Análise de Algoritmos

Parte destes slides são adaptações de slides

do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

Árvore geradora mínima

CLRS Cap 23

Árvore geradora mínima

Seja G um grafo conexo.

Uma árvore T em G é geradora
se contém todos os vértices de G.

Árvore geradora mínima

Seja G um grafo conexo.

Uma árvore T em G é geradora
se contém todos os vértices de G.

Problema: Dado G conexo com custo c_e para cada aresta e, encontrar árvore geradora em G de custo mínimo.

Árvore geradora mínima

Seja G um grafo conexo.

Uma árvore T em G é geradora
se contém todos os vértices de G.

Problema: Dado G conexo com custo c_e para cada aresta e,
encontrar árvore geradora em G de custo mínimo.

O custo de uma árvore é a soma do custo de suas arestas.

Árvore geradora mínima

Seja G um grafo conexo.

Uma árvore T em G é geradora
se contém todos os vértices de G.

Problema: Dado G conexo com custo c_e para cada aresta e, encontrar árvore geradora em G de custo mínimo.

O custo de uma árvore é a soma do custo de suas arestas.

Tal árvore é chamada de árvore geradora mínima em G.

Árvore geradora mínima

Seja G um grafo conexo.

Uma árvore T em G é geradora
se contém todos os vértices de G.

Problema: Dado G conexo com custo c_e para cada aresta e, encontrar árvore geradora em G de custo mínimo.

O custo de uma árvore é a soma do custo de suas arestas.

Tal árvore é chamada de árvore geradora mínima em G.

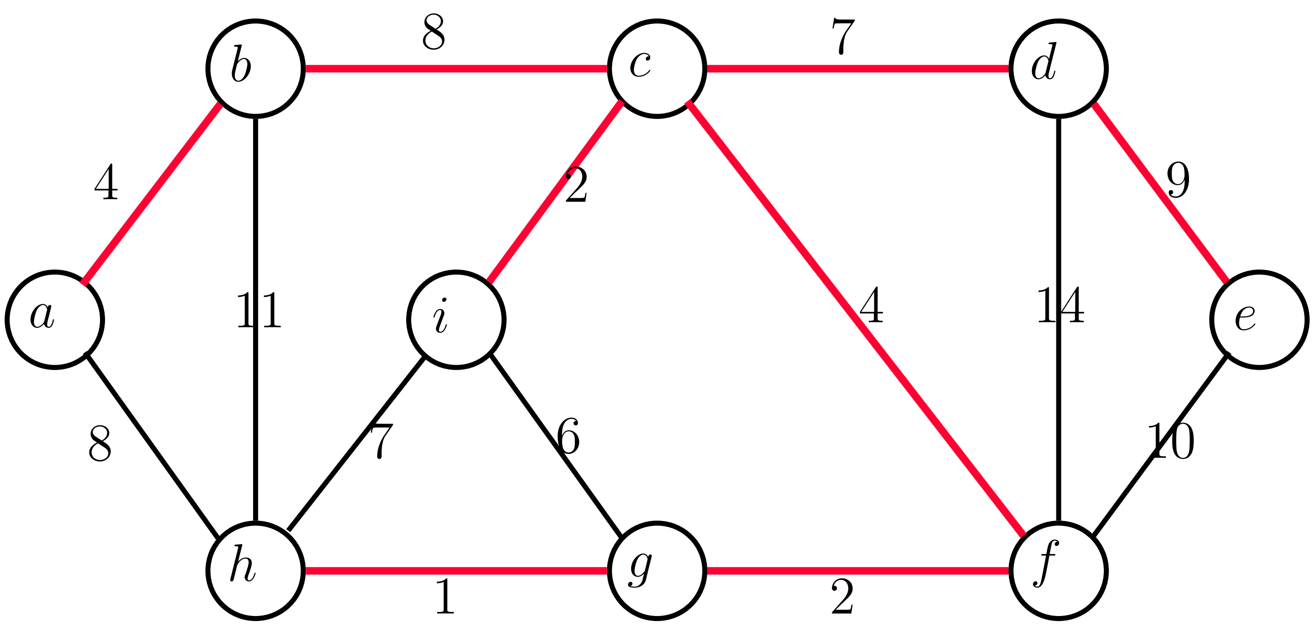
MST: minimum spanning tree

Árvore geradora mínima

Seja G um grafo conexo.

Uma árvore T em G é geradora
se contém todos os vértices de G.

Problema: Dado G conexo com custo c_e para cada aresta e,
encontrar árvore geradora mínima em G.



Arestas seguras

Seja $G = (V, E)$ um grafo conexo.
Função c que atribui um custo c_e para cada aresta $e \in E$.

$A \subseteq E$ contido em alguma MST de (G, c).

Arestas seguras

Seja $G = (V, E)$ um grafo conexo.
Função c que atribui um custo c_e para cada aresta $e \in E$.

$A \subseteq E$ contido em alguma MST de (G, c).

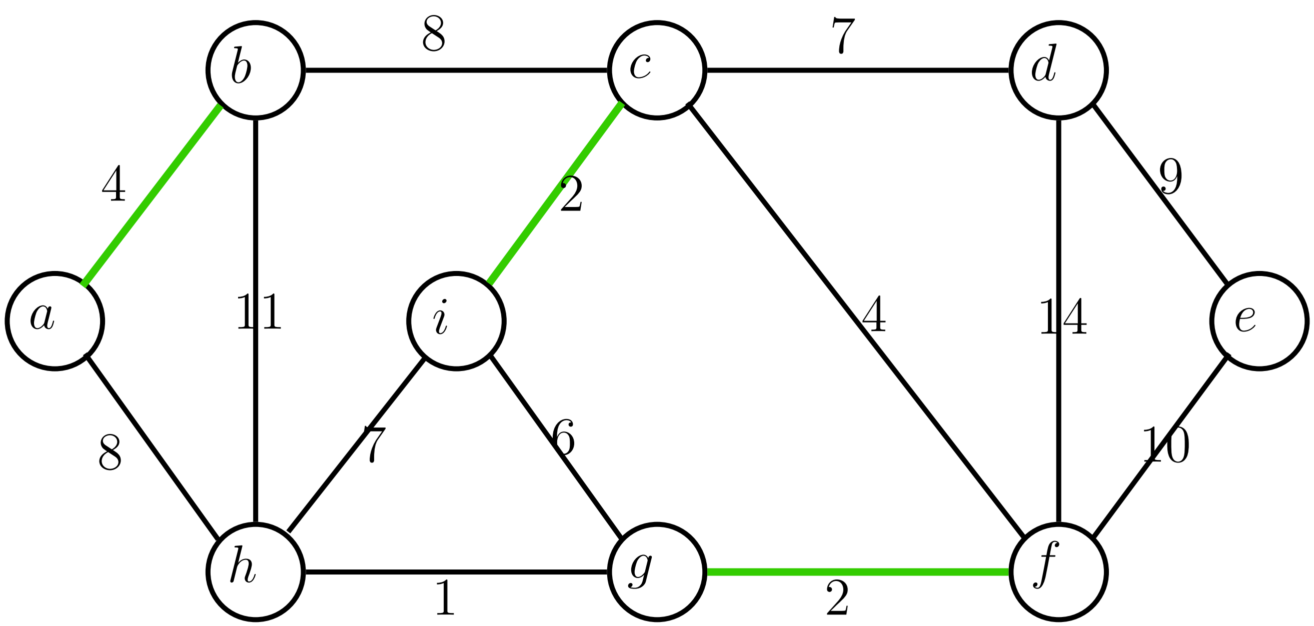
Aresta $e \in E$ é segura para A se
$A \cup \{e\}$ está contido em alguma MST de (G, c).

Arestas seguras

Seja $G = (V, E)$ um grafo conexo.
Função c que atribui um custo c_e para cada aresta $e \in E$.

$A \subseteq E$ contido em alguma MST de (G, c).

Aresta $e \in E$ é segura para A se
$A \cup \{e\}$ está contido em alguma MST de (G, c).

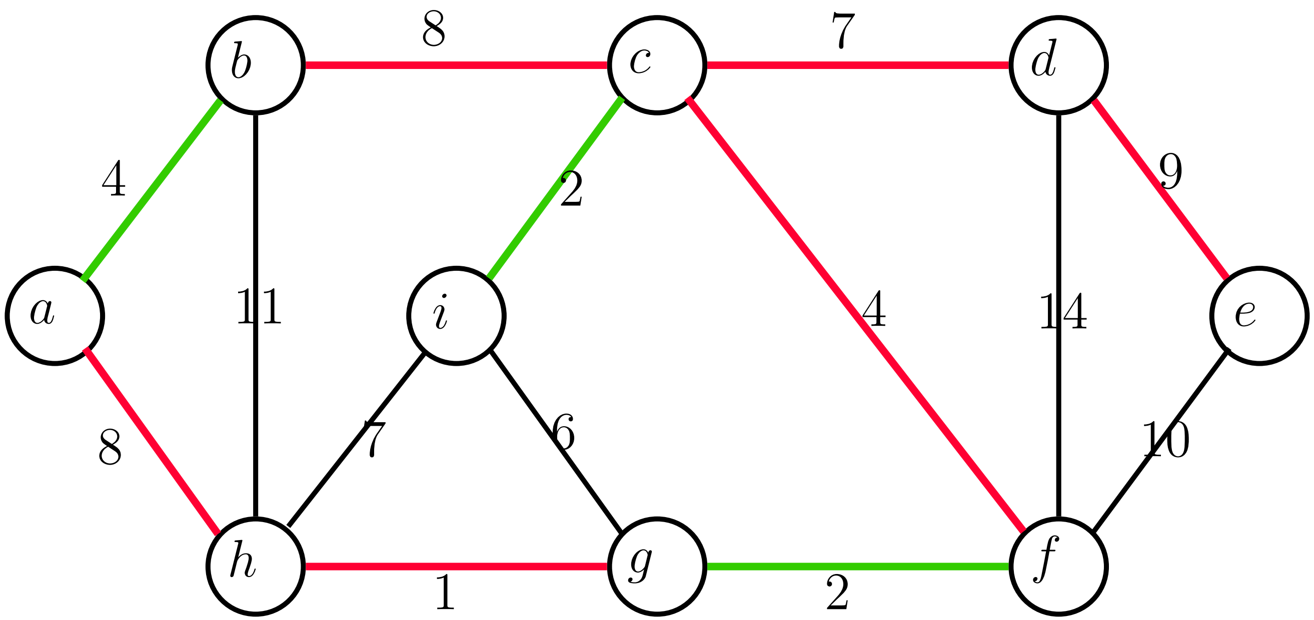


Arestas seguras

Seja $G = (V, E)$ um grafo conexo.
Função c que atribui um custo c_e para cada aresta $e \in E$.

$A \subseteq E$ contido em alguma MST de (G, c).

Aresta $e \in E$ é segura para A se
$A \cup \{e\}$ está contido em alguma MST de (G, c).



Arestas seguras

Seja $G = (V, E)$ um grafo conexo.
Função w que atribui um custo c_e para cada aresta $e \in E$.

$A \subseteq E$ contido em alguma MST de (G, c).

Aresta $e \in E$ é segura para A se
$A \cup \{e\}$ está contido em alguma MST de (G, c).

Se A não é uma MST, então existe aresta segura para A.

Arestas seguras

Seja $G = (V, E)$ um grafo conexo.
Função w que atribui um custo c_e para cada aresta $e \in E$.

$A \subseteq E$ contido em alguma MST de (G, c).

Aresta $e \in E$ é segura para A se
$A \cup \{e\}$ está contido em alguma MST de (G, c).

Se A não é uma MST, então existe aresta segura para A.

GENÉRICO (G, c)
1 $A \leftarrow \emptyset$
2 **enquanto** A não é geradora **faça**
3   encontre aresta segura e para A
4     $A \leftarrow A \cup \{e\}$
5 **devolva** A

Arestas seguras

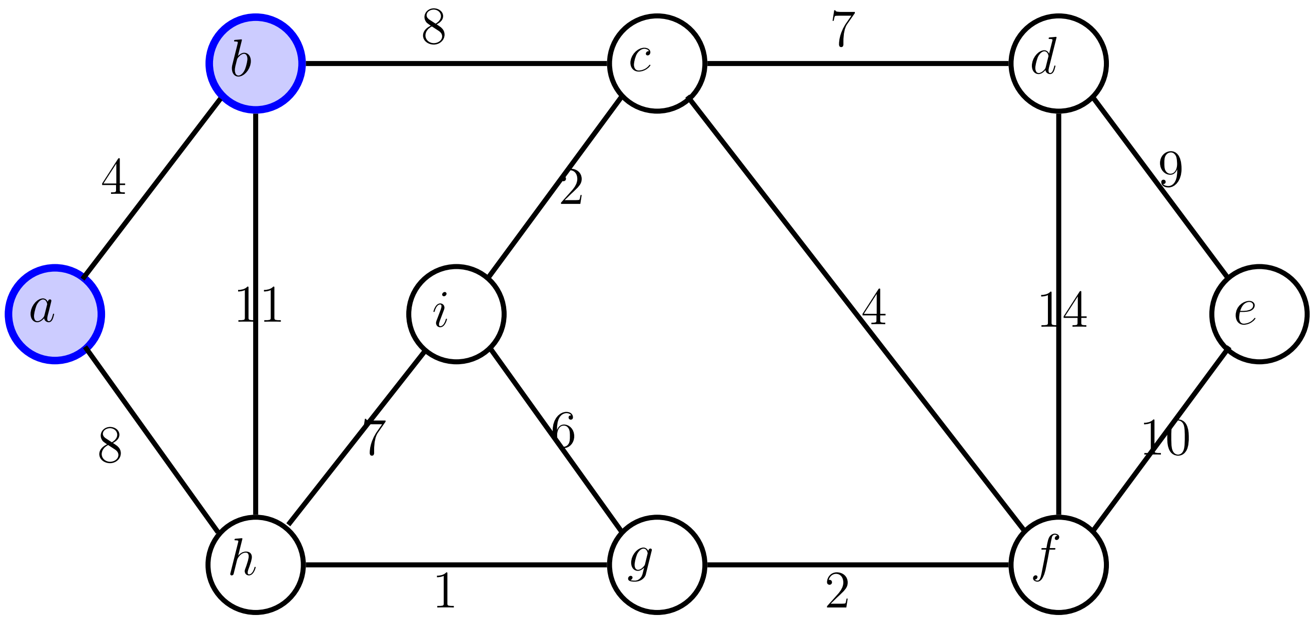
Corte em G: partição $(S, V \setminus S)$.

Aresta e cruza o corte $(S, V \setminus S)$ se
exatamente um de seus extremos está em S.

Arestas seguras

Corte em G: partição $(S, V \setminus S)$.

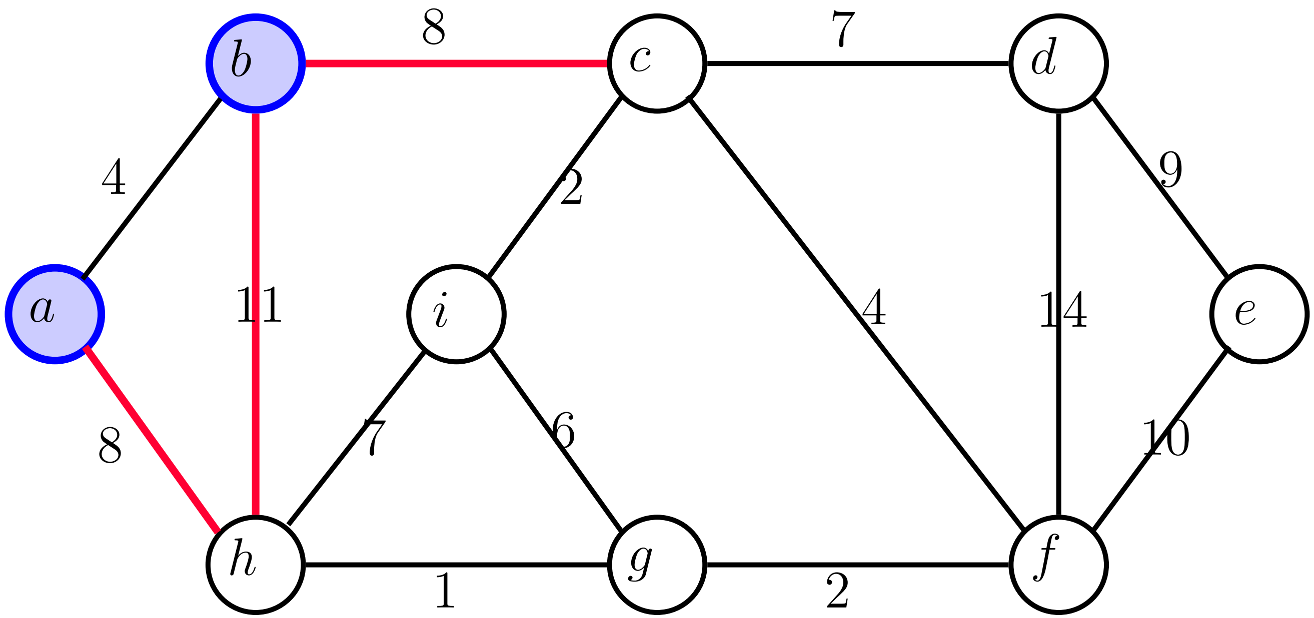
Aresta e cruza o corte $(S, V \setminus S)$ se
exatamente um de seus extremos está em S.



Arestas seguras

Corte em G: partição $(S, V \setminus S)$.

Aresta e cruza o corte $(S, V \setminus S)$ se
exatamente um de seus extremos está em S.



Arestas seguras

Corte em G: partição $(S, V \setminus S)$.

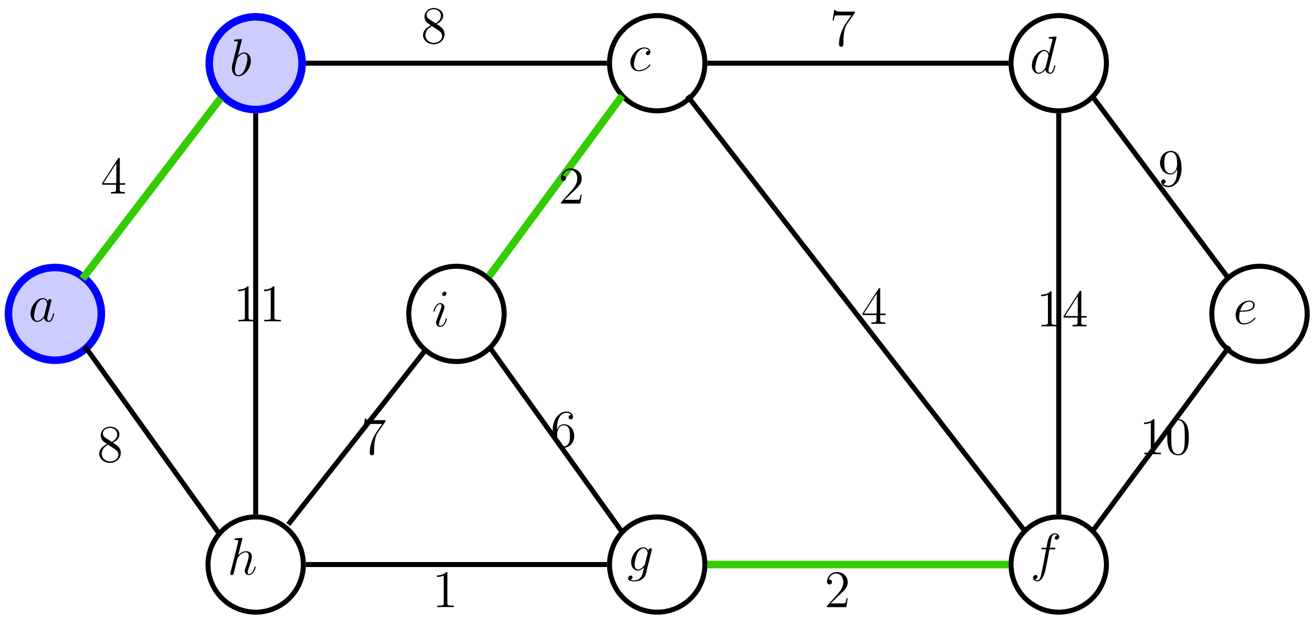
Aresta e cruza o corte $(S, V \setminus S)$ se
exatamente um de seus extremos está em S.

Arestas seguras

Corte em G: partição $(S, V \setminus S)$.

Aresta e cruza o corte $(S, V \setminus S)$ se
exatamente um de seus extremos está em S.

Corte respeita $A \subseteq E$: nenhuma aresta de A o cruza.

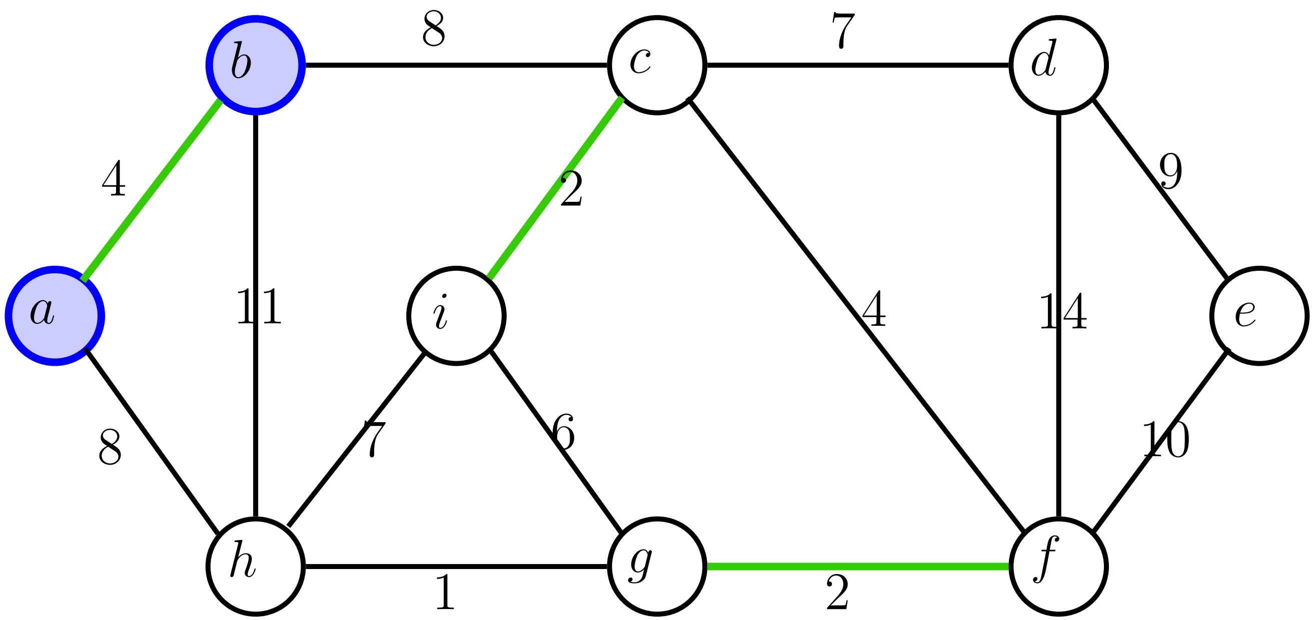


Arestas seguras

Corte em G: partição $(S, V \setminus S)$.

Aresta e cruza o corte $(S, V \setminus S)$ se
exatamente um de seus extremos está em S.

Corte respeita $A \subseteq E$: nenhuma aresta de A o cruza.



Teorema: Se A está contida em MST de (G, c), então e com $c(e)$ mínimo em corte que respeita A é segura para A.

Arestas seguras

Corte em G: partição $(S, V \setminus S)$.

Aresta e cruza o corte $(S, V \setminus S)$ se
exatamente um de seus extremos está em S.

Corte respeita $A \subseteq E$: nenhuma aresta de A o cruza.

Teorema: Se A está contida em MST de (G, c), então e com $c(e)$ mínimo em corte que respeita A é segura para A.

Arestas seguras

Corte em G: partição $(S, V \setminus S)$.

Aresta e cruza o corte $(S, V \setminus S)$ se
exatamente um de seus extremos está em S.

Corte respeita $A \subseteq E$: nenhuma aresta de A o cruza.

Teorema: Se A está contida em MST de (G, c), então e com $c(e)$ mínimo em corte que respeita A é segura para A.

Prova: Seja T uma MST em (G, c) que contém A.
Se e está em T, não há nada mais a provar.

Arestas seguras

Corte em G: partição $(S, V \setminus S)$.

Aresta e cruza o corte $(S, V \setminus S)$ se
exatamente um de seus extremos está em S.

Corte respeita $A \subseteq E$: nenhuma aresta de A o cruza.

Teorema: Se A está contida em MST de (G, c), então e com $c(e)$ mínimo em corte que respeita A é segura para A.

Prova: Seja T uma MST em (G, c) que contém A.
Se e está em T, não há nada mais a provar.

Se e não está em T, seja C o único circuito em $T + e$, e seja $f \in C$ com $c(f)$ máximo que cruza o mesmo corte (distinta de e em caso de empate).

Arestas seguras

Corte em G: partição $(S, V \setminus S)$.

Aresta e cruza o corte $(S, V \setminus S)$ se
exatamente um de seus extremos está em S.

Corte respeita $A \subseteq E$: nenhuma aresta de A o cruza.

Teorema: Se A está contida em MST de (G, c), então e com $c(e)$ mínimo em corte que respeita A é segura para A.

Prova: Seja T uma MST em (G, c) que contém A.
Se e está em T, não há nada mais a provar.

Se e não está em T, seja C o único circuito em $T + e$, e seja $f \in C$ com $c(f)$ máximo que cruza o mesmo corte (distinta de e em caso de empate).

Então $T' := T + e - f$ é MST e contém $A \cup \{e\}$. ∎

Árvore geradora mínima

Os dois próximos algoritmos se enquadram no genérico.

Árvore geradora mínima

Os dois próximos algoritmos se enquadram no genérico.

O primeiro é o algoritmo de Kruskal que, em cada iteração, escolhe uma aresta segura mais barata possível.

O algoritmo de Kruskal vai aumentando uma floresta.

Árvore geradora mínima

Os dois próximos algoritmos se enquadram no genérico.

O primeiro é o algoritmo de Kruskal que, em cada iteração, escolhe uma aresta segura mais barata possível.

O algoritmo de Kruskal vai aumentando uma floresta.

O segundo é o algoritmo de Prim,
que mantém uma árvore T que contém um vértice s,
acrescentando em cada iteração uma aresta segura a T.

Árvore geradora mínima

Os dois próximos algoritmos se enquadram no genérico.

O primeiro é o algoritmo de Kruskal que, em cada iteração, escolhe uma aresta segura mais barata possível.

O algoritmo de Kruskal vai aumentando uma floresta.

O segundo é o algoritmo de Prim,
que mantém uma árvore T que contém um vértice s,
acrescentando em cada iteração uma aresta segura a T.

Os dois produzem uma MST de (G, c).

$n := |V(G)|$ e $m := |E(G)|$.

Algoritmo de Kruskal

KRUSKAL (G, c)

$A \leftarrow \emptyset$
sejam $e_1, \ldots, e_m$ as arestas de G ordenadas por c
para cada $u \in V(G)$ **faça** MAKESET(u)
para $i \leftarrow 1$ **até** m **faça**
    sejam u e v as pontas de e_i
    se FINDSET$(u) \neq$ FINDSET(v)
        então $A \leftarrow A \cup \{e_i\}$
                UNION(u, v)
devolva A

Algoritmo de Kruskal

```
KRUSKAL (G, c)
    A ← ∅
    sejam e_1, ..., e_m as arestas de G ordenadas por c
    para cada u ∈ V(G) faça MAKESET(u)
    para i ← 1 até m faça
        sejam u e v as pontas de e_i
        se FINDSET(u) ≠ FINDSET(v)
            então A ← A ∪ {e_i}
                  UNION(u, v)
    devolva A
```

- MAKESET(x): cria conjunto unitário com elemento x;
- FINDSET(x): devolve identificador do conjunto da partição que contém x;
- UNION(x, y): substitui os conjuntos da partição que contêm x e y pela união deles.

Algoritmo de Kruskal

KRUSKAL (G, c)
1 $A \leftarrow \emptyset$
2 sejam $e_1, \ldots, e_m$ as arestas de G ordenadas por c
3 **para cada** $u \in V(G)$ **faça** MAKESET(u)
5 **para** $i \leftarrow 1$ **até** m **faça**
6   sejam u e v as pontas de e_i
7   **se** FINDSET$(u) \neq$ FINDSET(v)
8   **então** $A \leftarrow A \cup \{e_i\}$
9   UNION(u, v)
10 **devolva** A

Correção:
Note que e_i na linha 8 é uma aresta segura para A.

Union-Find

ED boa para representar uma partição de um conjunto, e as seguintes operações sobre a partição:

Union-Find

ED boa para representar uma partição de um conjunto, e as seguintes operações sobre a partição:

- MAKESET(x): cria conjunto unitário com elemento x;
- FINDSET(x): devolve identificador do conjunto da partição que contém x;
- UNION(x, y): substitui os conjuntos da partição que contêm x e y pela união deles.

Union-Find

ED boa para representar uma partição de um conjunto, e as seguintes operações sobre a partição:

- MAKESET(x): cria conjunto unitário com elemento x;
- FINDSET(x): devolve identificador do conjunto da partição que contém x;
- UNION(x, y): substitui os conjuntos da partição que contêm x e y pela união deles.

Custo de pior caso de cada operação: $\mathrm{O}(\lg n)$.

Union-Find

ED boa para representar uma partição de um conjunto, e as seguintes operações sobre a partição:

- MAKESET(x): cria conjunto unitário com elemento x;
- FINDSET(x): devolve identificador do conjunto da partição que contém x;
- UNION(x, y): substitui os conjuntos da partição que contêm x e y pela união deles.

Custo de pior caso de cada operação: $\mathrm{O}(\lg n)$.

Custo *amortizado* de cada operação: $\mathrm{O}(\lg^* n)$.

Union-Find

ED boa para representar uma partição de um conjunto, e as seguintes operações sobre a partição:

- MAKESET(x): cria conjunto unitário com elemento x;
- FINDSET(x): devolve identificador do conjunto da partição que contém x;
- UNION(x, y): substitui os conjuntos da partição que contêm x e y pela união deles.

Custo de pior caso de cada operação: $\mathrm{O}(\lg n)$.

Custo *amortizado* de cada operação: $\mathrm{O}(\lg^* n)$.

Análise amortizada: tópico a ser visto depois da prova.

Algoritmo de Kruskal

KRUSKAL (G, c)

$A \leftarrow \emptyset$
sejam $e_1, \ldots, e_m$ as arestas de G ordenadas por c
para cada $u \in V(G)$ **faça** MAKESET(u)
para $i \leftarrow 1$ **até** m **faça**
    sejam u e v as pontas de e_i
    se FINDSET$(u) \neq$ FINDSET(v)
        então $A \leftarrow A \cup \{e_i\}$
            UNION(u, v)
devolva A

Consumo de tempo do union-find:

MAKESET : $\mathrm{O}(1)$ FINDSET e UNION : $\mathrm{O}(\lg n)$

Algoritmo de Kruskal

KRUSKAL (G, c)

$A \leftarrow \emptyset$
sejam $e_1, \ldots, e_m$ as arestas de G ordenadas por c
para cada $u \in V(G)$ **faça** MAKESET(u)
para $i \leftarrow 1$ **até** m **faça**
    sejam u e v as pontas de e_i
    se FINDSET$(u) \neq$ FINDSET(v)
        então $A \leftarrow A \cup \{e_i\}$
            UNION(u, v)
devolva A

Consumo de tempo do union-find:
MAKESET : $O(1)$ FINDSET e UNION : $O(\lg n)$

Consumo de tempo do Kruskal: $O(m \lg n)$