATISFAZ

$$\chi(G) = \min\{\chi(G+uv), \chi(G-uv)\} - 1$$

PROVA: PARA  $u, v$  NÃO ADJACENTES:

SE  $c(u) = c(v)$ :

$$\chi(G+uv) \geq \chi(G)$$

$$\chi(G-uv) = \chi(G)$$

SE  $c(u) \neq c(v)$ :

$$\chi(G+uv) = \chi(G)$$

—

ENTÃO, A AFIRMATIVA I) VALE ■

AULA 4 - 22/01/2019

MÉTODO GULOSO: SEMPRE ESCOLHA O MELHOR ELEMENTO DISPONÍVEL DE ACORDO COM O CRITÉRIO GULOSO (QUE NUNCA MUDA), E NÃO SE ARREPENDE.

PROBLEMA DA ÁRVORE GERADORA MÍNIMA: DADO  $G = (V, E)$ , ENCONTRAR  $T = (V, A)$  TAL QUE  $T$  É UMA ÁRVORE GERADORA,  $A \subseteq E$ , COM  $\sum_{e \in A} p(e)$  MÍNIMO.

ALGORITMO: KRUSKAL

ENTRADA:  $G = (V, E)$  GRAFO CONEXO,  $p: E \rightarrow \mathbb{R}$  FUNÇÃO PESO

SAÍDA:  $T = (V, A)$  ÁRVORE GERADORA DE PESO MÍNIMO  
 $P = \sum_{e \in A} p(e)$ , PESO DE  $T$

0 PARA TODO  $v \in V(G)$ , SET( $v$ )  
1 ORDENE  $E(G)$  TAL QUE  $p(e_1) \le \dots \le p(e_m)$   
2  $P = 0$ ,  $A = \emptyset$   
3 PARA TODA  $(u, v) \in E(G)$ :  
4      $X = \text{FIND}(u)$ ,  $Y = \text{FIND}(v)$   
5     SE  $X \neq Y$ :  
6          $A = A \cup \{(u, v)\}$ ,  $P = P + p(u, v)$   
7         UNIR( $X, Y$ )

$\text{FIND}(A)$ :

1 SE  $A == \text{CONS}[A]$ , RETORNE  $A$  ]  $O(\log m)$   
2 RETORNE  $\text{CONS}[A] = \text{FIND}(\text{CONS}[A])$

$\text{SET}(v)$

1  $\text{CONS}[v] = v$  ]  $O(1)$   
2  $\text{TAM}[v] = 1$

UNIR(U,V)

1  $A = \text{FIND}(U)$

2  $B = \text{FIND}(V)$

3 SE  $\text{TAM}[A] > \text{TAM}[B]$ :

4  $\text{CONJ}[B] = A$

5  $\text{TAM}[A] = \text{TAM}[A] + \text{TAM}[B]$

6 SENÃO:

7  $\text{CONJ}[A] = B$

8  $\text{TAM}[B] = \text{TAM}[B] + \text{TAM}[A]$

$O(\log m)$

## COMPLEXIDADE DE TEMPO:

LINHA 1:  $O(m \log m) \rightarrow$  ORDENAÇÃO POR COMPARAÇÃO

LINHAS 3-7: REALIZA PARA CADA ARESTA, 2 OPERAÇÕES FIND(), QUE POR SUA VEZ GASTAM  $O(\log m)$ .  
POR CONSEQUENTE:

$$\sum_{j=1}^m 2 \log(m) = 2m \cdot \log(m) = O(m \log(m))$$

TOTAL:  $O(m \log(m))$

## COMPLEXIDADE DE ESPAÇO:

ENTRADA: GRAFO  $G = (V, E) \rightarrow O(n+m)$

SÁIDA:  $A \subseteq E$ , ARESTAS DA A,  $G, M \rightarrow O(m)$   
VETORES AUXILIARES  $\text{CONJ}[ ]$  E  $\text{TAM}[ ] \rightarrow O(m)$

TOTAL:  $O(n+m)$

CORRETUDE:

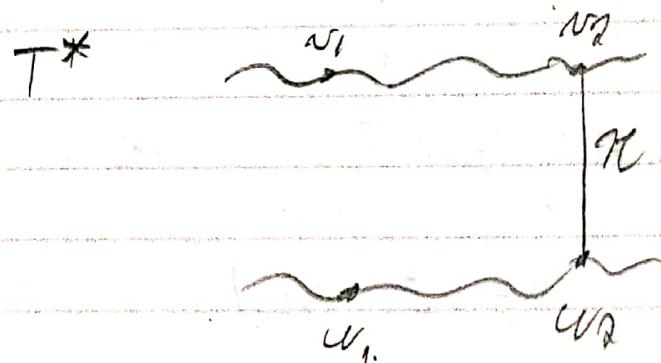
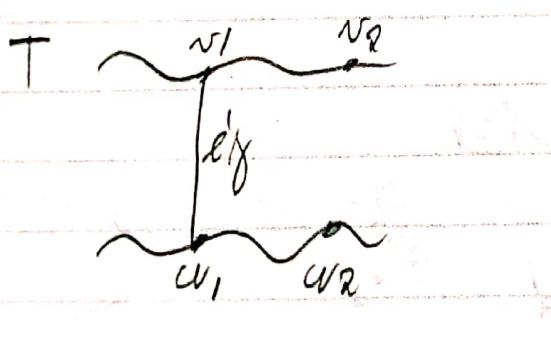
SEJA  $T = (V, A)$  A ÁRVORE GERADORA CONSTRUÍDA PELO ALGORITMO, LOGO  $P = \text{PESO}(T) = \sum p(e)$ , SEJAM  $e'_1, e'_2, \dots, e'_{m-1}$  AS ARESTAS NA ORDEM EM QUE FORAM ESCOLHIDAS PELO ALGORITMO.  
LOGO  $p(e'_1) \leq p(e'_2) \leq \dots \leq p(e'_{m-1})$

SUPONHA POR ABSURDO QUE  $T$  NÃO É MÍNIMA.

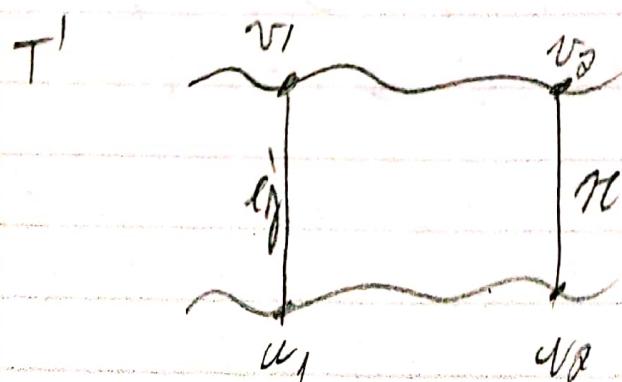
SEJA  $T^*$  UMA ÁRVORE GERADORA MÍNIMA DE  $G$  COM SEQUÊNCIA DE ARESTAS INICIAS (ESCOLHIDAS PELO ALGORITMO) EM COMUM COM  $T$  O MAIOR POSSÍVEL.

$$T = -e'_1, e'_2, \dots, e'_{j-1}, e'_j, \dots, e'_{m-1}$$

$$T^* = -e'_1, e'_2, \dots, e'_{j-1}, \cancel{e'_j}, \dots, e'_{m-1}$$



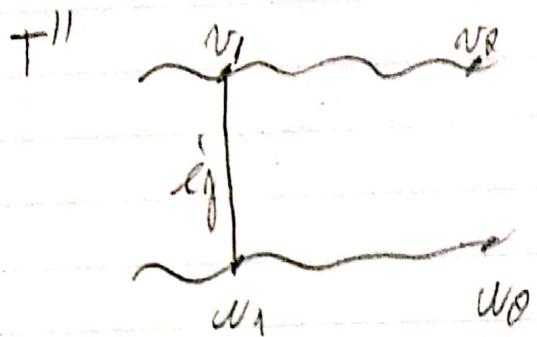
$T' = T^* + e'_j$  TEM EXATAMENTE 1 CICLO



$\exists \pi \in E(T^*) \setminus E(T)$  NO CICLO

$\pi$  É ALGUMA DAS ARESTAS

ENTÃO  $T'' = T^* + e'j - \pi$  É ÁRVORE GERADORA DE  $G$



$\pi$  É UMA DAS ARESTAS DE  $f_1 \wedge f_{n-1}$  EM  $T^*$ . ENTÃO  $P(\pi) \geq P(e'j)$ , LOGO, PESO( $T''$ )  $\leq$  PESO( $T^*$ ). NO entanto,  $T$  É MÍNIMA, ENTÃO PESO( $T''$ ) = PESO( $T^*$ ), O QUE É UMA CONTRADIÇÃO, POIS

$T' = e'_1, e'_2, \dots, e'_{j-1}, e'j, f_{j+1}, \dots, f_{n-1}$  TEM MAIS ARESTAS EM COMUM COM  $T$  DO QUE  $T^*$ .

ALGORITMO : PRIM

ENTRADA : GRAFO CONEXO  $G = (V, E)$

SAIDA :  $T = (V, A)$  ÁRVORE GERADORA MÍNIMA DE  $G$

$$P = \sum_{e \in A} P(e) = P(T)$$

1 PARA TODO  $v \in V(G)$ ,  $q[v] = +\infty$ ,  $\pi[v] = -1$

2 SEJA  $v \in V(G)$ , SEJA  $Q = \{v\}$  HEAP DE MÍNIMO

3  $q[v] = 0$ ,  $\pi[v] = v$ ,  $P(v \cap [v]) = 0$ ,

4  $P = 0$ , CONSTRUA  $Q$  COM PRIORIDADE  $q$ ,  $A = \emptyset$

5 ENQUANTO  $Q \neq \emptyset$ :

6 SEJA  $v$  O VERTICE DE MÍNIMA PRIORIDADE EM  $Q$ ,

7  $Q = Q \setminus \{v\}$ ,  $P = P + P(v \cap [v])$ ,  $A = A \cup \{v \cap [v]\}$

8 PARA TODO  $w \in A \setminus \{v\}$  TAL QUE  $w \in Q$ :

9 SE  $P(vw) < q[w]$ :

$$q[w] = P(vw), \pi[w] = v$$

CORRETUDE! DIFÍCIL!

COMPLEXIDADE DE TEMPO:

LINHA 4:  $O(m)$

LINHA 7:  $O(\log(m))$  REMOVE MÍNIMO

LINHA 10:  $O(\log(m))$  DIMINUI PRIORIDADE

LAÇO 8-10:  $O(\log(m))$

LINHAS 5-7:  $O(\log(m))$

LAÇO 5-10:

CADA VÉRTICE DE VG É CONSIDERADO EXATAMENTE 1 VEZ. ENTÃO:

$$\sum_{i=1}^n \left[ O(1) + O(\log(m)) + g(v_i) \cdot (O(1) + O(\log(m))) \right]$$

$$O(n) + O(n \log(m)) + O(n \log(m)) = O(n \log(m))$$

TOTAL:  $O(n \log(m))$

COMPLEXIDADE DE ESPAÇO:

ENTRADA: GRAFO  $G = (V, E) \rightarrow O(n+m)$

SAIDA:  $A \in E \rightarrow O(m)$

VETOR DE PRIORIDADES  $g \rightarrow O(n)$

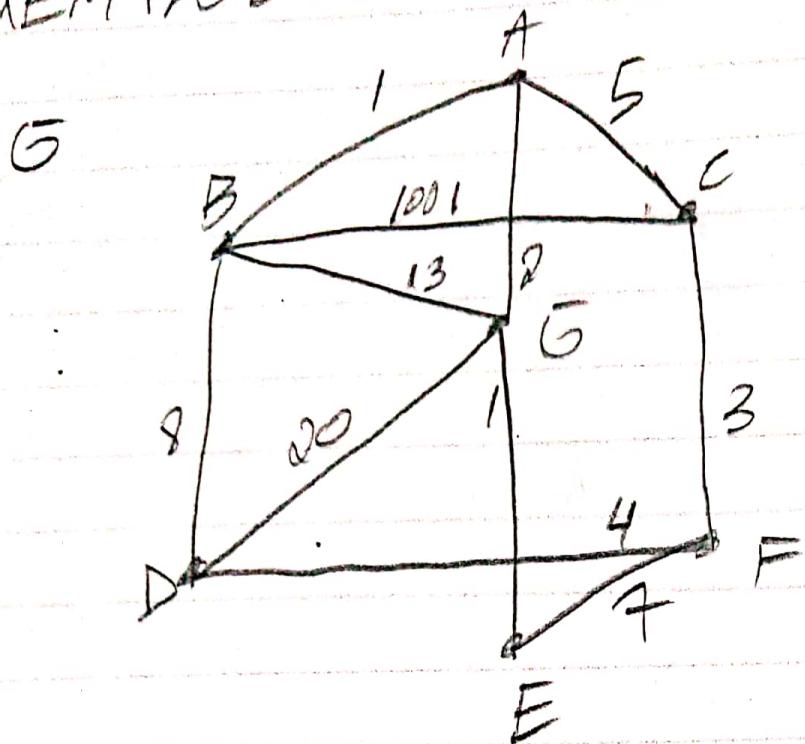
HEAP DE MÍNIMO:  $O(m)$

TOTAL:  $O(n+m)$

EJ: PARA CADA: CADA 1 DE UMA VEZ

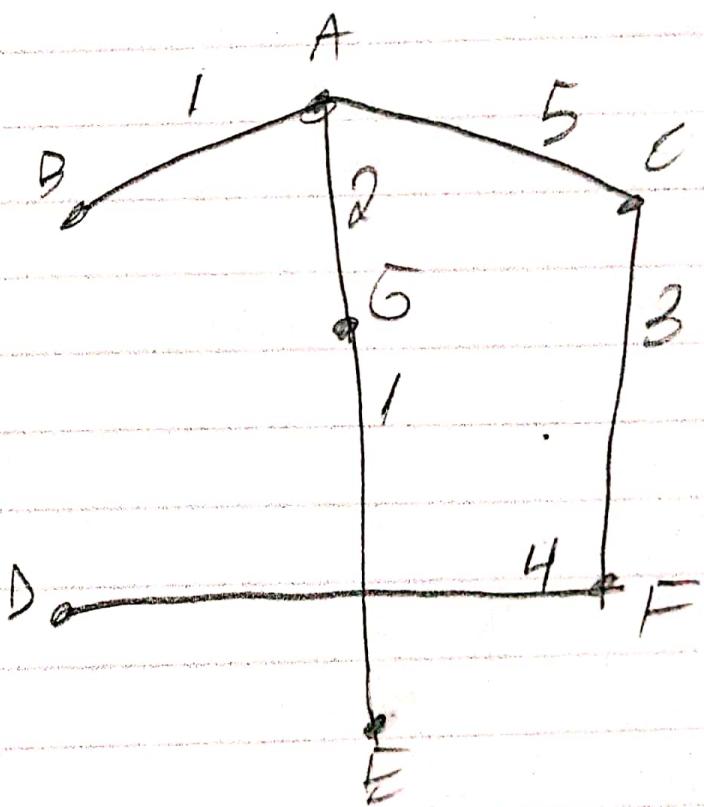
PARA TODO: TODOS DE UMA VEZ

EXEMPLOS:



COM KRUSKAL:

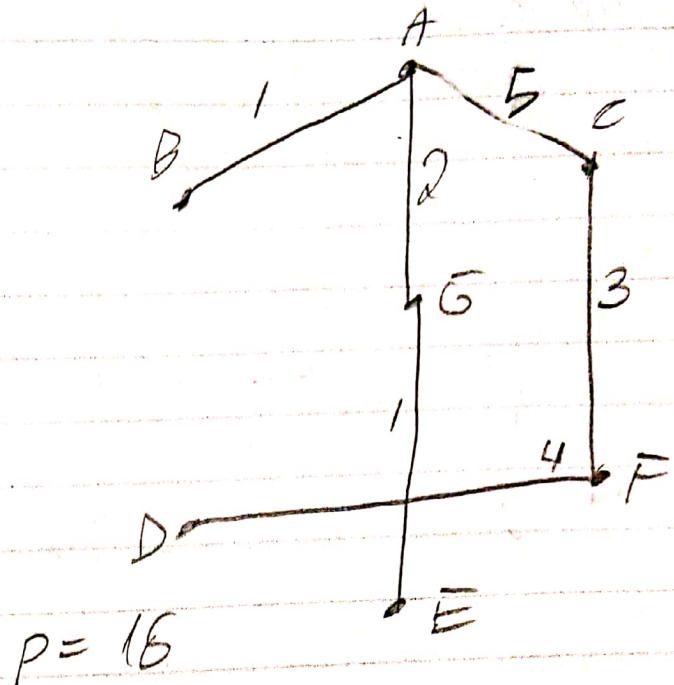
$e$	$P(e)$
AB	1
EG	1
AG	2
CF	3
DF	4
AC	5
EF	7
BD	8
BG	13
DG	20
BE	1001



$$P = 16$$

COM PRIM

V	G	N[V]
A	0	A
B	1	A
C	5	A
D	4	F
E	1	G
F	3	C
G	2	A

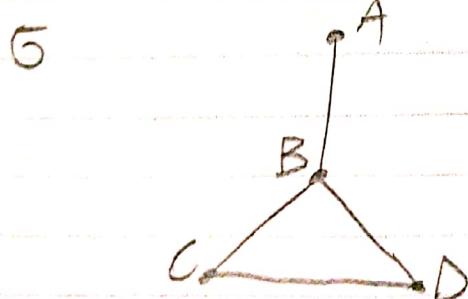


AULA 5 - 25/01/19

REVISÃO

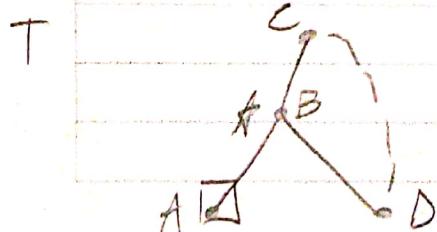
COMPONENTES BICONEXAS: SUBGRAFO TAL QUE  
É NECESSÁRIA A RETIRADA DE PELÔ MENOS  
QUE VERTICES PARA DESCONECTÁ-LO OU DEI-  
XÁ-LO TRIVIAL.

obs: TODO FILHO DE RAIZ É DEMARCADOR

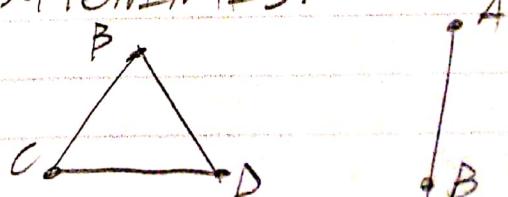


V	A	B	C	D
g	A	B	C	C
raiz	A	C	C	C

NÍVEL

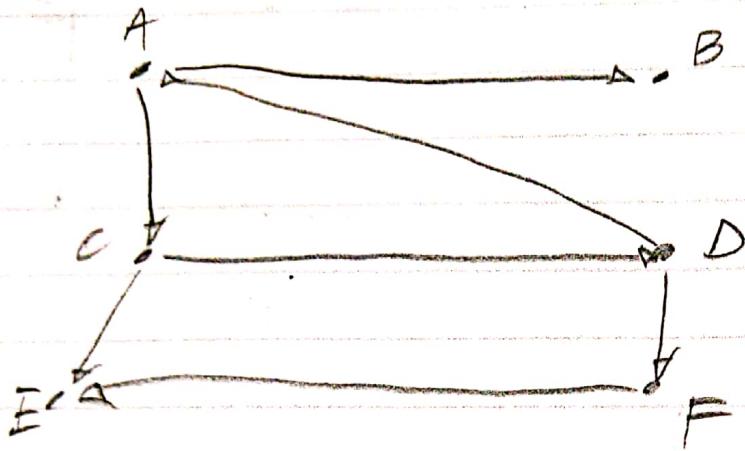


COM PONENTES:



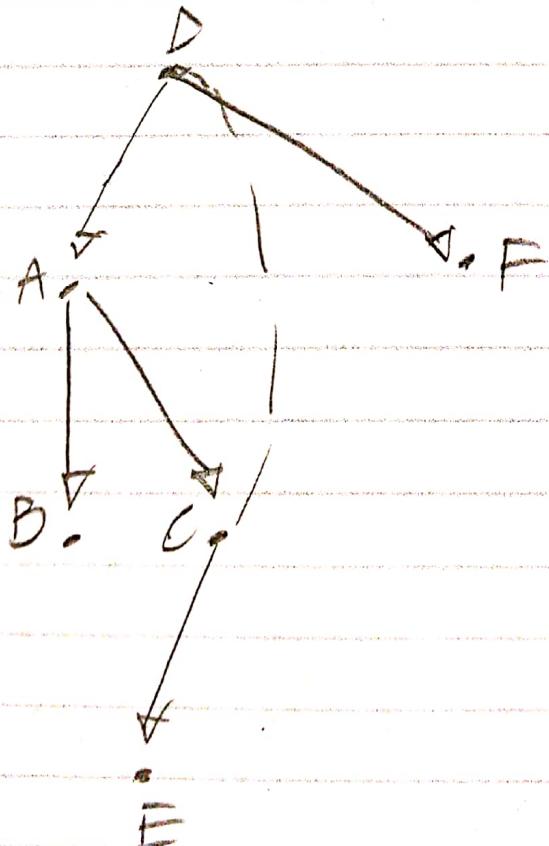
COMPONENTES FORTEMENTE CONEXAS:  
 SUBGRAFO  $G' = (V' \cup E')$  TAL QUE HAY CARRILHO ENTRE TODOS  $v, v' \in V'(G')$ .

Ex:



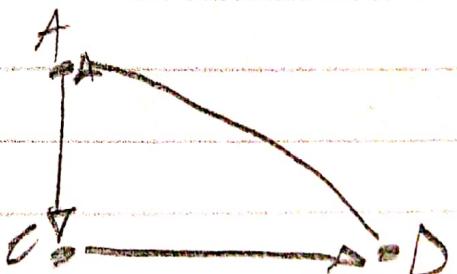
$v$	A	B	C	D	E	F
Con	1	3			5	6
PE	2	3	4	1	5	6

T



COMPONENTES:

$B - E - F$



## ALGORITMO DE KOSARAJU

1) FAÇA UMA BUSCA EM PROFUNDIDADE

2) INVERTA A DIREÇÃO DAS ARESTAS

3) FAÇA OUTRA BUSCA EM PROFUNDIDADE, ENRAIZANDO POR ORDEM DECREScente DE PS

4) AS ÁRVORES NA FLORESTA DE BUSCA SÃO AS CFC'S.

COLORAÇÃO: FUNÇÃO  $c: V \rightarrow \{1, 2, \dots, k\}$   
TAL QUE SE  $uv \in E(G)$ ,  $c(u) \neq c(v)$

B-COLORÍVEL:  $G = (V, E)$  É B-COLORÍVEL  
SE TEM COLORAÇÃO COM B-CORES

B-CROMÁTICO:  $G = (V, E)$  É B-CROMÁTICO QUANDO PÉ MENOR NÚMERO DE CORES EM QUE  $G$  TEM UMA COLORAÇÃO.

PROBLEMA DO NÚMERO CROMÁTICO: ENCONTRAR B

Ex 8 LISTA 2

⑧ MOSTRE QUE  $G$  É FORTEMENTE CONEXO SSE TODA BUSCA EM PROFUNDIDADE GERA UMA (ÚNICA) ÁRVORE DE BUSCA.

( $\Rightarrow$ ) SEJA  $G = (V, E)$  FORTEMENTE CONEXO. SUPONHA QUE UMA BUSCA DE RAIZ INICIAL N QUE GERE UMA FLORESTA.

SÓ HA VERÁ UMA NOVA RAIZ NA BUSCA SE EXISTE  $v \in V(G)$  TAL QUE  $v$  NÃO FOI MARCADO

NA BUSCA DE RAIZ N, LOGO, NÃO EXISTE CAMINHO  
DE M ATÉ N EM G, POIS É UMA CONTRADIÇÃO,  
POIS G É FORTEMENTE CONEXO.

AULA 6 - 29/01/2019

## SELEÇÃO DE ATIVIDADES

DEFINIÇÃO DO PROBLEMA: DADOS  $C = \{(i, j)\}_{i=1}^m$   
 $(c_i, t_i)$  ATIVIDADES COM HORÁRIOS DE INI-  
CIO  $c_i$  E DE TÉRMINO  $t_i$ . QUERO ENCONTRAR O  
MAIOR CONJUNTO DE ATIVIDADES QUE SÃO COM-  
PATÍVEIS.

CRITÉRIO GULOSO: PEGUE A ATIVIDADE QUE  
TERMINA PRIMEIRO

ALGORITMO: SELEÇÃO DE ATIVIDADES

ENTRADA:  $C = \{(i, j)\}_{i=1}^m$  CONJUNTO DE  
ATIVIDADES

SADA:  $S$ , MAIOR CONJUNTO DE ATIVIDADES  
COMPATÍVEIS.

1 ORDENE  $C$  TAL QUE  $j_1 < \dots < j_m$

2  $S = \{A_1\}$  ( $A_1 =$ ATIVIDADE 1)

3 PARA  $i$  DE 2 ATÉ  $m$ :

4 SEJA  $B$  A ÚLTIMA ATIVIDADE EM  $S$

5  $A_i =$ ATIVIDADE  $i$

6 SE  $A_i \cdot t > B \cdot e$ :

7  $S = S \cup \{A_i\}$

## COMPLEXIDADE DE TEMPO:

LINHA 1  $\rightarrow O(n \log(m))$  ORDENAÇÃO  
LAGO  $\varnothing - \overline{V}$   $\rightarrow O(m)$

TOTAL:  $O(n \log(m))$

## CAMINHOS MÍNIMOS

DEFINIÇÃO DO PROBLEMA: DADO UM GRAFO

$G = (V, E)$  E  $P: E \rightarrow \mathbb{R}^+$ , QUERO O PESO DO  
MENOR CAMINHO DE  $u$  ATÉ  $v$ ,  $\forall v \in V(G)$

ALGORITMO: DIJKSTRA

ENTRADA: GRAFO CONEXO  $G = (V, E)$ ,  $P: E \rightarrow \mathbb{R}^+$  PESO

SÁIDA:  $q[n] =$  PESO DO MENOR CAMINHO DE  $u$

ATÉ  $v$ ,  $\forall v \in V(G)$

1 SEJA  $u \in V(G)$

2  $q[u] = 0$ ,  $\pi[u] = u$

3 PARA TODO  $v \in V(G) \setminus \{u\}$ ,  $q[v] = \infty$ ,  $\pi[v] = -1$  [0m]

4 SEJA  $Q$  UMA HEAP DE MÍNIMO COM PRIORIDADE  $q$  [0m]

5 ENQUANTO  $Q \neq \emptyset$ :

6  $u =$  VÉRTICE DE MENOR PRIORIDADE EM  $Q$

7  $Q = Q \setminus \{u\}$

8 PARA CADA  $v \in \text{Adj}(u)$  COM  $v \in Q$ :

9 SE  $q[v] > p(u, v) + q[u]$

10  $q[v] = q[u] + p(u, v)$   $O(\log(m))$

11  $\pi[v] = u$

RES: DIJKSTRA NÃO FUNCIONA COM PESOS NEGATIVOS

COMPLEXIDADE:

$$O(n) + O(n) + \sum_{i=1}^n (O(\log(n)) + O(\log(n))) = \\ O(n) + O(n \log(n)) + O(n \log(n)) \\ = O(n \log(n))$$

TOTAL:  $O(n \log(n))$

## FLOYD-WARSHALL

DADO  $V(G) = \{v_1, v_2, \dots, v_n\}$

$w_{ij}$  = PESO DO MENOR CAMINHO DE  $v_i$  ATÉ  $v_j$   
EM G

INITIALIZE W COM

$$w_{ij} = \begin{cases} \infty, & \text{SE } v_i \neq v_j \in E(G) \\ p(v_i v_j), & \text{SE } v_i = v_j \in V(G) \\ 0, & \text{SE } i = j \end{cases}$$

A CADA ITERAÇÃO K, COMPARE SE

$$w_{ij}^k = \min \{ w_{ij}^{k-1}, w_{ik}^{k-1} + w_{kj}^{k-1} \}$$

APÓS n ITERAÇÕES:  $w_{ij}^n$  CONTÉM O PESO DO CAMINHO MÍNIMO DE  $v_i$  ATÉ  $v_j$ .

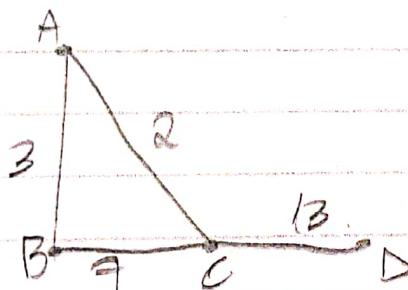
# ALGORITMO: FLOYD-MARSHALL

ENTRADA:  $G = (V, E)$  GRAFO CONEXO,  $P: E \rightarrow \mathbb{Q}^+$ , PESO

SAIDA:  $W$ , MATRIZ EM QUE  $w_{ij} =$  PESO DO MENOR CAMINHO DE  $v_i$  ATÉ  $v_j$ ,  $v_i, v_j \in V(G)$ .

- 1 PARA  $i$  DE 1 ATÉ  $n$ :
- 2 PARA  $j$  DE 1 ATÉ  $n$ :
- 3 SE  $i = j$ ,  $w_{ij} = 0$
- 4 SENÃO SE  $v_j \in V(G)$ ,  $w_{ij} = p(v_iv_j)$
- 5 SENÃO  $w_{ij} = \infty$
- 6 PARA  $k$  DE 1 ATÉ  $n$ :
- 7 PARA  $i$  DE 1 ATÉ  $n$ :
- 8 PARA  $j$  DE 1 ATÉ  $n$ :
- 9  $w_{ij} = \min\{w_{ij}, w_{ik} + w_{kj}\}$

Ex:



$W$	A	B	C	D
A	0	3	2	15
B	3	0	5	18
C	2	5	0	13
D	15	18	13	0

COMO DETECTAR CICLOS NEGATIVOS? SE ALGUM  $w_{ii} < 0$ ,  $1 \leq i \leq n$

COMO SABER QUAIS OS CAMINHOS?

GUARDAR NA TABELA  $\pi$

$\pi_{ij} =$  VERTICE IMEDIATAMENTE ANTERIOR

A  $v_j$  NO CAMINHO DE  $v_i$  ATÉ  $v_j$

ALTERAÇÃO NO ALGORITMO:

SE  $w_{ik} + w_{kj} < w_{ij}$

$\pi_{ij} = k$ ,  $w_{ij} = w_{ik} + w_{kj}$



NO EXEMPLO ANTERIOR:

H	A	B	C	D
A	-	A	A	C
B	B	-	A	C
C	C	A	-	C
D	C	A	D	-

## PROGRAMAÇÃO DINÂMICA

- PROPRIEDADE DA SUBESTRUTURA ÓTIMA:
  - o PROBLEMA PODE SER RESOLVIDO COMBINANDO A SOLUÇÃO DOS SEUS PROBLEMAS MENORES (DIVIDIR PARA CONQUISTAR)
- "OVERLAPPING SUBPROBLEMS"  
SUBPROBLEMAS IGUAIS SÃO RESOLVIDOS MULTIPHAS VEZES

## PROBLEMA DA MAIOR SUBSEQUÊNCIA COMUM:

DADAS DUAS STRINGS A E B, DESCUBRA QUAL OTAMANHO DA MAIOR SUBSEQUÊNCIA COMUM ENTRE AS DUAS

Ex:  $A = AXYT$        $B = AYZX$   
 $|A| = m$        $|B| = n$

INICIALIZO  $M_{0j} = M_{i0} = 0$

M	-	A	Y	Z	X
-	0	0	0	0	0
A	0	1	1	1	1
X	0	1	1	1	2
Y	0	1	2	2	2
T	0	1	2	2	②

→ Tamanho da maior subseqüência comum está em  $M_{mn}$

## RELAÇÃO DE RECORRÊNCIA:

$$M_{ij} = \begin{cases} M_{i-1, j-1} + 1, & \text{SE } A_i = B_j \\ \max\{M_{i-1, j}, M_{i, j-1}\}, & \text{SE } A_i \neq B_j \end{cases}$$

ALGORITMO: MAIOR SUBSEQUÊNCIA COMUM (LCS)

ENTRADA:  $X = (x_1, \dots, x_m) \in V = (y_1, \dots, y_n)$  DOTS  
STRINGS

SÁIDA:  $N$ , TAMANHO DA LCS

```

1 SE  $SAM$   $m = |X| \equiv m = |Y|$  O(1)
2 PARA  $i$  DE 1 ATÉ  $m$ ,  $M_{i0} = 0$  O(m)
3 PARA  $j$  DE 1 ATÉ  $n$ ,  $M_{0j} = 0$  O(n)
4 PARA  $i$  DE 1 ATÉ  $m$ :
5   PARA  $j$  DE 1 ATÉ  $n$ :
6     SE  $x_i == y_j$ ,  $M_{ij} = M_{i-1, j-1} + 1$  O(mn)
7     SENÃO  $M_{ij} = \max\{M_{i-1, j}, M_{i, j-1}\}$  O(mn)
8  $N = M_{mn}$ 

```

COMPLEXIDADE DE TEMPO:  $O(m) + O(n) + O(mn) = O(mn)$

PROBLEMA DO TROCO: DADOS VALORES DE MOEDAS  $v = \{v_1, v_2, \dots, v_k\}$  DISTINTOS E UM TROCO  $T$ , QUEREMOS FORMAR  $T$  COM O MÉIOR NÚMERO DE MOEDAS DE  $V$  POSSÍVEL.

MONTAREMOS A MATRIZ  $M$ , ONDE  $M_{ij} =$  O NÚMERO DE MOEDAS NECESSÁRIA PARA FORMAR O TROCO  $j$  UTILIZANDO MOEDAS DE VALORES  $v_1, \dots, v_j$

EX:  $T = 7$ ,  $V = \{1, 2, 3, 5, 7\}$

$M$	1	2	3	5	7
1	1	1	1	1	1
2	2	1	1	1	1
3	3	2	1	1	1
4	4	2	2	2	2
5	5	3	2	1	1
6	6	3	2	2	2
7	7	4	3	2	1

RELAÇÃO DE RECORRÊNCIA:

$$M_{ij} = \begin{cases} \min \{M_{i-v_j,j} + 1, M_{i,j-1}\} & \text{SE } i \geq v_j \\ M_{i,j-1} & \text{SE } i \leq v_j \\ 0 & \text{SE } i = 0 \end{cases}$$

COMPLEXIDADE:  $O(VT)$

PD DO TROCO EM OFI: FAZENDO ORDENAÇÃO  
IN-PLACE

1	1
2	2
3	"
"	"
T	T

NA ITERAÇÃO 1

AULA 7 - 04/02/2018

REDE: DIGRAFO  $D = (V, E)$  EM QUE A CADA ARESTA  $e \in E(D)$  É ASSOCIADO UM VALOR REAL POSITIVO CHAMADO CAPACIDADE DA ARESTA (DENOTADO POR  $c(e)$ ).

TODA REDE POSSUI DOIS VERTICES ESPECIAIS A E T, TAL QUE

A É A ORIGEM  $\rightarrow$  UMA FONTE QUE ALCANÇA TODOS OS VERTICES DE D.

T É O DESTINO  $\rightarrow$  UM SUMIDOURO QUE É ALCANGADO POR TODOS OS VERTICES DE D.

EBS: 1 ÚNICO S, 1 ÚNICO T

FLUXO: FUNÇÃO QUE ASSOCIA UM VALOR REAL  $f(e)$  A CADA ARESTA  $e \in E(D)$ , TAL QUE:

1)  $0 \leq f(e) \leq c(e)$   $\forall e \in E(D)$ , OU SEJA, O FLUXO EM CADA ARESTA NÃO ULTRAPASSA A CAPACIDADE.

$$2) \sum_{v \in E(D)^+} f(vv) = \sum_{v \in E(D)^-} f(vv) \quad \forall v \in V(D)$$

JA, O FLUXO SE CONSERVA EM TODO VÉRTICE DIFERENTE DE  $s$  E  $t$ . O SOMATORIO DO FLUXO NAS ARESTAS CONVERGENTES PARA  $s$  É IGUAL AO SOMATORIO NAS ARESTAS DIVERGENTES DE  $t$ .

NOTAÇÃO:

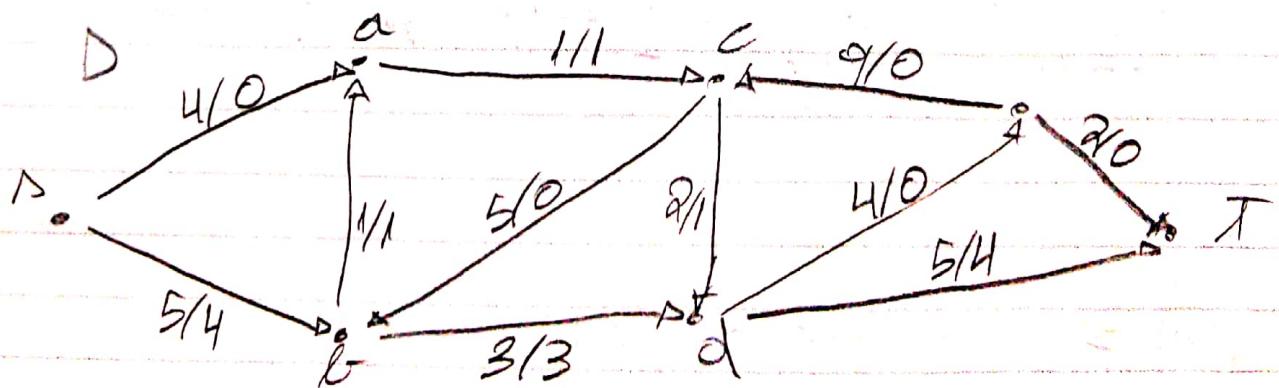
$$\text{val}(f) = \text{VALOR DO FLUXO} = \sum_{v \in V(D)} f(vv)$$

SE PARA UMA ARESTA  $uv$ ,  $c(uv) = f(uv)$ , DIZEMOS QUE  $uv$  ESTÁ SATURADA.

FLUXO MAXIMAL: UM FLUXO  $f$  É DITO MAXIMAL SE TODO CAMINHO DE  $s$  ATÉ  $t$  POSSUI AO MENOS 1 ARESTA SATURADA.

FLUXO MÁXIMO:  $f(D)$  É MÁXIMO SE NÃO EXISTE  $f'(D)$  TAL QUE  $f'(D) > f(D)$

Ex.



$f$  É MAXIMAL E MÁXIMO

CORTE: DADO  $S \subset V(D)$ , TAL QUE  $\Delta S \in T\delta S$ , O CONJUNTO DE ARESTAS COM UM EXTREMO EM  $S$  E OUTRO EM  $\bar{S}$  É UM CORTE DE ARESTAS.

$$S_1 = \{\Delta\} \quad (\bar{S}_1, \bar{S}_1) = (\bar{S}_1, \bar{S}_1)^T = \{\Delta_{\bar{S}_1}, \Delta_{\bar{S}_1}\}$$

CAPACIDADE DE UM CORTE:

$$c(S, \bar{S}) = \sum_{e \in (S, \bar{S})^+} c(e)$$

FLUXO EM UM CORTE:

$$\chi(S, \bar{S}) = \sum_{e \in (S, \bar{S})^+} \chi(e) - \sum_{e \in (S, \bar{S})^-} \chi(e)$$

QDS: QUANTOS CORTES DIFERENTES EM UMA REDE?  
 $2^{n-2}$  CORTES

FATO:  $\chi(S, \bar{S}) \leq c(S, \bar{S})$ ,  $\forall (S, \bar{S})$

$$\chi(S, \bar{S}) \stackrel{\text{def}}{=} \sum_{e \in (S, \bar{S})^+} \chi(e) - \sum_{e \in (S, \bar{S})^-} \chi(e) \leq \sum_{e \in (S, \bar{S})^+} \chi(e) \leq \sum_{e \in (S, \bar{S})^+} c(e) \stackrel{\text{def}}{=} c(S, \bar{S})$$

LEMMA: PARA TODO CORTE  $(S, \bar{S})$  EM UMA REDE  $D$  COM FLUXO  $\chi$ ,  $\text{val}(\chi) = \chi(S, \bar{S})$

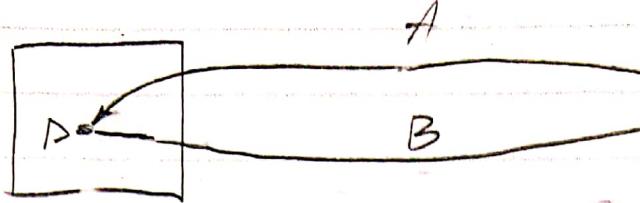
PROVA: INDUÇÃO EM  $|S|$

BASE: COMO  $S = \{\Delta\}$ , TEMOS

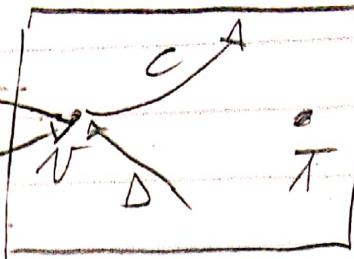
$$\chi(S, \bar{S}) = \sum_{e \in (S, \bar{S})^+} \chi(e) - \sum_{e \in (S, \bar{S})^-} \chi(e) = \sum_{e \in (S, \bar{S})^+} \chi(e) = \text{val}(\chi)$$

PASSO INDUTIVO: SEJA  $S'$  TAL QUE  $\lambda(S') \geq \lambda(S)$

$S$



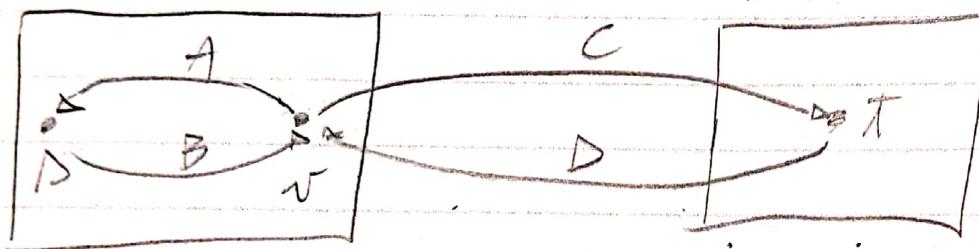
$\bar{S}$



$$A = \sum_{v \in S} f(vv)$$

$$S' = S \cup \{v\}$$

$\bar{S}'$



$$C = \sum_{v \in S'} f(vv)$$

$$D = \sum_{v \in S'} f(vv)$$

$$\lambda(S, \bar{S}) = \lambda(S, \bar{S}) - (B - A) + C - D$$

$$= \lambda(S, \bar{S}) - B + A + C - D$$

$$= \lambda(S, \bar{S}) + (A + C) - (C + D)$$

ENTRA  
EM  
 $v$

SAI  
DE  
 $v$

$$( \sum_{v \in S} f(vv) = \sum_{v \in S'} f(vv) )$$

$$\lambda(S, \bar{S}) = \lambda(S, \bar{S}).$$

TEOREMA DO FLUXO MÁXIMO - CORTE MÍNIMO:

SE  $\lambda(S, \bar{S}) = c(S, \bar{S})$ ,  $\lambda$  É O FLUXO MÁXIMO.

CHAMAMOS  $(S, \bar{S})$  DE CORTE MÍNIMO DE  $D$ .

SUPONHA QUE  $(S, \bar{S})$  NÃO SEJA MÍNIMO. ENTÃO, EXISTE OUTRO CORTE  $(\bar{S}, \bar{S})$  TAL QUE  $c(\bar{S}, \bar{S}) < c(S, \bar{S})$ . COMO  $\lambda(S, \bar{S}) = \lambda(\bar{S}, \bar{S})$ , TERÍAMOS QUE  $\lambda(\bar{S}, \bar{S}) > c(S, \bar{S})$ , O QUE É IMPOSSÍVEL. LOGO,  $(S, \bar{S})$  É MÍNIMO.

# ALGORITMO: FLUXO MAXIMAL

ENTRADA: REDE  $D = (V, E)$ ,  $c: E \rightarrow \mathbb{R}^+$

SAÍDA:  $f: E \rightarrow \mathbb{R}^+$  TAL QUE  $f$  É FLUXO MAXIMAL

1 PARA TODO  $e \in E(D)$ ,  $f(e) = 0$

2  $\text{val}(f) = 0$

3 ENQUANTO EXISTIR CAMINHO  $P$  DE SATE  $T$   
SEM ARESTAS SATURADAS:

4  $m = \min_{e \in P} \{c(e) - f(e)\}$

5 PARA TODO  $e \in P$ ,  $f(e) = f(e) + m$

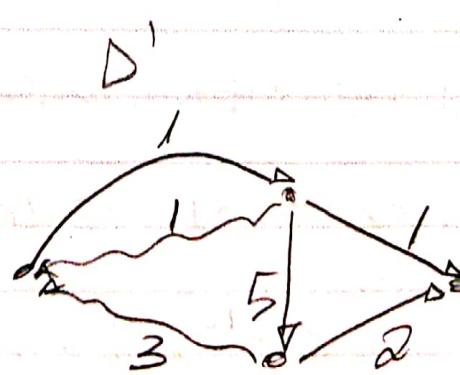
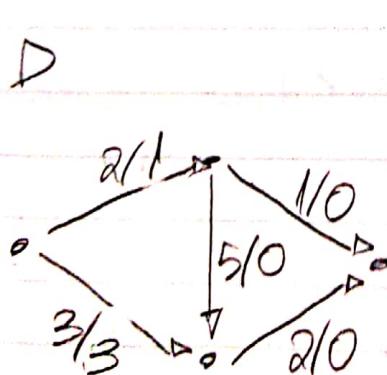
6  $\text{val}(f) = \text{val}(f) + m$

REDE RESIDUAL: A REDE RESIDUAL  $D'$  DE  $D$  é  
 $D'$ , DAI, É O DIGRAFO TAL QUE  $V' = V \setminus E$

SE  $c(uv) - f(uv) > 0$ , ENTÃO  $uv \in E'$ ,  
 $c(uv)' = c(uv) - f(uv)$ .  
(ARESTA DIRETA)

SE  $f(uv) > 0$ , ENTÃO  $vu \in E'$ ,  $c(vu)' = f(vu)$ .  
(ARESTA CONTRÁRIA)

Ex:



— DIRETA  
— CONTRÁRIA

# ALGORITMO: FORD-FULKERSON

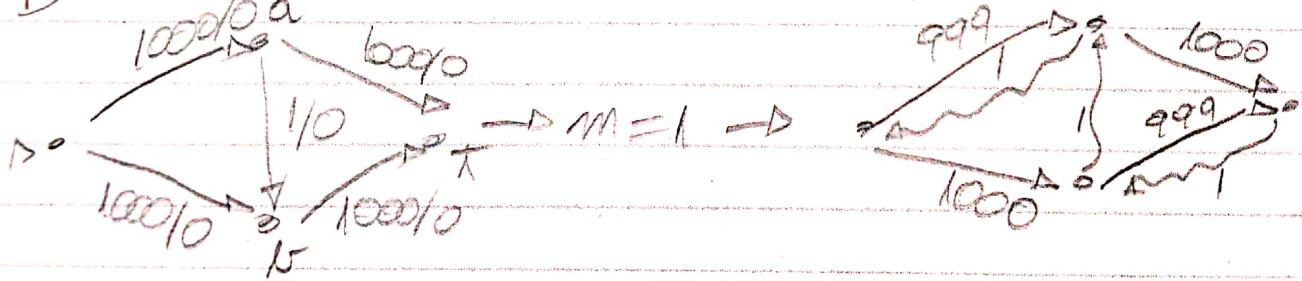
ENTRADA:  $D = (V, E)$ ,  $c: E \rightarrow N$

SÍLDA:  $f: E \rightarrow N$ , VM FLUXO MÁXIMO

- 1 PARA TODO  $e \in E(D)$ ,  $f(e) = 0$   $O(m)$
- 2  $\text{val}(f) = 0$   $O(1)$
- 3  $D' = D$   $O(m+m)$   $O(m)$
- 4 ENQUANTO EXISTIR CAMINHO  $P$  DE  $S$  ATÉ  $T$  EM  $D'$ :  $\boxed{7}$
- 5  $m = \min_{e \in P} \{c(e)\}$   $O(n)$
- 6  $f = f \oplus m$ ,  $\forall e \in P$   $O(n)$
- 7  $\text{val}(f) = \text{val}(f) + m$   $O(1)$
- 8 CONSTRUA  $D'(f)$   $O(m)$

$$f(e) \oplus m = \begin{cases} f(e) + m, & \text{SE } e \in \text{DIRETA} \\ f(e) - m, & \text{SE } e \in \text{CONTRÁRIA} \end{cases}$$

$D$



FLUXO MÁXIMO

NÚMERO DE ITERAÇÕES NO PIOR CASO:  $O(\text{val}(f^*))$

COMPLEXIDADE DE TEMPO:

$$\begin{aligned} & 2m + \text{val}(f^*) [n+m+2n+m+1] = \\ & 2m + n\text{val}(f^*) + m\text{val}(f^*) + 2n\text{val}(f^*) + m\text{val}(f^*) + \text{val}(f^*) \\ & \text{COMO } D \text{ É CONEXO} | O(m\text{val}(f^*)) \end{aligned}$$

TOTAL:  $O(m\text{val}(f^*))$