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

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 peso w_e para cada aresta e,
encontrar árvore geradora em G de peso 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 peso w_e para cada aresta e,
encontrar árvore geradora em G de peso mínimo.

O peso de uma árvore é a soma do peso 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 peso w_e para cada aresta e,
encontrar árvore geradora em G de peso mínimo.

O peso de uma árvore é a soma do peso 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 peso w_e para cada aresta e,
encontrar árvore geradora em G de peso mínimo.

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

Tal árvore é chamada de árvore geradora mínima em G.

MST: minimum spanning tree

Arestas seguras

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

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

Arestas seguras

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

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

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

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 peso w_e para cada aresta $e \in E$.

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

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

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

GENÉRICO (G, w)

```
1  A ← ∅
2  enquanto A não é geradora faça
3      encontre aresta segura e para A
4      A ← A ∪ {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.

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, w), então e com $w(e)$ 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, w), então e com $w(e)$ mínimo em corte que respeita A é segura para A.

Prova: Seja T uma MST em (G, w) que contém A.
Se e está em corte que respeita A, então e não está em T.

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, w), então e com $w(e)$ mínimo em corte que respeita A é segura para A.

Prova: Seja T uma MST em (G, w) que contém A.
Se e está em corte que respeita A, então e não está em T.

Seja C o único circuito em $T + e$, e seja $f \in C$ com $w(f)$ máximo, 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, w), então e com $w(e)$ mínimo em corte que respeita A é segura para A.

Prova: Seja T uma MST em (G, w) que contém A.
Se e está em corte que respeita A, então e não está em T.

Seja C o único circuito em $T + e$, e seja $f \in C$ com $w(f)$ máximo, 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 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 Prim,
que mantém uma árvore T que contém um vértice s,
acrescentando em cada iteração uma aresta segura a T.

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

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

O algoritmo de Kruskal vai aumentando uma floresta.

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

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

Algoritmo de Prim

PRIM (G, w)

1 seja s um vértice arbitrário de G
2 **para** $v \in V(G) \setminus \{s\}$ **faça** $\text{key}[v] \leftarrow \infty$
3 $\text{key}[s] \leftarrow 0 \quad \pi[s] \leftarrow \text{nil}$
4 $Q \leftarrow V(G)$
5 **enquanto** $Q \neq \emptyset$ **faça**
6 $u \leftarrow$ EXTRACT-MIN(Q)
7 **para cada** $v \in \text{adj}(u)$ **faça**
8 **se** $v \in Q$ e $w(uv) < \text{key}(v)$
9 **então** $\pi[v] \leftarrow u \;\; \text{key}[v] \leftarrow w(uv)$
▷ DecreaseKey implícito na alteração do $\text{key}[v]$
10 **devolva** π

Algoritmo de Prim

```
PRIM (G, w)
 1  seja s um vértice arbitrário de G
 2  para v ∈ V(G) \ {s} faça key[v] ← ∞
 3  key[s] ← 0     π[s] ← nil
 4  Q ← V(G)
 5  enquanto Q ≠ ∅ faça
 6      u ← EXTRACT-MIN(Q)
 7      para cada v ∈ adj(u) faça
 8          se v ∈ Q e w(uv) < key(v)
 9              então π[v] ← u  key[v] ← w(uv)
                    ▷ DecreaseKey implícito na alteração do key[v]
10  devolva π
```

Consumo de tempo das operações na fila de prioridade:

Linha 4: $\Theta(n)$ EXTRACT-MIN e DECREASE-KEY : $\mathrm{O}(\lg n)$

Algoritmo de Prim

```
PRIM (G, w)
 1  seja s um vértice arbitrário de G
 2  para v ∈ V(G) \ {s} faça key[v] ← ∞
 3  key[s] ← 0    π[s] ← nil
 4  Q ← V(G)
 5  enquanto Q ≠ ∅ faça
 6      u ← EXTRACT-MIN(Q)
 7      para cada v ∈ adj(u) faça
 8          se v ∈ Q e w(uv) < key(v)
 9              então π[v] ← u  key[v] ← w(uv)
                  ▷ DecreaseKey implícito na alteração do key[v]
10  devolva π
```

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

Algoritmo de Prim

```
PRIM (G, w)
 1  seja s um vértice arbitrário de G
 2  para v ∈ V(G) \ {s} faça key[v] ← ∞
 3  key[s] ← 0    π[s] ← nil
 4  Q ← V(G)
 5  enquanto Q ≠ ∅ faça
 6      u ← EXTRACT-MIN(Q)
 7      para cada v ∈ adj(u) faça
 8          se v ∈ Q e w(uv) < key(v)
 9              então π[v] ← u  key[v] ← w(uv)
                  ▷ DecreaseKey implícito na alteração do key[v]
10  devolva π
```

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

Consumo de tempo com Fibonacci heap: $O(m + n \lg n)$