Análise e Síntese de Algoritmos
Árvores abrangentes de menor custo (MSTs). Algoritmo de Prim.

Prof. Pedro T. Monteiro

IST - Universidade de Lisboa

2024/2025

Contexto

- Revisão [CLRS, Cap.1-13]
  - Fundamentos; notação; exemplos
- Técnicas de Síntese de Algoritmos [CLRS, Cap.15-16]
  - Programação dinâmica
  - Algoritmos greedy
- Algoritmos em Grafos [CLRS, Cap.21-26]
  - Algoritmos elementares
  - Caminhos mais curtos [CLRS, Cap.22,24-25]
  - Árvores abrangentes [CLRS, Cap.23]
  - Fluxos máximos [CLRS, Cap.26]
- Programação Linear [CLRS, Cap.29]
  - Algoritmos e modelação de problemas com restrições lineares
- Tópicos Adicionais [CLRS, Cap.32-35]
  - Complexidade Computacional

Resumo

Motivação

Problema
Suponha que pretende instalar uma nova rede de fornecimento para um serviço (TV por cabo, gás natural, ...) numa urbanização

Para estabelecer a rede é necessário fazer obras na via pública para instalar a infraestrutura (colocação de cabos de fibra óptica ou novas condutas de gás)

Objectivo
Fornecer o serviço a todas as casas da urbanização através de uma rede.
No entanto, cada possível ligação na urbanização tem um custo e pretende-se minimizar o custo total da instalação

Motivação

Solução

- Cada casa da urbanização é modelada como um vértice num grafo
- Cada possível ligação entre casas corresponde a um arco pesado cujo peso indica o custo da ligação
- A solução do problema corresponde à árvore abrangente de menor custo (*Minimum Spanning Tree (MST)*) do grafo

Árvores Abrangentes de Menor Custo

Árvore Abrangente

- Um grafo não dirigido $G = (V, E)$, diz-se ligado se para qualquer par de vértices existe um caminho que liga os dois vértices
- Dado grafo não dirigido $G = (V, E)$, ligado, uma árvore abrangente é um sub-conjunto acíclico $T \subseteq E$, que liga todos os vértices
- O tamanho da árvore é $|T| = |V| - 1$

Árvores Abrangentes de Menor Custo

Árvore Abrangente de Menor Custo

Dado grafo $G = (V, E)$, ligado, não dirigido, com uma função de pesos $w : E \to R$, identificar uma árvore abrangente T, tal que a soma dos pesos dos arcos de T é minimizada

$$\min\ w(T) = \sum_{(u,v)\in T} w(u, v)$$

Algoritmo (greedy) genérico

Abordagem Greedy

- Manter conjunto A que é um sub-conjunto de uma MST T
- A cada passo do algoritmo identificar arco (u, v) que pode ser adicionado a A sem violar a invariante
- $A \cup \{(u, v)\}$ é sub-conjunto de uma MST T
  - (u, v) é declarado um arco seguro para A

Invariante

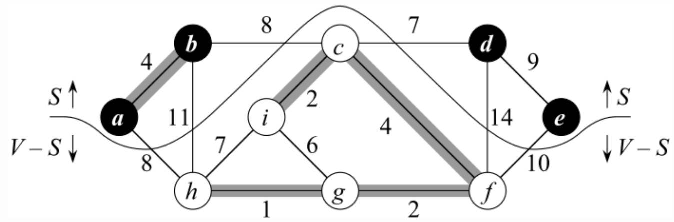
Antes de cada iteração, A é um sub-conjunto de uma MST T

Algoritmo (greedy) genérico

ST-Genérico(G,w) (para já ainda sem Minimum, apenas Spanning Tree)

```
A = ∅
while A não forma árvore abrangente do
  identificar arco seguro (u,v) para A
  A = A ∪ {(u,v)}
end while
return A
```


Algoritmo (greedy) genérico



Definições

- Um corte $(S, V-S)$ de um grafo não dirigido $G=(V,E)$ é uma partição de V
- Um arco $(u,v) \in E$ cruza o corte $(S, V-S)$ se um dos extremos está em S e o outro está em $V-S$
- Um corte respeita um conjunto de arcos A se nenhum arco de A cruza o corte
- Um arco diz-se um arco leve que cruza um corte se o seu peso é o menor de todos os arcos que cruzam o corte

Algoritmo (greedy) genérico

Definições

- Seja $G=(V,E)$ um grafo não dirigido, ligado, com função de pesos w
- Seja A um sub-conjunto de E incluído numa MST T $(A \subseteq T \subseteq E)$

$\Rightarrow$ Então (u,v) é um arco seguro para A

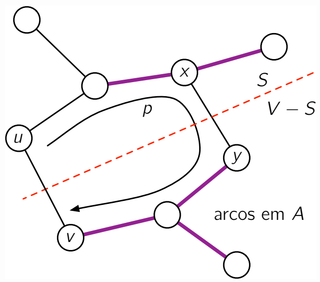
Prova

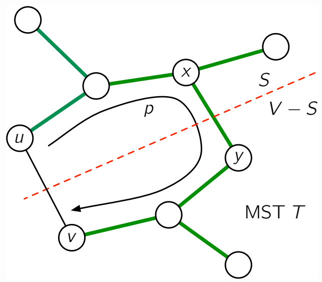
- MST T, com $A \subseteq T$, e arco leve $(u,v) \notin T$
- Objectivo: Construir outra MST T' que inclui $A \cup \{(u,v)\}$
- (u,v) é um arco seguro para A

Algoritmo (greedy) genérico

Critérios de Optimalidade

- O arco (u,v) forma ciclo com arcos do caminho p, definido em T, que liga u a v
- Dado u e v estarem nos lados opostos do corte $\{S\}/\{V-S\}$, então existe pelo menos um arco (x,y) do caminho p em T que cruza o corte



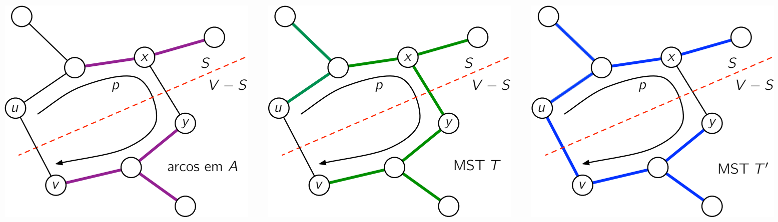


Algoritmo (greedy) genérico

Critérios de Optimalidade (cont.)

Arco (x, y)

- $(x, y) \notin A$, porque corte $\{S\}/\{V - S\}$ respeita A
- Remoção de (x, y) divide T em dois componentes
- Inclusão de (u, v) permite formar $T' = T \setminus \{(x, y)\} \cup \{(u, v)\}$
- Dado que (u, v) é um arco leve que cruza o corte $\{S\}/\{V - S\}$, e porque (x, y) também cruza o corte: $w(u, v) \le w(x, y)$



Algoritmo (greedy) genérico

Critérios de Optimalidade (cont.)

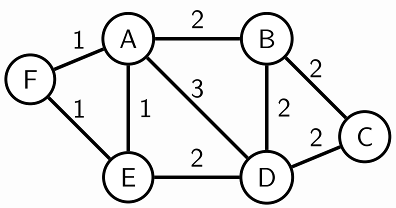
- $w(T') = w(T) - w(x, y) + w(u, v)$
- $w(T') \le w(T)$ porque $w(u, v) \le w(x, y)$
- Mas T é MST, pelo que $w(T) \le w(T')$, por definição de MST
- Logo, $w(T') = w(T)$, e T' também é MST

(u, v) é seguro para A:

- Verifica-se $A \subseteq T'$, dado que por construção $A \subseteq T$, e $(x, y) \notin A$
- Assim, verifica-se também $A \cup (u, v) \subseteq T'$
- T' é MST, pelo que (u, v) é seguro para A

Algoritmo (greedy) genérico

Exercício: Quantas MST's existem?



Algoritmo de Prim

Algoritmo de Prim

- Algoritmo greedy
- MST construída a partir de um vértice raíz r
- Algoritmo mantém sempre uma árvore A $(A \subseteq T)$
- Árvore A é extendida a partir do vértice r
- A cada passo é escolhido um arco leve, seguro para A
- Utilização de fila de prioridade Q

Notação

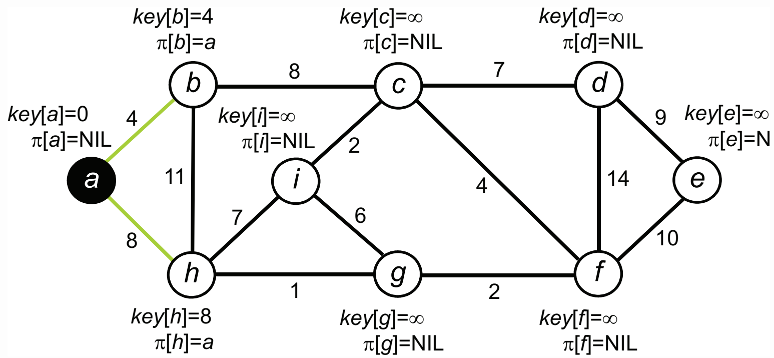
- key$[v]$: menor peso de qualquer arco que ligue v a um vértice na árvore
- $\pi[v]$: antecessor de v na árvore

MST-Prim(G,w,r)

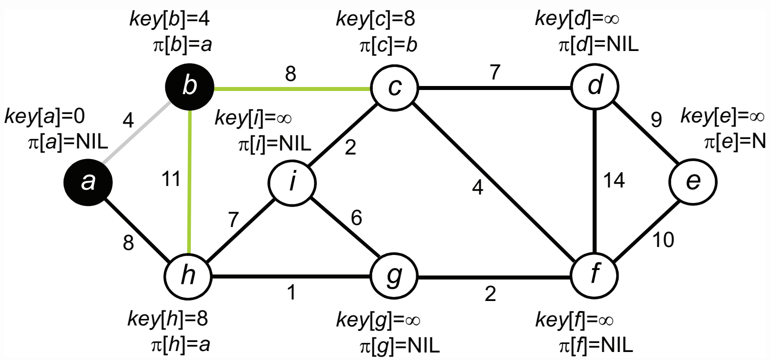
```
for each u ∈ G.V do
  key[u] = ∞                                   {Inicialização}
  π[u] = NIL
end for
key[r] = 0
Q = G.V                                        {Fila de prioridade}
while Q ≠ ∅ do
  u = Extract-Min(Q)
  for each v ∈ Adj[u] do
    if v ∈ Q and w(u,v) < key[v] then
      π[v] = u
      key[v] = w(u,v)                          {Actualização de Q}
    end if
  end for
end while
```

Invariante

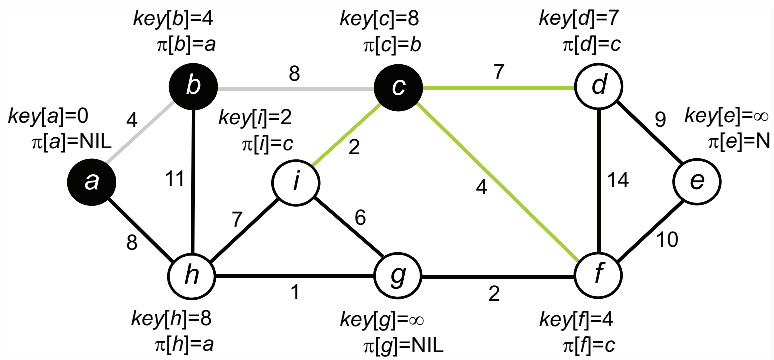
$\forall v \in Q, key[v] = min(\{w(u,v) | u \notin Q\})$



Q: b, h, c, d, e, f, g, i

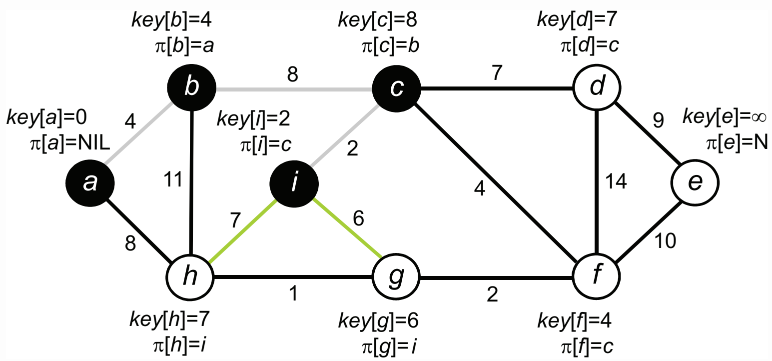


Q: c, h, d, e, f, g, i



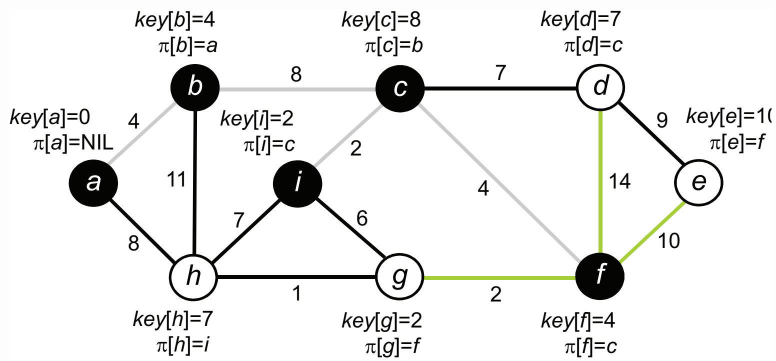
Q: i, f, d, h, e, g

Algoritmo de Prim



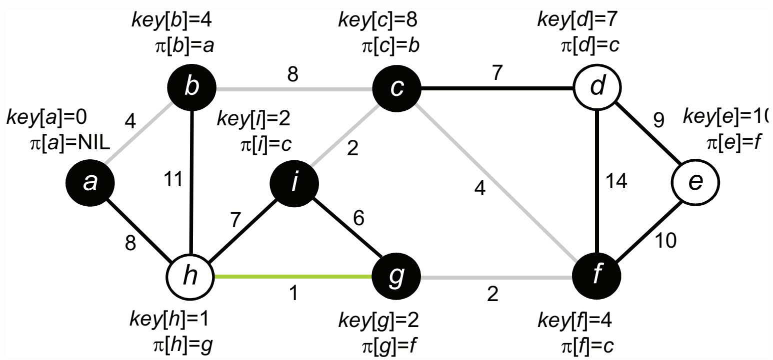
Q: f, g, d, h, e

Algoritmo de Prim



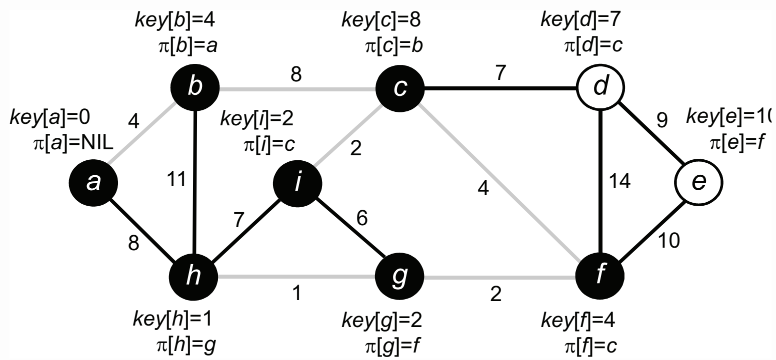
Q: g, d, h, e

Algoritmo de Prim



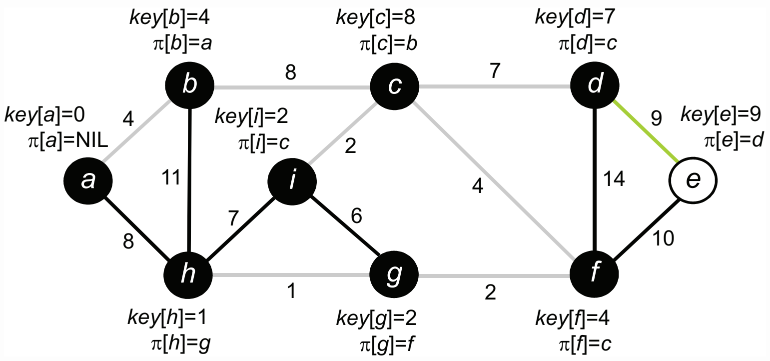
Q: h, d, e

Algoritmo de Prim



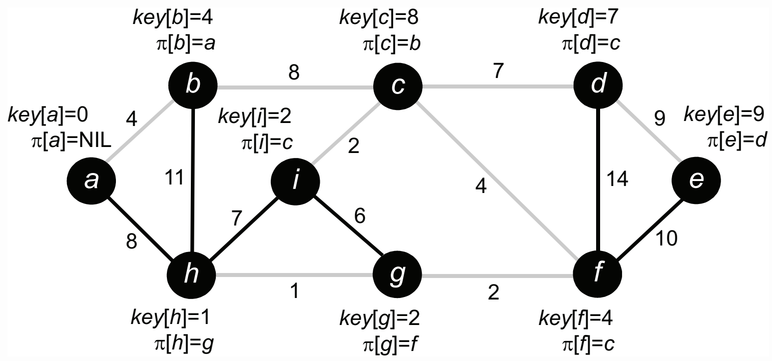
Q: d, e

Algoritmo de Prim



Q: e

Algoritmo de Prim



Q: $\emptyset$

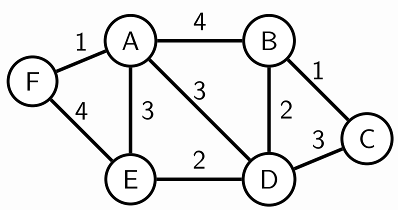
Algoritmo de Prim

Complexidade

- Fila de prioridade baseada em amontoados (heap)
- Extração de vértice da fila Q, implica actualização de Q
  - Cada vértice é extraído apenas 1 vez: $\Theta(V)$
  - Actualização de Q: $O(log\ V)$
  - Logo: $O(V\ log\ V)$
- Para cada arco (i.e. $\Theta(E)$) existe no pior-caso uma actualização de Q em $O(log\ V)$
- Complexidade algoritmo Prim: $O(V\ log\ V + E\ log\ V)$
- Logo, é possível assegurar $O(E\ log\ V)$ porque grafo é ligado
  - Grafo ligado: $|E| \geq |V| - 1$

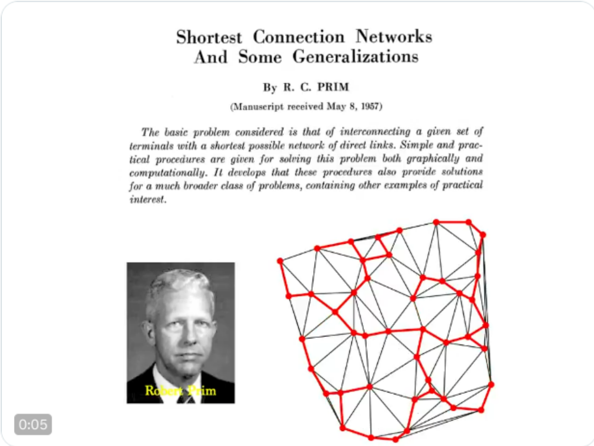
Algoritmo de Prim

Exercício: Calcule a MST usando o algoritmo de Prim (nó raiz F)



Gabriel Peyré
@gabrielpeyre

Oldies but goldies: R. C. Prim, Shortest connection networks and some generalizations, 1957. A greedy algorithm to compute the minimum spanning tree of a graph.



Questões?

Dúvidas?