Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Programa de Pós-Graduação em Ciência da Computação

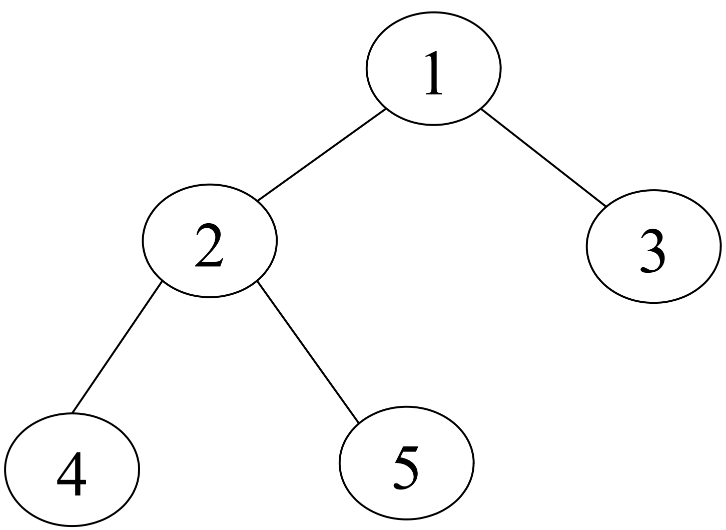
GRAFOS

Árvores Geradoras Mínimas

Nelson Cruz Sampaio Neto
nelsonneto@ufpa.br

Definição

- Árvore: é um grafo **G = (V, E)** que seja acíclico e conexo.
- O conceito "acíclico" refere-se a grafos sem ciclos simples de comprimento maior que 2.
- Toda árvore com **v** vértices possui exatas **v − 1** arestas.

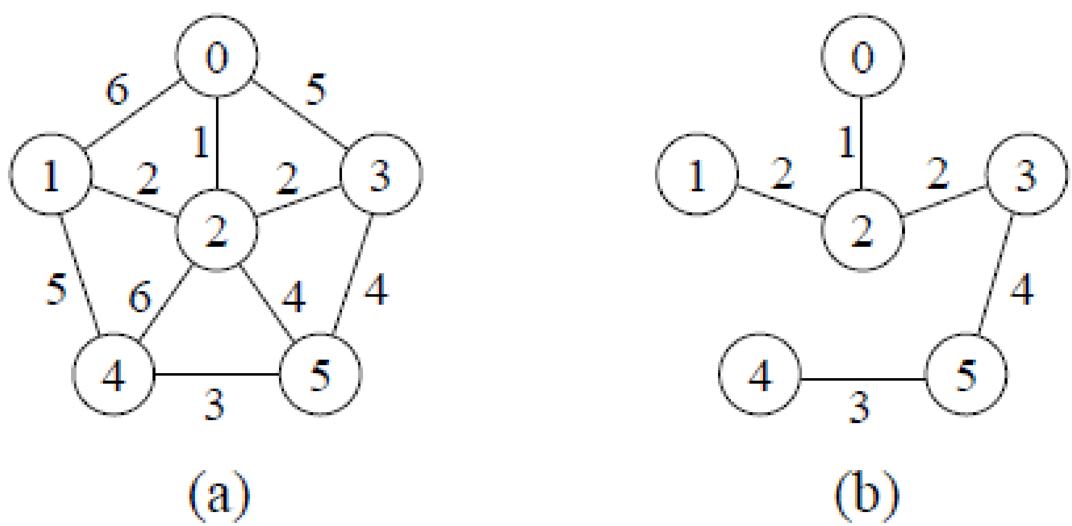


Note que a adição de mais uma aresta na árvore ao lado provoca o surgimento de um ciclo simples.

Motivação

- Árvores constituem uma classe extremamente importante na Teoria de Grafos, principalmente devido sua aplicação nas mais diversas áreas.

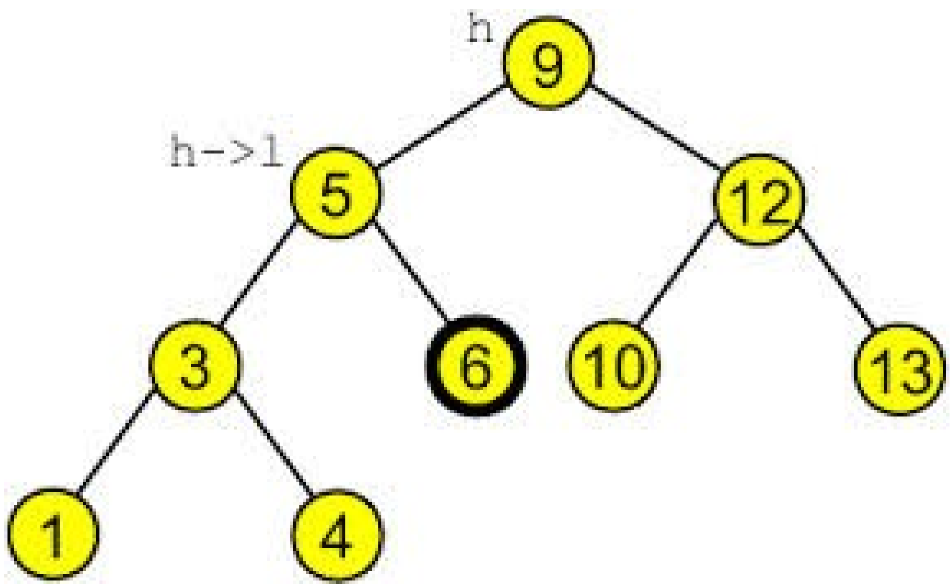
- <u>Exemplo</u>: Um projeto de redes de comunicação conectando diversas localidades. Como realizar essa conexão usando a menor quantidade de metros de cabo possível?



Motivação

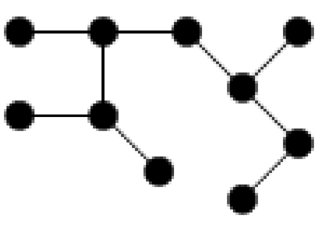
- <u>Árvore Binária de Pesquisa</u>: os vértices da subárvore esquerda possuem um valor inferior ao vértice raiz e os vértices da subárvore direita possuem um valor superior ao vértice raiz.

- O objetivo desta árvore é estruturar os dados de forma flexível, permitindo pesquisa binária.

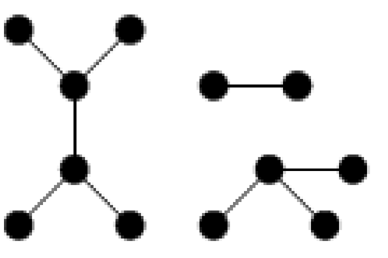


Conceitos

- Árvore Enraizada: quando algum vértice **v** $\in$ **V** é escolhido como especial. Esse vértice é chamado de raiz da árvore.
- Árvore Livre: termo usado quando a raiz da árvore não encontra-se definida. Exemplo: Figura (a).
- Floresta: é um grafo necessariamente acíclico, podendo ou não ser conexo. Exemplo: Figura (b).



(a)



(b)

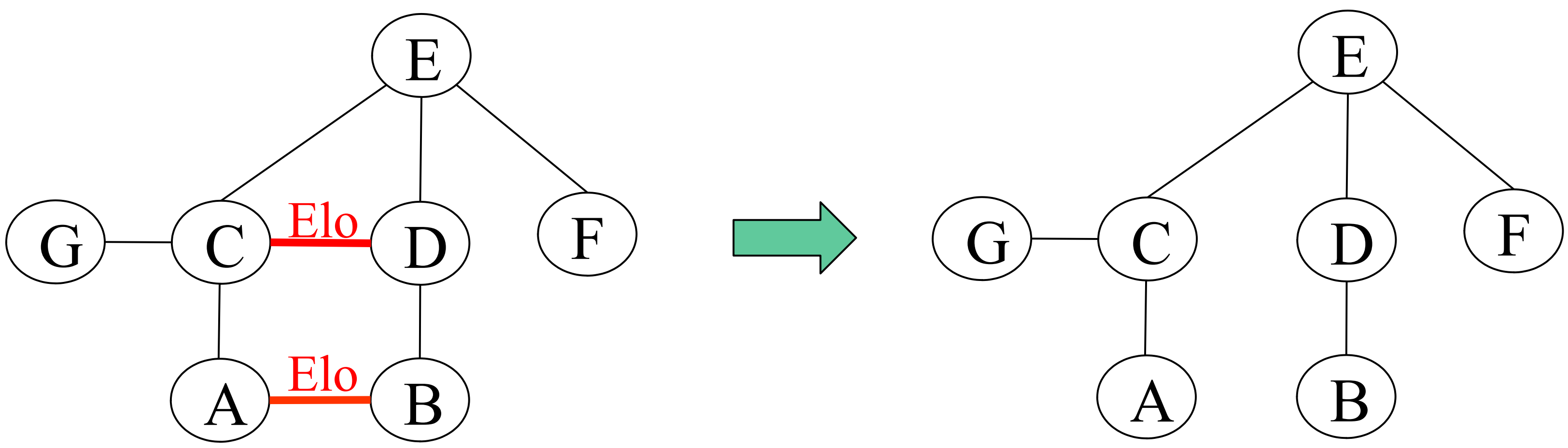
Conceitos

- Se **(v, w)** é uma aresta da árvore **T**, então considera-se que o vértice **v** é pai de **w**; e **w** é filho de **v**.
- O vértice raiz de uma árvore não possui pai, assim como um vértice folha não possui filhos.
- O nível de um vértice é dado pelo comprimento do caminho da raiz até ele. O nível da raiz é zero.
- A altura de uma árvore é igual ao valor máximo de nível entre os vértices que a formam.

Conceitos

- Árvore geradora de um grafo conexo **G = (V, A)** é um subgrafo que contém todos os vértices de **G** e forma uma árvore.

- Floresta geradora de um grafo **G = (V, A)** é um subgrafo que contém todos os vértices de **G** e forma uma floresta.



Conceitos

- Elo: são as arestas que devem ser retiradas do grafo conexo **G** para a obtenção de uma árvore geradora.
- O número total de elos distintos de um grafo conexo **G** é igual a **e – v + 1**. Também chamado de posto.
- O grafo do slide anterior tem posto igual a 2, ou seja, possui dois elos (arestas marcadas em vermelho, por exemplo).
- A adição de um elo à uma árvore provoca o surgimento de um único ciclo simples, chamado de ciclo fundamental.

Conceitos

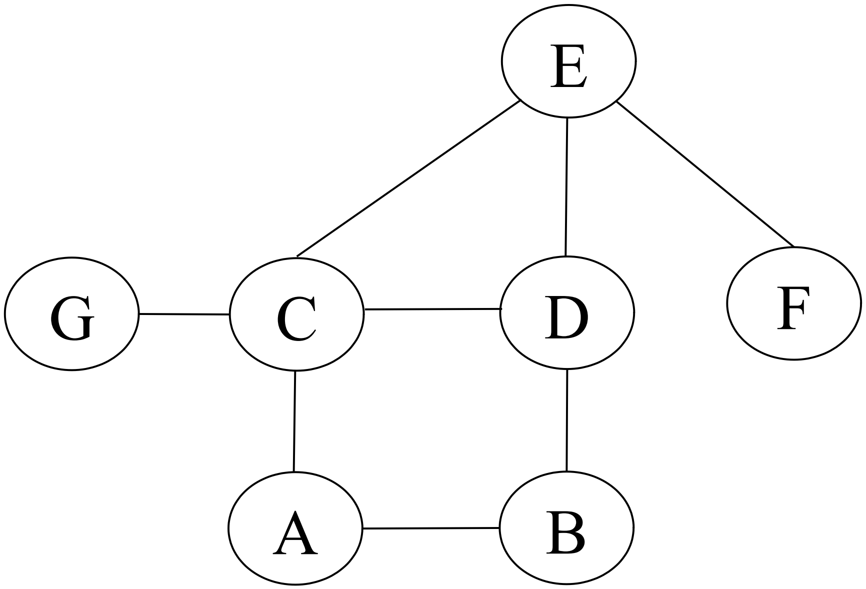
- Excentricidade de um vértice $\mathbf{v} \in \mathbf{V}$ é o valor da distância[1] máxima entre **v** e **w**, para todo $\mathbf{w} \in \mathbf{V}$.

- Para obter a excentricidade em grafos não valorados basta realizar busca em largura a partir do vértice em questão.

- O centro de um grafo **G** é o subconjunto dos vértices de excentricidade mínima.

1. Distância é o comprimento do menor caminho entre dois vértices.

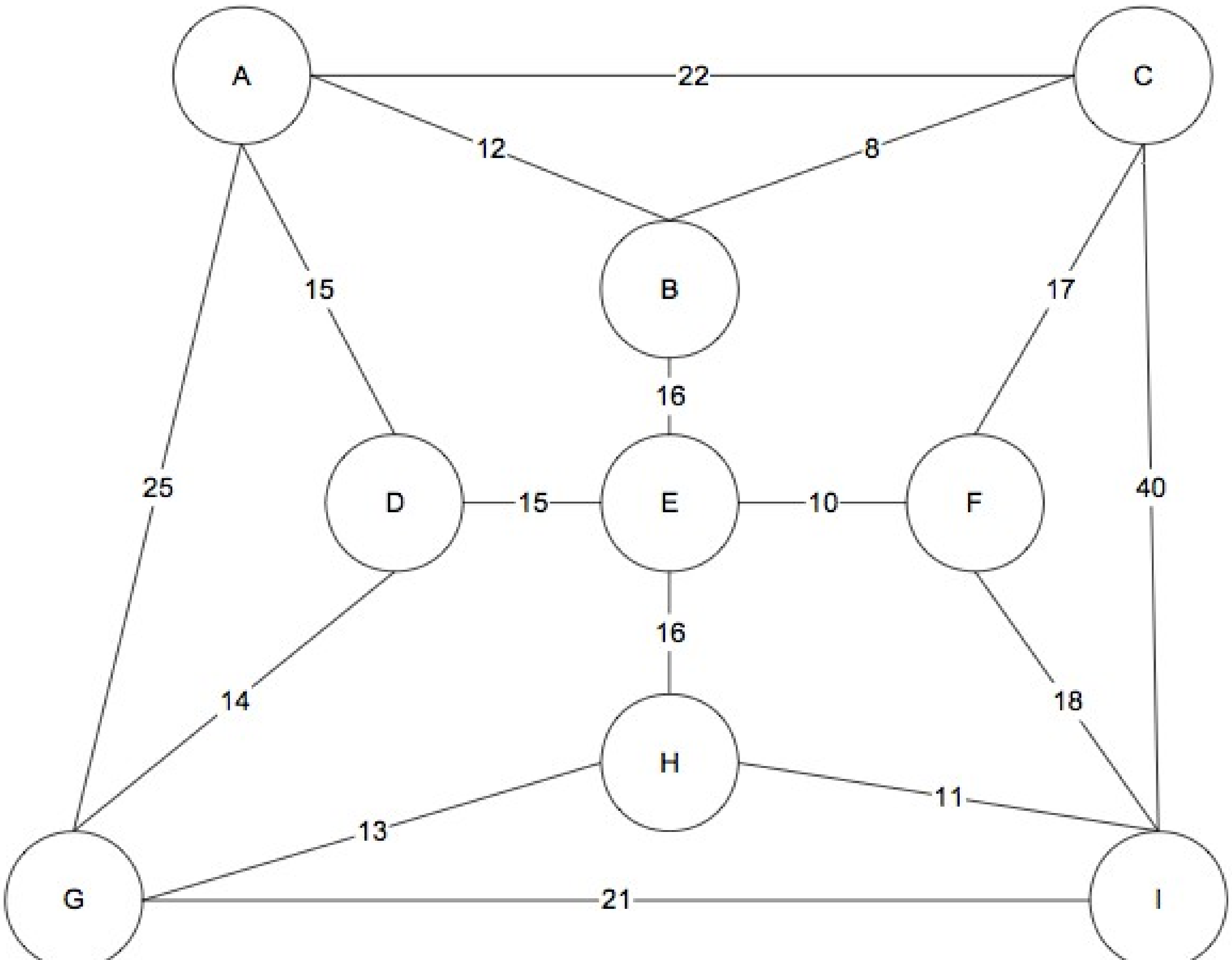
Exemplo



Vértice	Excentricidade
A	3
B	3
C	2
D	2
E	2
F	3
G	3

- O centro do grafo é o subconjunto dos vértices {C, D, E}
- O raio = 2 e o diâmetro = 3

Exemplo



Exemplo

- Matriz de caminho mínimo pelo algoritmo de Floyd-Warshall:

	A	B	C	D	E	F	G	H	I
A		12	20	15	28	37	25	38	**46**
B	12		8	27	16	25	37	32	**43**
C	20	8		35	24	17	**45**	40	35
D	15	27	**35**		15	25	14	27	**35**
E	28	16	24	15		10	**29**	16	27
F	37	25	17	25	10		**39**	26	18
G	25	37	**45**	14	29	39		13	21
H	38	32	**40**	27	16	26	13		11
I	**46**	43	35	35	27	18	21	11	

- Centro do grafo é o vértice **E**
- Raio = 29 km
- Diâmetro = 46 km

Exemplo

- O centro é (ou são) o vértice que, sob o ponto de vista de distância, oferece as melhores condições para a instalação de um equipamento coletivo. Por exemplo, se o quartel do corpo de bombeiros estivesse em “E”, todas as suas intervenções se situariam, no máximo, a 29 km.

Conceitos

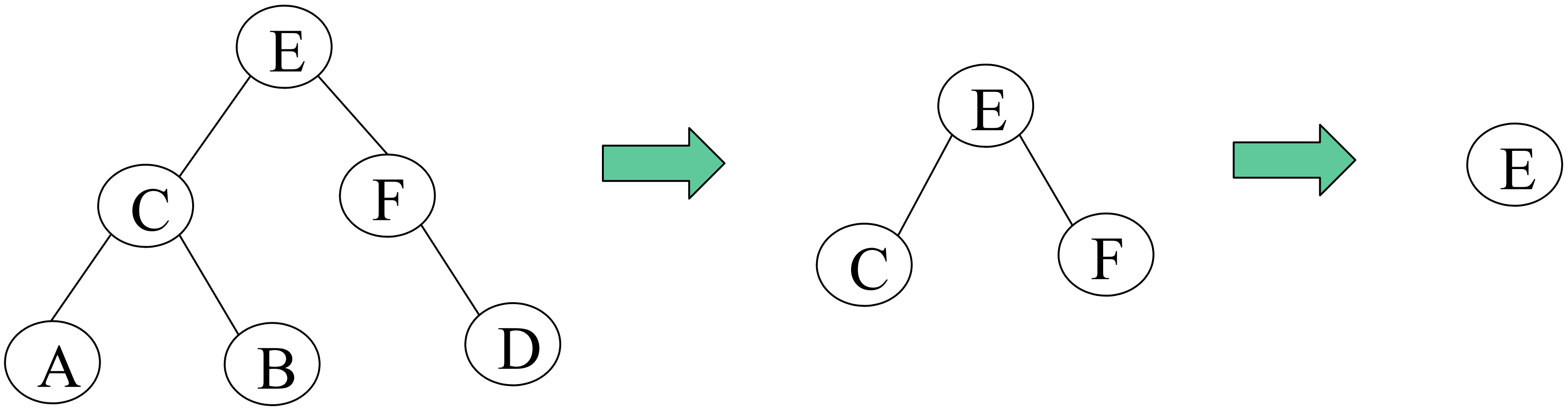
- O centro de um grafo **G** possui de **1** a **|V|** vértices. Já o centro de uma árvore possui não mais do que **2** vértices.

- Seja **T** uma árvore com **|V| > 2**. Se retirarmos todos os vértices de grau igual a 1, a árvore resultante **T'** possui o mesmo centro de **T**.

- Algoritmo para determinar o centro de uma árvore:

```
Dados: árvore T(V, E)
Enquanto |V| > 2 faça
    excluir os vértices de grau = 1
```

Exemplo



Vértice	Excentricidade
A	4
B	4
C	3
D	4
E	**2**
F	3

Árvore Geradora Mínima (AGM)

- Projeto de redes de comunicações conectando **n** localidades.
- Arranjo de **n − 1** conexões, conectando duas localidades cada.
- Objetivo: dentre as possibilidades de conexões, achar a que usa a menor quantidade de cabos.
- Modelagem:
  - Grafo conexo e não-direcionado **G = (V, E)**
  - **V**: conjunto de cidades
  - **E**: conjunto de possíveis conexões
  - **p (u, v)**: peso da aresta **(u, v)** $\in$ **E**

Árvore Geradora Mínima (AGM)

- Solução: encontrar uma árvore **T = (V, E_T)**, onde **E_T** $\subseteq$ **E**, que conecta todos os vértices de **G** e minimiza o peso total:

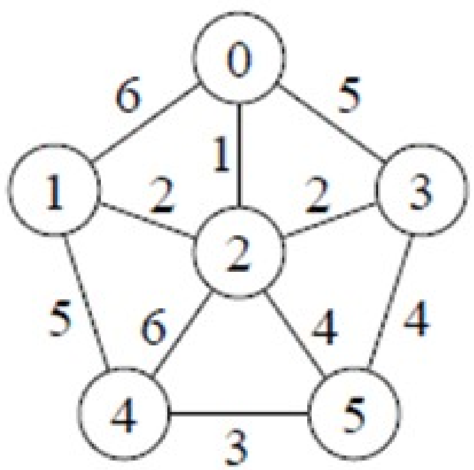
$$p(T) = \sum_{(u,v) \in E_T} p(u, v)$$

- A ideia é escolher **E_T** de tal modo que **(V, E_T)** seja uma árvore geradora. O critério de otimização corresponde a minimizar a soma dos pesos das arestas de **E_T** .

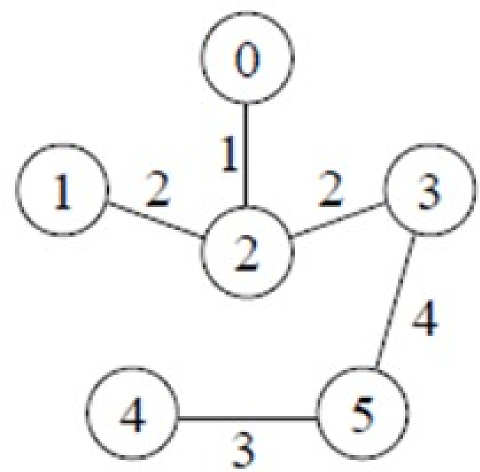
- O problema de obter a árvore **T** é conhecido como Árvore Geradora Mínima ou AGM.

Exemplo

- Árvore geradora mínima **T**, obtida do grafo **G**, com peso total igual a 12. A árvore **T** não é única, pode-se substituir a aresta (3, 5) pela aresta (2, 5) obtendo outra árvore geradora de custo 12.



Grafo Conexo



Árvore geradora mínima T

AGM – Algoritmo Genérico

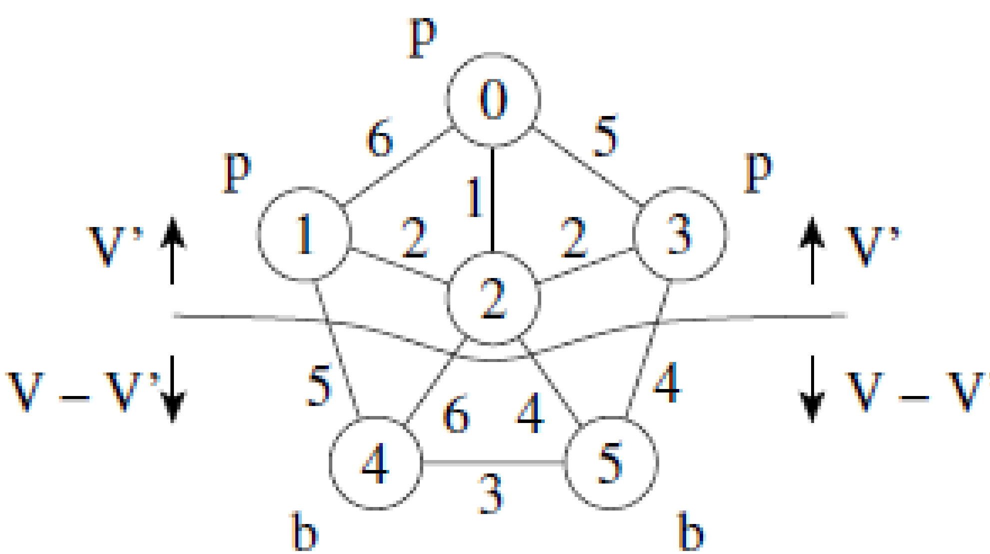
- Uma estratégia gulosa permite obter a AGM adicionando uma aresta de cada vez.
- Invariante: Antes de cada iteração, **T** é um subconjunto de uma árvore geradora mínima.
- A cada passo adicionamos a **T** uma aresta **(u, v)** que não viola o invariante, essa recebe o nome de <u>aresta segura</u>.

1. T = 0
2. enquanto (T não constitui uma árvore geradora mínima)
3.   (u, v) = seleciona (E)
4.   se (u, v) é segura para T
5.   então T = T + { (u, v) }
6. retorne T

AGM – Definição de Corte

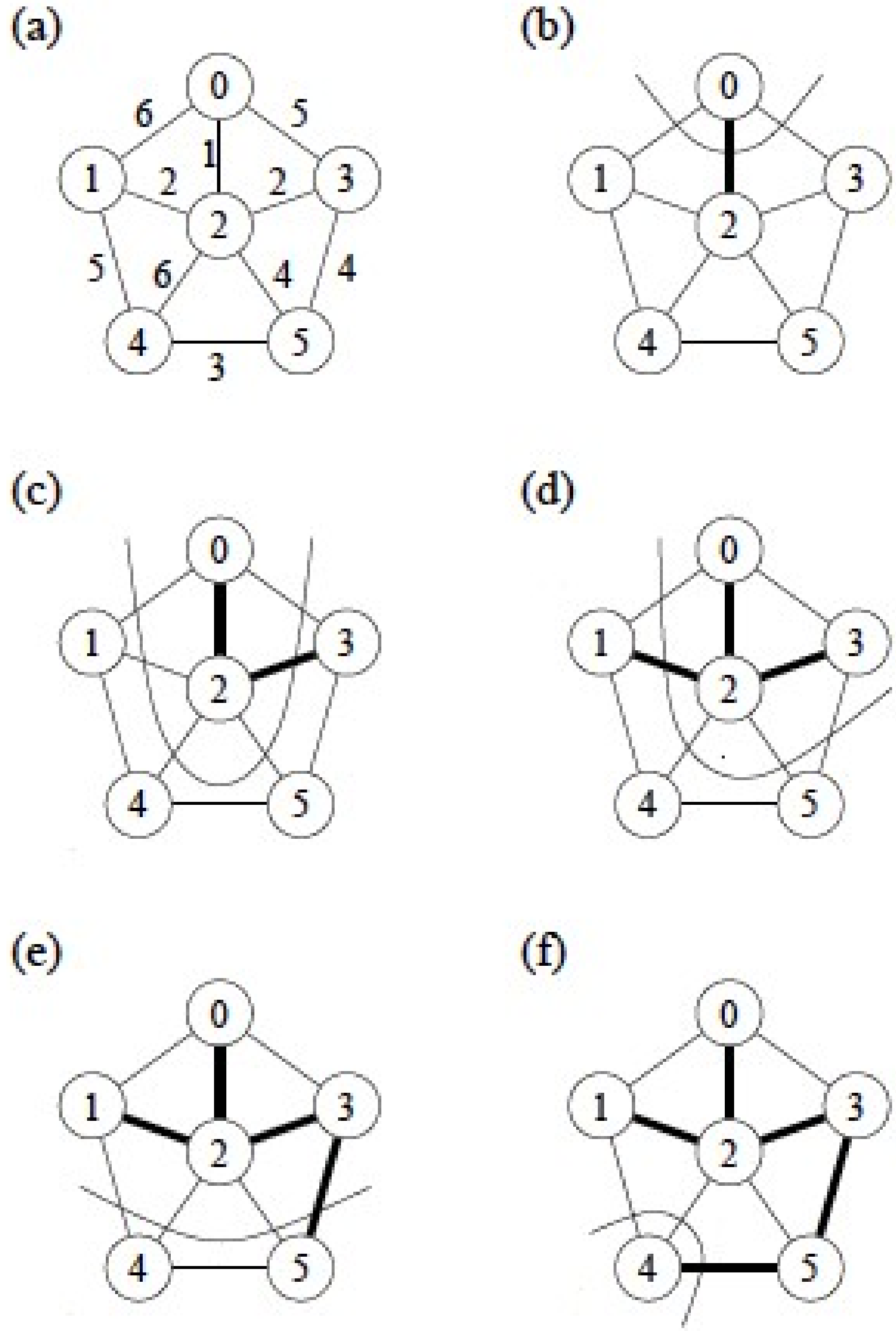
- Dificuldade: Como encontrar uma aresta segura?

- Teorema do Corte Mínimo. A prova desse teorema pode ser obtida em Cormen et al. (Capítulo 23).



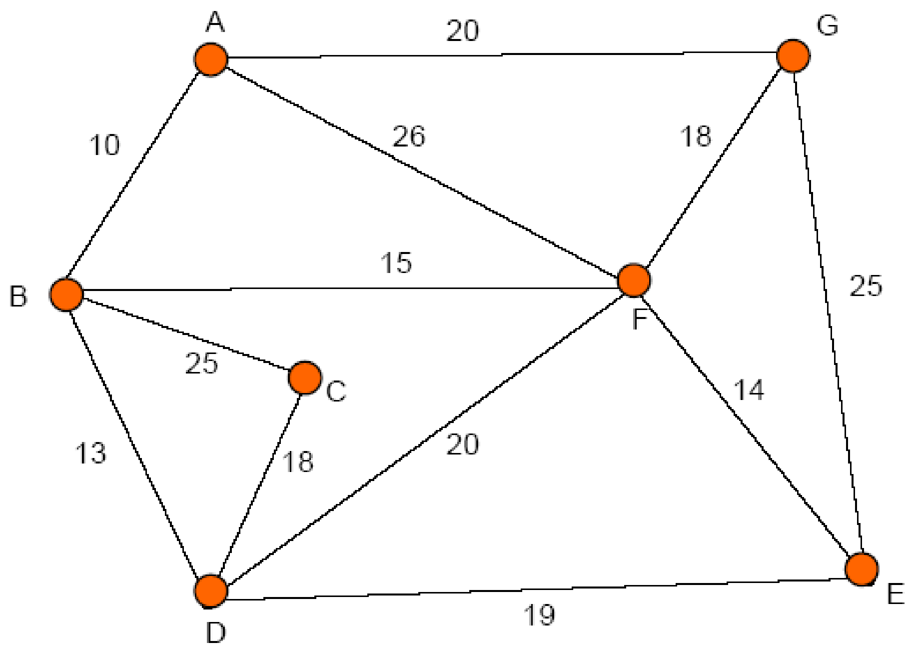
AGM – Definição de Corte

- Um corte **(V’, V − V’)** de um grafo não direcionado **G = (V, A)** é uma partição de **V**.
- Uma aresta **(u, v)** $\in$ **E** cruza o corte **(V’, V − V’)** se um de seus vértices pertence a **V’** e o outro vértice pertence a **V − V’**.
- Um corte respeita um conjunto **T** se não existirem arestas em **T** que o cruzem.
- Uma aresta cruzando o corte **(V’, V − V’)** que tenha custo mínimo sobre todas as arestas que o cruzam é uma <u>aresta segura</u>*.*


(a)
0
1
2
3
4
5
6
5
1
2
2
5
6
4
4
3
(b)
(c)
(d)
(e)
(f)

Exercício

A prefeitura de uma cidade quer pavimentar algumas ruas para que se possa ir de qualquer bairro para qualquer bairro só por rua pavimentada. Porém, o prefeito quer minimizar ao máximo o total de quilômetros a pavimentar. O mapa da cidade (bairros, ruas e quilometragem das ruas) é dado abaixo. Resolva o problema informando o conjunto de ruas e o total de quilômetros a pavimentar.



AGM – Algoritmo de Prim

- O algoritmo de Prim é derivado do algoritmo genérico e faz uso do Teorema do Corte Mínimo.

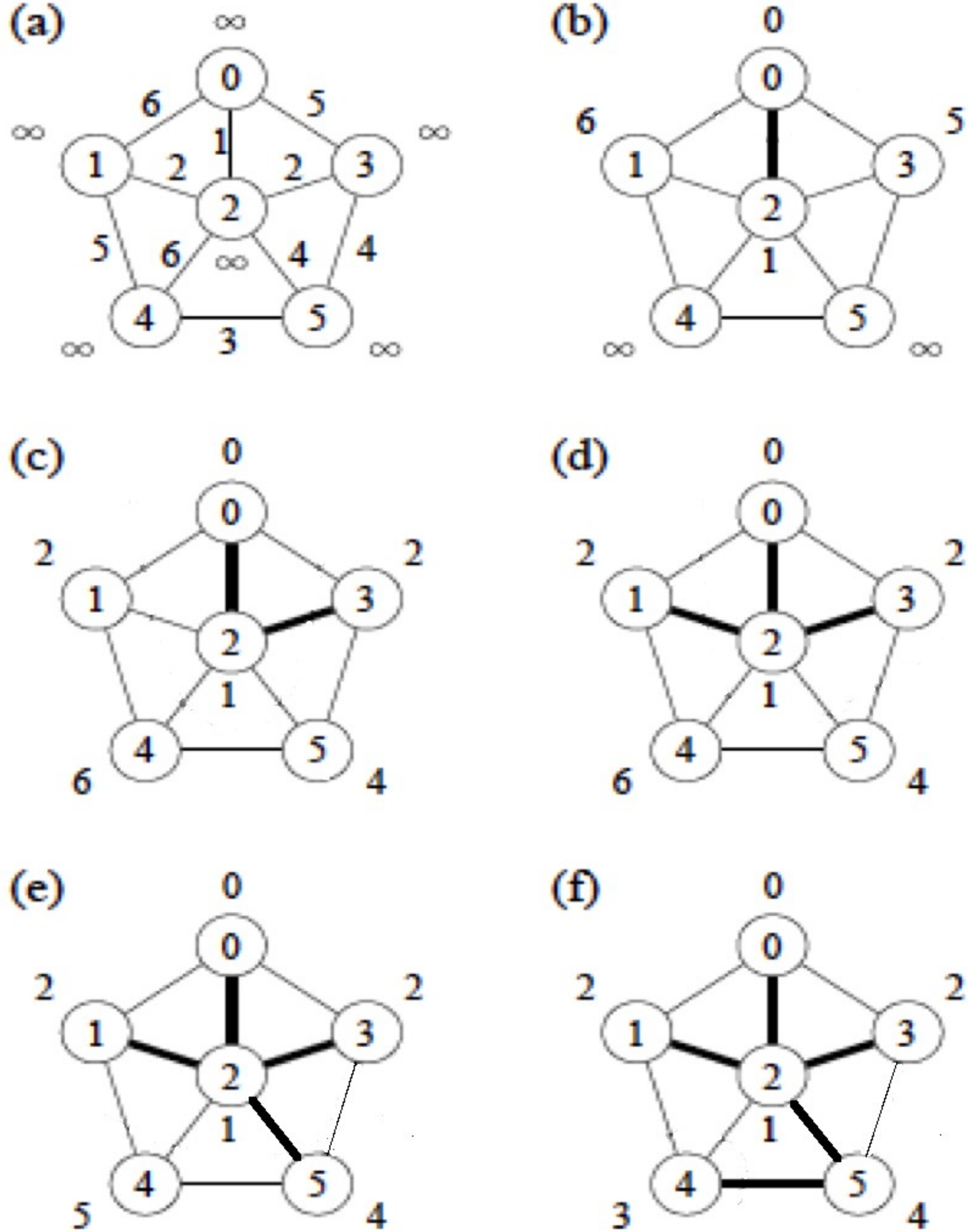
- No algoritmo de Prim, a árvore **T** tem raíz em **r** (escolhida arbitrariamente). Inicialmente, a árvore **T** é vazia.

- Toda aresta segura (e de peso mínimo) adicionada sempre conecta a árvore **T** a um vértice que não esteja na árvore.

AGM – Algoritmo de Prim

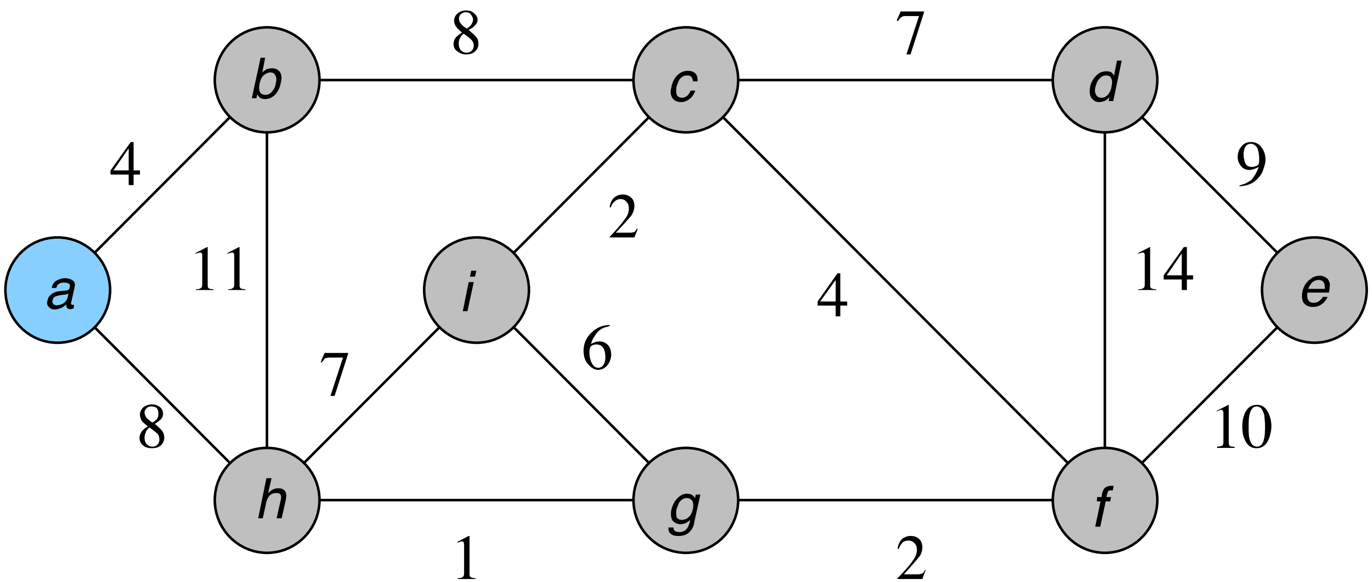
AGM-PRIM(G, w, r)

para cada $u \in V[G]$
    faça $key[u] \leftarrow \infty$
        $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V[G]$
enquanto $Q \neq \emptyset$ **faça**
    $u \leftarrow$ EXTRACT-MIN(Q)
    para cada $v \in \mathrm{Adj}[u]$
        se $v \in Q$ **e** $w(u, v) < key[v]$
            então $\pi[v] \leftarrow u$
                $key[v] \leftarrow w(u, v)$
retorne π

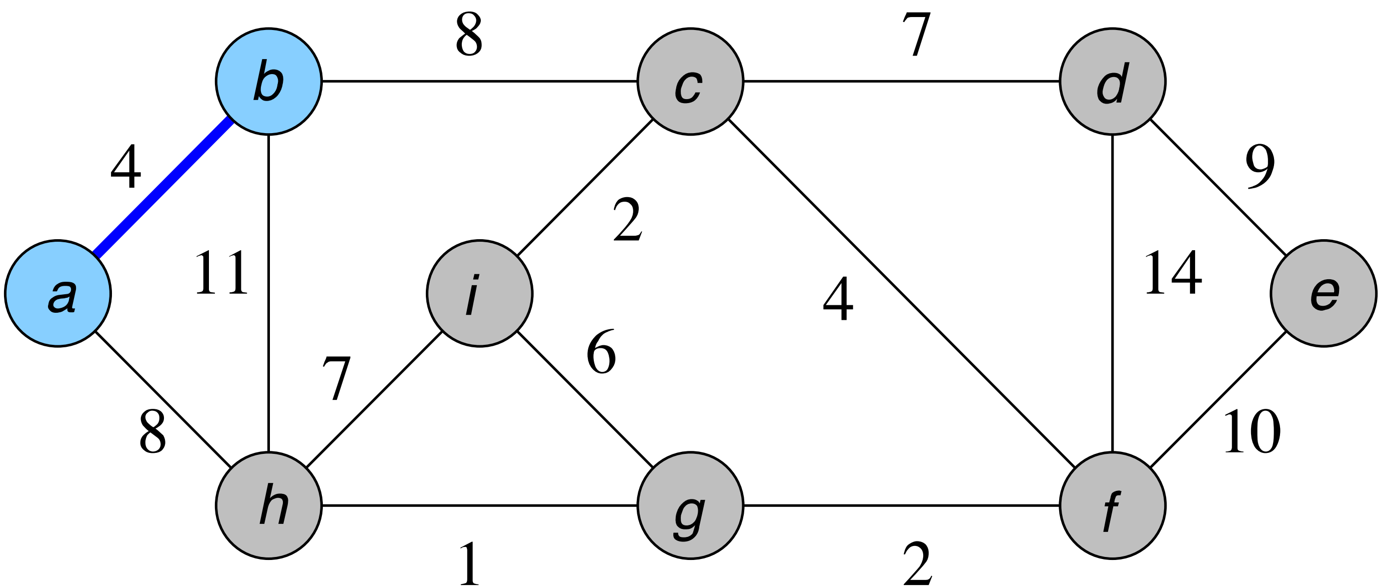


(a)
(b)
(c)
(d)
(e)
(f)

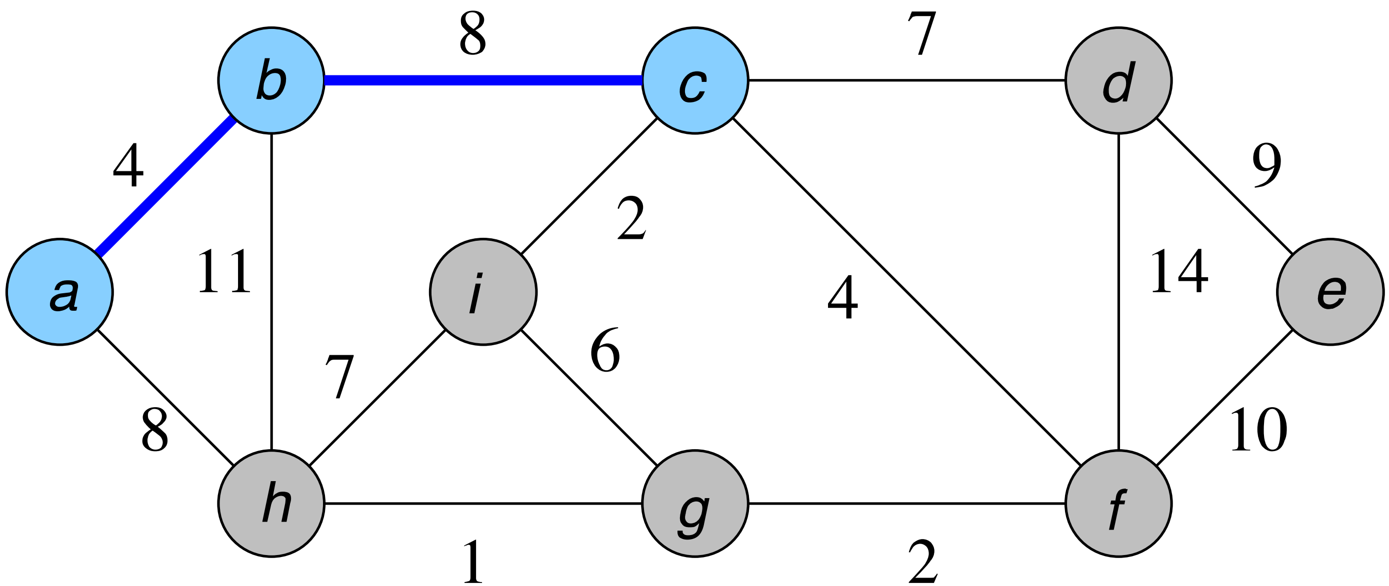
AGM – Algoritmo de Prim



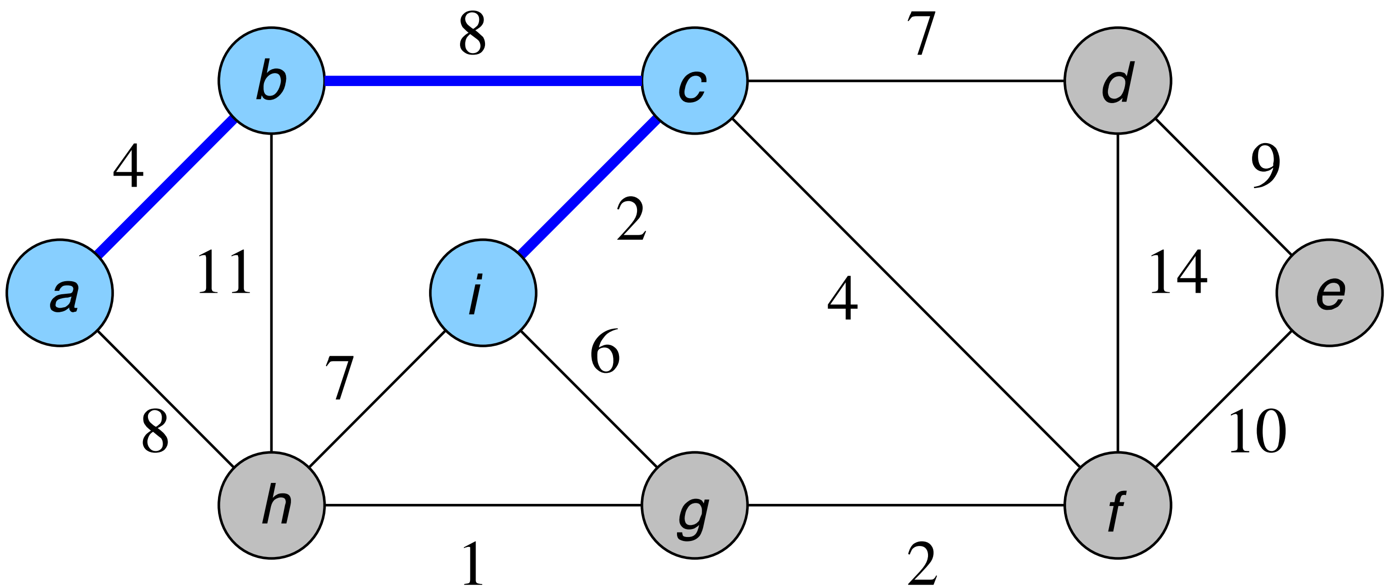
AGM – Algoritmo de Prim



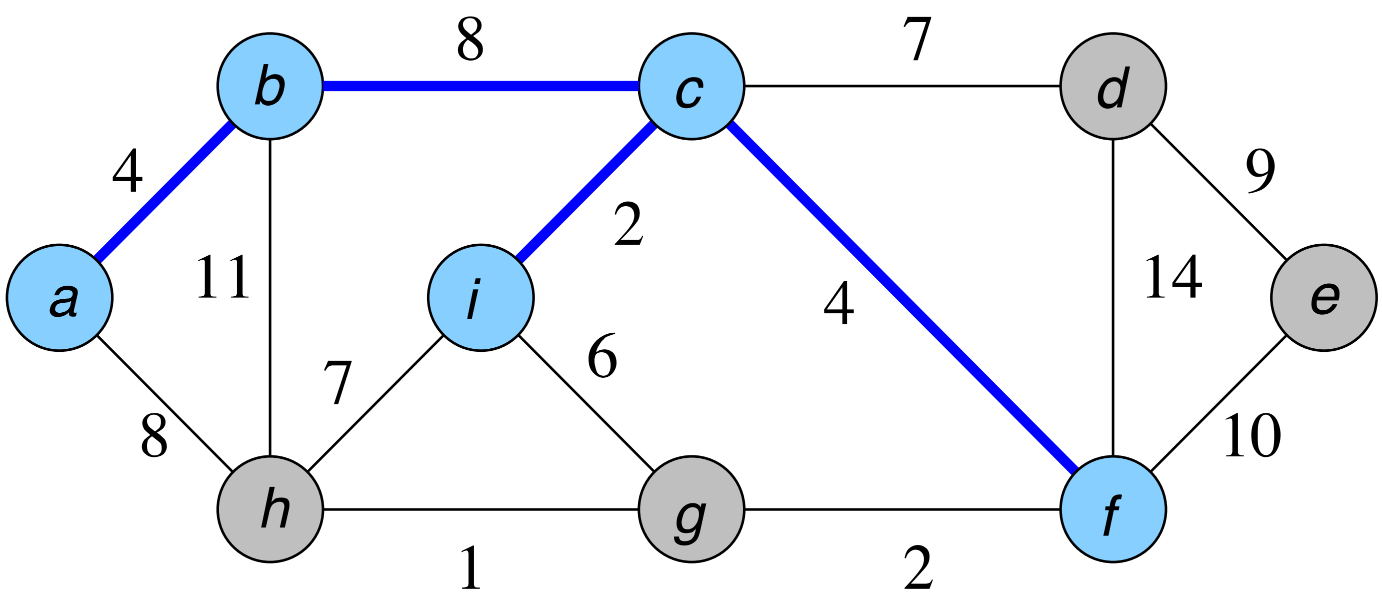
AGM – Algoritmo de Prim



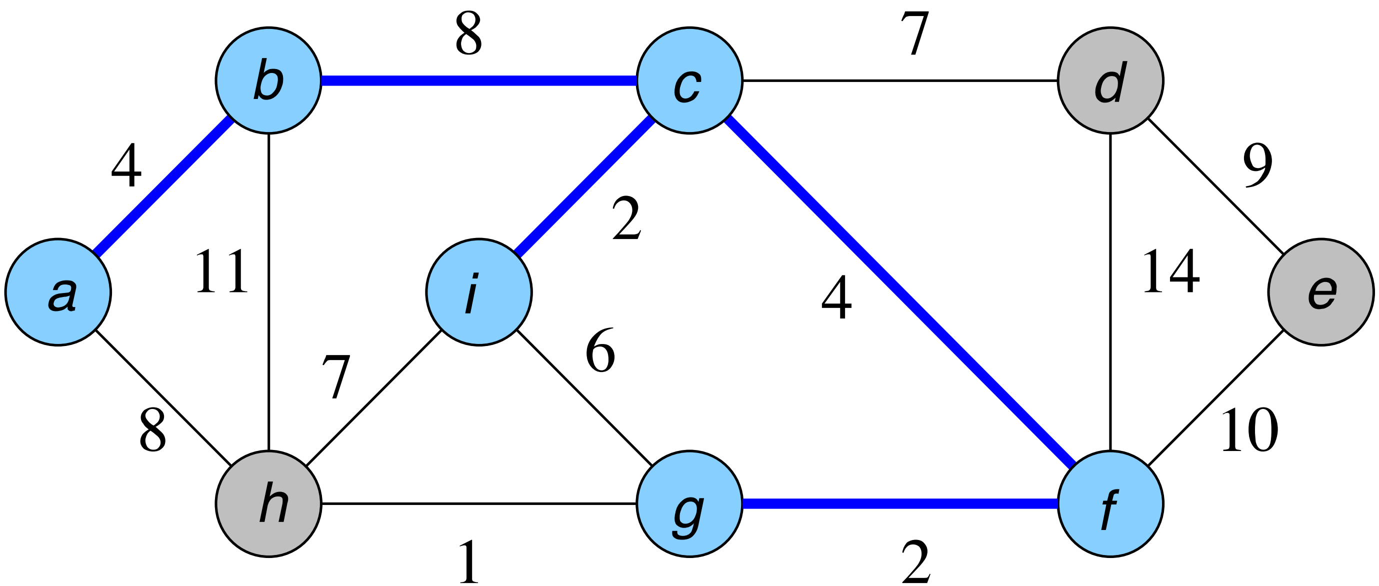
AGM – Algoritmo de Prim



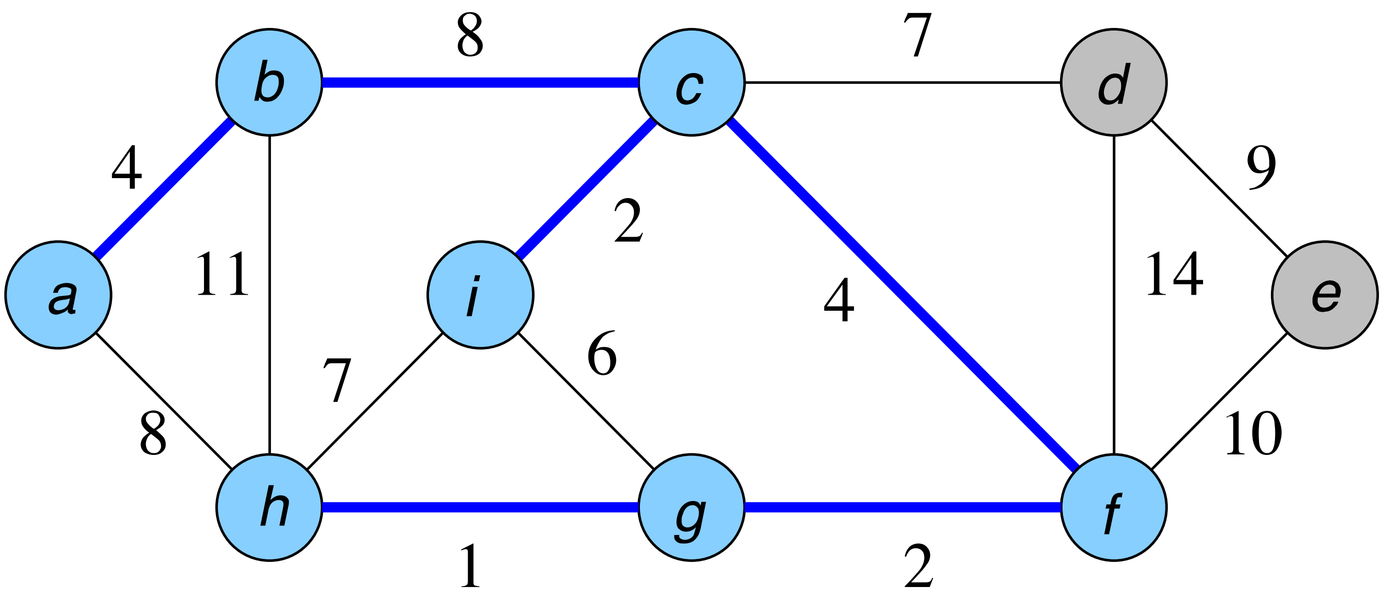
AGM – Algoritmo de Prim



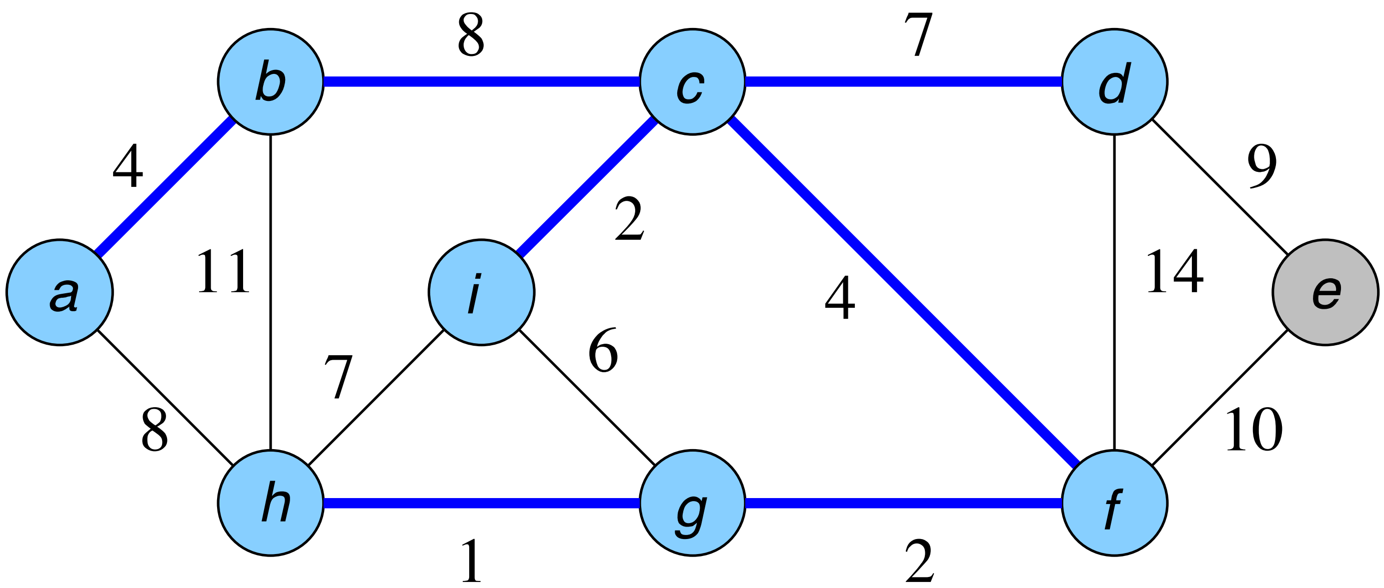
AGM – Algoritmo de Prim



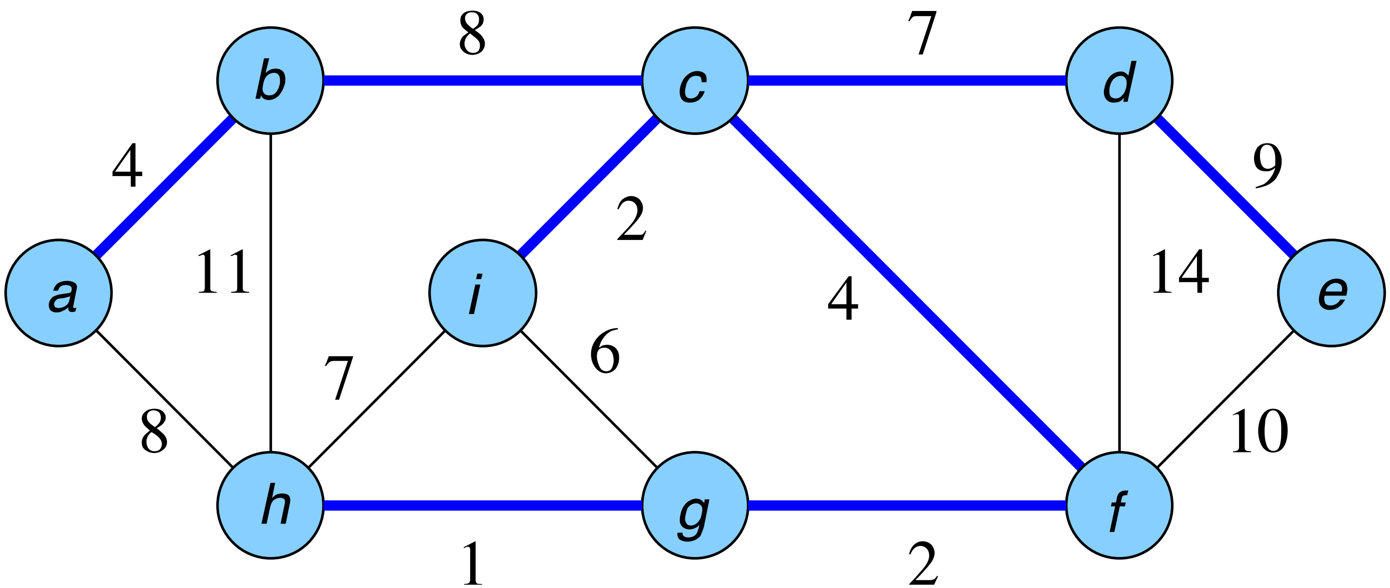
AGM – Algoritmo de Prim



AGM – Algoritmo de Prim



AGM – Algoritmo de Prim



AGM – Algoritmo de Prim

- O algoritmo de Prim usando *heap* binário e lista de adjacência tem complexidade no tempo igual a O(E log V).
  - As linhas 1 – 5 são executadas em tempo: O(V);
  - A linha 7 é executada |V| vezes e cada operação EXTRAIR-MIN consome O(log V). Resultando em um tempo: O(V log V);
  - As linhas 8 – 11 são executadas O(E) e cada operação de decremento consome O(log V). Resultando em um tempo: O(E log V); e
  - O teste de pertinência à fila **Q** pode ser feito em tempo constante usando um vetor booleano.

AGM – Algoritmo de Prim

- Pode-se fazer melhor usando a estrutura de dados *heap* de Fibonacci que guarda **|V|** elementos e suporta as seguintes operações:
  - EXTRACT-MIN: $O(\log V)$.
  - DECREASE-KEY: tempo amortizado $O(1)$.

- Usando um *heap* de Fibonacci para implementar a fila de prioridade **Q** melhoramos o tempo para $O(E + V \log V)$.

AGM – Algoritmo de Kruskal

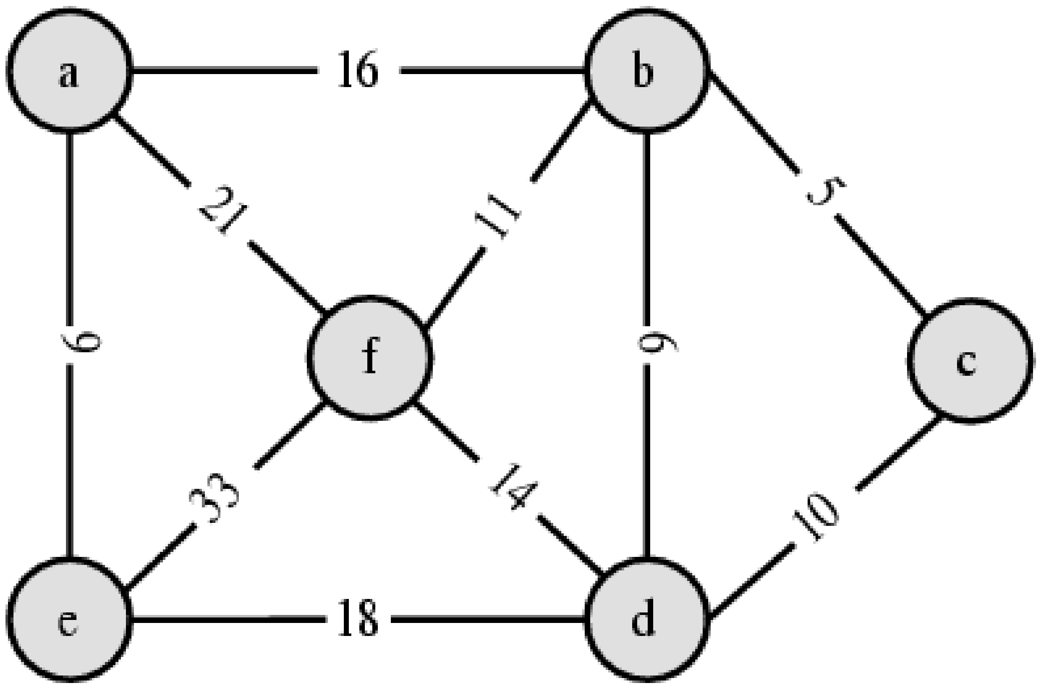
- Outro algoritmo muito conhecido para resolver o problema de encontrar uma AGM é o de Kruskal.

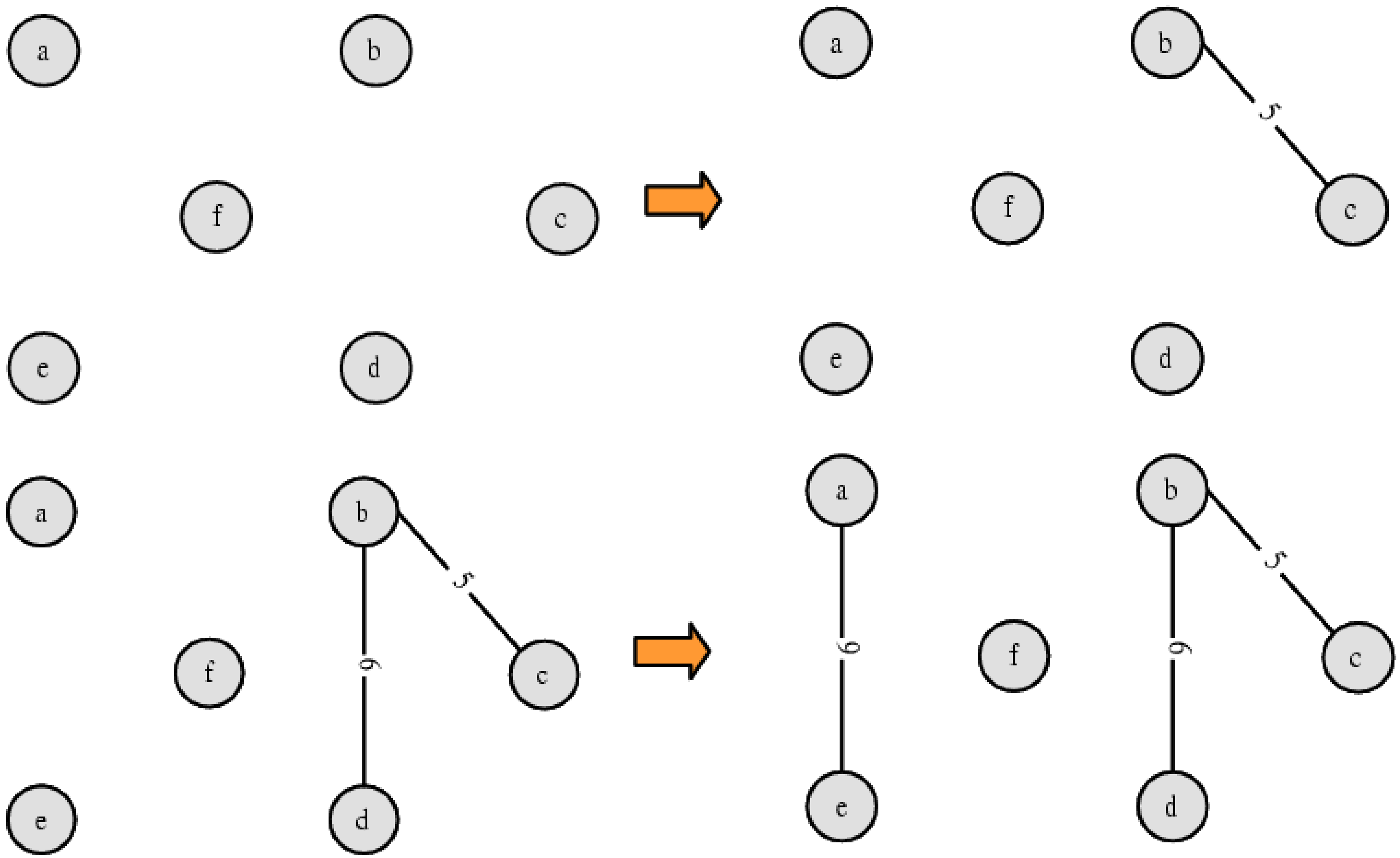
- Esse algoritmo, assim como Prim, é uma especialização do algoritmo genérico e faz uso do Teorema do Corte Mínimo.

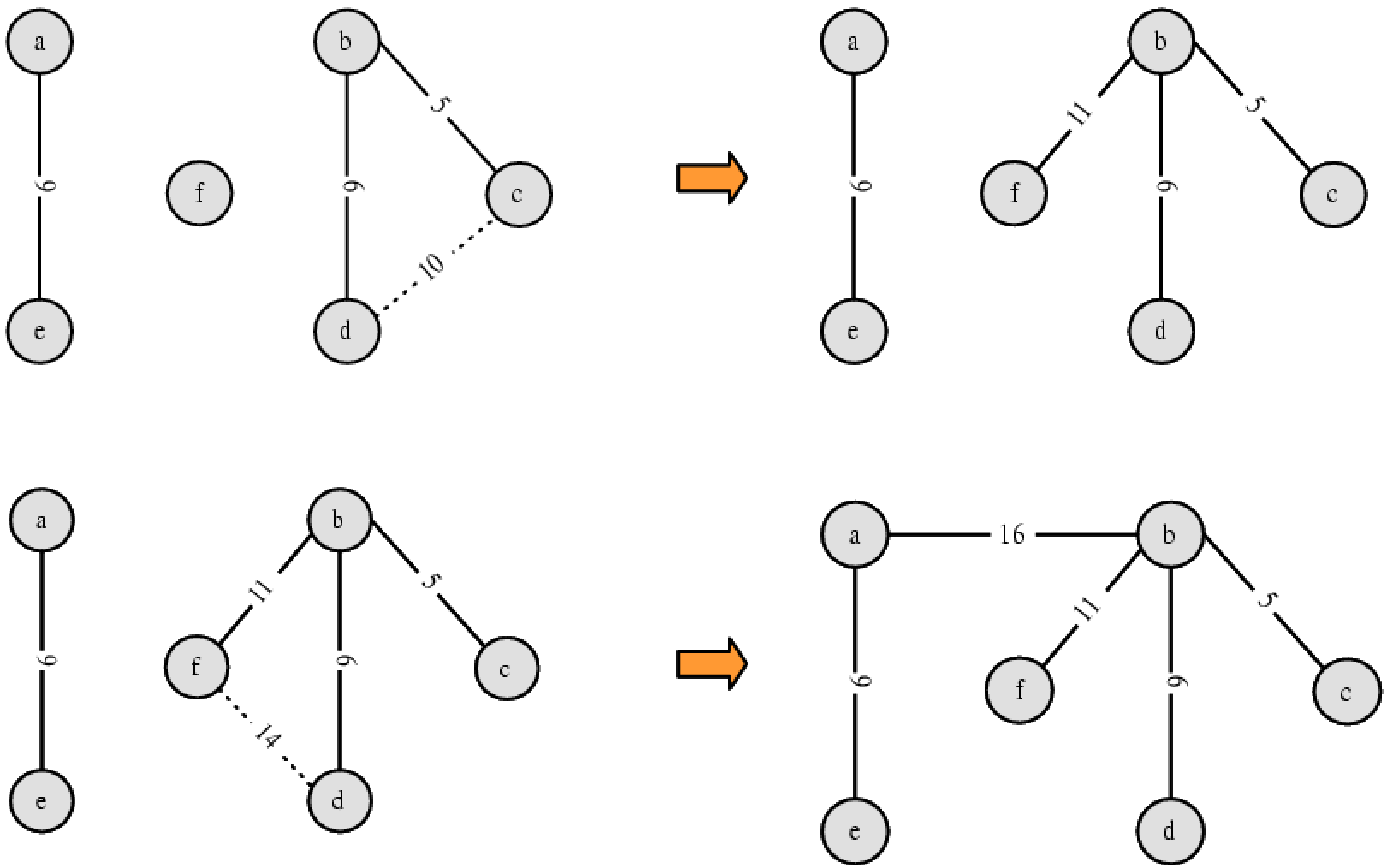
- O conjunto **A** é uma floresta e a aresta segura adicionada a **A** é uma aresta de menor peso que conecta duas árvores distintas.

- O processo que une duas árvores é repetido até que exista apenas uma árvore na floresta.

AGM – Algoritmo de Kruskal




a
b
f
c
e
d
5


a
b
c
d
e
f
5
10
11
14
16

AGM – Algoritmo de Kruskal

AGM-KRUSKAL(G, w)

$A \leftarrow \emptyset$
para cada $v \in V[G]$ **faça**
    MAKE-SET(v)
Ordene as arestas em ordem não-decrescente de peso
para cada $(u, v) \in E$ nessa ordem **faça**
    se FIND-SET$(u) \neq$ FIND-SET(v)
        então $A \leftarrow A \cup \{(u, v)\}$
            UNION(u, v)
devolva A

Complexidade:

- Ordenação: $O(E \lg E)$
- $|V|$ chamadas a MAKE-SET
- $O(E)$ chamadas a UNION e FIND-SET

AGM – Algoritmo de Kruskal

Usando a representação *disjoint-set forests* com union by rank e path compression, o tempo gasto com as operações é $O((V+E)\alpha(V)) = O(E\alpha(V))$.

Como $\alpha(V) = O(\lg V) = O(\lg E)$ o passo que consome mais tempo no algoritmo de Kruskal é a ordenação.

Logo, a complexidade do algoritmo é $O(E \lg E) = O(E \lg V)$.

Mais detalhes sobre as estruturas de dados para conjuntos disjuntos podem ser encontrados em Cormen et al. (Capítulo 21).