8. Árvores Geradoras Mínimas

→ Entrada: um grafo não-dirigido e ponderado $G = (V, E, w)$.

→ Saída: uma árvore geradora mínima (AGM) de G.

AGM: é um subconjunto do conjunto de arestas, que componha uma árvore (acíclico) com todos os vértices de G, tal que o peso total do subconjunto seja mínimo.

→ Duas ideias centrais: → Acíclica: aresta segura
→ Custo mínimo: aresta leve

8.2 Algoritmo de Kruskal

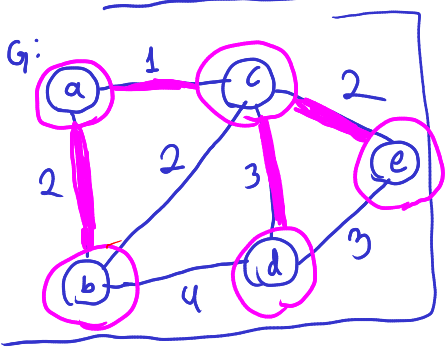
→ para o grafo G, início com $|V|$ árvores (todo vértice é raiz de uma árvore)

→ iterativamente, selecionando uma aresta p/ unir duas árvores (segura e leve)

segura → união de conjuntos disjuntos de vértices ajuda.

leve → prioriza ordem a escolha das arestas por peso

Teste de Mesa: $A = \{\{a,c\}, \{a,b\}, \{c,e\}, \{c,d\}\}$



$S_a = \{a, c, b, e, d\}$

$S_b = \{b, a, c, e, d\}$

$S_c = \{a, c, b, e, d\}$

$S_d = \{d, a, c, b, e\}$

$S_e = \{e, a, c, b, d\}$

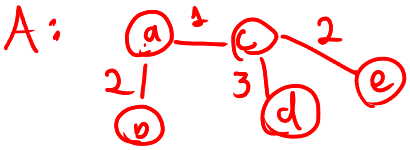
Algoritmo 21: Algoritmo de Kruskal.

Input : um grafo $G = (V, E, w)$

1 $A \leftarrow \{\}$
2 $S \leftarrow$ vetor de $|V|$ elementos vazios
3 **foreach** $v \in V$ **do**
4 $\quad S_v \leftarrow \{v\}$
5 $E' \leftarrow$ lista de arestas ordenadas por ordem crescente de peso
6 **foreach** $\{u, v\} \in E'$ **do**
7 $\quad$ **if** $S_u \neq S_v$ **then**
8 $\quad\quad A \leftarrow A \cup \{\{u, v\}\}$
9 $\quad\quad x \leftarrow S_u \cup S_v$
10 $\quad\quad$ **foreach** $y \in x$ **do**
11 $\quad\quad\quad S_y \leftarrow x$
12 **return** A

$E' = \langle \{a,c\}, \{a,b\}, \{b,c\}, \{c,e\}, \{c,d\}, \{d,e\}, \{b,d\} \rangle$ (✓ ✓ ✗ ✓ ✓ ✗ ✗)

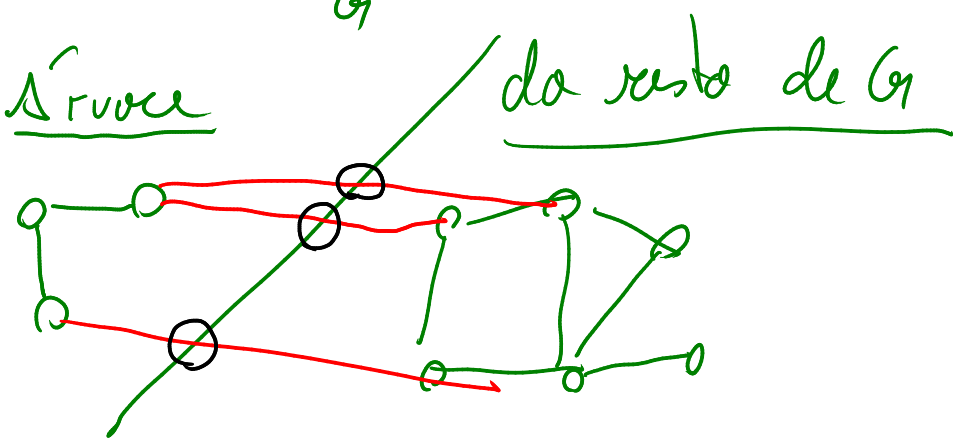
A:



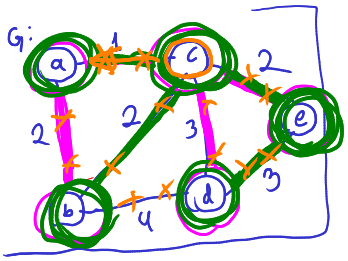
custo: 8

8.3 Algoritmo de Prim

→ Constroímos uma árvore iterativamente, computando as arestas candidatas a partir do "conhecimento" das adjacências dos vértices que já pertencem a árvore.



Teste de Mesa!

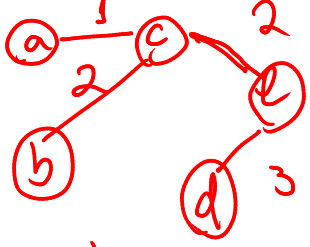


r = d

	A	K	Q (✓=sim, ✗=não)
a	~~*~~ c	~~*~~ 1	✗
b	~~v~~ ~~*~~ c	~~*~~ ~~*~~ 2	✗
c	~~v~~ ~~*~~ e	~~*~~ ~~*~~ 2	✗
d	v	~~*~~ 0	✗
e	v d	~~*~~ 3	✗

u = ~~d~~ ~~c~~ ~~e~~ ~~c~~ b

A:



N(u) = {~~a~~, c, ~~d~~}

Custo: 8

Algoritmo 22: Algoritmo de Prim.

Input : um grafo $G = (V, E, w)$

1 $r \leftarrow$ selecionar um vértice arbitrário em V

// Definindo o vetor dos antecessores A e uma chave para cada vértice K

2 $A_v \leftarrow$ **null** $\forall v \in V$

3 $K_v \leftarrow \infty \; \forall v \in V$

4 $K_r \leftarrow 0$

// Definindo a estrutura de prioridade de chave mínima Q

5 $Q \leftarrow (V, K)$

6 **while** $Q \neq \{\}$ **do**

7 $\quad u \leftarrow \arg\min_{v \in Q} \{K_v\}$

8 $\quad Q \leftarrow Q \backslash \{u\}$

9 $\quad$ **foreach** $v \in N(u)$ **do**

10 $\quad\quad$ **if** $v \in Q \wedge w(\{u, v\}) < K_v$ **then**

11 $\quad\quad\quad A_v \leftarrow u$

12 $\quad\quad\quad K_v \leftarrow w(\{u, v\})$

13 **return** A