Pontifícia Universidade Católica de Minas Gerais
Bacharelado em Ciência da Computação
Teoria dos Grafos

Teoria dos Grafos

Prof.: Felipe Domingos
felipe@pucminas.br

Árvores
Árvores Geradoras Mínimas

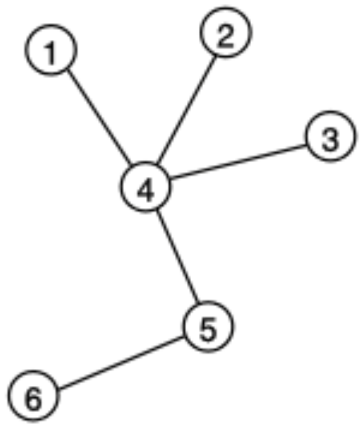
Cormen – páginas 445 até 458

e

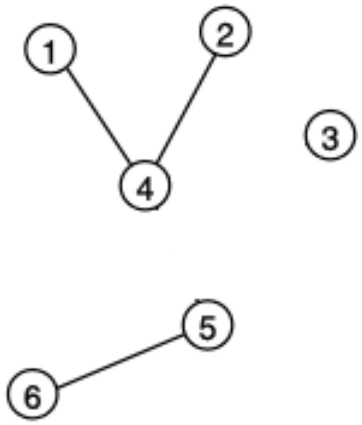
Deo – páginas 39 até 44 e 55 até 57 e 61 até 64

Árvore

- Uma árvore é um grafo conexo (existe caminho entre quaisquer dois de seus vértices) e acíclico (não possui ciclos)
- Grafo acíclico mas não conexo é chamado de floresta
- Uma floresta também é definida como uma união disjunta de árvores



árvore com 6 vértices e 5 arestas



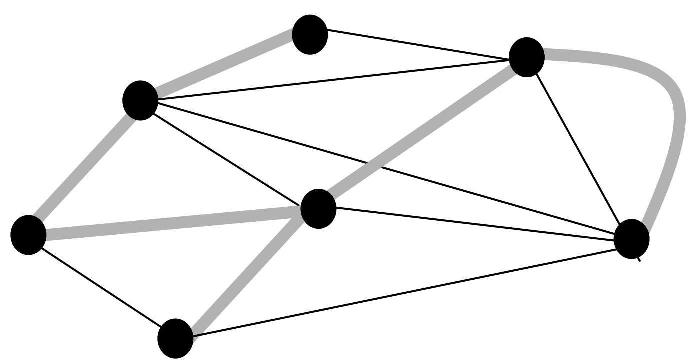
floresta 6 vértices e 3 arestas

Árvore

- Propriedades:
  - Em uma árvore existe um e apenas um caminho entre cada par de vértices.
  - Uma árvore com n vértices tem $n-1$ arestas
  - Um floresta com n vértices e k componentes possui $n-k$ arestas
  - Um grafo G é uma árvore se, e somente se, a remoção de qualquer aresta o desconectar (grafo minimamente conectado)
  - Um grafo G com n vértices, $n-1$ arestas e nenhum ciclo é conexo
  - Toda árvore é um grafo bipartido e planar
  - Em qualquer árvore com pelo menos 2 vértices existem pelo menos 2 vértices pendentes

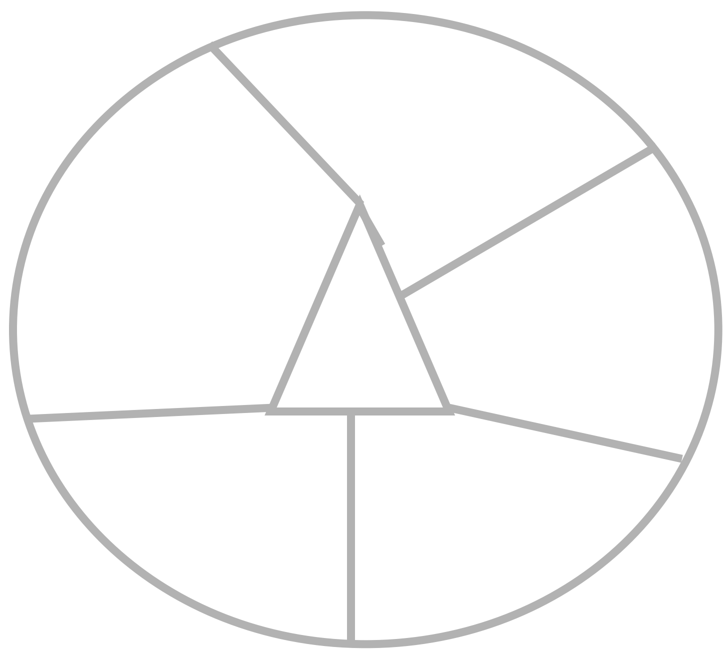
Árvore Geradora (Árvore Espalhada ou Árvore de Extensão)

- Uma <u>árvore geradora </u>de um grafo conexo G é um subgrafo de G que contém todos os vértices de G e é uma árvore
- <u>Árvore geradora </u>só é definida para grafos conexo porque toda árvore é conexa. Grafos não conexo possuem florestas geradoras



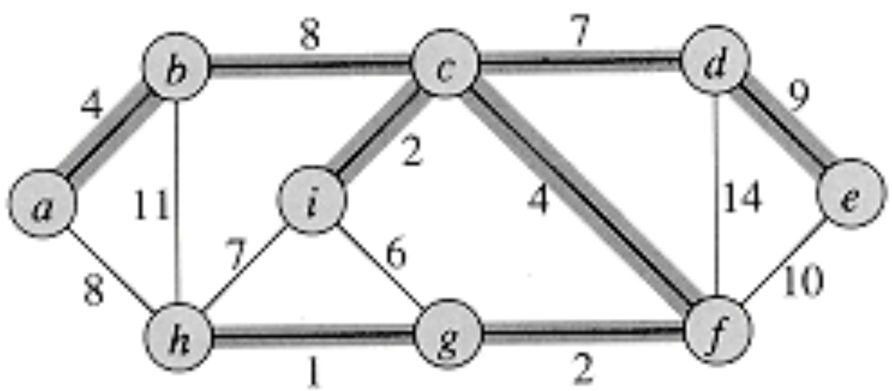
Árvore Geradora

- Um fazendeiro possui 6 lotes murados como mostrado na figura abaixo. Os lotes estão cheios de água. Quantos muros devem ser removidos para que toda a água seja liberada?



Árvore Geradora Mínima

- Árvore Geradora Mínima é a árvore geradora de menor peso de G
- Dado um grafo G com pesos associados às arestas, encontrar uma árvore geradora mínima de G



Árvore Geradora Mínima

- A partir de um grafo conectado não orientado $G=(V,E)$ com uma função peso $w:E\rightarrow R$, desejamos encontrar uma árvore geradora mínima correspondente a G
- Algoritmos:
  - Algoritmo de Kruskal
  - Algoritmo de Prim

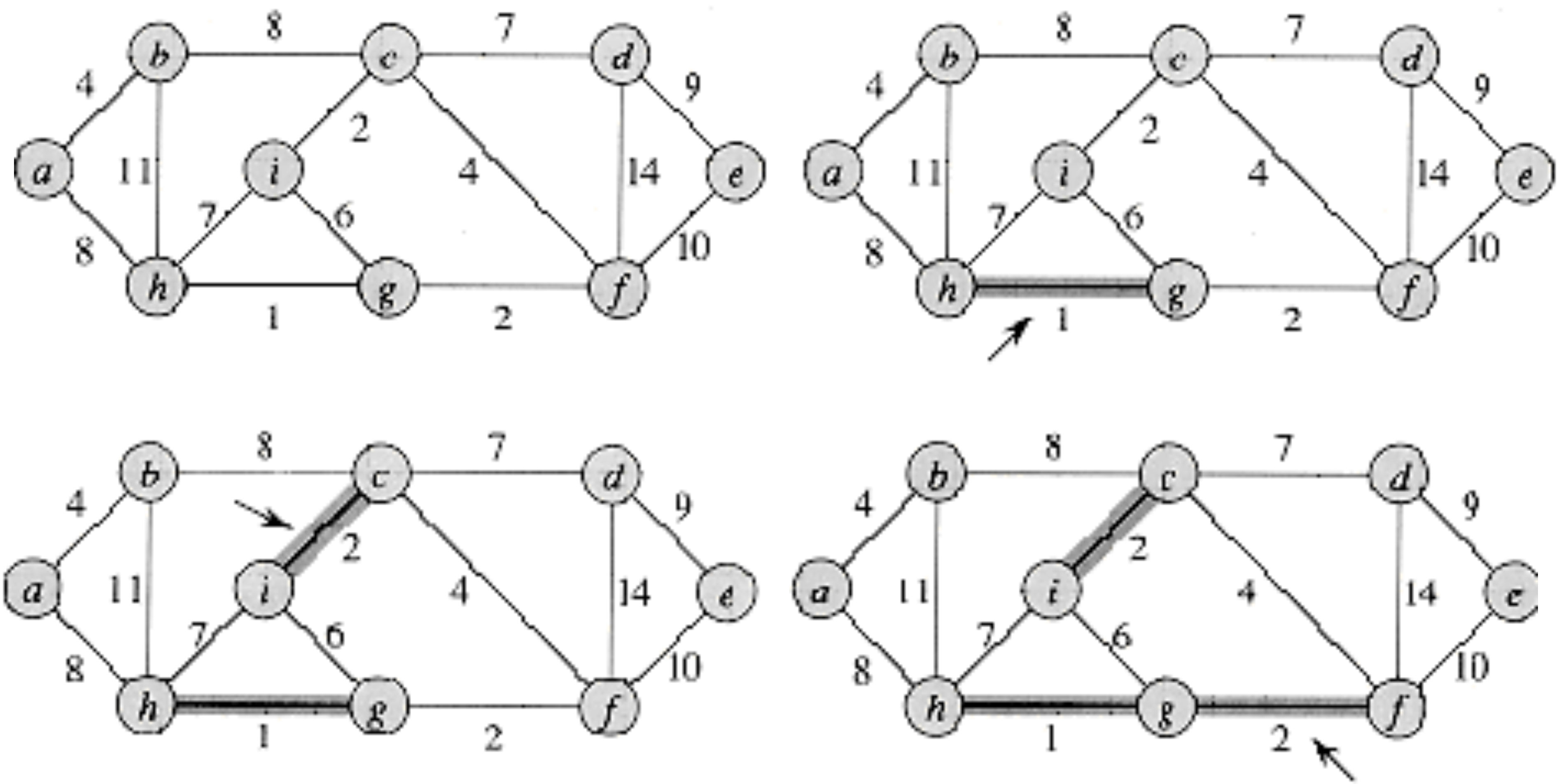
Algoritmo de Kruskal

- Encontre uma aresta segura para adicionar à floresta crescente encontrando, de todas as arestas que conectam duas árvores quaisquer na floresta, uma aresta `(u,v)` de peso mínimo
- Utiliza um estrutura de dados de conjuntos disjuntos para manter vários conjuntos disjuntos de elementos
- Cada conjunto contém os vértices em uma árvore da floresta atual
- A operação `FIND-SET(u)` retorna um elemento representativo do conjunto que contém `u`
- A combinação de árvores é realizada pelo procedimento `UNION`

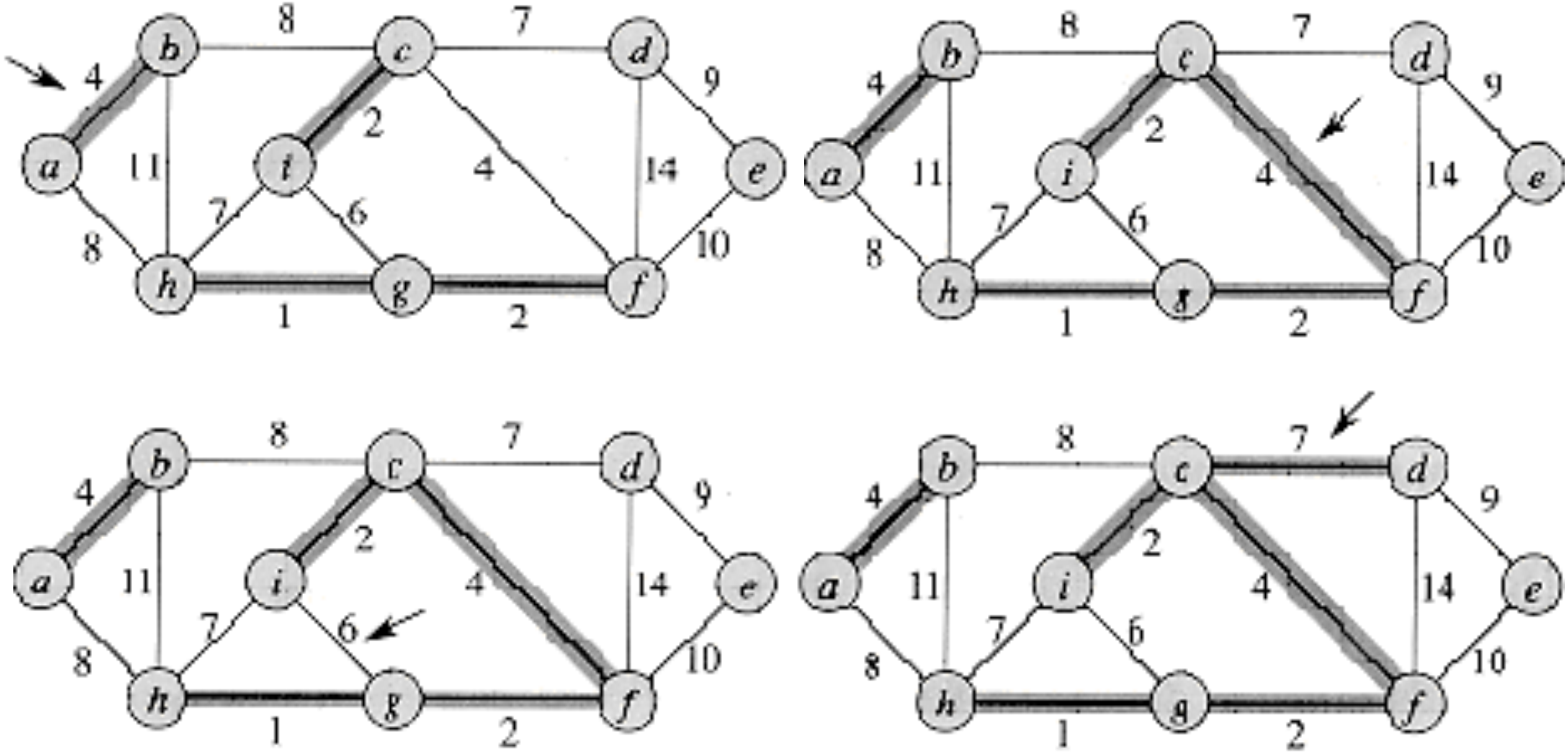
Algoritmo de Kruskal

```
MST-KRUSKAL(G,w)
A ← ∅
for cada vértice v ∈ V[G]
    do MAKE-SET(v)
ordenar as arestas de E por peso w não decrescente
for cada aresta (u,v)∈E, em ordem de peso não decrescente do
    if FIND-SET(u) ≠ FIND-SET(v) then
        A ← A ∪ {(u,v)}
        UNION(u,v)
return A
```

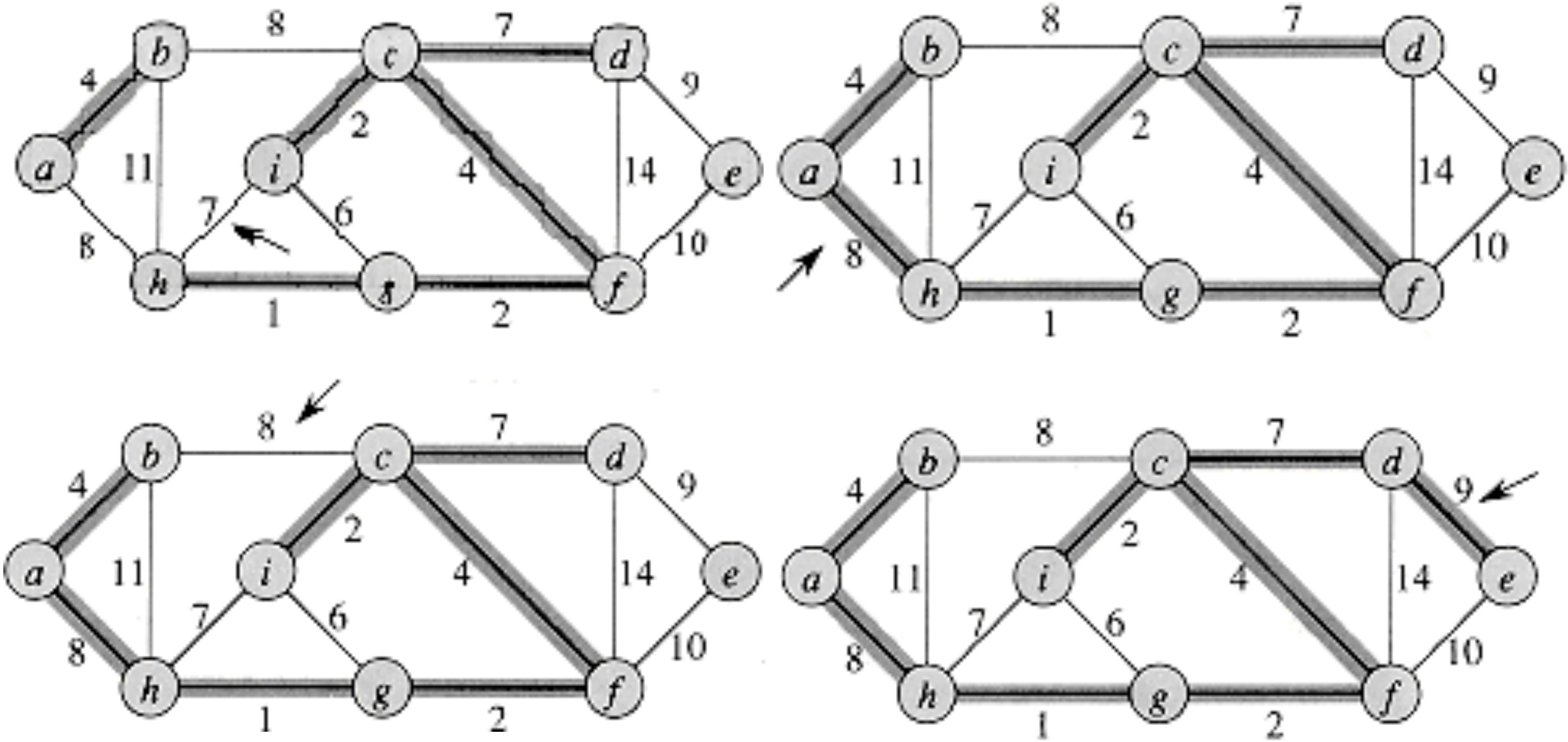
Algoritmo de Kruskal



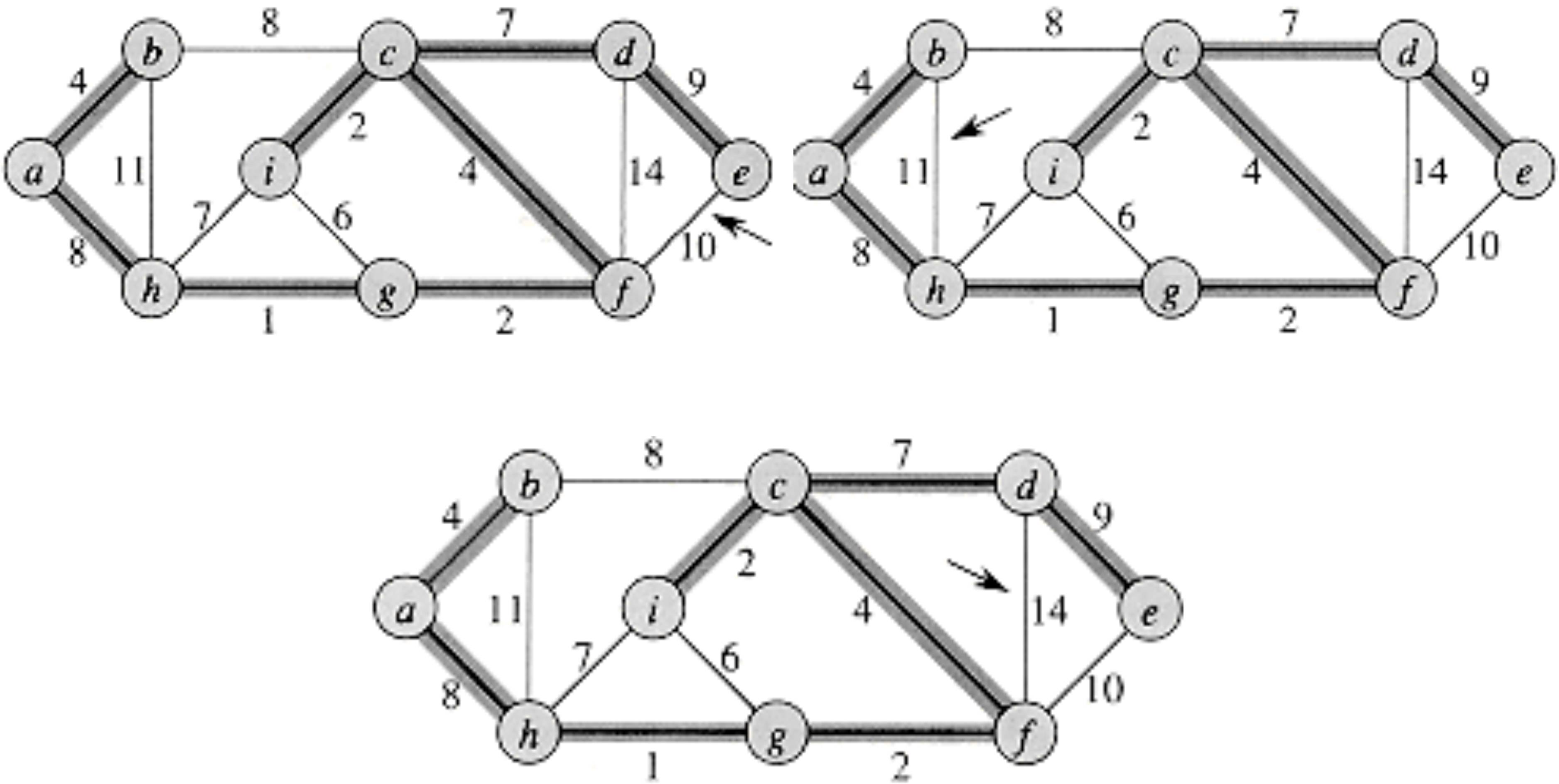
Algoritmo de Kruskal



Algoritmo de Kruskal



Algoritmo de Kruskal



Algoritmo de Kruskal

- Análise de Complexidade (considerando que a implementação da floresta de conjuntos disjuntos seja feita com as heurísticas de união por ordenação e compressão de caminho):
  - Tempo para ordenar as arestas é $O(e \log e)$
  - O segundo for executa $O(e)$ operações FIND-SET e UNION sobre a floresta de conjuntos disjuntos que juntamente com as n operações MAKE-SET, demoram ao todo o tempo $O((n+e)\alpha(n))$, onde α é a função de crescimento muito lento
  - Como G é supostamente conectado, $e \geq n-1$, e assim as operações de conjuntos disjuntos demoram o tempo $O(e\,\alpha(n))$
  - Como $\alpha(n)=O(\log n)=O(\log e)$, o tempo total é $O(e \log e)$
  - Considerando que $e \leq n^2$, $\log e = O(\log n)$, e portanto podemos redefinir o tempo de execução do algoritmo de Kruskal como $O(e \log n)$

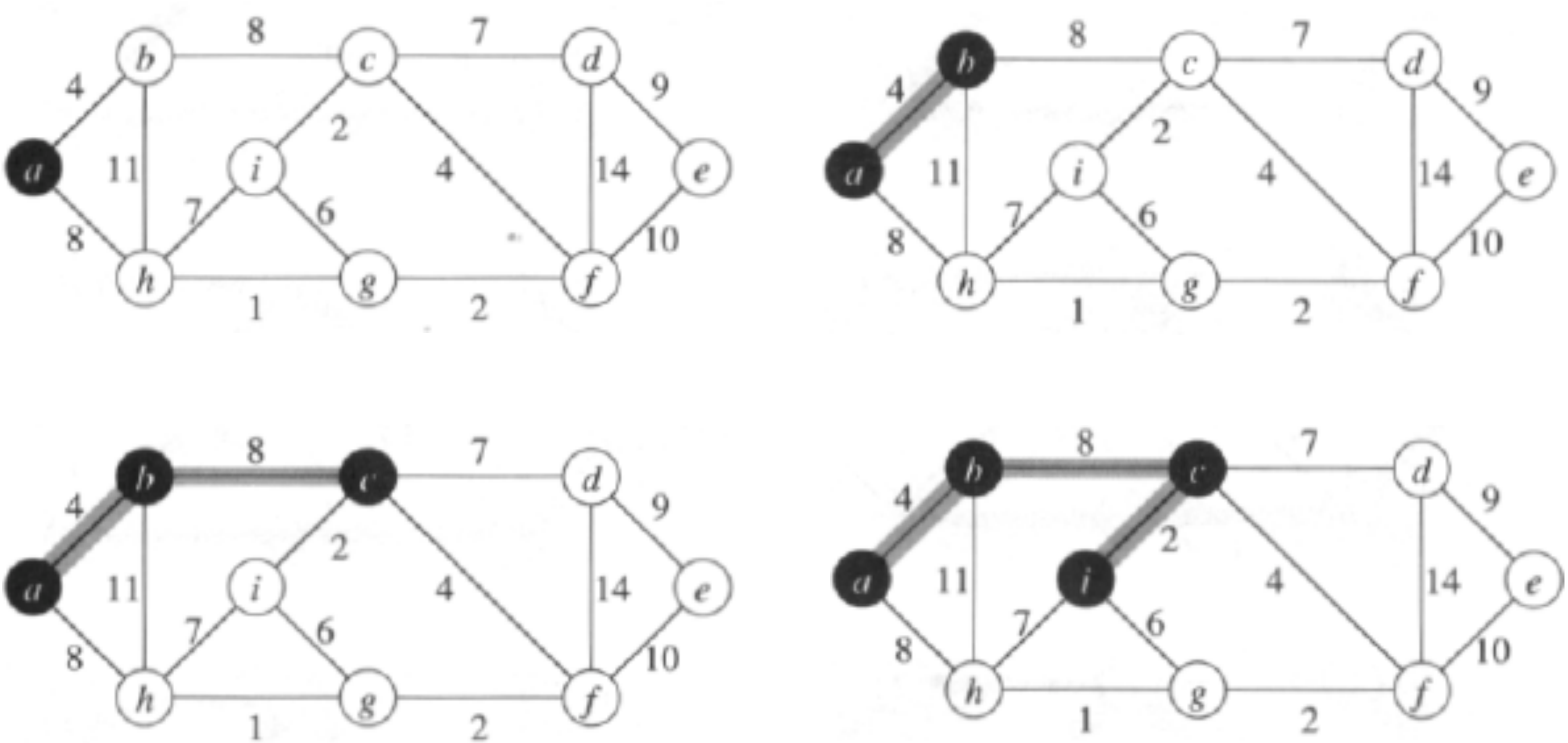
Algoritmo de Prim

- As arestas no conjunto `A` sempre formam uma árvore única
- A árvore começa a partir de um vértice de raiz arbitrária r e aumenta até a árvore alcançar todos os vértices em `V`
- Em cada etapa, uma aresta leve conectando um vértice de `A` a um vértice em `V−A` é adicionada à árvore
- Quando o algoritmo termina, as arestas em `A` formam uma árvore geradora mínima
- Durante a execução do algoritmo, todos os vértices que não estão na árvore residem em uma fila de prioridade mínima `Q` baseada em um campo chave
- Para cada vértice `v`, `chave[v]` é o peso mínimo de qualquer aresta que conecta `v` a um vértice na árvore
- `π[v]` é o pai de `v` na árvore

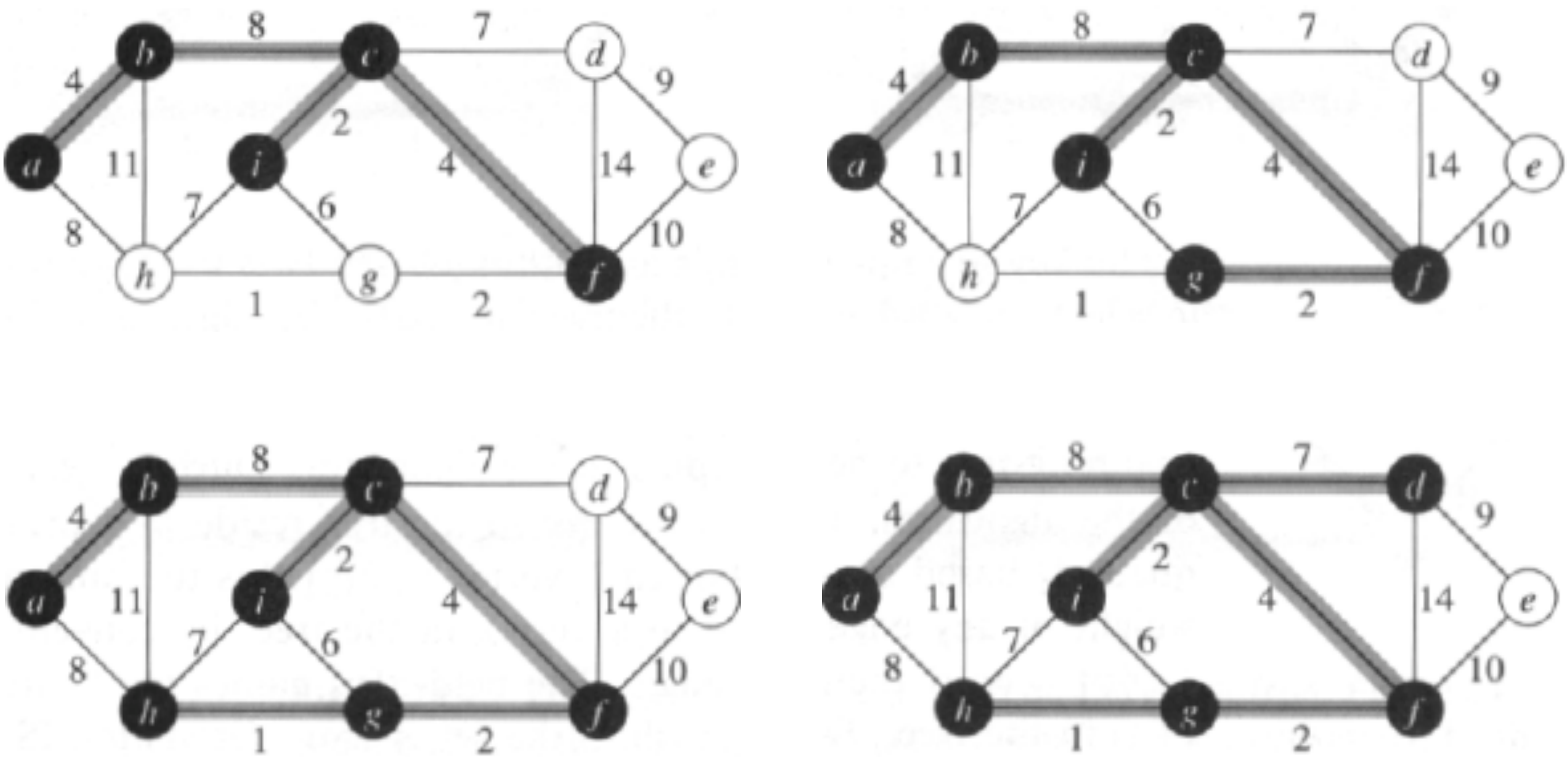
Algoritmo de Prim

```
MST-PRIM (G,w,r)
for cada u ∈ V[G] do
    chave[u] ← ∞
    π[u] ← NIL                        BUILD-MIN-HEAP
chave[r] ← 0
Q ← V[G]
while Q ≠ 0 do
    u ← EXTRACT-MIN(Q)      INSERE NA AGM
    for cada v ∈ Adj[u] do
        if v ∈ Q e w(u,v) < chave[v] then
            π[v] ← u
            chave[v] ← w(u,v)       DECREASE-KEY
```

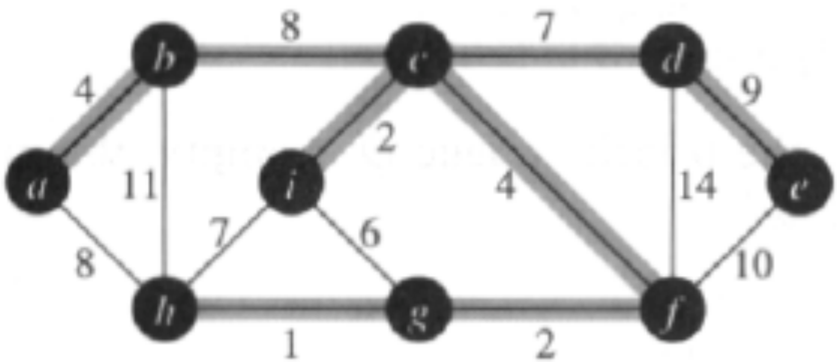
Algoritmo de Prim



Algoritmo de Prim



Algoritmo de Prim



Algoritmo de Prim

- Análise de Complexidade
  - BUILD-MIN-HEAP é executado em tempo `O(n)`
  - O corpo do **while** é executado `n` vezes, e como cada operação de EXTRACT-MIN demora `O(log n)`, o tempo total para todas as chamadas a EXTRACT-MIN é `O(n log n)`
  - O **loop for** é executado completamente `O(e)` vezes, pois a soma dos comprimentos de todas as listas de adjacências é `2*e`
  - Dentro do **loop for**, o teste de pertinência a `Q` pode ser implementado em tempo constante, mantendo-se um bit para cada vértice que informa se ele está ou não em `Q`, e atualizando-se o bit quando o vértice é removido de `Q`
  - A operação DECREASE-KEY sobre o heap mínimo pode ser implementada em tempo `O(log n)`
  - Portanto, o tempo total é `O(n log n + e log n) = O(e log n)`

Exercícios

16. Seja G um grafo conexo. O que podemos dizer sobre:
    - Uma aresta de G que aparece em todas as árvores geradoras?
    - Uma aresta de G que não aparece em nenhuma árvore geradora?

17. Em um dado grafo simples e conexo G, existe uma aresta es cujo peso é menor do que o peso de qualquer outra aresta de G. Podemos dizer toda AGM de G conterá es? Justifique.

16. Em um dado grafo simples e conexo G, existe uma aresta eb cujo peso é maior do que o peso de qualquer outra aresta de G. Podemos dizer que nenhuma AGM conterá eb? Justifique.