

## Árvores Abrangentes de Menor Custo

- Definições Elementares
- Teorema do Corte
- Algoritmo de Kruskal
- Algoritmo

①

## Árvores Abrangentes de Menor Custo

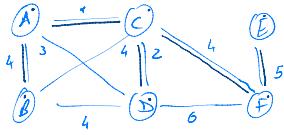
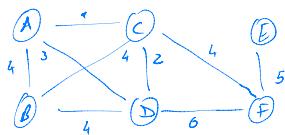
### Definição 1] Árvore Abrangente & Árvore Abrangente de Menor Custo]

Seja  $G = (V, E, w)$  um grafo não dirigido ponderado, uma árvore abrangente de  $G$  é um subconjunto das arestas de  $G$ ,  $T \subseteq E$ , tal q.  $T$  não contém ciclos.

Dada uma árvore abrangente  $T$ , o peso de  $T$  é definido como a soma dos pesos das arestas de  $T$ :  $w(T) = \sum_{(u,v) \in T} w(u,v)$

- Dado  $G = (V, E, w)$ , aínc AA de menor custo de  $G$  é uma AA com menor peso.

### Exemplo 1] AA



### Definição 2] Anco Spanning

Seja  $G = (V, E, w)$  um grafo ponderado, não-dirigido, ligado e seja  $A$  um subconjunto de um AAC em  $G$ ; um anco  $(u, v)$  é segredo para  $A$  se  $AV\{u, v\}$  é um subconjunto de um AAC.

### Algoritmo 1] Algoritmo fenômeno pr AACs

MST( $G$ )

let  $A = \emptyset$

while ( $A$  does not touch all vertices)

  | pick  $(u, v) \in E$  st  $(u, v)$  is safe for  $A$   
  |  $A = A \cup \{(u, v)\}$

return  $A$

### Definição 2] Corte q. respeita A / Anco lige

Seja  $G = (V, E, c)$  um grafo ponderado, ligado e não-dirigido e seja  $A$  um subconjunto de uma MST em  $G$ ; então:

- Um corte  $(S, V-S)$  respeita  $A$  se nenhum das arestas de  $A$  atravessa o corte
- Um anco  $(u, v)$  é lige para o corte  $(S, V-S)$  se  $(u, v)$  se  $(x, y)$  atravessa o corte e  $w(x, y) = \min \{w(x, y) \mid (x, y) \text{ atravessa } (S, V-S)\}$

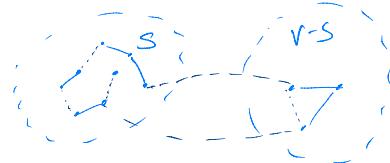
(2)

### Teorema 1 [ Arco ligeiro $\rightarrow$ Arco seguro / Propriedade de Coorte ]

Seja  $G = (V, E, w)$  um grafo pesado, não dirigido e ligado e seja  $A$  um subconjunto de vértices  $S \subseteq V$ . Seja  $T$  uma MST de  $G$  e  $(S, V-S)$  um corte que interessa  $A$ ; então se  $(u, v) \in E$  é um arco ligeiro de  $(S, V-S)$  então  $(u, v)$  é seguro para  $A$ .

Prova:

- Suponhamos que  $(S, V-S)$  é um corte em  $G$ ,  $A$  é uma MST,  $(u, v)$  é um arco ligeiro que atravessa  $(S, V-S)$ .



- Seja  $T$  a MST que contém  $A$ ; então:

$$T = T_S \cup T_{\bar{S}} \cup \{(u, v)\} \text{ atravessa o corte}$$

- 2 hipóteses:

- $(u, v) = (u, r) \Rightarrow (u, r)$  é seguro para  $A$
- $(u, v) \neq (u, r)$ 
  - $w(u, v) = w(u, r)$  porque  $(u, v)$  é ligeiro e  $T$  é MST
  - Seja  $T' = T_S \cup T_{\bar{S}} \cup \{(v, r)\}$
  - $T'$  é uma MST de  $G$  e  $T' \supseteq A \cup \{(u, r)\}$

■

### Lema 1 [ Kruskal ]

Seja  $G = (V, E, w)$  um grafo pesado, ligado e não dirigido, seja  $A$  um subconjunto de vértices  $S \subseteq V$  de  $G$  e seja  $C = (V_C, E_C)$  uma qualquer componente ligada no interior de  $G_A = (V_A, E_A)$ ; então o arco de menor peso a ligar  $C$  a outra componente de  $G_A$  é seguro para  $A$ .

Prova:

- O corte  $(V_C, V - V_C)$  é seguro para  $A$ .
- $(u, v)$  é um arco ligeiro que atravessa o corte
- O resultado segue do teorema 1

### Algoritmo 2 [ Kruskal ]

Kruskal( $G, w$ )for each  $v \in G.V$     MakeSet( $v$ )     $E^l = \text{sort}(E)$  // increasing weight     $A = \emptyset$     for each  $(u, v) \in E^l$         if ( $\text{FindSet}(u) \neq \text{FindSet}(v)$ )             $A = A \cup \{(u, v)\}$             Union( $u, v$ )    return  $A$ 

Complexidade:

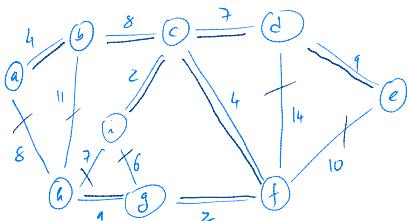
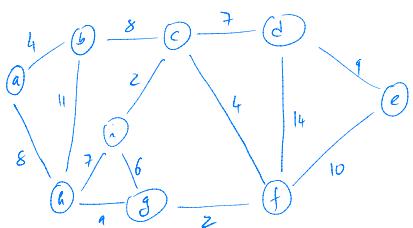
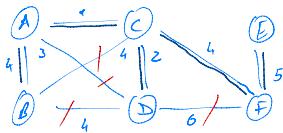
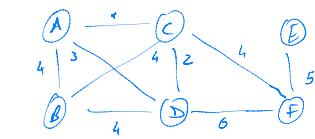
• Criar os conjuntos:  $O(V)$ • Unir os vértices:  $O(E \lg E) \approx O(E \lg V)$ 

• Custo principal:

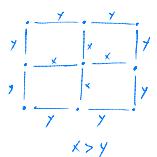
 $O(E \lg V)$ MakeSet:  $O(1)$ Find:  $O(\lg V)$ Union:  $O(\lg V)$ 
 $\left. \begin{array}{l} \\ \\ \end{array} \right\} \text{próximo arco}$ 
 $O(E \lg V)$

(3)

## Exemplo 2 [kruskal]



## Exemplo 3 [Grafos de árvores - 1]



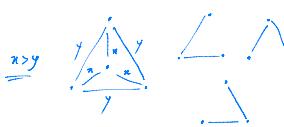
Um lado vir falso incompleto: 3 lados completos e 1 lado incompleto  
 $\square \quad \square \quad \square \quad \square \quad$  6 casos

$$\text{Nº total de hipóteses} = 4 \times 2 \times 4$$

Analisamos apenas 1 caso:  $\square \quad \square \quad \square \quad \square \quad$  2 casos

$\boxed{- \cdot -}$  - 4 hipóteses para figura  
 para figura o vértice central)

## Exercício 4 [Grafos binários - 2]



- 3 casos



- Para cada caso: 3 hipóteses

Total: 9 casos

(4)

## Algoritmo 3 [Prim]

 $\text{Prim}(G, w, \alpha)$ for each  $v \in G.V$      $v.\text{key} = \infty$      $v.\pi = \text{Nil}$      $\alpha.\text{key} = 0$      $\alpha.\pi = \text{Nil}$ while  $Q \neq []$     let  $u = \text{ExtractMin}(Q)$     for each  $v \in G.\text{Adj}[u]$         if  $w(u, v) < v.\text{key}$   $\wedge$   $v \in Q$              $v.\text{key} = w(u, v)$              $v.\pi = u$ Complexidade:

- Análise agregada

- Quantas vezes é q um vértice é extraído da fila?

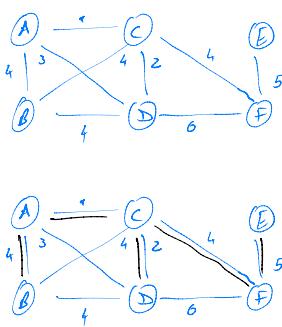
| V | (põe um vértice nunca é recolocado na fila)  $\Rightarrow$ usto é  $O(1)$

- O corpo do ciclo interno é executado 2 vezes por arco

$\hookrightarrow$  Alterar a prioridade é  $\log V$

- $O(E \log V)$

## Exemplo 4 [Prim]



	A	B	C	D	E	F
1	$\text{Nil}/0$	$\text{Nil}/\infty$	$\text{Nil}/\infty$	$\text{Nil}/\infty$	$\text{Nil}/\infty$	$\text{Nil}/\infty$
2	$\text{Nil}/0$	$4/A$	$1/A$	$3/A$	$\text{Nil}/\infty$	$\text{Nil}/\infty$
3	$\text{Nil}/0$	$4/A$	$1/A$	$2/C$	$\text{Nil}/\infty$	$4/C$
4	$\text{Nil}/0$	$4/A$	$1/A$	$2/C$	$\text{Nil}/\infty$	$4/C$
5	$\text{Nil}/0$	$4/A$	$1/A$	$2/C$	$\text{Nil}/\infty$	$4/C$
6	$4/B$	$4/A$	$1/A$	$2/C$	$4/F$	$4/C$
7	$4/B$	$4/A$	$1/A$	$2/C$	$4/F$	$4/C$

Garantia:

Observação:  $A = \{(v, \pi, v) \mid v \in V \setminus Q\}$  é ponto de uma MST

$\forall v \in Q. v.\text{key} = \min\{w(u, v) \mid v \in V \setminus Q\} \wedge (v.\pi \neq \text{Nil} \Rightarrow w(v, \pi, v) = v.\text{key})$

vertices(A) =  $V \setminus Q$

Iniciaização ] Fim da Primeira Iteração]

$A = \emptyset$

$\forall v \in N(e). v.\text{key} = w(e, v) = w(v, \pi, v)$

$\forall v \notin N(e). v.\text{key} = \infty \wedge v.\pi = \text{Nil}$

## [Passo]

- O conteúdo  $(V \setminus Q, Q)$  é seguro para  $A$  e  $(\alpha, \pi, \alpha)$  é um arco leve q abavesse o arco. Logo aplicando o Teorema 1, concluímos q  $A \cup \{(v, \pi, v) \mid v \in V \setminus Q\}$  é ponto de uma MST de  $G$ .
- A segunda parte da invariante é satisfeita porque o algoritmo actualiza as chaves das vizinhas de  $u$ .

