Árvores geradoras mínima

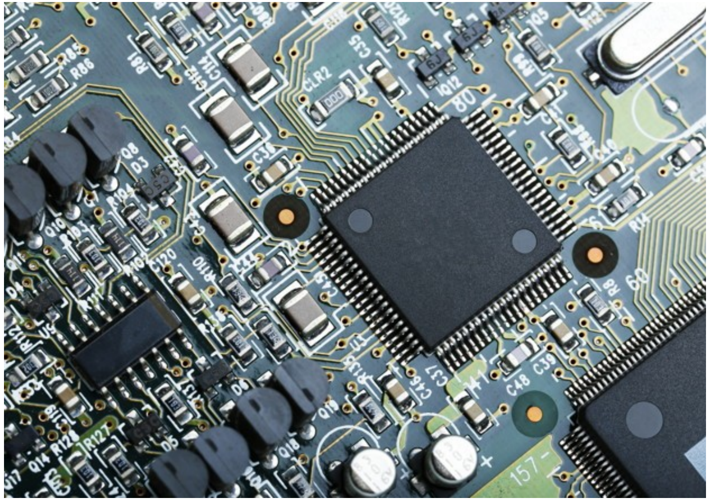
prof. Dr. Wesin Ribeiro

Neste capítulo

- Introdução
- A árvore geradora mínima
- O método genérico
- O algoritmo de Kruskal
- O algoritmo de Prim
- Revisão

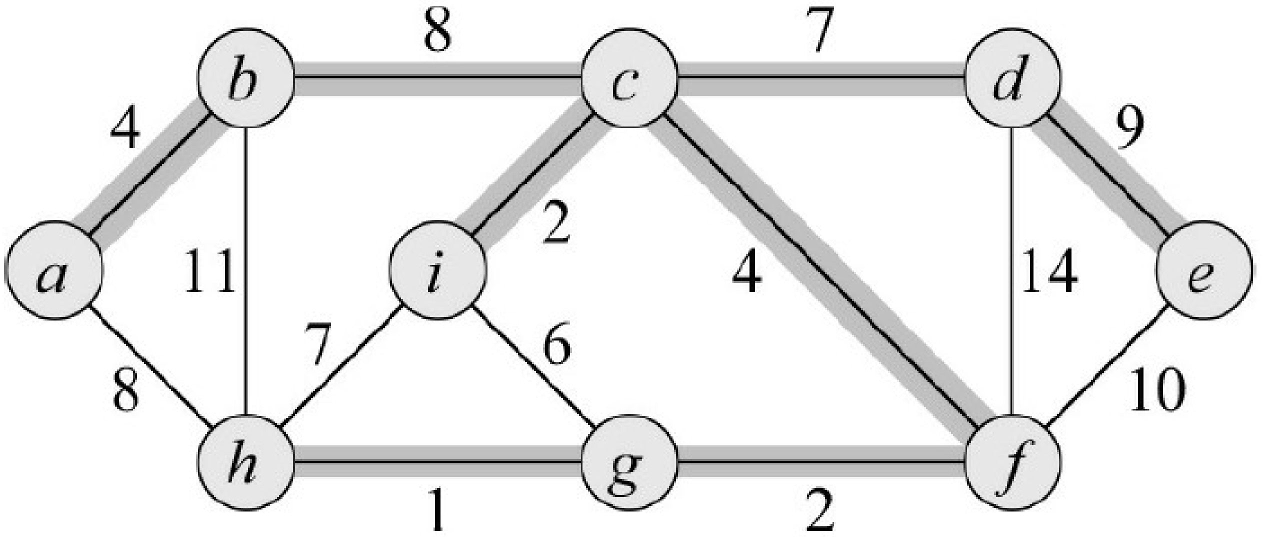
Introdução

No capítulo anterior entramos no universo dos grafos conhecendo seus conceitos e algoritmos básicos. Neste capítulo iremos começar a expandir esse universo apresentando dois algoritmos (Kruskal e Prim) usados para resolver o problema da árvore geradoras mínimas. Esse problema consiste basicamente em encontrar o caminho com menor custo em um grafo conexo não dirigido ponderado.



A árvore geradora mínima

Os dois algoritmos são algoritmos gulosos. Cada etapa de um algoritmo guloso deve fazer uma entre várias opções possíveis. A estratégia gulosa faz a escolha que é a melhor no momento.



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

$$T \subseteq E$$

Método genérico

Antes de cada iteração, A é um subconjunto de alguma árvore geradora mínima. Em cada etapa, determinamos uma aresta (u, v) que pode ser adicionada a A sem violar esse invariante.

```
GENERIC-MST(G, w)
    A = ∅
    while A não formar uma árvore geradora
        encontre uma aresta (u, v) que seja segura para A
        A = A ∪ {(u, v)}
    return A
```

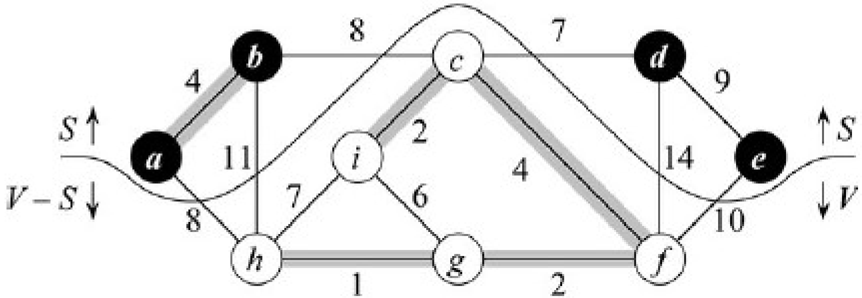
Como encontrar a aresta segura?

Corte (S, V-S) é uma partição dos vértices V contidos no grafo não dirigido.

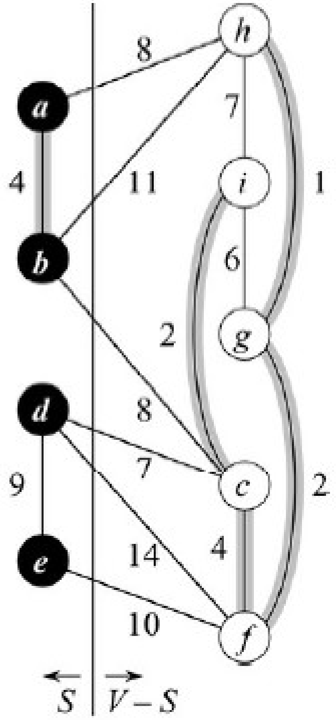
Cruzar o corte (S, V-S) ocorre quando um aresta ultrapassa o limite do corte.

Respeitar o corte (S, V-S) ocorre quando a aresta não ultrapassa o corte.

Uma aresta é uma **aresta leve** que cruza o corte se seu peso é mínimo.



(a)



(b)

O Algoritmo de Kruskal

O algoritmo de Kruskal acha uma aresta segura para adicionar à floresta que está sendo desenvolvida encontrando, entre todas as arestas que conectam quaisquer duas árvores na floresta, uma aresta (*u, v) de peso* mínimo.

- O príncipio do algorítimo é a geração de uma floresta, antes de gerar a árvore geradora mínima
  - ✓ Floresta = grafo não orientado acíclico
  - ✓ Várias árvores
- Busca a aresta segura
- Aresta de peso mínimo que conecta duas árvores na floresta
- Garantia de ser árvore geradora mínima apenas depois da ultima iteração
- Curiosidades
  - ✓ Criado por Joseph Bernard Kruskal, Jr
  - ✓ Nascido em 1928
  - ✓ Terminou seu phd na Universidade de Princeton em 1956

Pseudo código

Utiliza uma estrutura de dados de conjuntos disjuntos para manter vários conjuntos disjuntos de elementos

MST-KRUSKAL(G, w)

$A = \emptyset$
for cada vértice $v \in G.V$
    MAKE-SET(v)
ordene as arestas de $G.E$ em ordem não decrescente de peso w
for cada aresta $(u, v) \in G.E$, tomada em ordem não decrescente de peso
    if FIND-SET$(u) \neq$ FIND-SET(v)
        $A = A \cup \{(u, v)\}$
        UNION(u, v)
return A

Detalhes do pseudo código

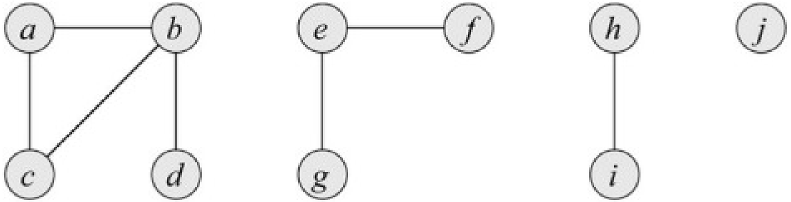
Utiliza uma estrutura de dados de conjuntos disjuntos para manter vários conjuntos disjuntos de elementos

MAKE-SET(x) cria um novo conjunto cujo único membro (e, portanto, o representante) é x. Visto que os conjuntos são disjuntos, exigimos que x ainda não esteja em algum outro conjunto.

UNION(x, y) une os conjuntos dinâmicos que contêm x e y, digamos Sx e Sy, em um novo conjunto que é a união desses dois conjuntos

FIND-SET(x) retorna um ponteiro para o representante do (único) conjunto que contém x.

A coleção de conjuntos disjuntos após o processamento de cada aresta.



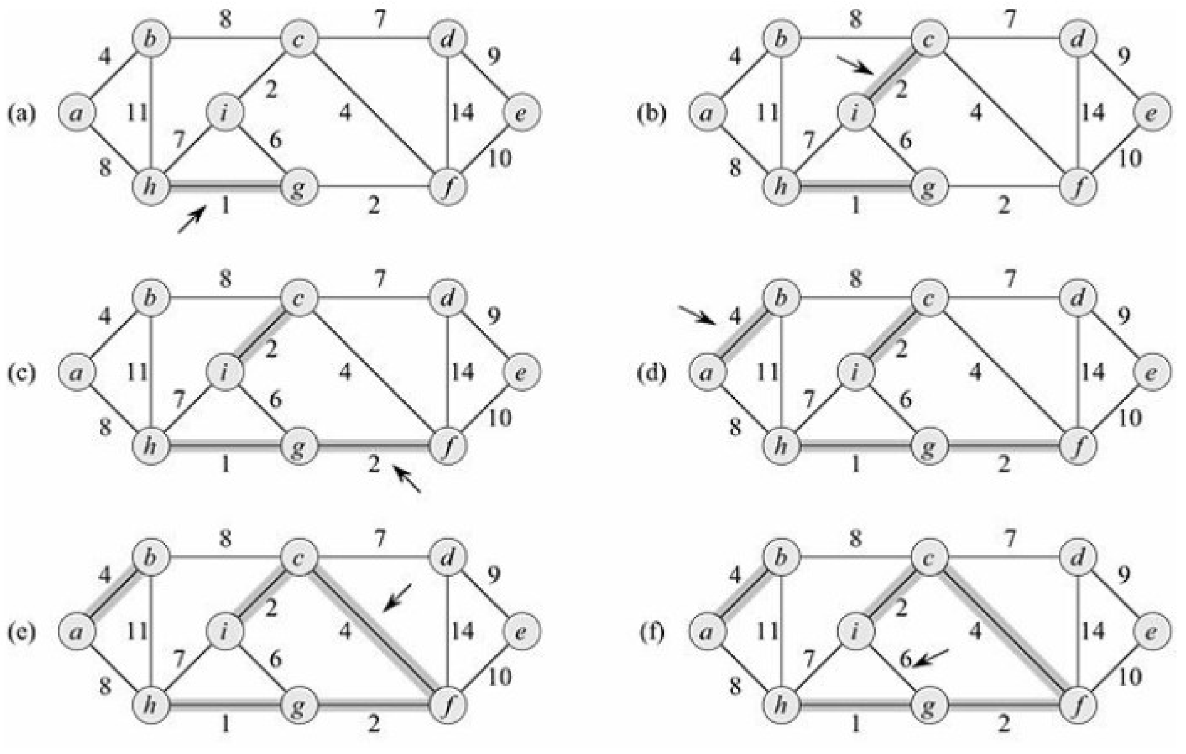
(a)

Aresta processada	Coleção de conjuntos disjuntos								
conjuntos iniciais	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{e\}$	$\{f\}$	$\{g\}$	$\{h\}$	$\{i\}$	$\{j\}$
(b,d)	$\{a\}$	$\{b,d\}$	$\{c\}$		$\{e\}$	$\{f\}$	$\{g\}$	$\{h\}$	$\{i\}$	$\{j\}$
(e,g)	$\{a\}$	$\{b,d\}$	$\{c\}$		$\{e,g\}$	$\{f\}$		$\{h\}$	$\{i\}$	$\{j\}$
(a,c)	$\{a,c\}$	$\{b,d\}$			$\{e,g\}$	$\{f\}$		$\{h\}$	$\{i\}$	$\{j\}$
(h,i)	$\{a,c\}$	$\{b,d\}$			$\{e,g\}$	$\{f\}$		$\{h,i\}$		$\{j\}$
(a,b)	$\{a,b,c,d\}$				$\{e,g\}$	$\{f\}$		$\{h,i\}$		$\{j\}$
(e,f)	$\{a,b,c,d\}$				$\{e,f,g\}$			$\{h,i\}$		$\{j\}$
(b,c)	$\{a,b,c,d\}$				$\{e,f,g\}$			$\{h,i\}$		$\{j\}$

(b)

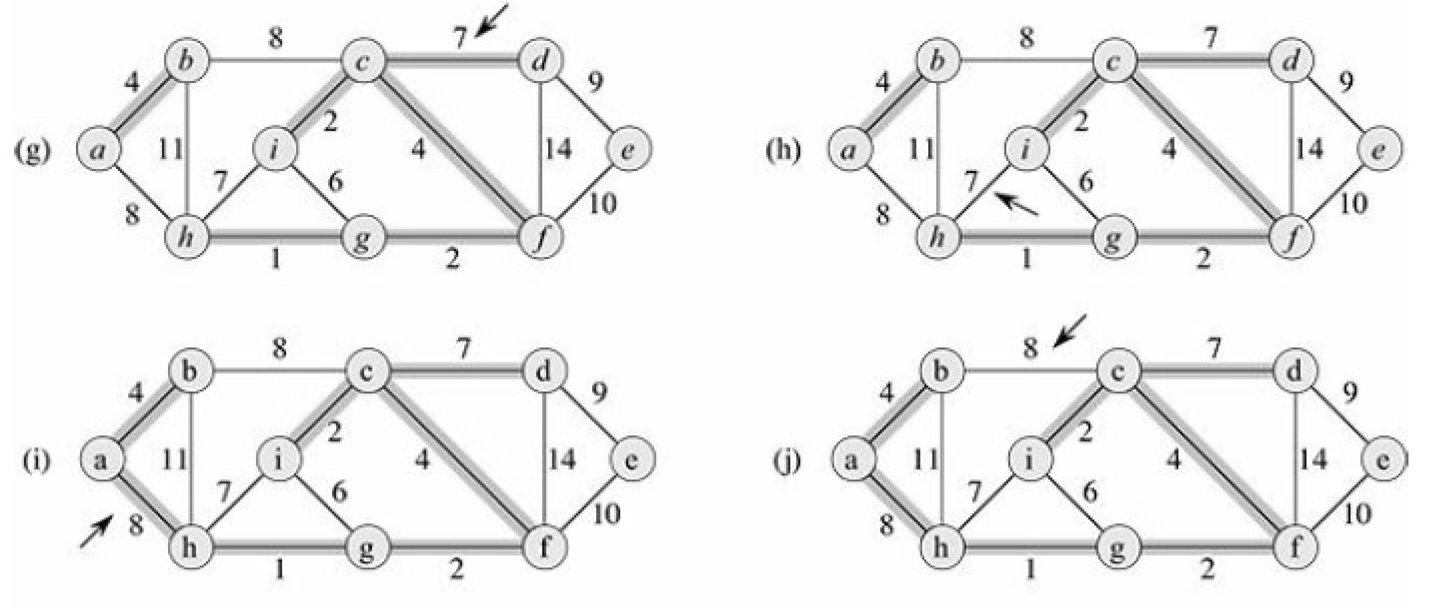
O Algoritmo de Kruskal

As arestas sombreadas pertencem a árvore geradora mínima que está sendo formada. O algoritmo ordena as arestas por peso de modo crescente.



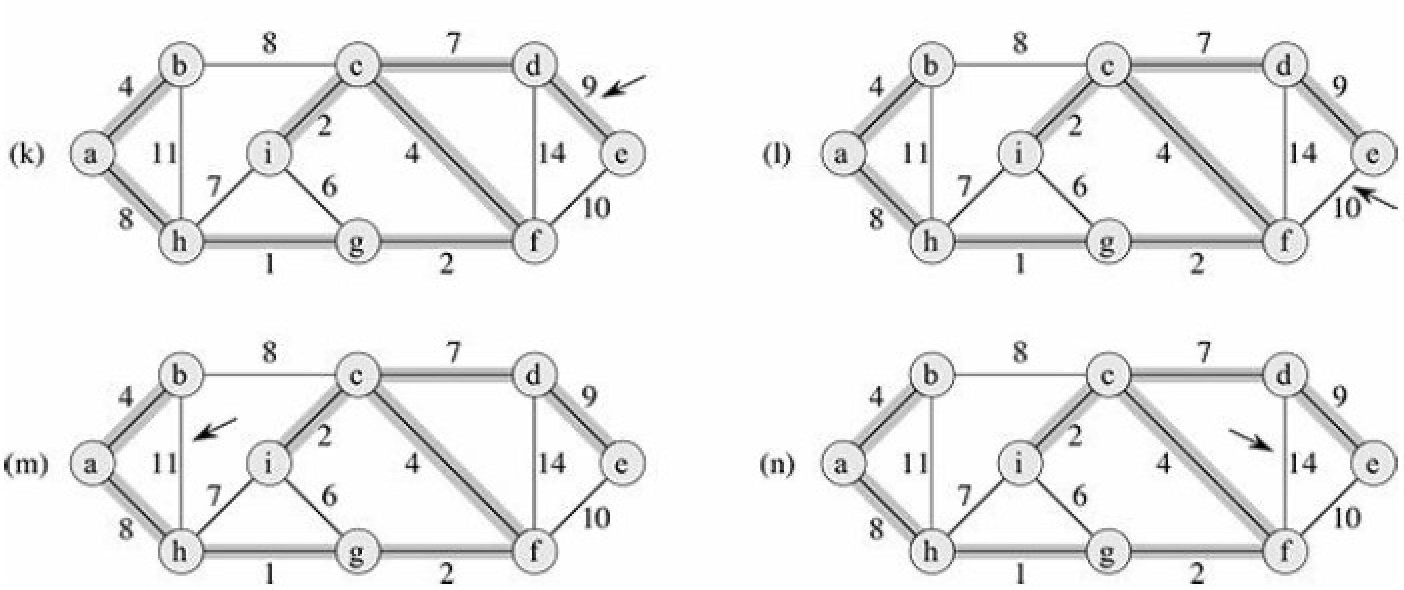
O Algoritmo de Kruskal

As arestas sombreadas pertencem a árvore geradora mínima que está sendo formada. O algoritmo ordena as arestas por peso de modo crescente.



O Algoritmo de Kruskal

Se a aresta une duas árvores distintas na floresta, ela é adicionada à floresta, juntando assim as duas árvores.



O Algoritmo de Prim

Como o algoritmo de Kruskal, o algoritmo de Prim é um caso especial do método genérico de árvore geradora mínima

- As arestas no conjunto A sempre formam uma árvore única
- Começa em um vértice arbitrário r
- Aumenta até que a árvore abranja todos os vértices em V
- O caminho não pode forma ciclos
- O caminho precisa ser conexo
- Reinicia sempre pelo caminho de menor peso
- Utiliza uma fila de prioridade mínima para auxiliar na construção.

Pseudo código

Como o algoritmo de Kruskal, o algoritmo de Prim é um caso especial do método genérico de árvore geradora mínima

Extract-Min(Q) elimina o elemento da fila Q cujo chave é mínima, retornando o ponteiro para o elemento extraído

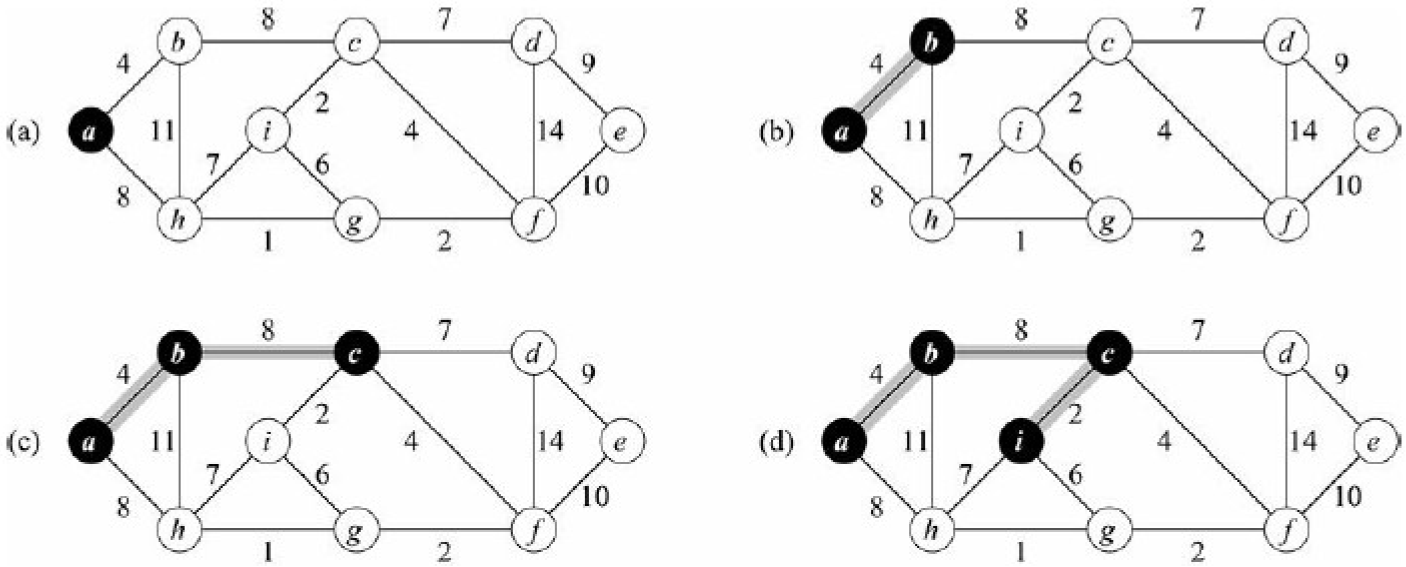
$$A = \{(v, v.\pi) : v \in V - \{r\}\} \;.$$

MST-PRIM(G, w, r)

```
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```

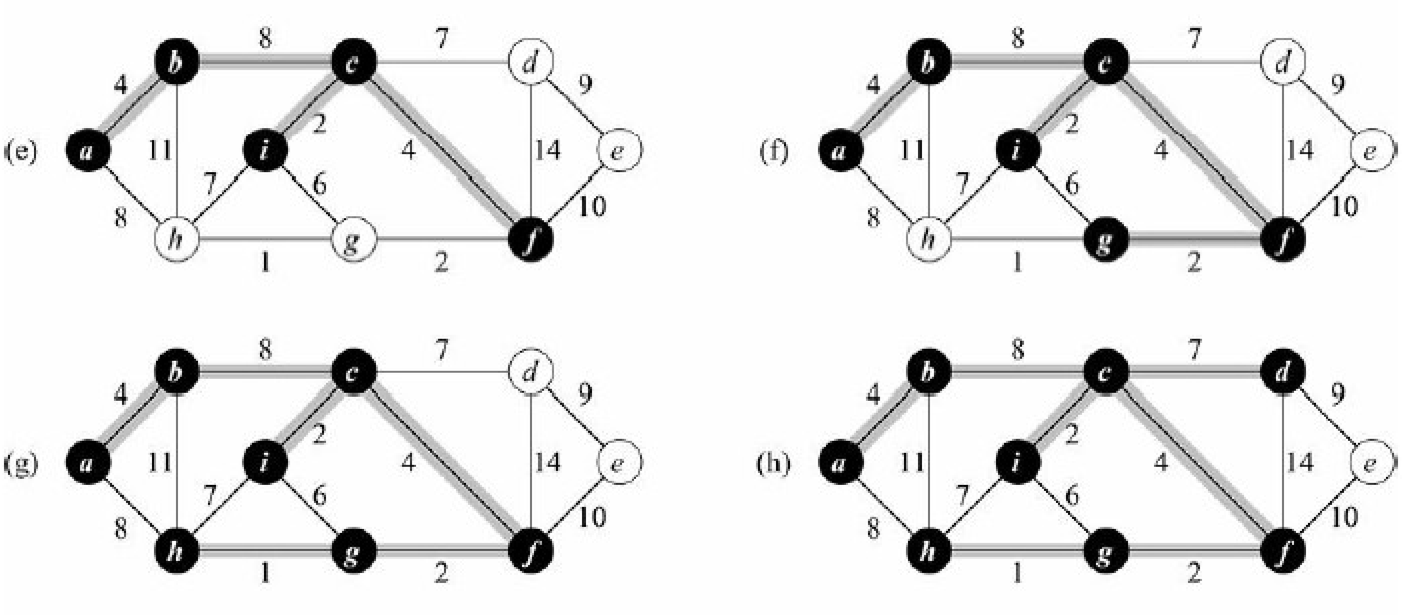
O Algoritmo de Prim

O vértice raiz é a. Arestas sombreadas estão na árvore que está sendo desenvolvida, e os vértices pretos estão na árvore.



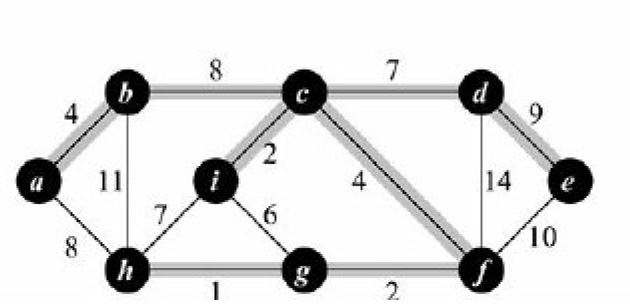
O Algoritmo de Prim

Em cada etapa do algoritmo, os vértices na árvore determinam um corte do grafo, e uma aresta leve que cruza o corte é acrescentada à árvore.



O Algoritmo de Prim

Na segunda etapa, por exemplo, o algoritmo tem a opção de adicionar a aresta (b, c) ou a aresta (a, h) à árvore, visto que ambas são arestas leves que cruzam o corte.



Exercícios

1) Seja (u,v) uma aresta de peso mínimo em um grafo conexo G. Mostre que (u, v) pertence a alguma árvore geradora mínima de G.

2) Mostre que, se uma aresta (u, v) está contida em alguma árvore geradora mínima, então ela é uma aresta leve que cruza algum corte do grafo.

3) O algoritmo de Kruskal pode devolver diferentes árvores geradoras para o mesmo grafo de entrada G, dependendo de como as ligações são rompidas quando as arestas são ordenadas. Mostre que, para cada árvore geradora mínima T de G, existe um modo de ordenar as arestas de G no algoritmo de Kruskal, de tal forma que o algoritmo retorne T.

4) Suponha que representamos o grafo G = (V, E) como uma matriz de adjacências. Dê uma implementação simples do algoritmo de Prim para esse caso que seja executada no tempo O(V2).

Revisão

Atenção, chegou a hora da revisão.