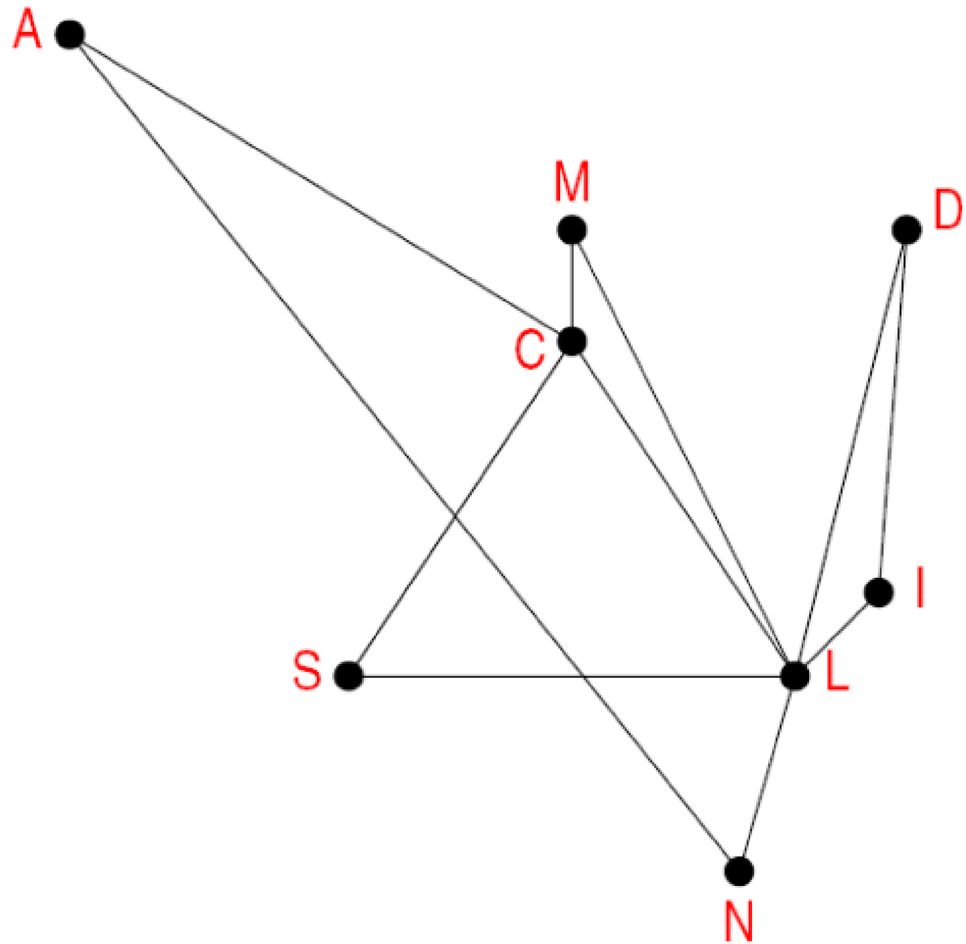
Árvore geradora mínima

Árvore Geradora Mínima (AGM)

- Uma companhia aérea recebeu permissão para voar nas rotas da figura à direita.

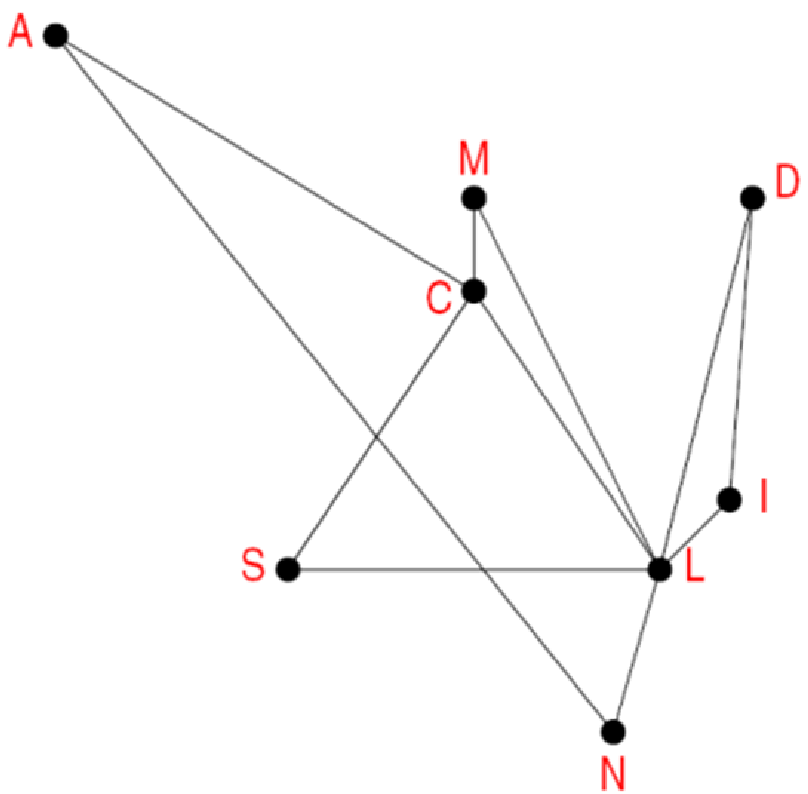
- A empresa **não** irá operar em **todas as vias** (questões economias).

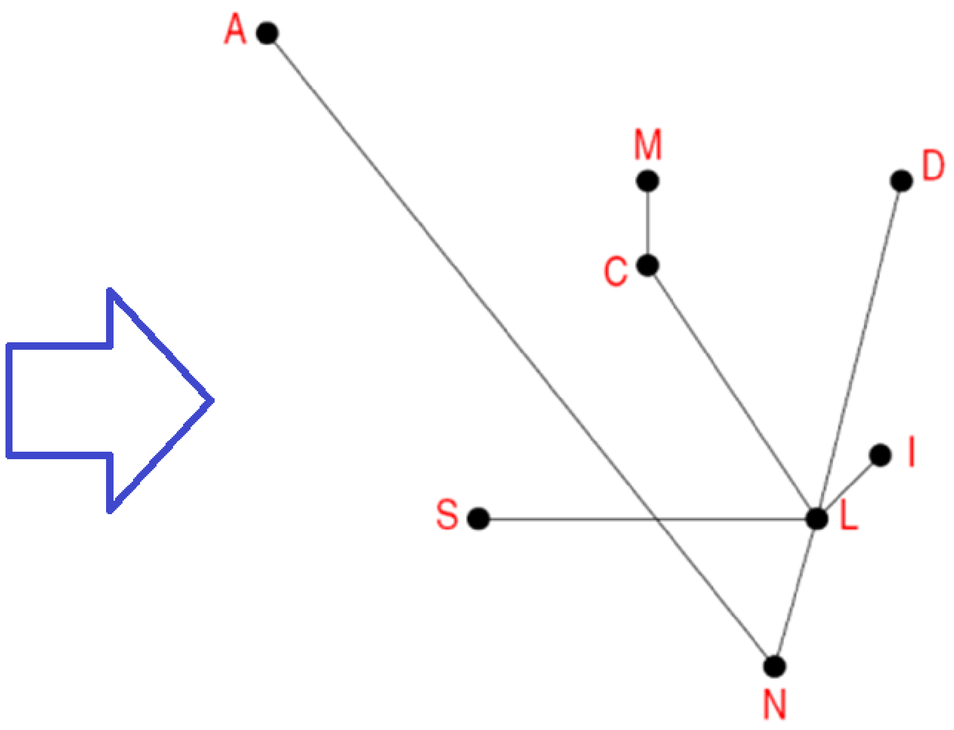
- Além disso, ela precisa atender **à toda demanda aérea** do grafo.



Árvore Geradora Mínima (AGM)

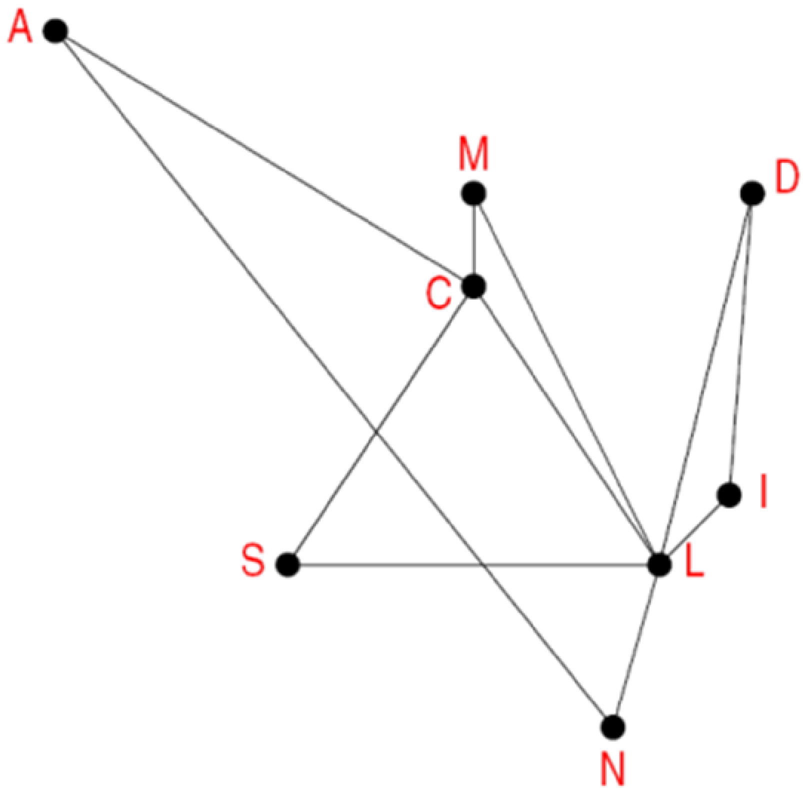
- Uma forma de atender à toda a demanda, isto é, interconectar todas as cidades, é a seguinte:

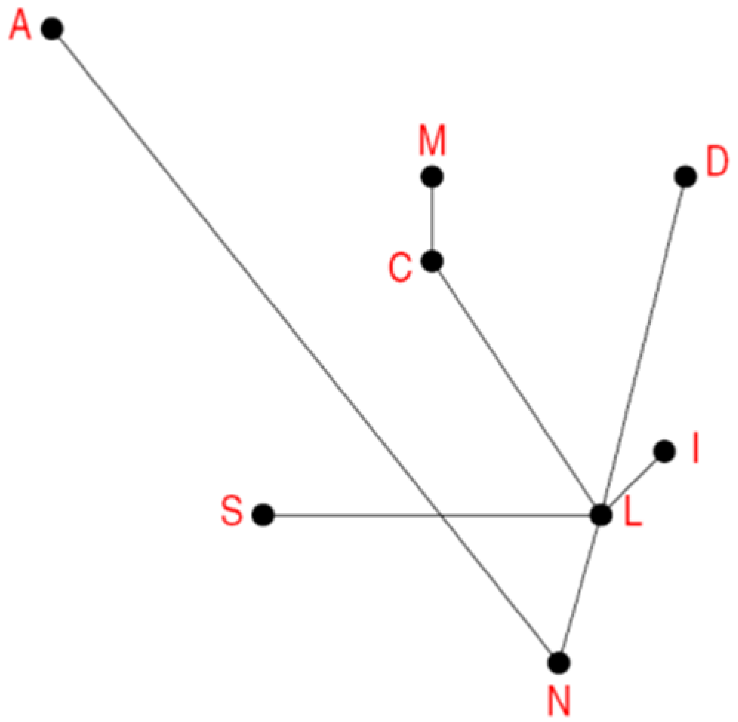




Árvore Geradora Mínima (AGM)

Q: Este conjunto de rotas é **MÍNIMO**?



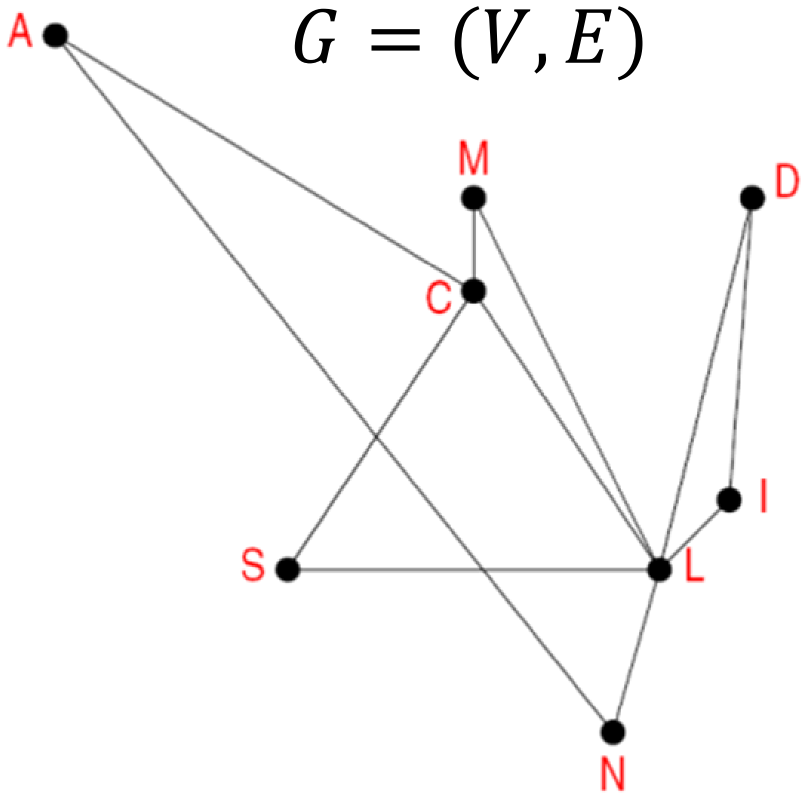


Árvore Geradora Mínima (AGM)

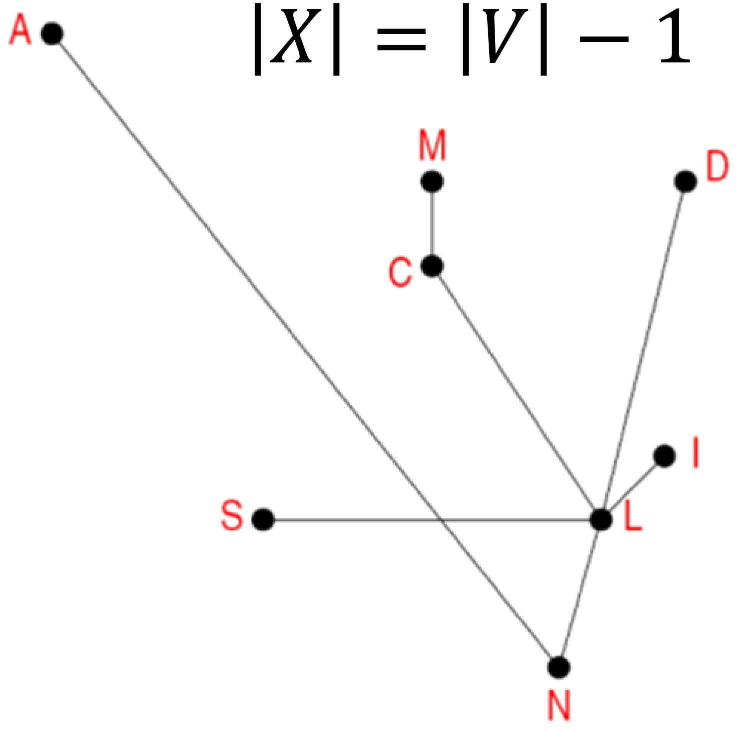
Q: Este conjunto de rotas é **MÍNIMO**?

Sim. Qualquer árvore extraída de um grafo com $|V|$ nós irá possuir $|V| - 1$ arestas.

$G = (V, E)$



$G' = (V, X)$, com $|X| = |V| - 1$

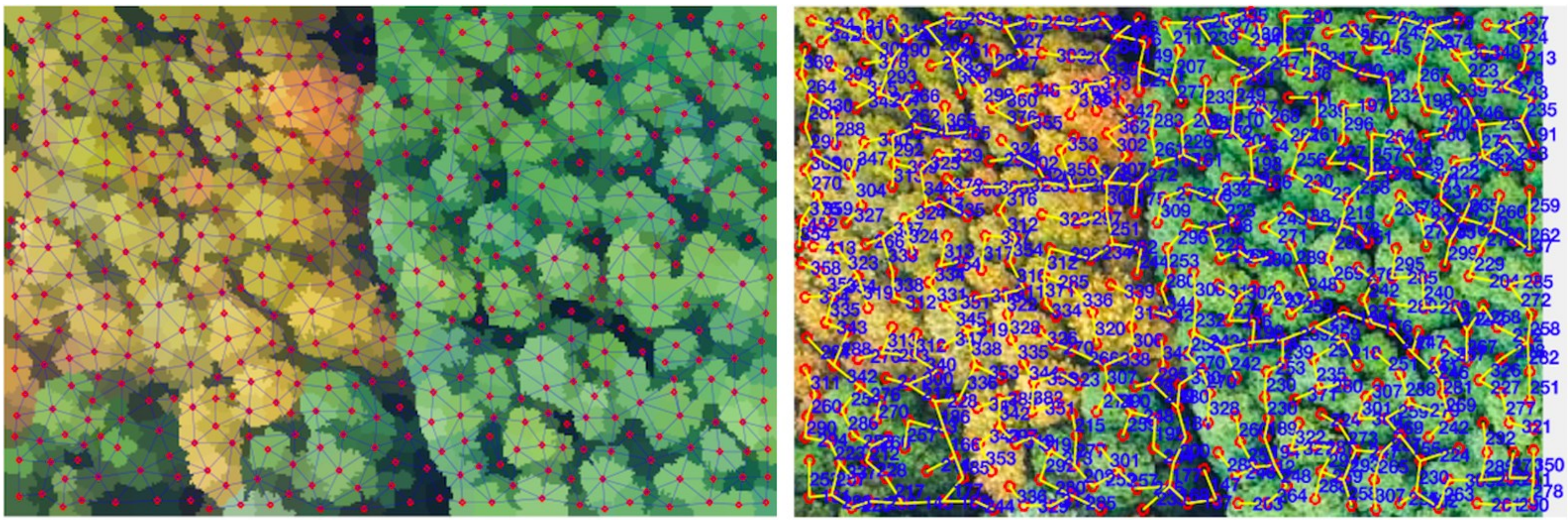


Árvore Geradora Mínima (AGM)

- O cientista checo *Otakar Borůvka* criou o primeiro algoritmo para encontrar uma AGM, em 1926.
- **Motivação:** resolver o problema de encontrar uma eficiente cobertura elétrica de Moravia (na Rep. Checa).
- Atualmente conhecido como *Algoritmo de Borůvka*

Árvore Geradora Mínima – Aplicações

- Transporte aéreo e terrestre.
- Redes de computadores.
- PDI, CG e IA.

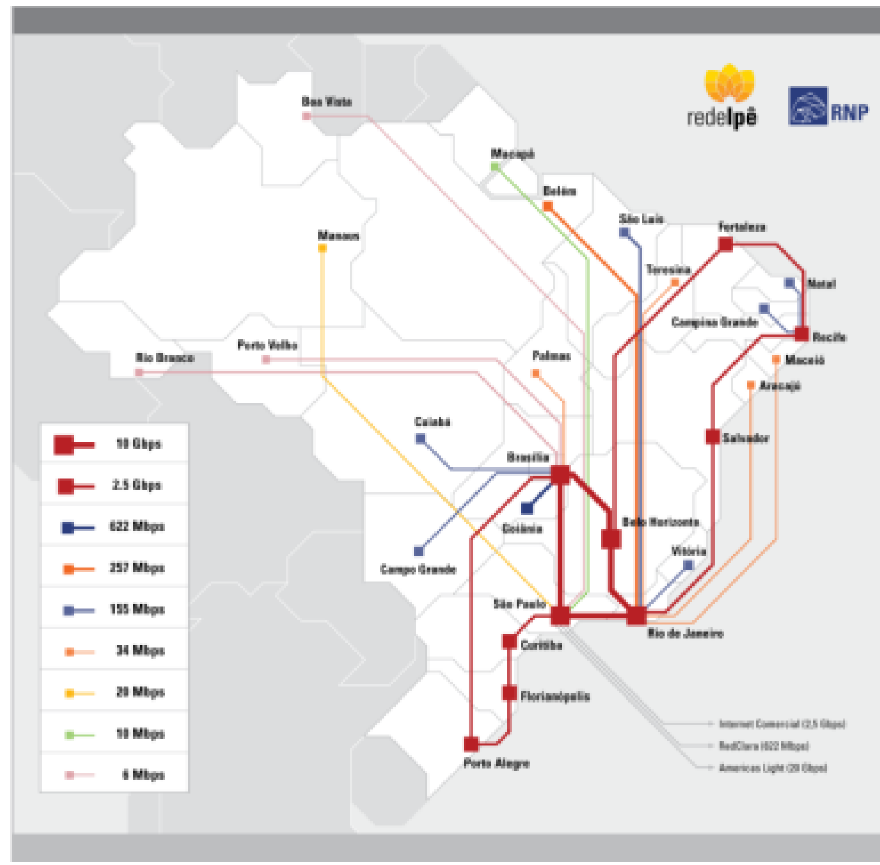


(a) (b)

Segmentação de imagens aéreas usando AGM (Yuhan et al. 2020 - https://doi.org/10.1117/1.JRS.14.022210)

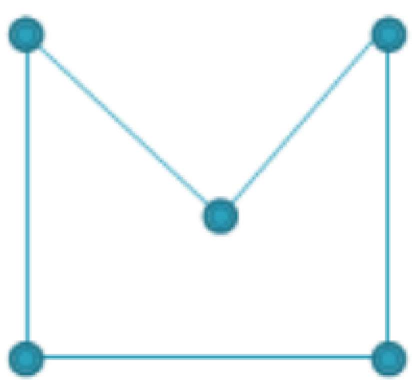
Árvore Geradora Mínima – Aplicações

- Circuitos.
- Redes de transmissão de energia.

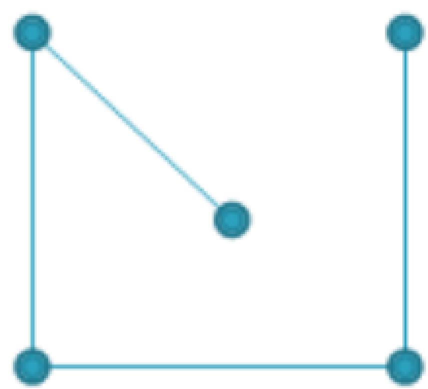


Árvores Geradoras

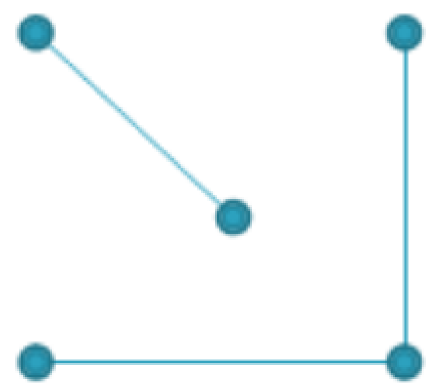
- **Definição (Subgrafo gerador)**: Seja $G = G(V, E)$ um grafo. Denomina-se subgrafo gerador um subgrafo $G_1 = (V_1, E_1)$ tal que $|V| = |V_1|$.



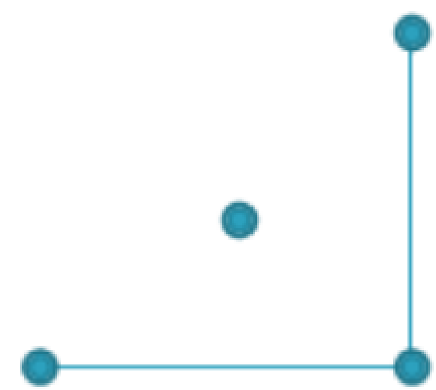
Grafo G



G_1 é um subgrafo gerador



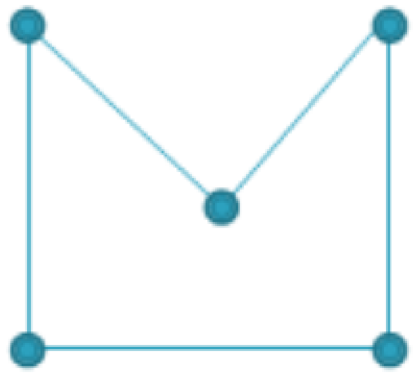
G_2 é um subgrafo gerador



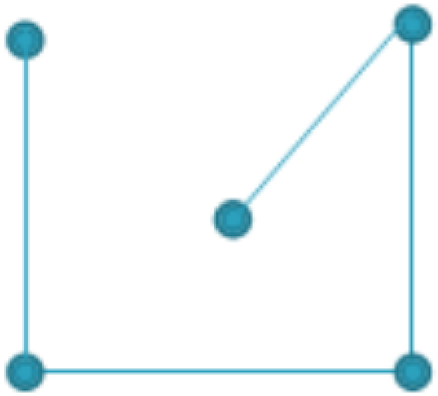
G_3 **NÃO** é um subgrafo gerador

Árvores Geradoras

- Quando o subgrafo gerador é uma **árvore**, o mesmo é denominado **árvore geradora**.



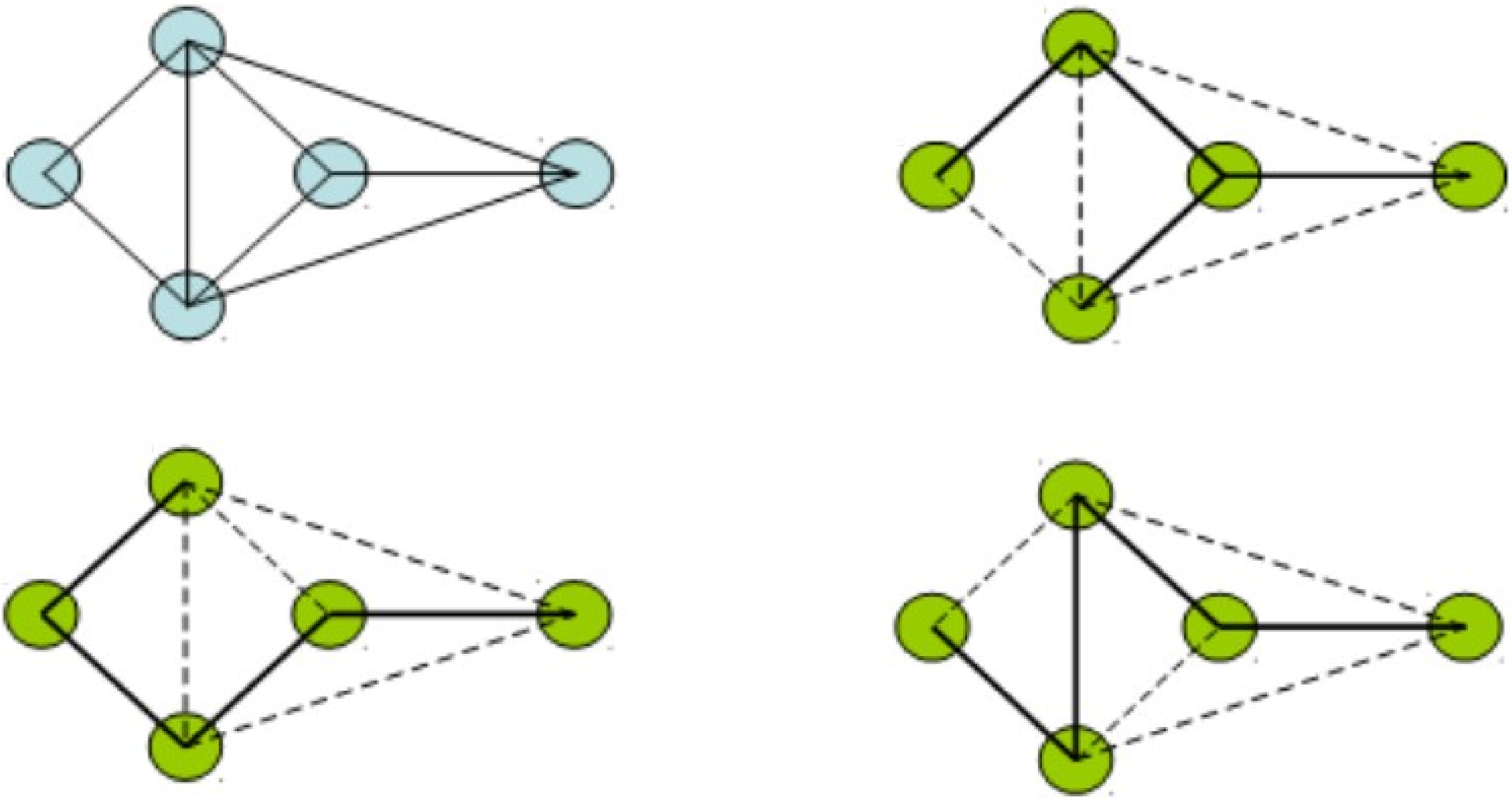
Grafo G



G_4 é um subgrafo gerador, ou seja, é uma árvore geradora de G.

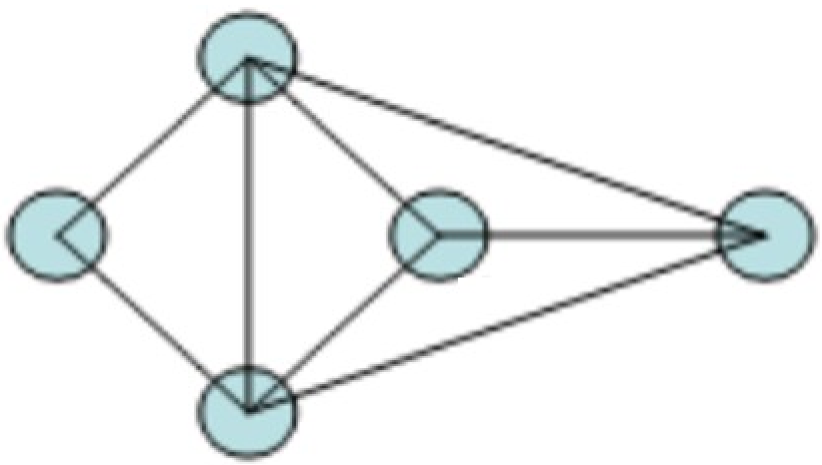
Árvores Geradoras

- Um grafo pode ter várias árvores geradoras

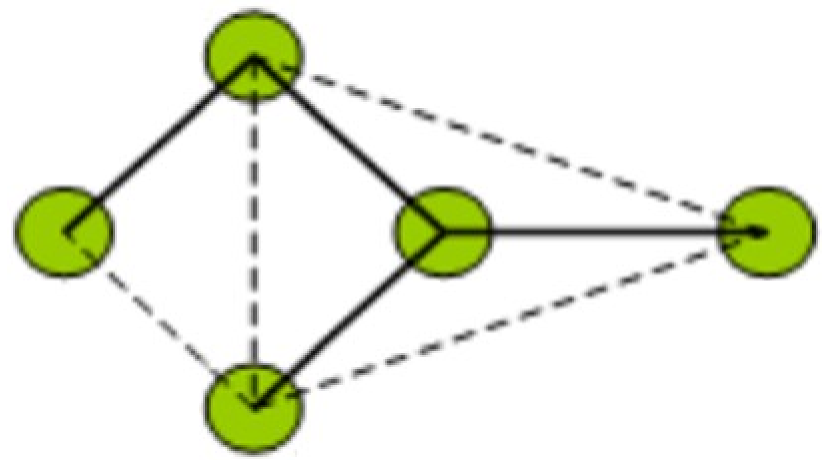


Árvores Geradoras

- Árvore geradora de um grafo conexo G é, portanto, um subgrafo gerador de G que, na verdade, é uma árvore.



Grafo G



Árvore geradora de G

- **Teorema**: todo grafo conexo possui pelo menos uma árvore geradora.

Árvore Geradora Mínima - Modelagem

- Seja um grafo conexo $G(V, E, W)$ não direcionado e com pesos nas arestas, isto é, para cada aresta $(u, v) \in E$, $w(u, v)$ é o peso (custo) da ligação de u com v.

- **Problema da AGM (MST – Minimum Spanning Tree):** Encontrar um subconjunto acíclico, $X \subseteq E$, que conecte todos os nós do grafo, e cuja soma dos pesos (peso total) seja a menor possível.

$$G = (V, E) \Longrightarrow G' = (V, X)$$

$$\min_{X \subset E} \{w(X)\}; \quad w(X) = \sum_{(u,v) \in X} w(u, v)$$

Árvore Geradora Mínima - Modelagem

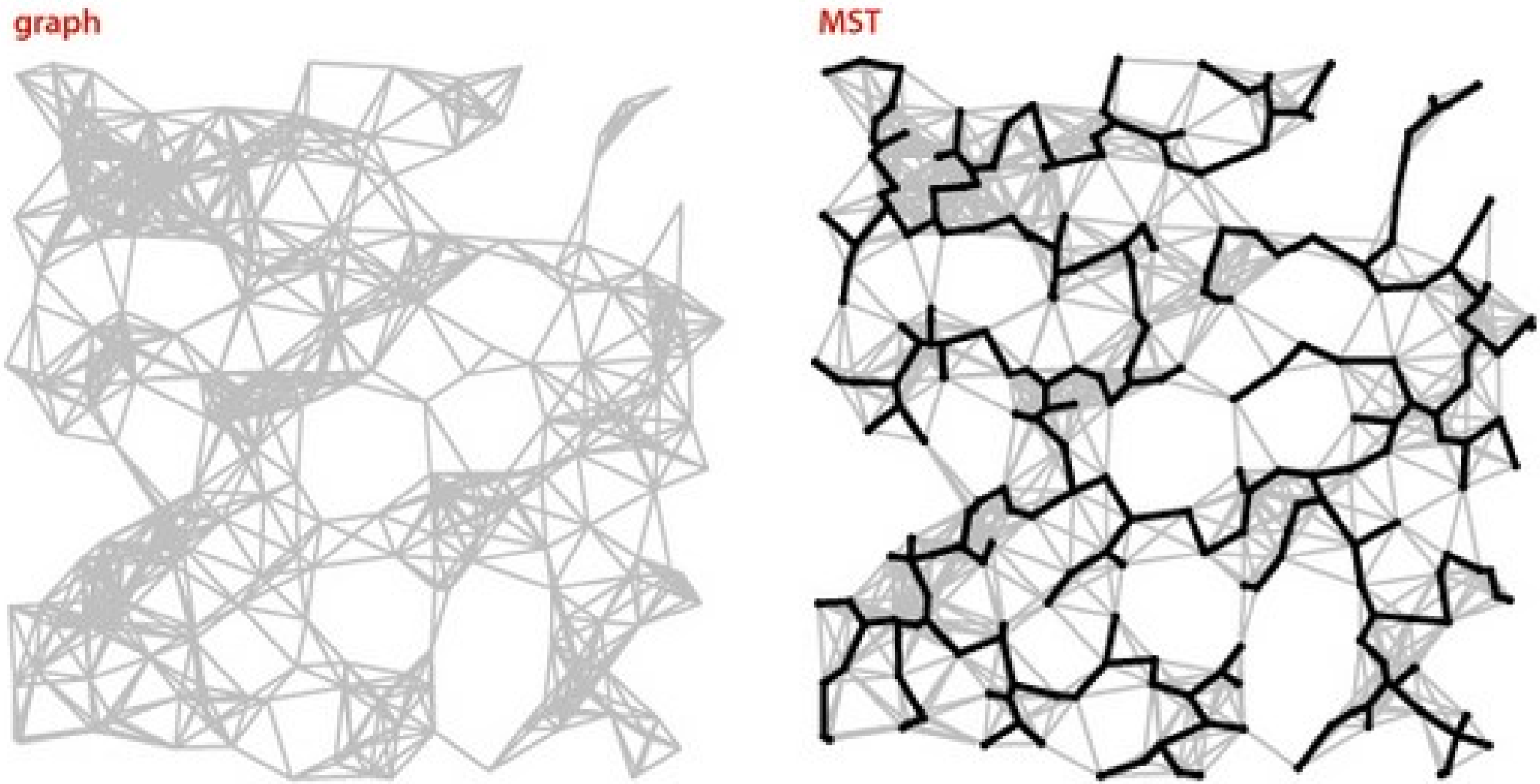
Observações:

- Tendo em vista que o conjunto X é **acíclico**, e que **conecta todos os nós**, ele é, na verdade, uma árvore, isto é, $X = T$. Tal árvore é chamada de:
  - Árvore geradora, ou Árvore de varredura, ou ainda Árvore de extensão.

- O problema de determinar a árvore geradora de **menor custo** é conhecido como:
  - Problema da Árvore Geradora Mínima (AGM), ou Problema da Árvore de Varredura Mínima, ou ainda Problema da Árvore de Extensão Mínima.

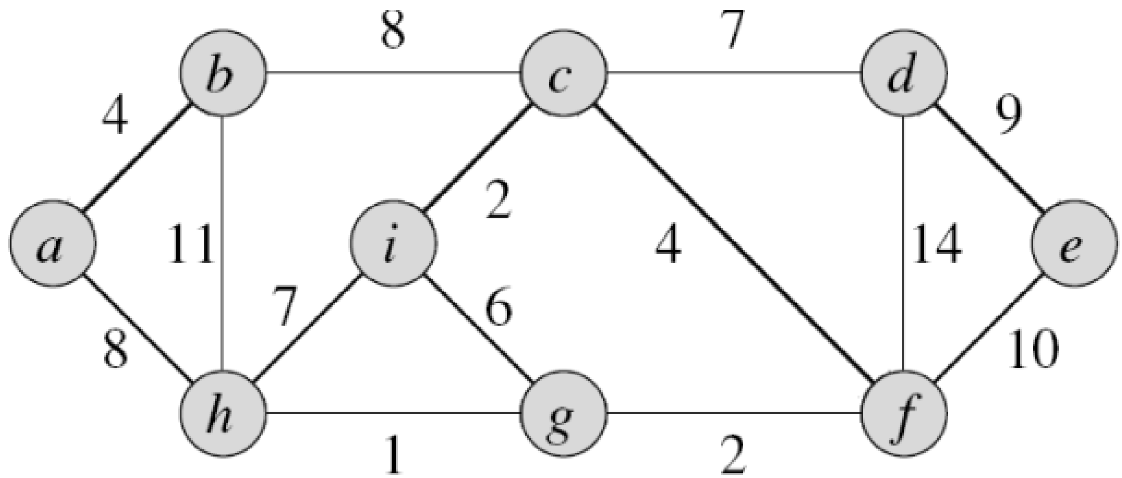
Árvore Geradora Mínima - Modelagem

- **Problema da AGM/MST:** Encontrar uma árvore geradora $X \subseteq E$ cuja soma dos pesos é a menor possível.

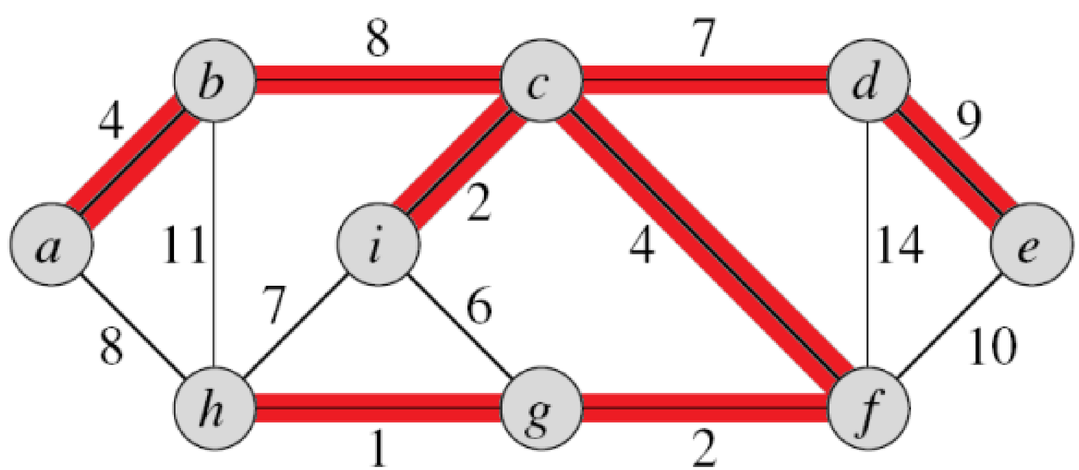


Observação: a AGM pode **não ser única** em um grafo.

$$G = (V, A)$$



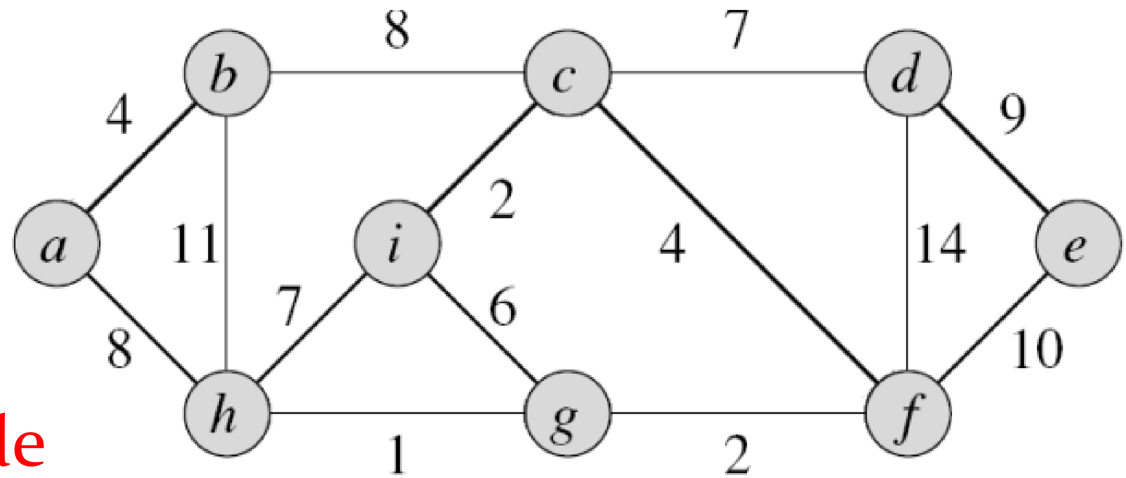
$$G' = (V, X)$$



Árvore Geradora Mínima - Modelagem

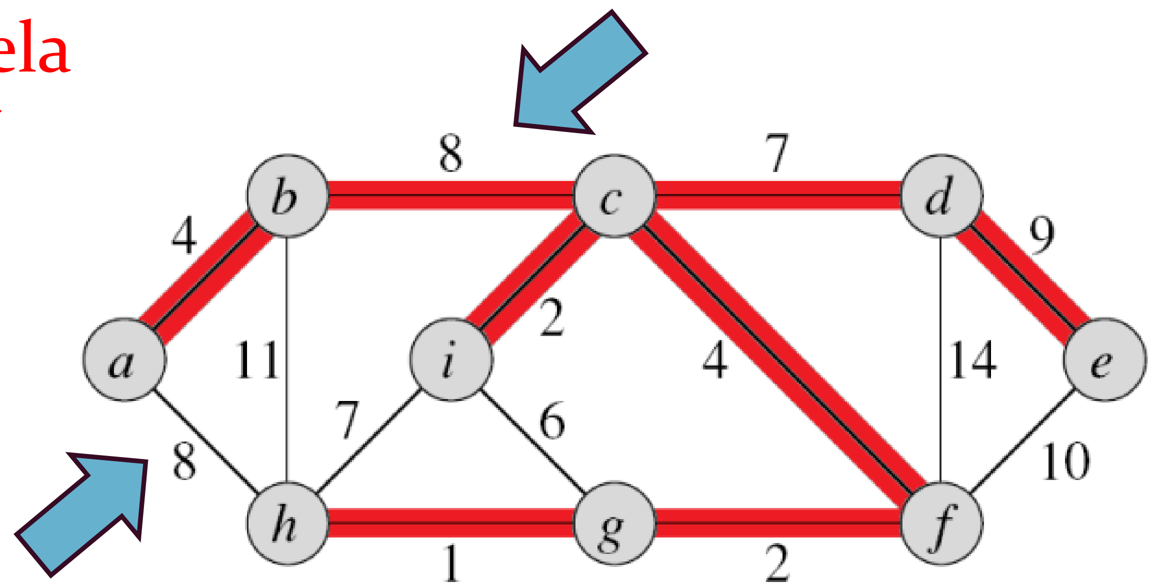
Observação: a AGM pode **não ser única.**

$$G = (V, A)$$



A aresta (b, c) pode ser substituída pela aresta (a, h) em X

$$G' = (V, X)$$



Árvore Geradora Mínima - Modelagem

- Existem dois algoritmos clássicos na literatura para resolver o problema da AGM:
  - Algoritmo de Kruskal;
  - Algoritmo de Prim;

- Ambos são considerados algoritmos "gulosos".
  - Nos algoritmos acima, a estratégia gulosa defende que **a escolha da menor aresta a cada passo deve ser sempre realizada**.
    - Melhor escolha imediata.

Árvore Geradora Mínima

Árvore Geradora Mínima – Estratégia Genérica

- Algoritmo genérico, que constrói uma árvore geradora mínima adicionando uma aresta de cada vez.

- Grafo não-orientado: $G = (V, E)$
- Peso nas arestas: $w: E \to \mathbb{R}$
- Antes de cada iteração, X representa o **subconjunto de arestas** de alguma AGM.
- A cada iteração, uma aresta $(u, v) \in E$ é adicionada ao conjunto X.

- **Problema:** Como definir qual aresta do grafo original G pode integrar alguma AGM?

$$X \leftarrow X \cup \{(u, v)\}$$

Algoritmo Genérico - AGM

Input: Grafo não-direcionado $G = G(V, E, W)$

AGM_GENERICO (G)

$X \leftarrow \{\ \}$

Enquanto $|X| \neq |V| - 1$ *faça*

Encontrar uma aresta (u,v) segura para X

$X \leftarrow X \cup \{(u, v)\}$

Fim_enquanto

Retorna X

Fim

Algoritmo Genérico - AGM

Input: Grafo não-direcionado $G = G(V, E, W)$

$AGM\_GENERICO\ (G)$

$X \leftarrow \{\ \}$

$Enquanto\ |X| \neq |V| - 1\ \ faça$

Encontrar uma aresta (u,v) segura para X

$X \leftarrow X \cup \{(u, v)\}$

$Fim\_enquanto$

$Retorna\ X$

Fim

É obvio que o ponto chave é a localização da aresta que pode fazer parte de alguma AGM.

Árvore Geradora Mínima – Estratégia Genérica

- Para identificar uma aresta segura para a AGM, precisamos de um conceito da **Teoria dos Grafos** denominado **CORTE**.

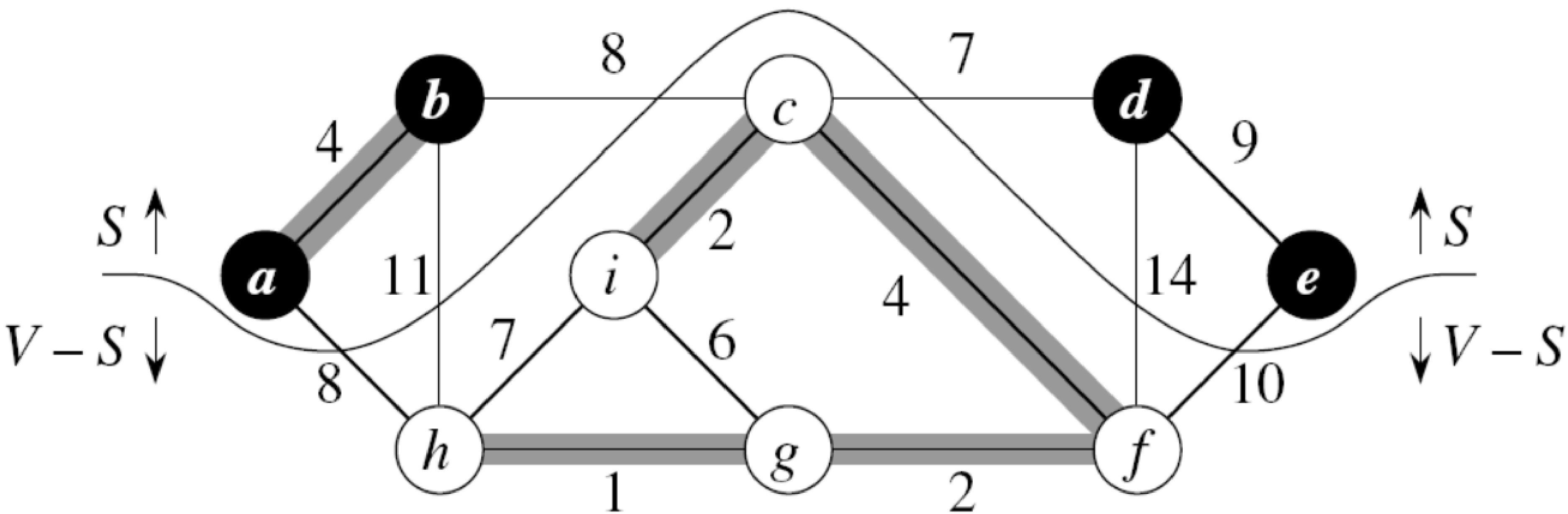
- Corte: é uma **partição do conjunto de nós**.

- Matematicamente
  - **Corte em um grafo** $G = (V, E, W)$**:**

$$Partição\text{: } (S, V - S)$$

Árvore Geradora Mínima – Estratégia Genérica

- Primeira forma de visualizar um corte:

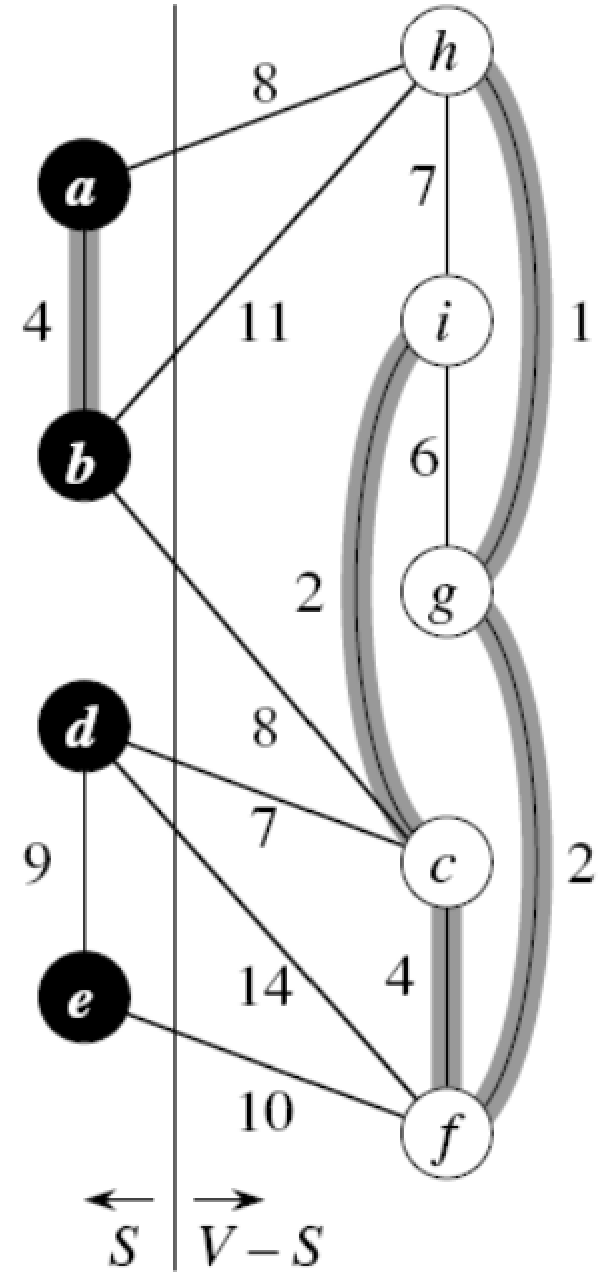


$$S = \{a, b, d, e\}$$
$$V - S = \{h, i, c, g, f\}$$

Árvore Geradora Mínima

- Segunda forma de visualizar o mesmo corte:

$$S = \{a, b, d, e\}$$
$$V - S = \{h, i, c, g, f\}$$



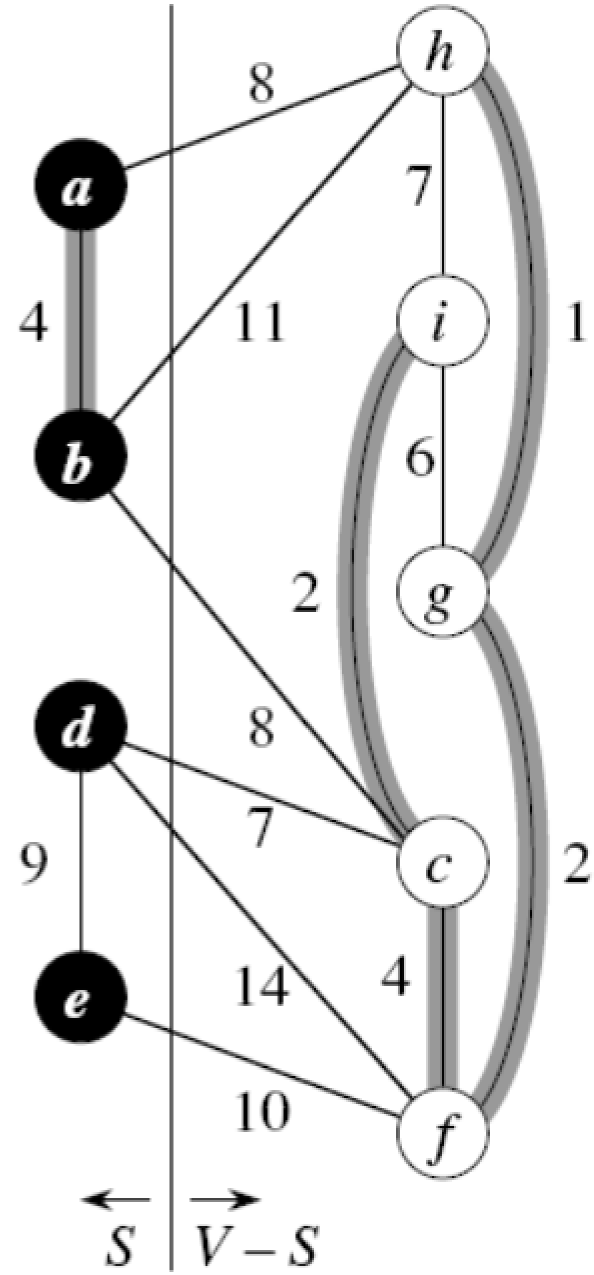
Árvore Geradora Mínima

Definição 1 (aresta que cruza o corte):

- Dizemos que uma aresta (u, v) **cruza o corte** se um dos seus nós está em S, e o outro está em $V - S$.

Definição 2 (corte respeita X):

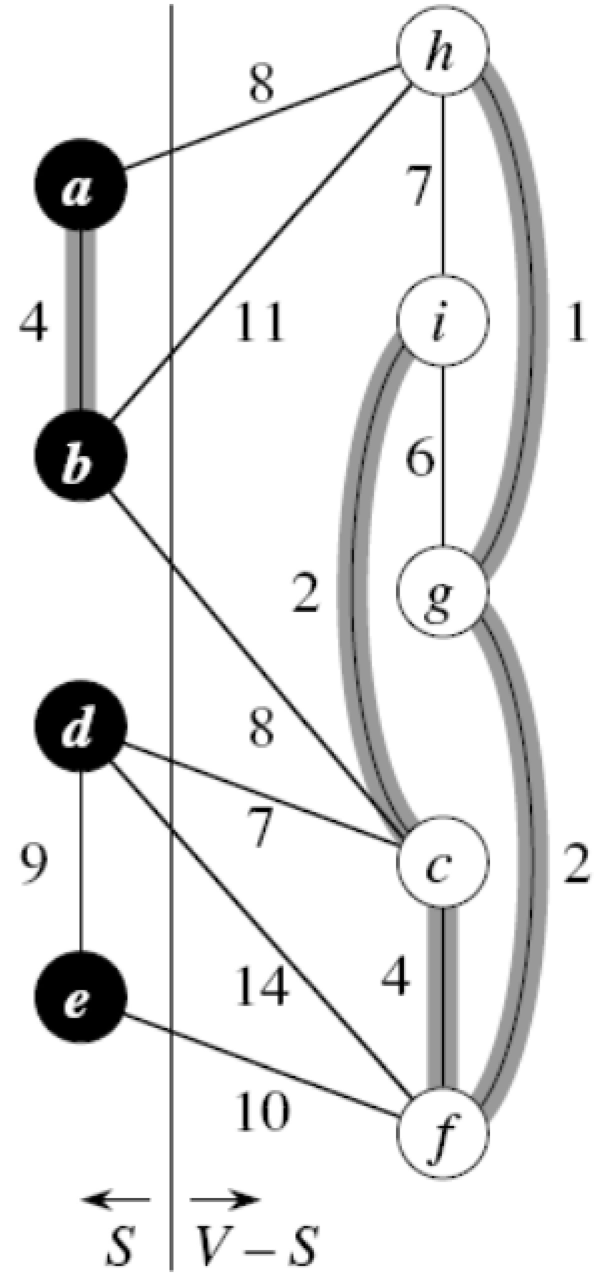
- Dizemos que um corte **respeita o conjunto** $X \subseteq E$ se nenhuma aresta de X cruza o corte.



Árvore Geradora Mínima

Definição 3 (aresta leve):

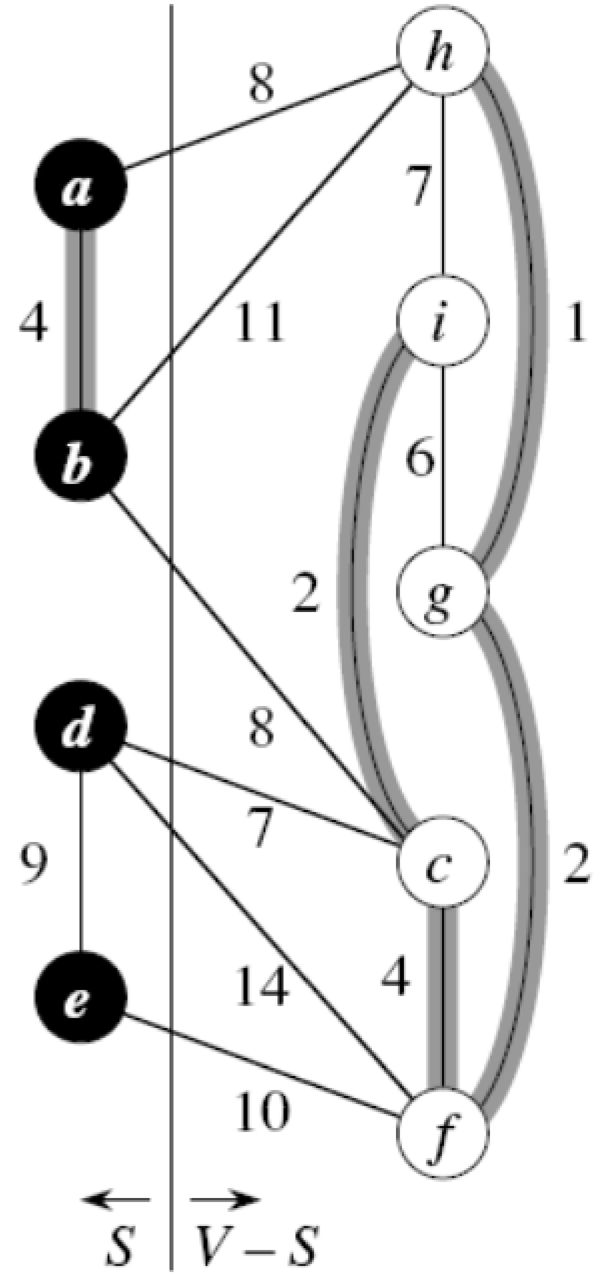
- Dizemos que uma aresta é uma **aresta leve** cruzando o corte se o seu peso é o menor quando comparado às outras arestas que cruzam o corte.



Árvore Geradora Mínima

Teorema (arestas seguras)

- Seja $(S, V - S)$ qualquer corte do grafo G que respeita $X \subseteq E$, e seja (u, v) uma aresta leve cruzando $(S, V - S)$, então a aresta (u, v) é segura para X.



Algoritmo Genérico - AGM

Input: Grafo não-direcionado $G = G(V, E, W)$

$AGM\_GENERICO\ (G)$

$X \leftarrow \{\ \}$

$Enquanto\ |X| \neq |V| - 1\ \ faça$

Encontrar uma aresta (u,v) segura para X

$X \leftarrow X \cup \{(u, v)\}$

$Fim\_enquanto$

$Retorna\ X$

Fim

AGM – Algoritmo de Kruskal

- Algoritmo de Kruskal: criado por Joseph Bernard Kruskal, Jr, em 1956.
- Nascimento: 1928 – Falecimento: 2010
- Terminou seu PhD na Universidade de Princeton em 1956.
- Orientado por Paul Erdos.
- Trabahou na Bell Laboratories (divisão de pesquisa da empresa de telecomunicações AT&T); Univ. de Princeton e NYU.

Árvore Geradora Mínima – Kruskal

- O algoritmo inicia com uma floresta, e vai adicionando arestas seguras à AGM.

- **Arestas seguras no Algoritmo de Kruskal:**
  - No algoritmo de Kruskal, a aresta segura será sempre uma **aresta de peso mínimo que conecta dois componentes distintos** (duas árvores distintas da floresta).

Árvore Geradora Mínima – Kruskal

- **Ponto Chave do Algoritmo**: A cada iteração, encontra uma aresta segura para adicionar à floresta.
- Kruskal é considerado um algoritmo guloso, porque em cada passo, ele adiciona à floresta uma **aresta de peso mínimo** (daquelas que ainda podem ser adicionadas).
  - Ou seja, a cada iteração, ele realiza uma avaliação dentre todas as possibilidades existentes.

AGM – Algoritmo de Kruskal

$AGM\_Kruskal\ (G = G(V, E, W))$

$X \leftarrow \{\ \}$ //AGM

$Para\ cada\ nó\ v \in V\ \ faça$

$Cria\_Árvore\ (v)$

Fim_para

$E' \longleftarrow ordenar\ as\ arestas\ de\ E\ por\ pesos\ crescentes$

$Para\ cada\ aresta\ (u, v) \in E'$

$Se\ Conjunto\_De(u) \neq Conjunto\_De(v)\ então$

$X \leftarrow X \cup \{(u, v)\}$

$Aplicar\_Uniao(u, v)$

Fim_se

Fim_para

$Retorne\ X$

AGM – Algoritmo de Kruskal

$AGM\_Kruskal\ (G = G(V, E, W))$

$X \leftarrow \{\ \}$ //AGM

$Para\ cada\ nó\ v \in V\ \ faça$

$Cria\_Árvore\ (v)$

Fim_para

|V| árvores auxiliares são criadas

$E' \leftarrow ordenar\ as\ arestas\ de\ E\ por\ pesos\ crescentes$

$Para\ cada\ aresta\ (u, v) \in E'$

$Se\ Conjunto\_De(u) \neq Conjunto\_De(v)\ então$

$X \leftarrow X \cup \{(u, v)\}$

$Aplicar\_Uniao(u, v)$

Fim_se

Fim_para

$Retorne\ X$

$AGM\_Kruskal\ (G = G(V, E, W))$

$X \leftarrow \{\ \}$ //AGM

$Para\ cada\ nó\ v \in V\ \ faça$

$Cria\_Árvore\ (v)$

Fim_para

$\boxed{E' \leftarrow ordenar\ as\ arestas\ de\ E\ por\ pesos\ crescentes}$

$Para\ cada\ aresta\ (u, v) \in E'$

$Se\ Conjunto\_De(u) \neq Conjunto\_De(v)\ então$

$X \leftarrow X \cup \{(u, v)\}$

$Aplicar\_Uniao(u, v)$

Fim_se

Fim_para

$Retorne\ X$

O conjunto de arestas é ordenado em função dos pesos. Condição necessária para a criação da AGM.

$AGM\_Kruskal\,(G = G(V, E, W))$

$X \leftarrow \{\ \}$ //AGM

$Para\ cada\ nó\ v \in V\ \ faça$

$Cria\_Árvore\ (v)$

Fim_para

$E' \leftarrow$ *ordenar as arestas de E por pesos crescentes*

$Para\ cada\ aresta\ (u, v) \in E'$

$Se\ Conjunto\_De(u) \neq Conjunto\_De(v)\ então$

$X \leftarrow X \cup \{(u, v)\}$

$Aplicar\_Uniao(u, v)$

Fim_se

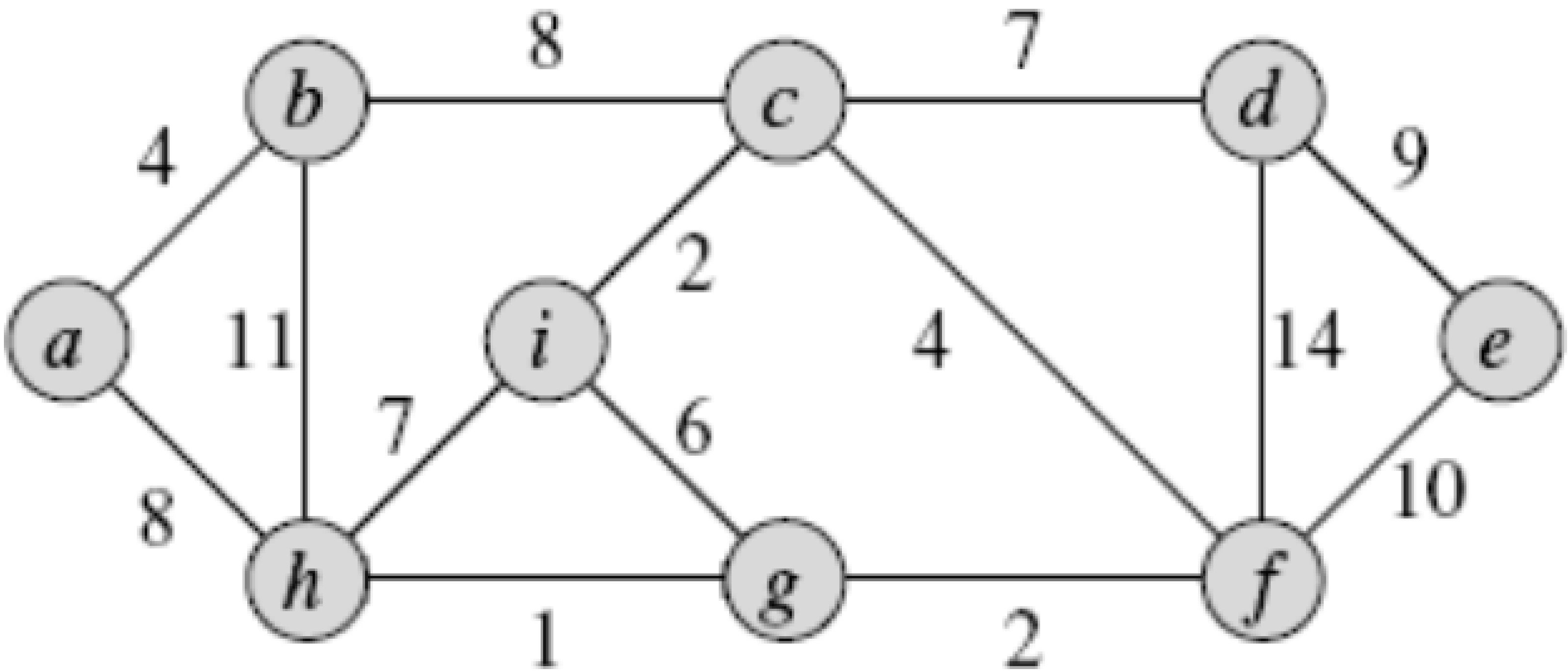
Fim_para

$Retorne\ X$

Se ***u*** e ***v*** estão em árvores distintas, a aresta (***u***,***v***) é adicionada em X, e então é aplicado uma união das árvores de ***u*** e ***v***.

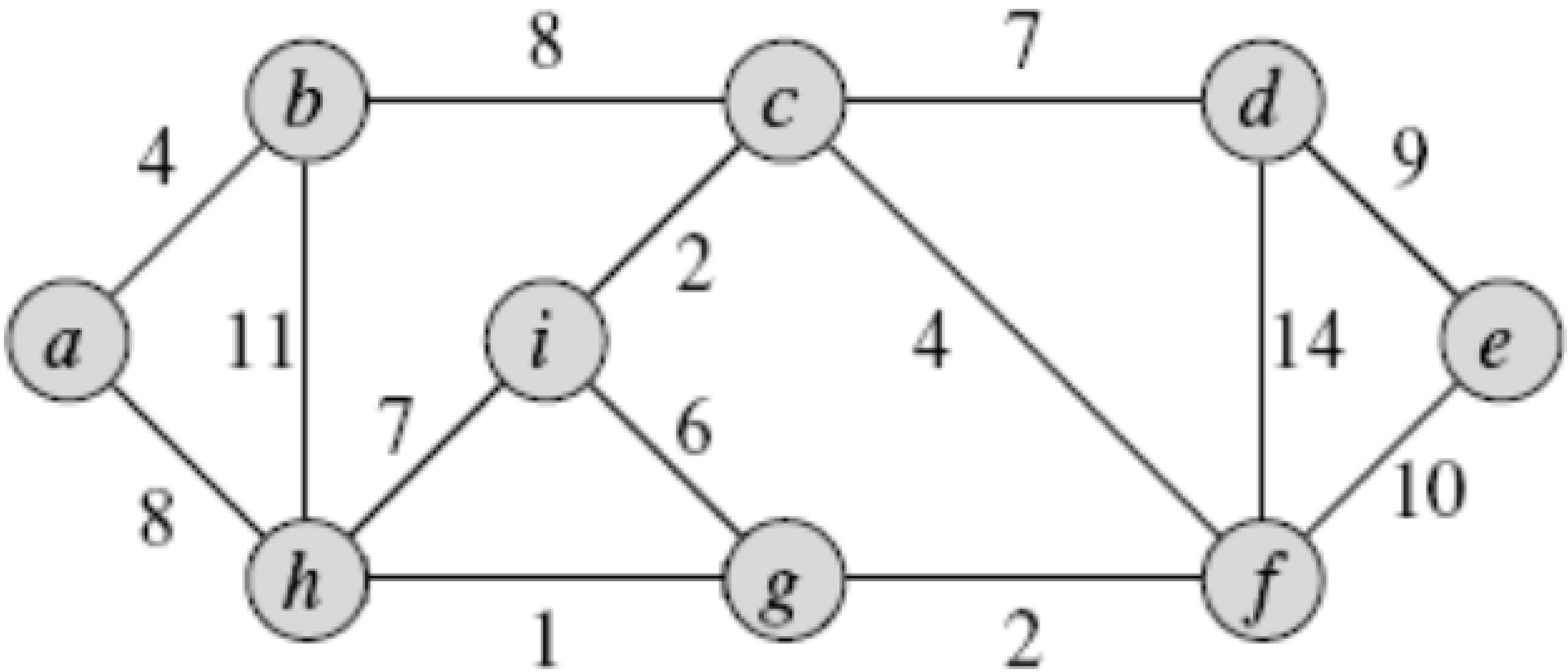
AGM – Algoritmo de Kruskal

Rodando um exemplo "na mão"



AGM – Algoritmo de Kruskal

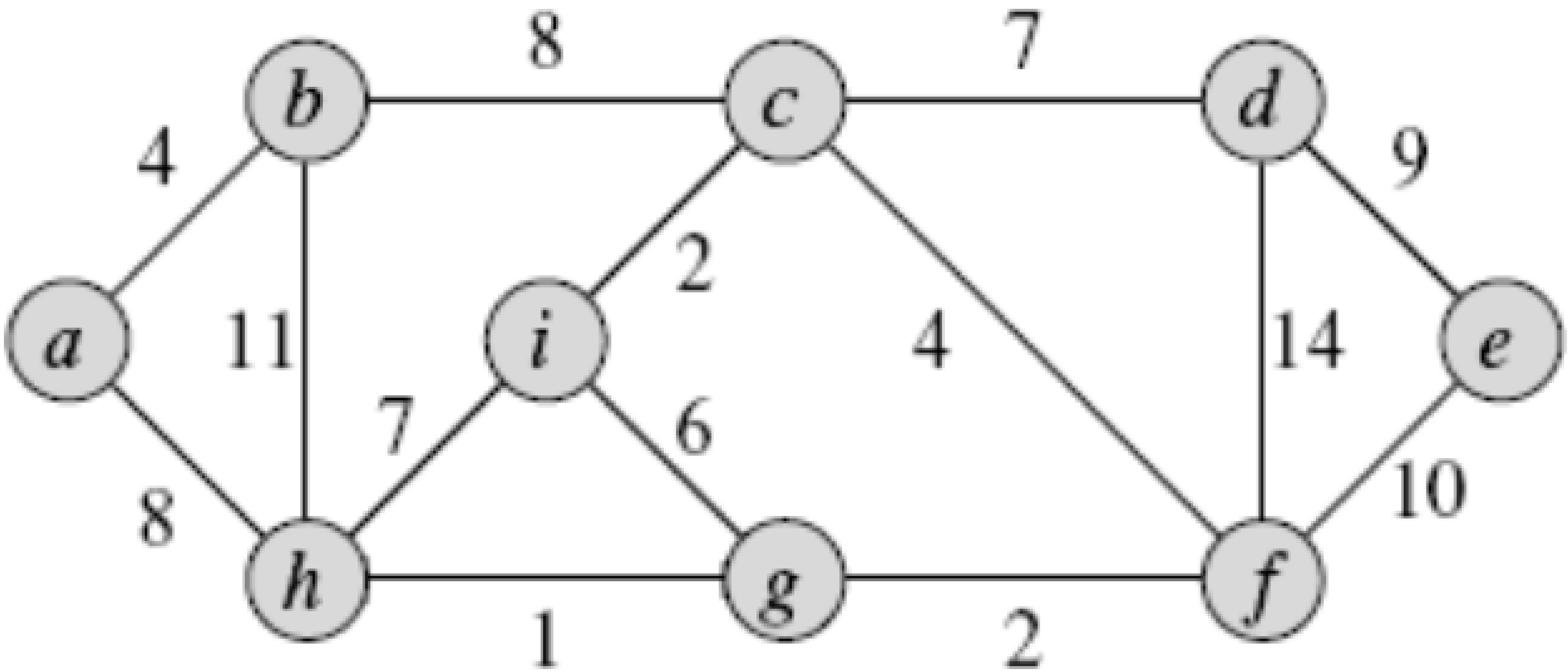
Passo 1: criar um conjunto (árvore) para cada nó.



$$\{\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\{g\},\{h\},\{i\}\}$$

AGM – Algoritmo de Kruskal

Passo 2: ordenar as arestas (por pesos) do conjunto E.

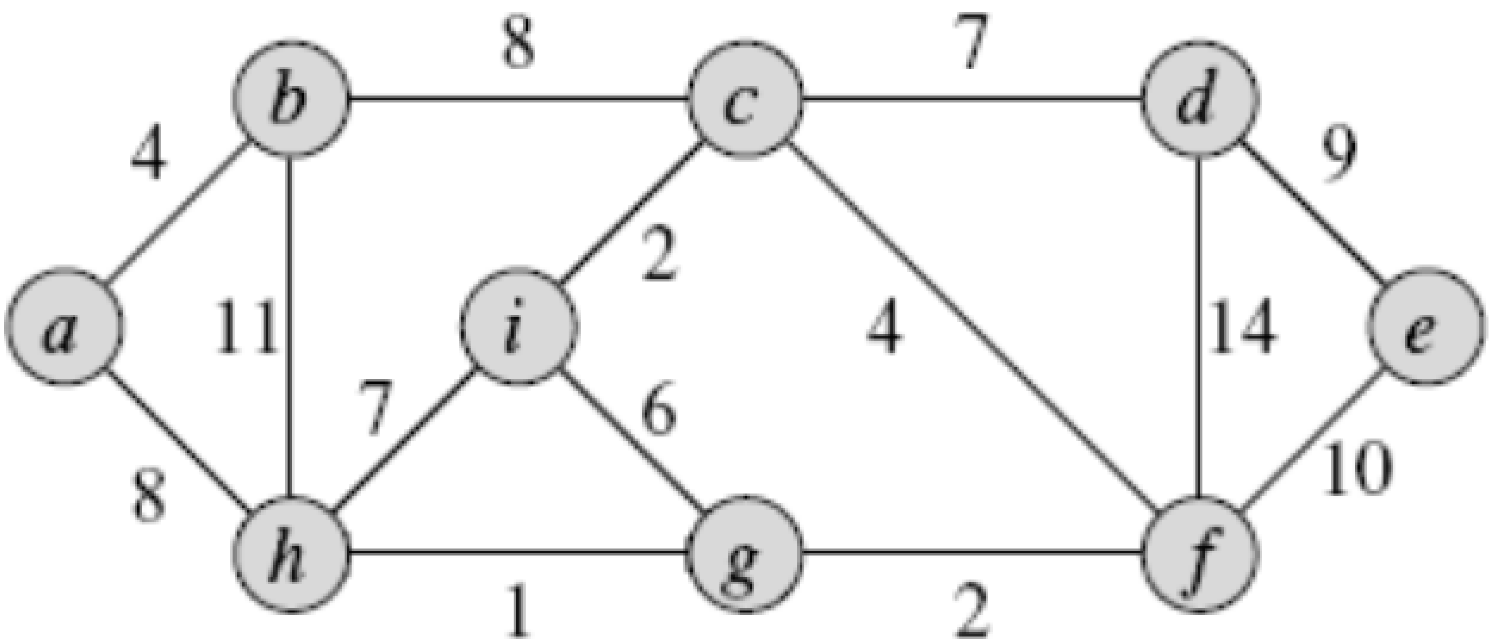


E': (g,h); (c,i); (f,g); (a,b); (c,f); (g,i); (c,d); (h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)

Passo 3: para cada aresta ordenada, faça

$$E': \boxed{(g,h);}\ (c,i); (f,g); (a,b); (c,f); (g,i); (c,d);$$
$$(h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)$$

g e ***h*** pertencem a mesma árvore na floresta?

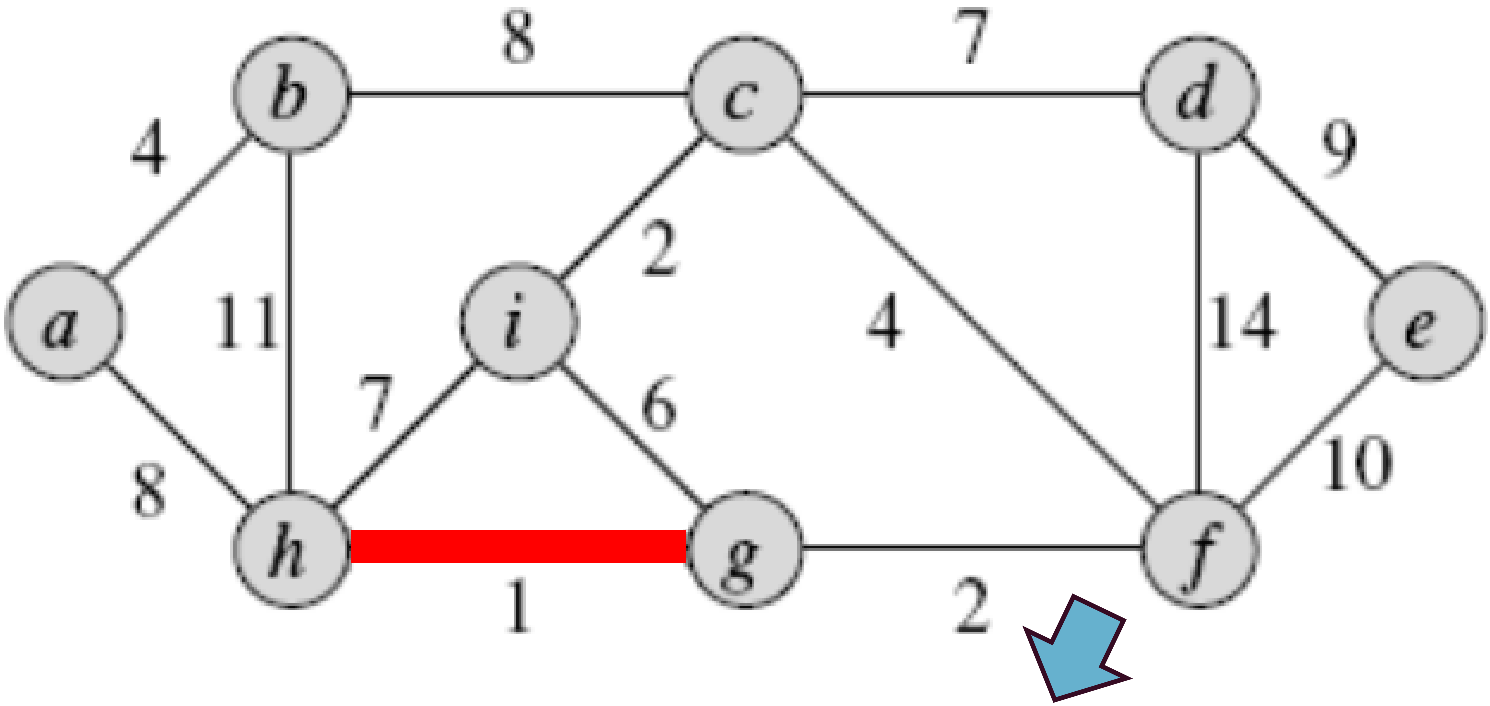


$$\{\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\{g\},\{h\},\{i\}\}$$

Passo 3: para cada aresta ordenada, faça

$$E': \boxed{(g,h)}; (c,i); (f,g); (a,b); (c,f); (g,i); (c,d);$$
$$(h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)$$

Não. Então união das árvores de ***g*** e ***h***; e adição na AGM



$$\{\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\boxed{\{g,h\}},\{i\}\}$$

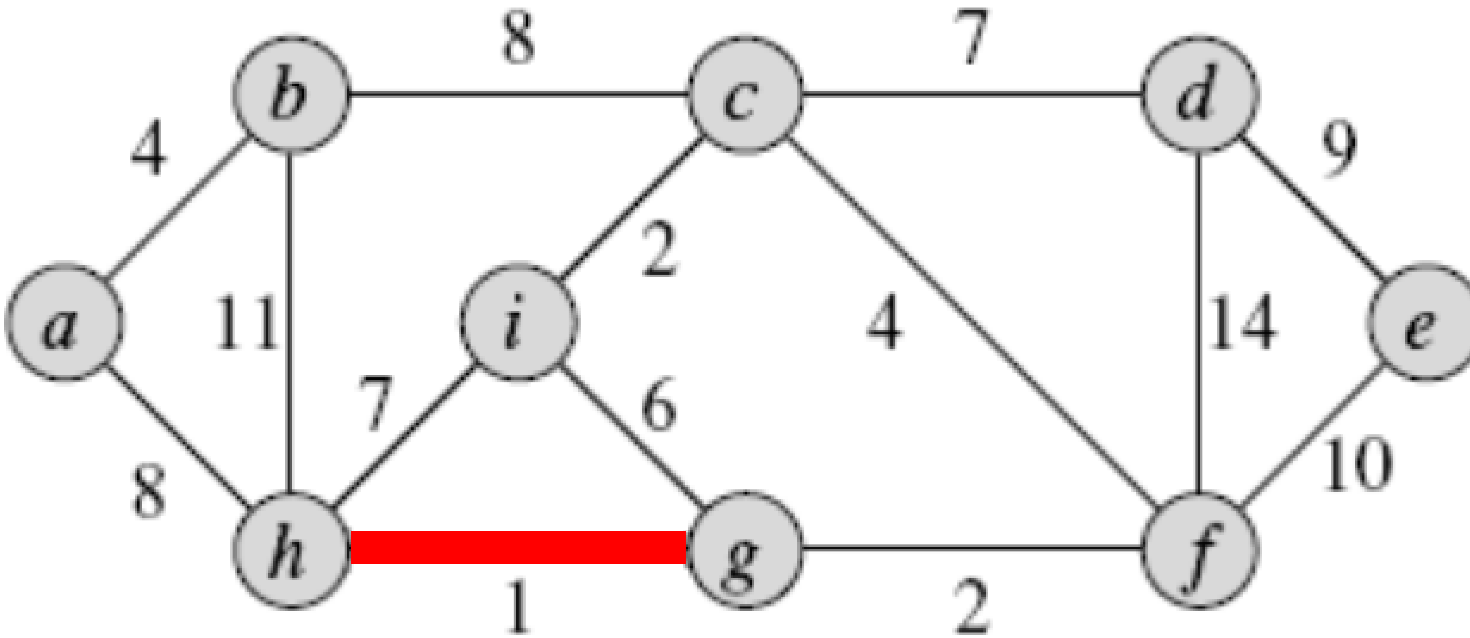
Algoritmo de Kruskal – auto-intutivido

AGM – Algoritmo de Kruskal

Passo 3: para cada aresta ordenada, faça

$E': (g,h); \boxed{(c,i)}; (f,g); (a,b); (c,f); (g,i); (c,d);$
$(h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)$

c **e** ***i*** **pertencem a mesma árvore na floresta?**

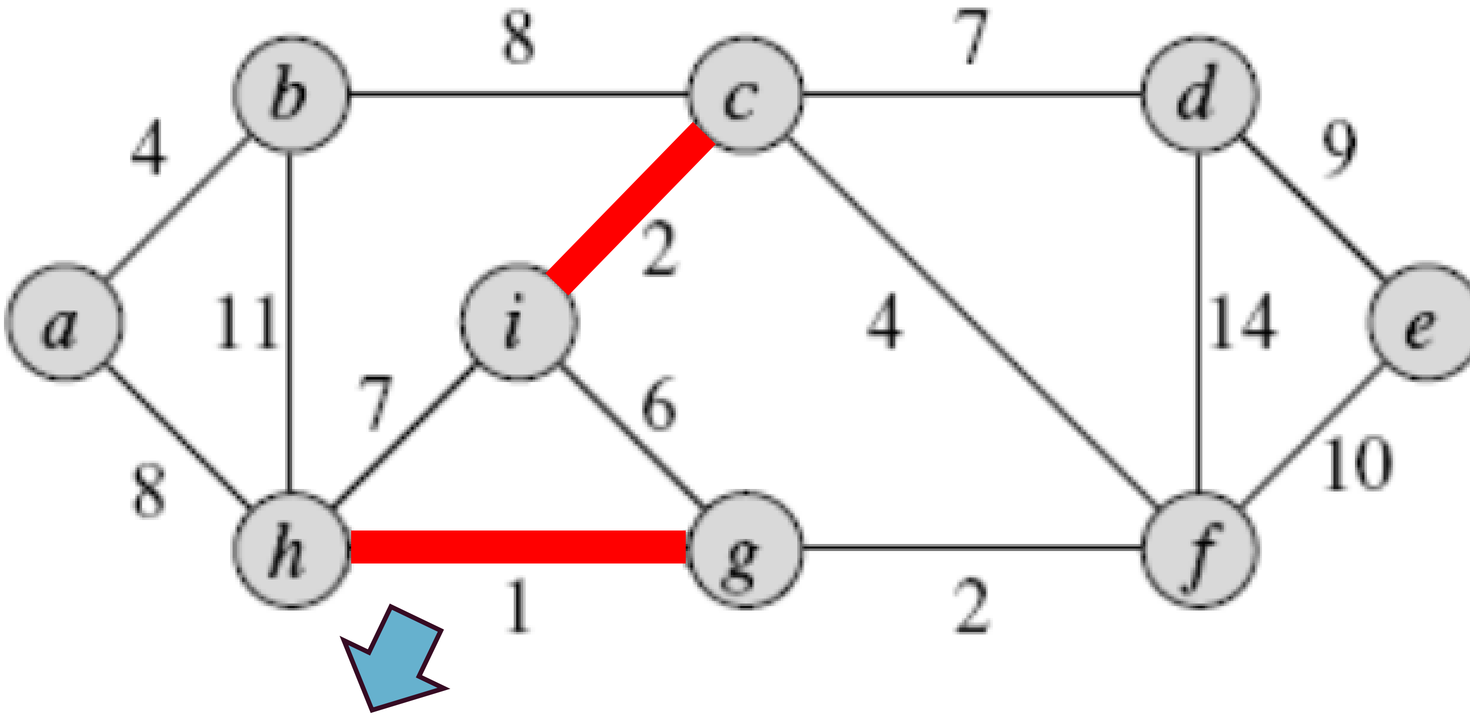


$$\{\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\{g,h\},\{i\}\}$$

Passo 3: para cada aresta ordenada, faça

$E': (g,h); \boxed{(c,i);} (f,g); (a,b); (c,f); (g,i); (c,d);$
$(h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)$

Não. Então união das árvores de ***c*** e ***i***; e adição na AGM



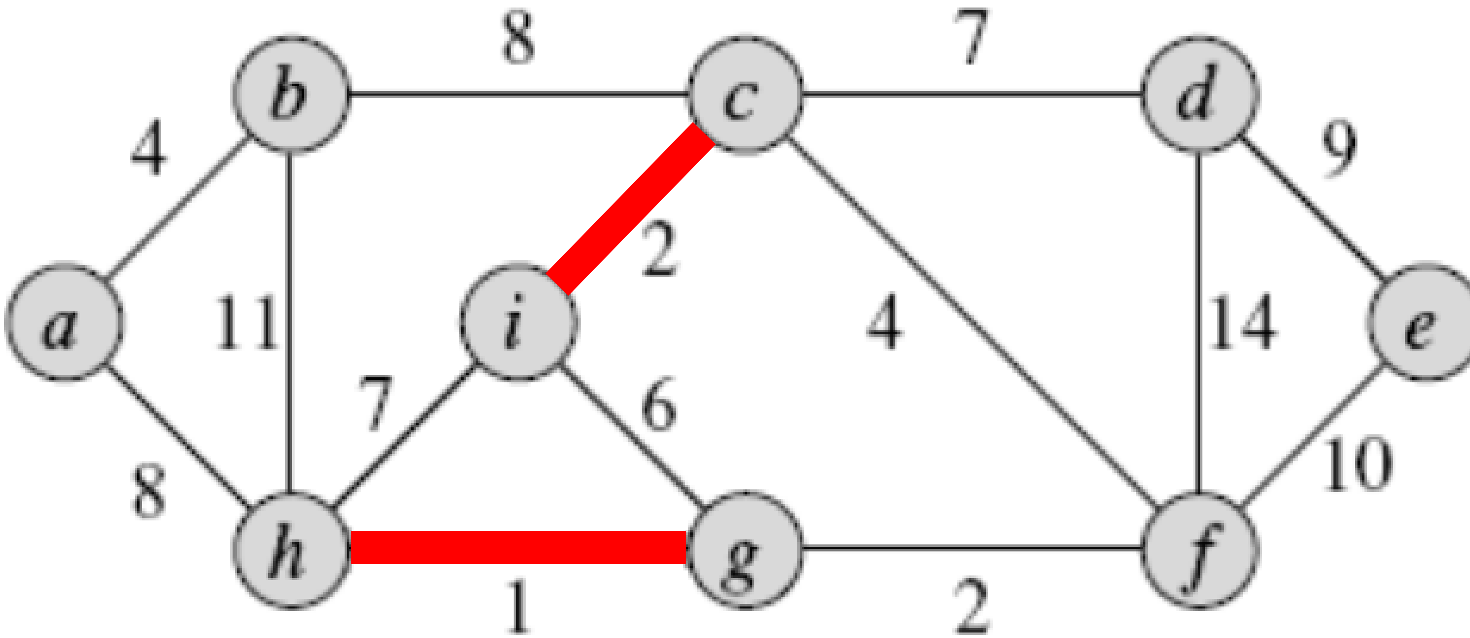
$$\{\{a\},\{b\},\boxed{\{c,i\},}\{d\},\{e\},\{f\},\{g,h\}\}$$

AGM – Algoritmo de Kruskal

Passo 3: para cada aresta ordenada, faça

E': (g,h); (c,i); $\boxed{(f,g)}$; (a,b); (c,f); (g,i); (c,d); (h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)

f e ***g*** pertencem a mesma árvore na floresta?

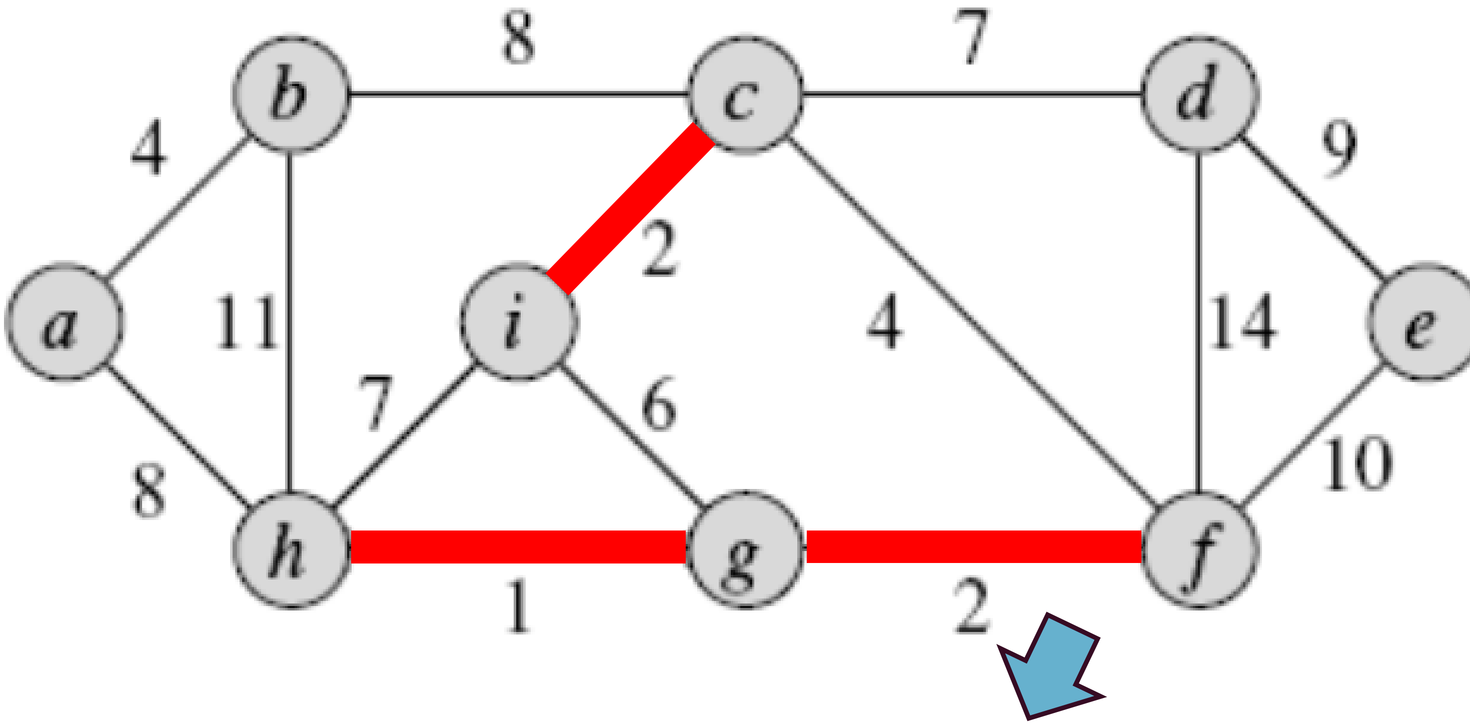


$$\{\{a\},\{b\},\{c,i\},\{d\},\{e\},\{f\},\{g,h\}\}$$

Passo 3: para cada aresta ordenada, faça

$$E': (g,h); (c,i); \boxed{(f,g)}; (a,b); (c,f); (g,i); (c,d);$$
$$(h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)$$

Não. Então união das árvores de ***f*** e ***g***; e adição na AGM

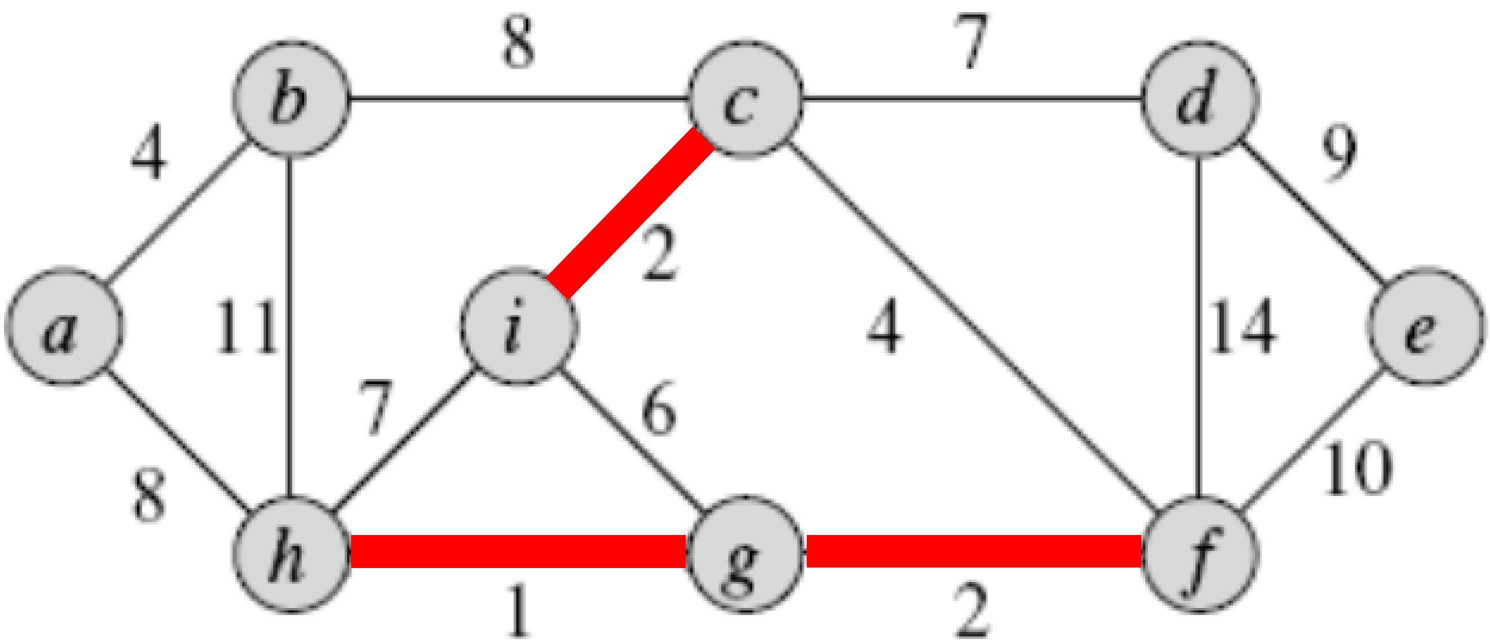


$$\{\{a\},\{b\},\{c,i\},\{d\},\{e\},\boxed{\{f,g,h\}}\}$$

Passo 3: para cada aresta ordenada, faça

E': (g,h); (c,i); (f,g); $\boxed{(a,b)}$; (c,f); (g,i); (c,d); (h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)

a **e** ***b*** pertencem a mesma árvore na floresta?



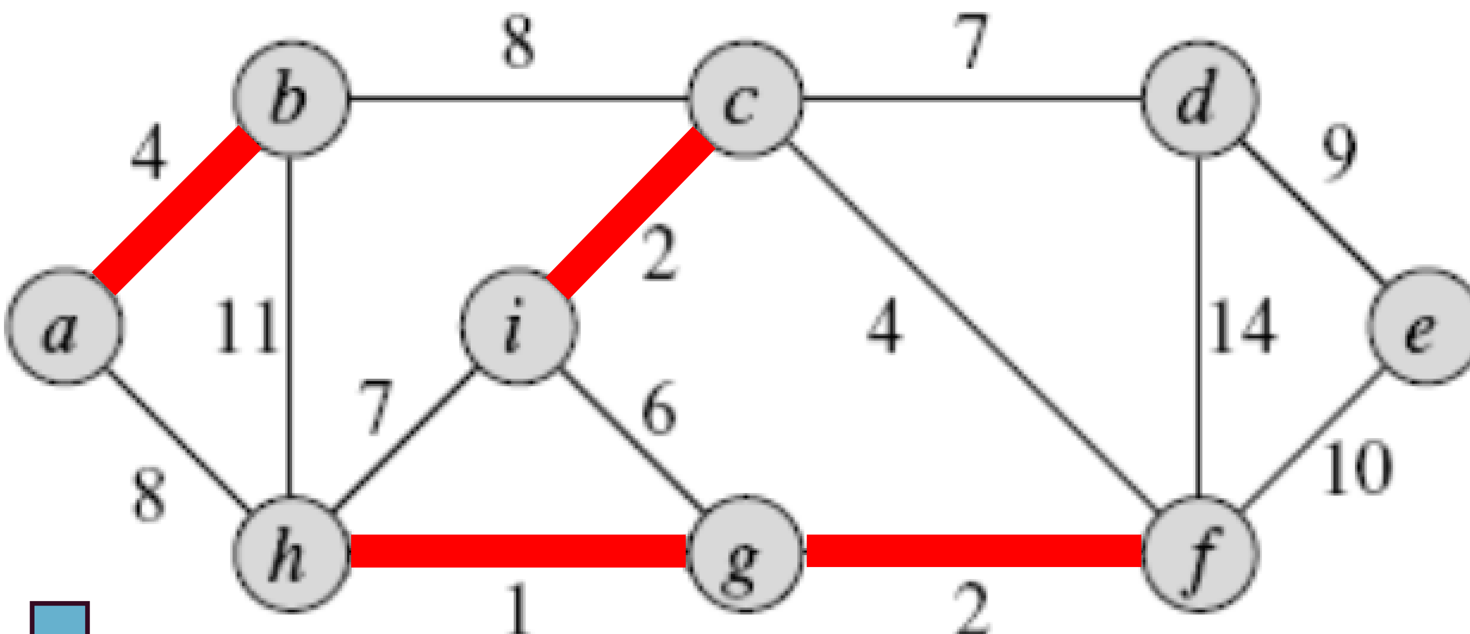
$$\{\{a\},\{b\},\{c,i\},\{d\},\{e\},\{f,g,h\}\}$$

AGM – Algoritmo de Kruskal

Passo 3: para cada aresta ordenada, faça

$$E': (g,h); (c,i); (f,g); \boxed{(a,b)}; (c,f); (g,i); (c,d);$$
$$(h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)$$

Não. Então união das árvores de ***a*** e ***b***; e adição na AGM



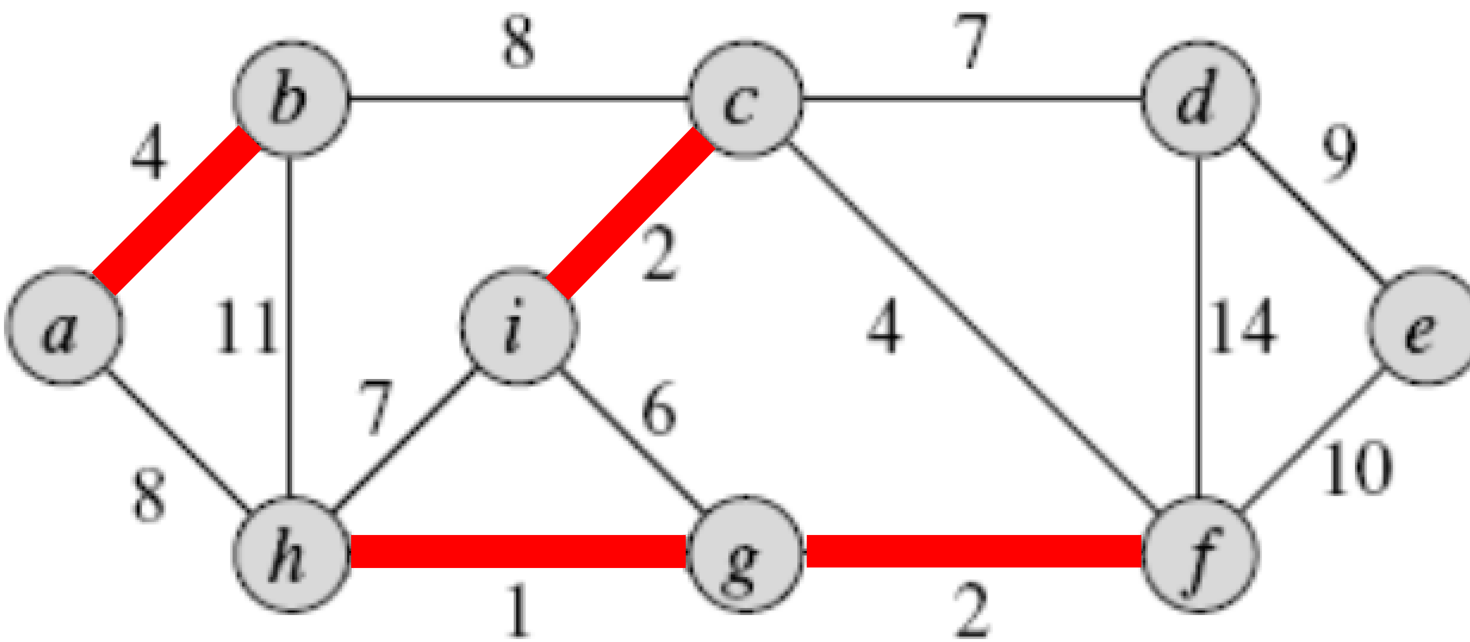
$$\{\boxed{\{a,b\}}, \{c,i\}, \{d\}, \{e\}, \{f,g,h\}\}$$

AGM – Algoritmo de Kruskal

Passo 3: para cada aresta ordenada, faça

$$E': (g,h); (c,i); (f,g); (a,b); \boxed{(c,f)}; (g,i); (c,d);$$
$$(h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)$$

c e f pertencem a mesma árvore na floresta?

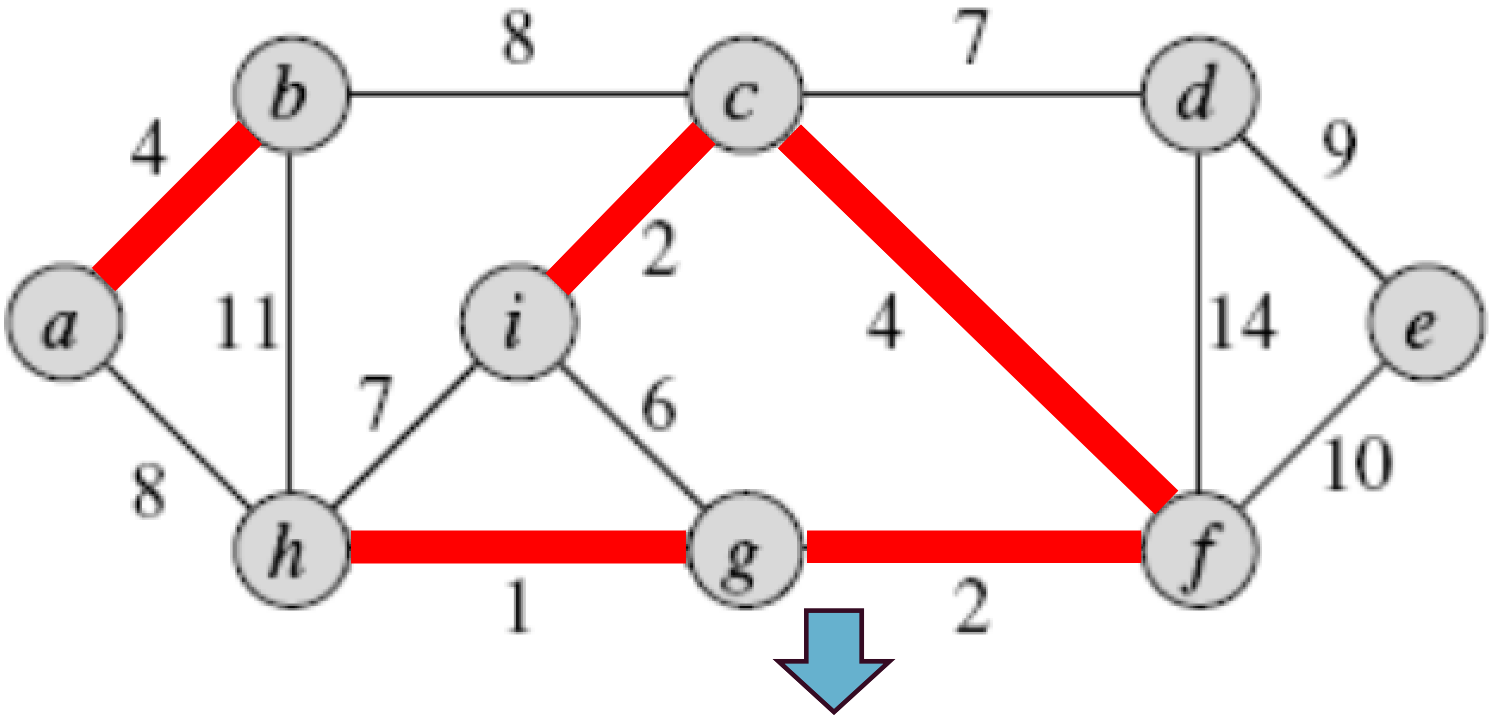


$$\{\{a,b\},\{c,i\},\{d\},\{e\},\{f,g,h\}\}$$

Passo 3: para cada aresta ordenada, faça

E': (g,h); (c,i); (f,g); (a,b); $\boxed{(c,f)}$; (g,i); (c,d); (h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)

Não. Então união das árvores de ***c*** e ***f***; e adição na AGM

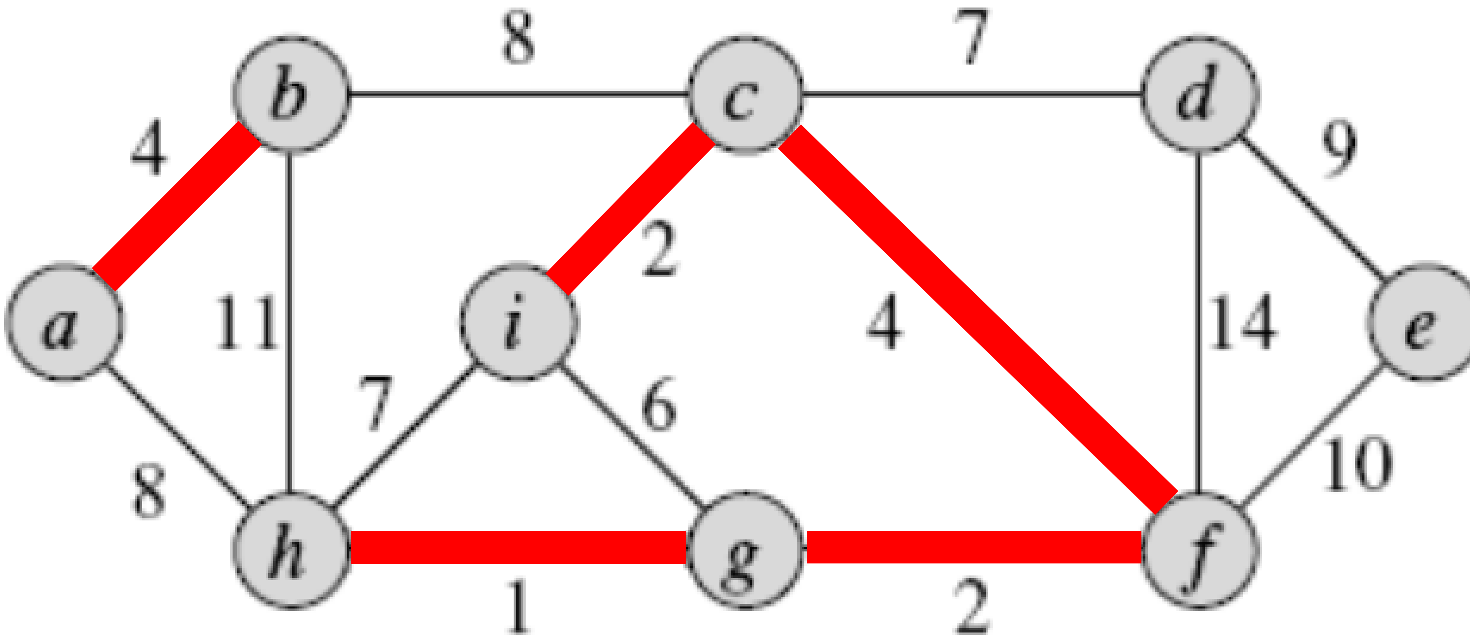


$$\{\{a,b\},\{d\},\{e\},\boxed{\{c,f,g,h,i\}}\}$$

Passo 3: para cada aresta ordenada, faça

E': (g,h); (c,i); (f,g); (a,b); (c,f); $\boxed{(g,i)}$; (c,d); (h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)

g e ***i*** pertencem a mesma árvore na floresta?



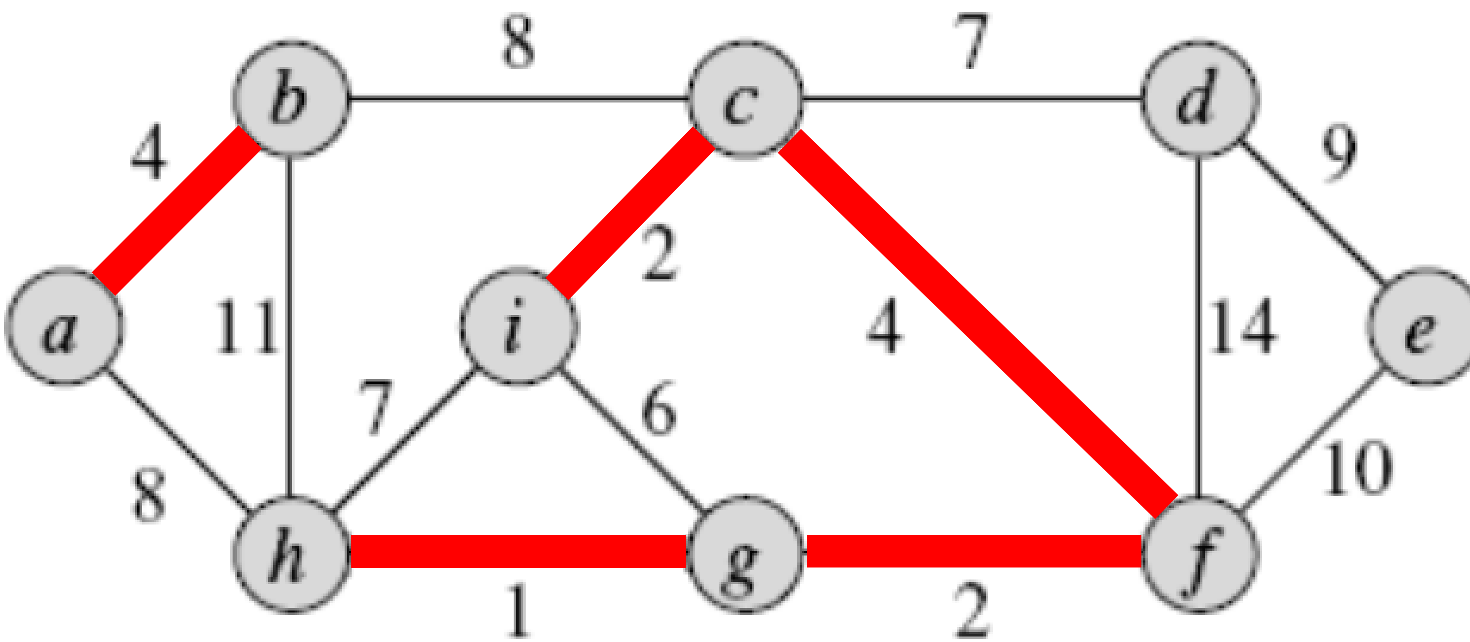
$$\{\{a,b\},\{d\},\{e\},\{c,f,g,h,i\}\}$$

AGM – Algoritmo de Kruskal

Passo 3: para cada aresta ordenada, faça

$$E': (g,h); (c,i); (f,g); (a,b); (c,f); \cancel{(g,i)}; (c,d);$$
$$(h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)$$

Sim. Logo ***g*** e ***i*** irá fechar um ciclo,então remove-se (***g***,***i***)

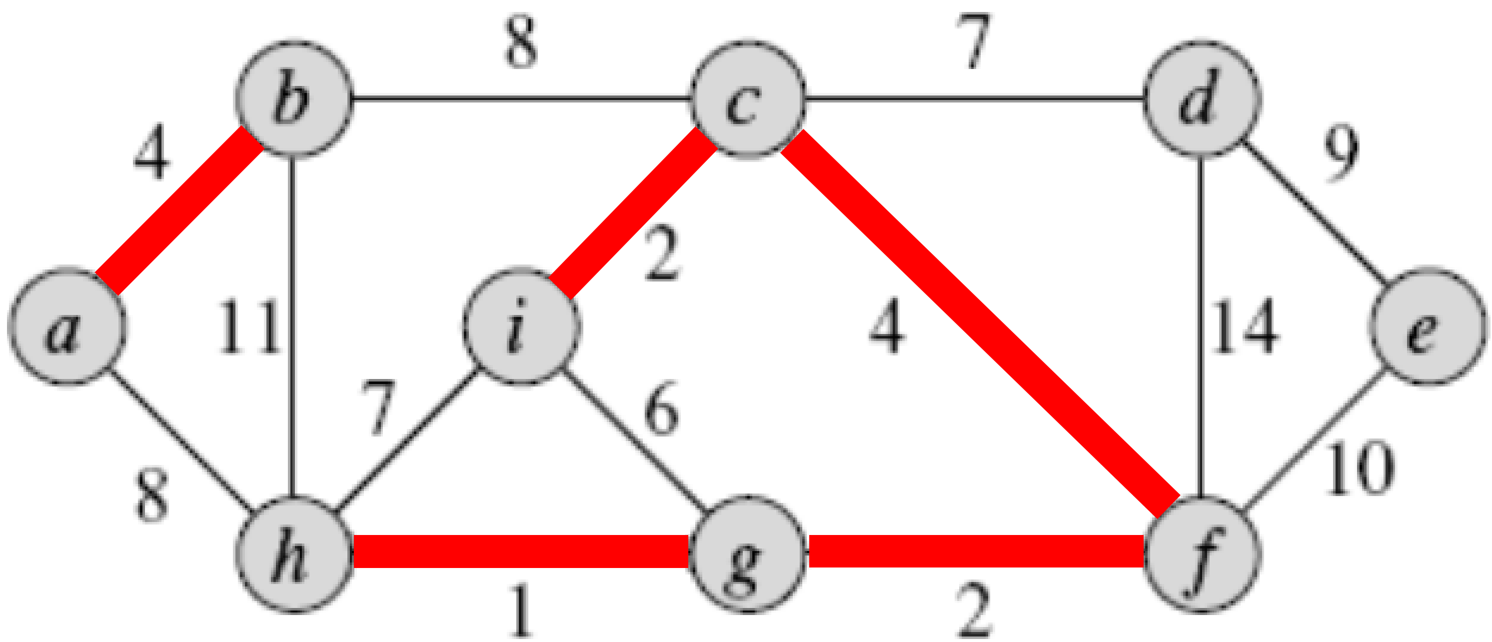


$$\{\{a,b\},\{d\},\{e\},\{c,f,g,h,i\}\}$$

Passo 3: para cada aresta ordenada, faça

E': $(g,h); (c,i); (f,g); (a,b); (c,f); \cancel{(g,i)}; \boxed{(c,d)};$
$(h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)$

c e ***d*** pertencem a mesma árvore na floresta?

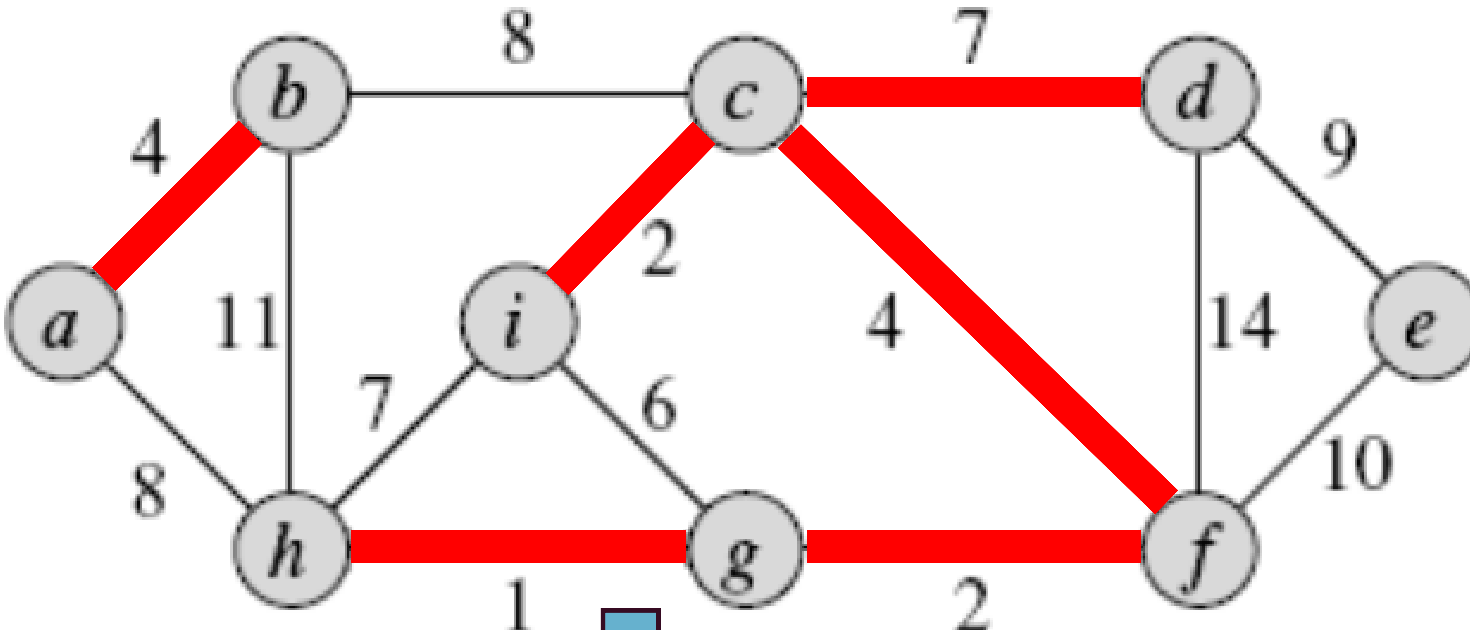


$$\{\{a,b\},\{d\},\{e\},\{c,f,g,h,i\}\}$$

Passo 3: para cada aresta ordenada, faça

$$E': (g,h); (c,i); (f,g); (a,b); (c,f); \cancel{(g,i)}; \boxed{(c,d)}; (h,i); (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)$$

Não. Então união das árvores de ***c*** e ***d***; e adição na AGM



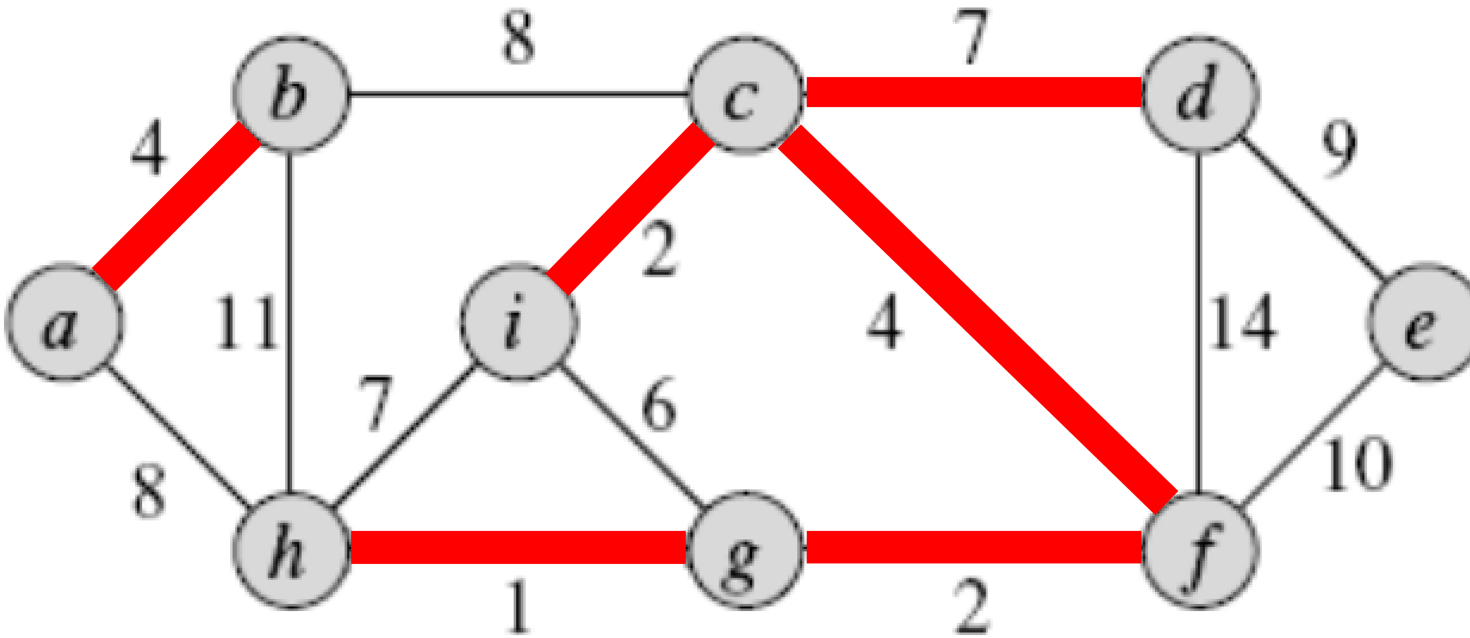
$$\{\{a,b\},\{e\},\boxed{\{c,d,f,g,h,i\}}\}$$

AGM – Algoritmo de Kruskal

Passo 3: para cada aresta ordenada, faça

$$E': (g,h); (c,i); (f,g); (a,b); (c,f); \cancel{(g,i)}; (c,d);$$
$$\boxed{(h,i)}; (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)$$

h e ***i*** pertencem a mesma árvore na floresta?

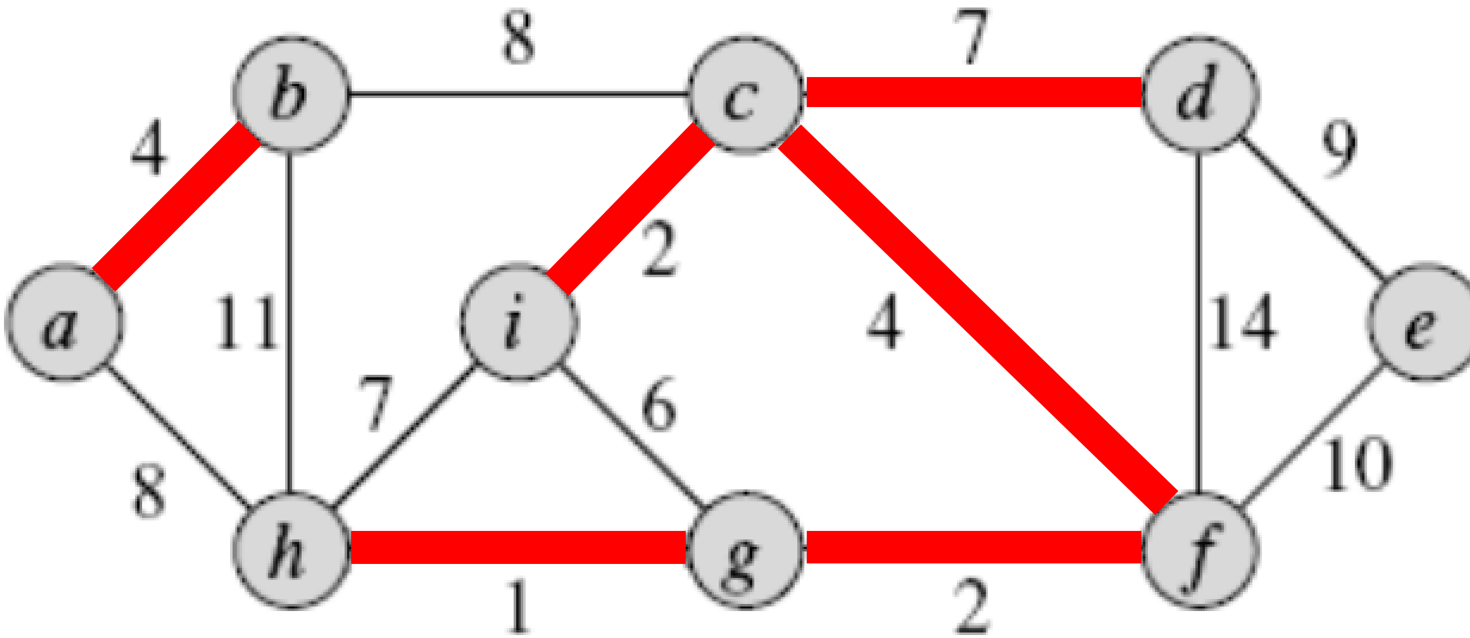


$$\{\{a,b\},\{e\},\{c,d,f,g,h,i\}\}$$

Passo 3: para cada aresta ordenada, faça

E': (g,h); (c,i); (f,g); (a,b); (c,f); $\cancel{(g,i)}$; (c,d); $\boxed{\cancel{(h,i)}}$; (a,h); (b,c); (d,e); (e,f); (b,h); (d,f)

Sim. Logo ***h*** e ***i*** irá fechar um ciclo,então remove-se (***h,i***)

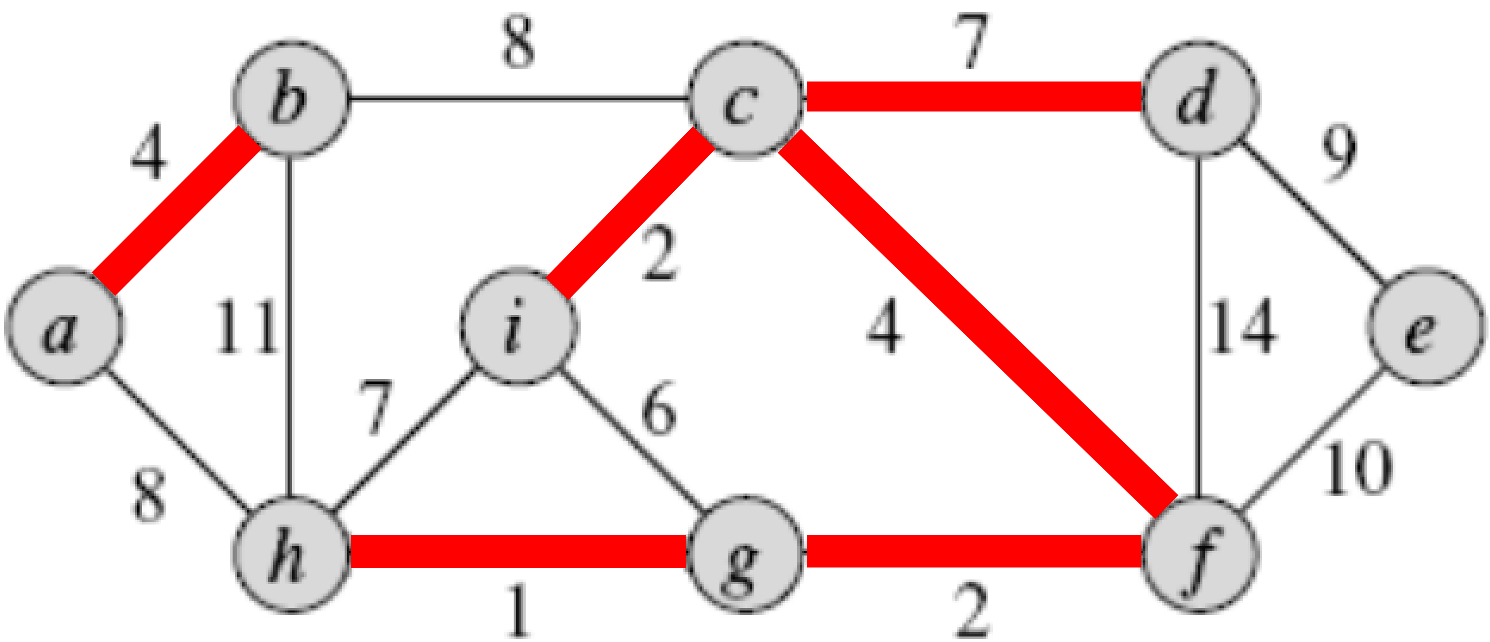


$$\{\{a,b\},\{e\},\{c,d,f,g,h,i\}\}$$

Passo 3: para cada aresta ordenada, faça

E': $(g,h); (c,i); (f,g); (a,b); (c,f); \cancel{(g,i)}; (c,d);$
$\cancel{(h,i)}; \boxed{(a,h)}; (b,c); (d,e); (e,f); (b,h); (d,f)$

a e ***h*** pertencem a mesma árvore na floresta?

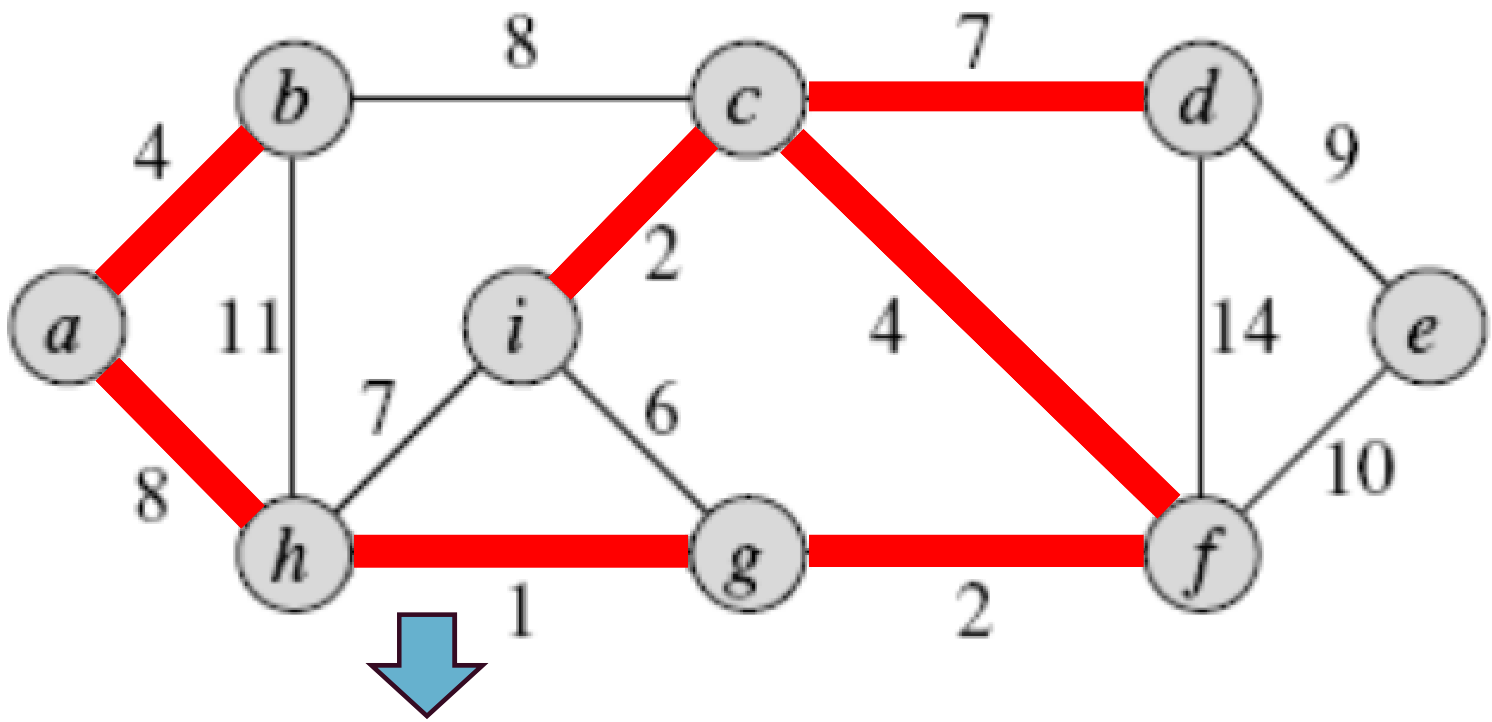


$$\{\{a,b\},\{e\},\{c,d,f,g,h,i\}\}$$

Passo 3: para cada aresta ordenada, faça

E': (g,h); (c,i); (f,g); (a,b); (c,f); $\cancel{(g,i)}$; (c,d); $\cancel{(h,i)}$; $\boxed{(a,h)}$; (b,c); (d,e); (e,f); (b,h); (d,f)

Não. Então união das árvores de ***a*** e ***h***; e adição na AGM

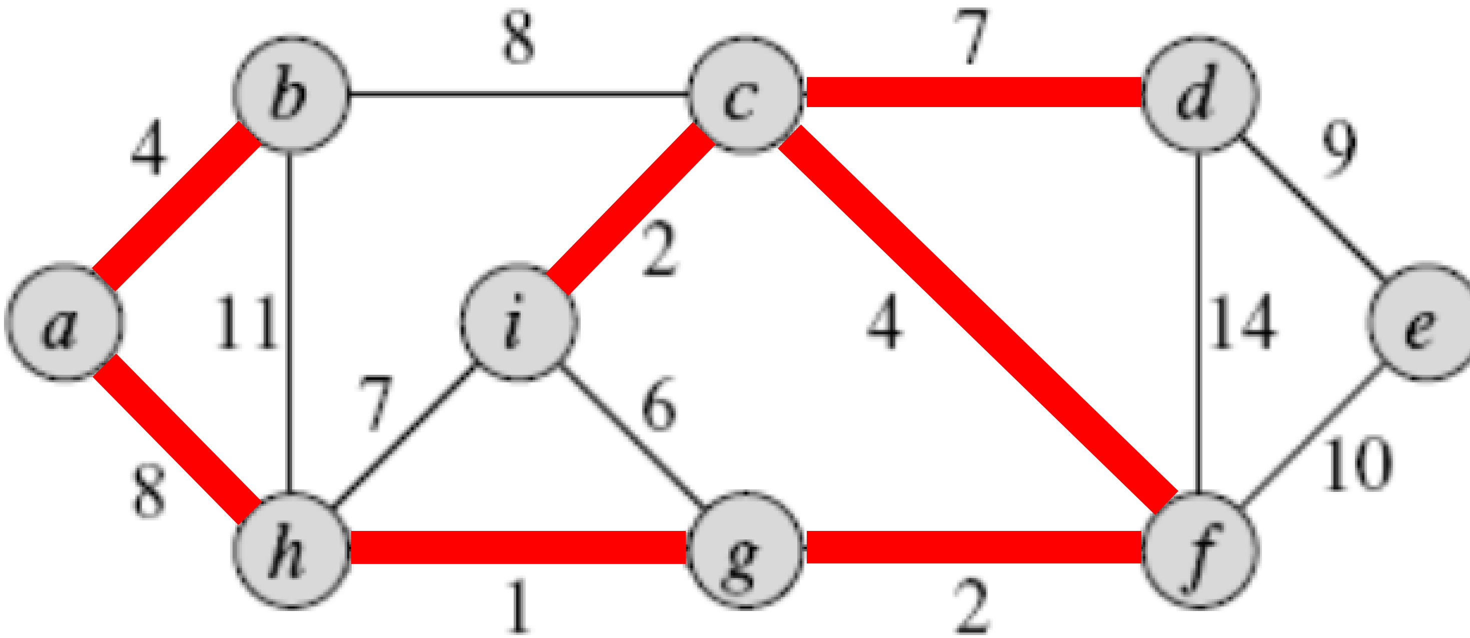


$\{\{e\}, \boxed{\{a, b, c, d, f, g, h, i\}}\}$

Passo 3: para cada aresta ordenada, faça

E': $(g,h); (c,i); (f,g); (a,b); (c,f); \cancel{(g,i)}; (c,d);$
$\cancel{(h,i)}; (a,h); \boxed{(b,c)}; (d,e); (e,f); (b,h); (d,f)$

b e ***c*** pertencem a mesma árvore na floresta?



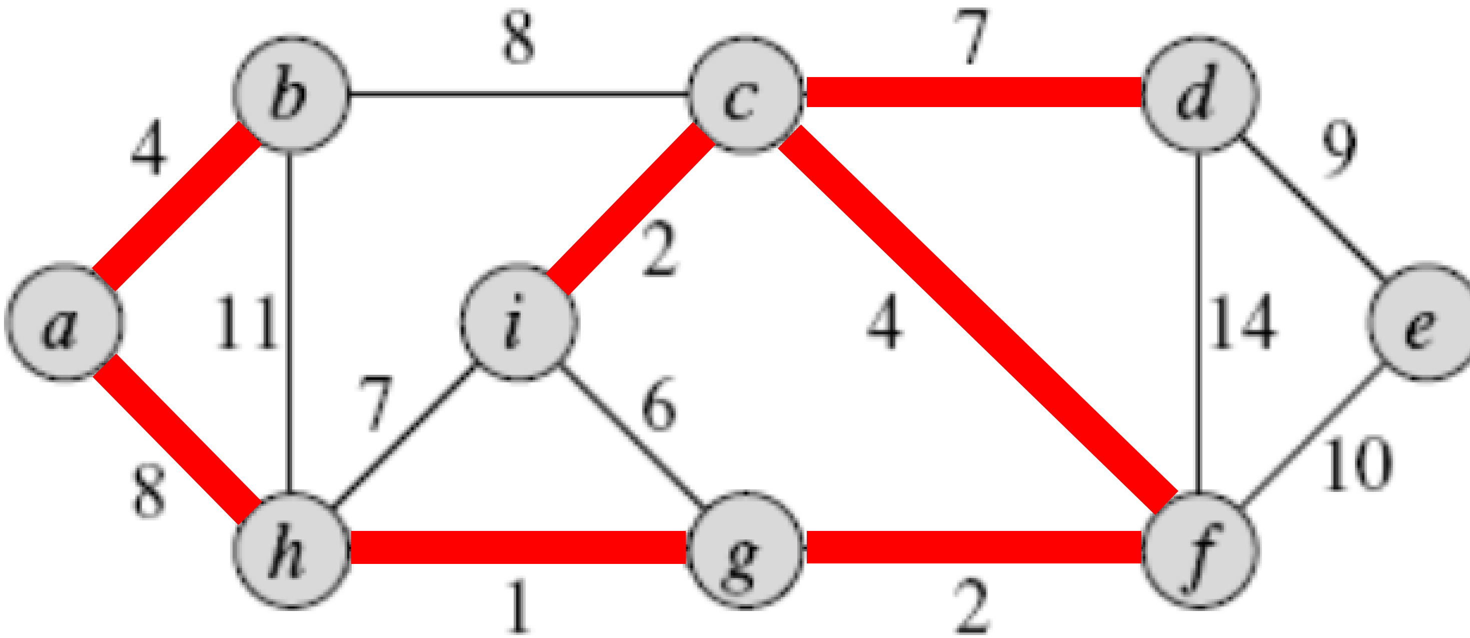
$$\{\{e\}, \{a, b, c, d, f, g, h, i\}\}$$

AGM – Algoritmo de Kruskal

Passo 3: para cada aresta ordenada, faça

$$E': (g,h); (c,i); (f,g); (a,b); (c,f); \cancel{(g,i)}; (c,d);$$
$$\cancel{(h,i)}; (a,h); \boxed{\cancel{(b,c)}}; (d,e); (e,f); (b,h); (d,f)$$

Sim, $(\boldsymbol{b},\boldsymbol{c})$ fecha ciclo. Então descarta

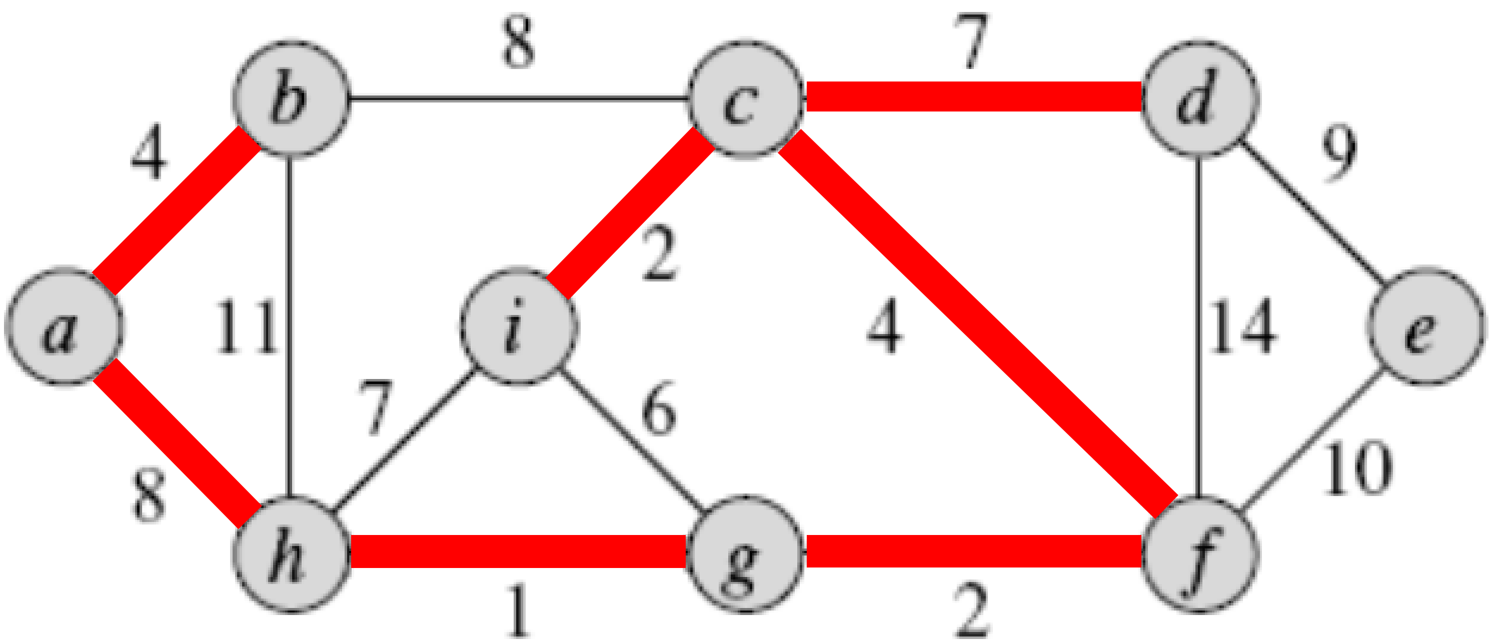


$$\{\{e\},\{a,b,c,d,f,g,h,i\}\}$$

Passo 3: para cada aresta ordenada, faça

E': (g,h); (c,i); (f,g); (a,b); (c,f); $\cancel{(g,i)}$; (c,d);
$\cancel{(h,i)}$; (a,h); $\cancel{(b,c)}$; $\boxed{(d,e)}$; (e,f); (b,h); (d,f)

d e ***e*** pertencem a mesma árvore na floresta?

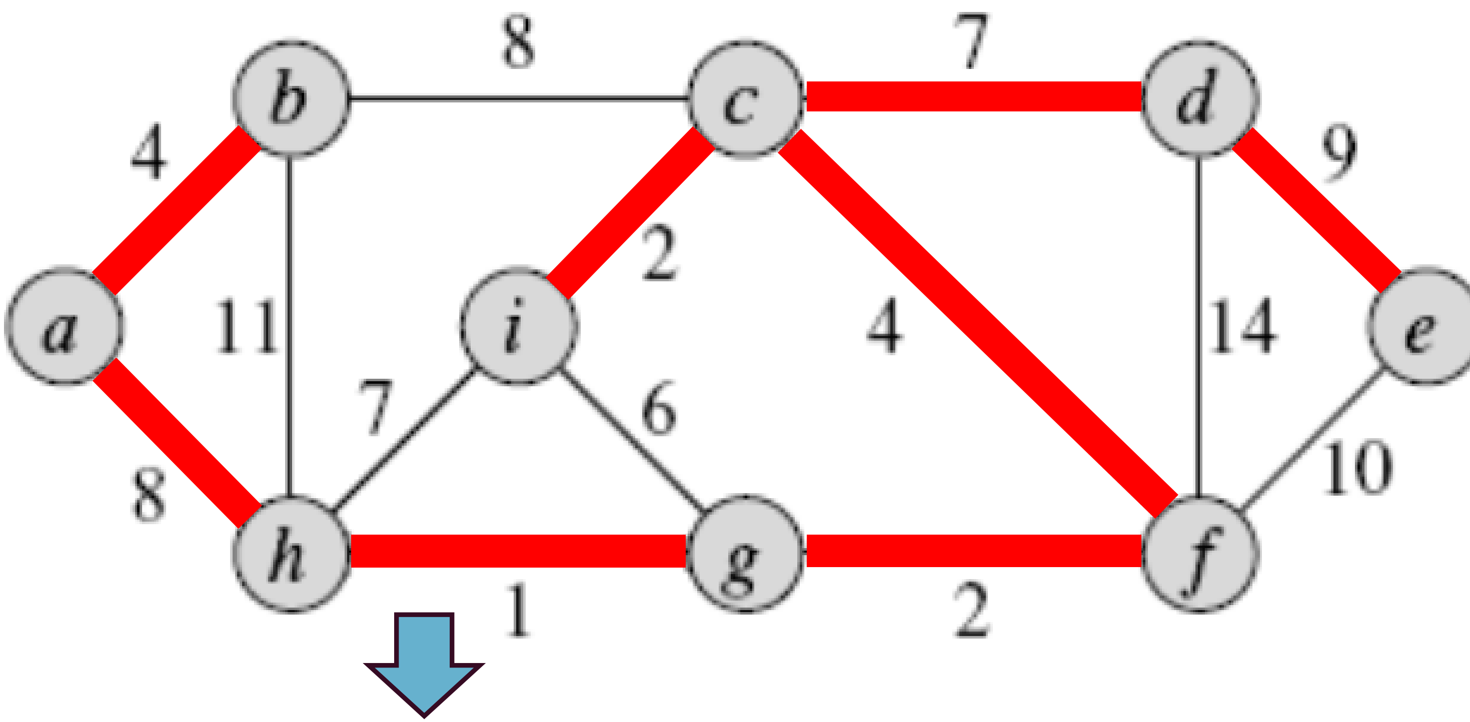


$$\{\{e\},\{a,b,c,d,f,g,h,i\}\}$$

Passo 3: para cada aresta ordenada, faça

E': (g,h); (c,i); (f,g); (a,b); (c,f); $\cancel{(g,i)}$; (c,d); $\cancel{(h,i)}$; (a,h); $\cancel{(b,c)}$; $\boxed{(d,e)}$; (e,f); (b,h); (d,f)

Não. Então união das árvores de $\boldsymbol{d}$ e $\boldsymbol{e}$; e adição na AGM



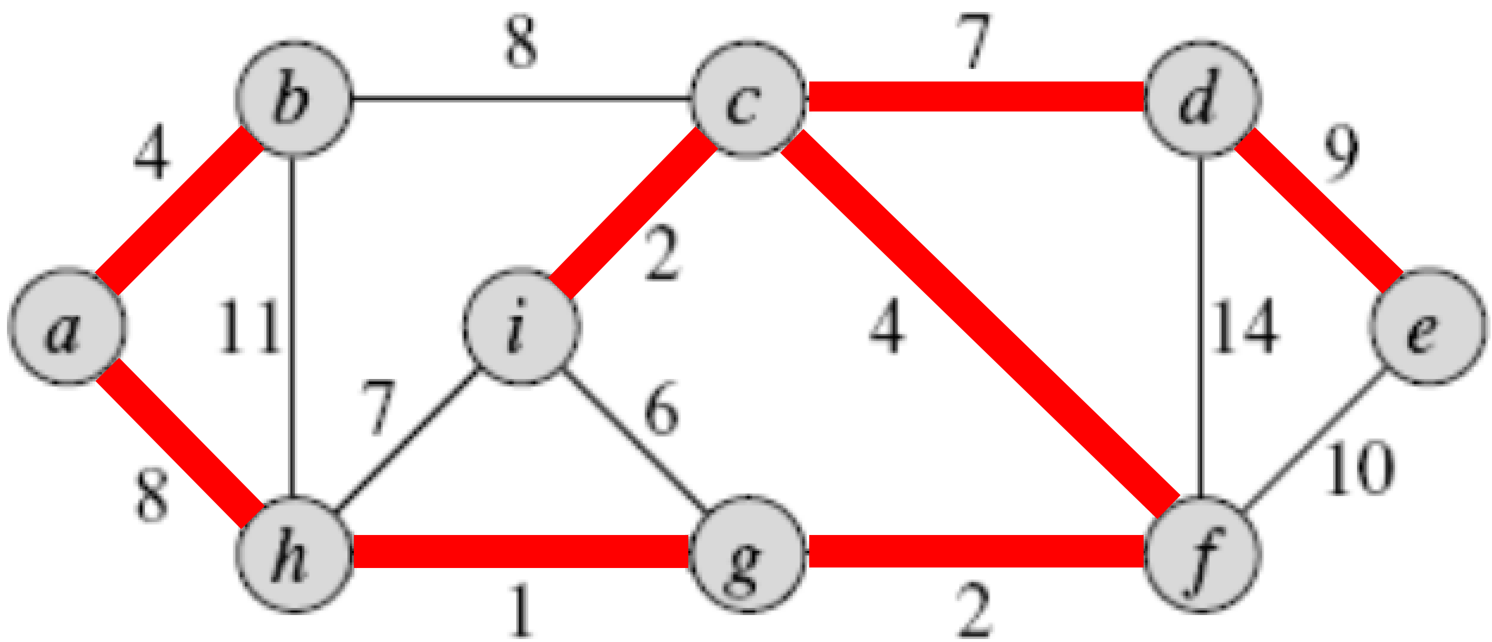
$\{\boxed{\{a,b,c,d,e,f,g,h,i\}}\}$

AGM – Algoritmo de Kruskal

Passo 3: para cada aresta ordenada, faça

E': (g,h); (c,i); (f,g); (a,b); (c,f); $\cancel{(g,i)}$; (c,d); $\cancel{(h,i)}$; (a,h); $\cancel{(b,c)}$; (d,e); $\boxed{(e,f)}$; (b,h); (d,f)

e e ***f*** pertencem a mesma árvore na floresta?



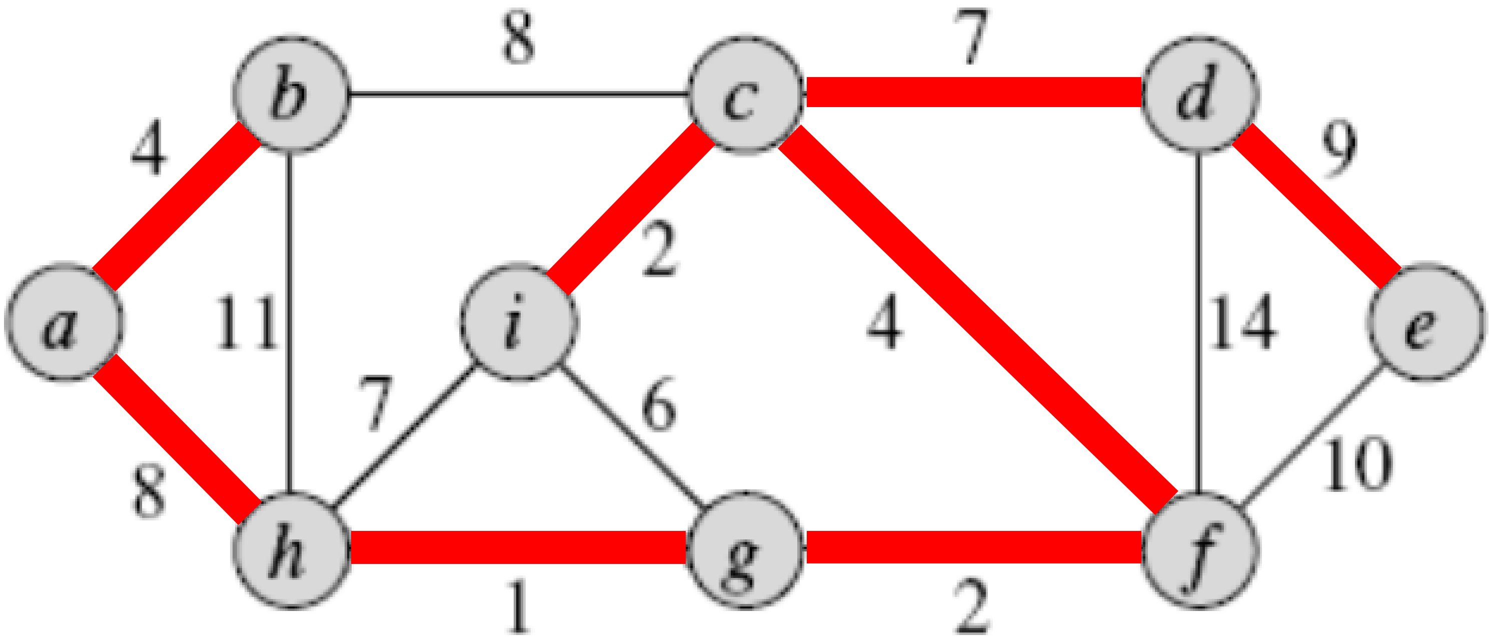
$$\{\{a,b,c,d,e,f,g,h,i\}\}$$

AGM – Algoritmo de Kruskal

Passo 3: para cada aresta ordenada, faça

$$E': (g,h); (c,i); (f,g); (a,b); (c,f); \cancel{(g,i)}; (c,d);$$
$$\cancel{(h,i)}; (a,h); \cancel{(b,c)}; (d,e); \boxed{\cancel{(e,f)}}; (b,h); (d,f)$$

Sim, (***e***,***f***) fecha ciclo. Então descarta



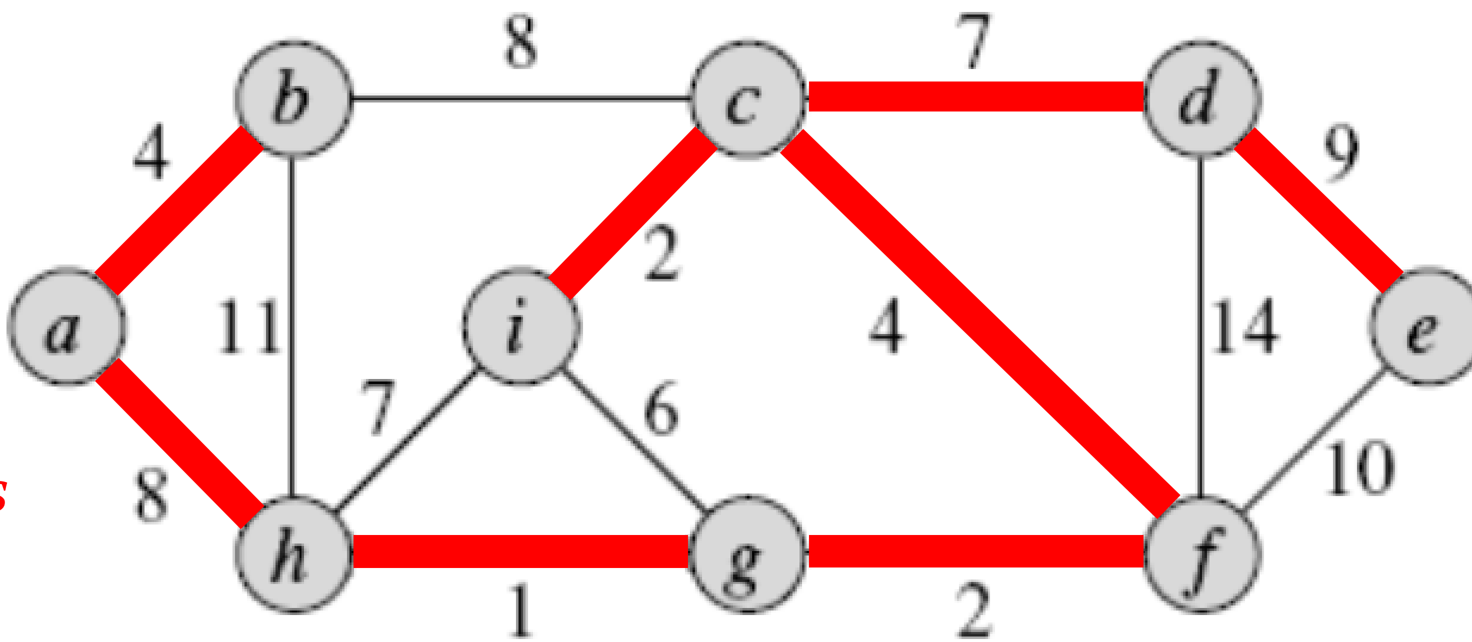
$$\{\{a,b,c,d,e,f,g,h,i\}\}$$

AGM – Algoritmo de Kruskal

Passo 3: para cada aresta ordenada, faça

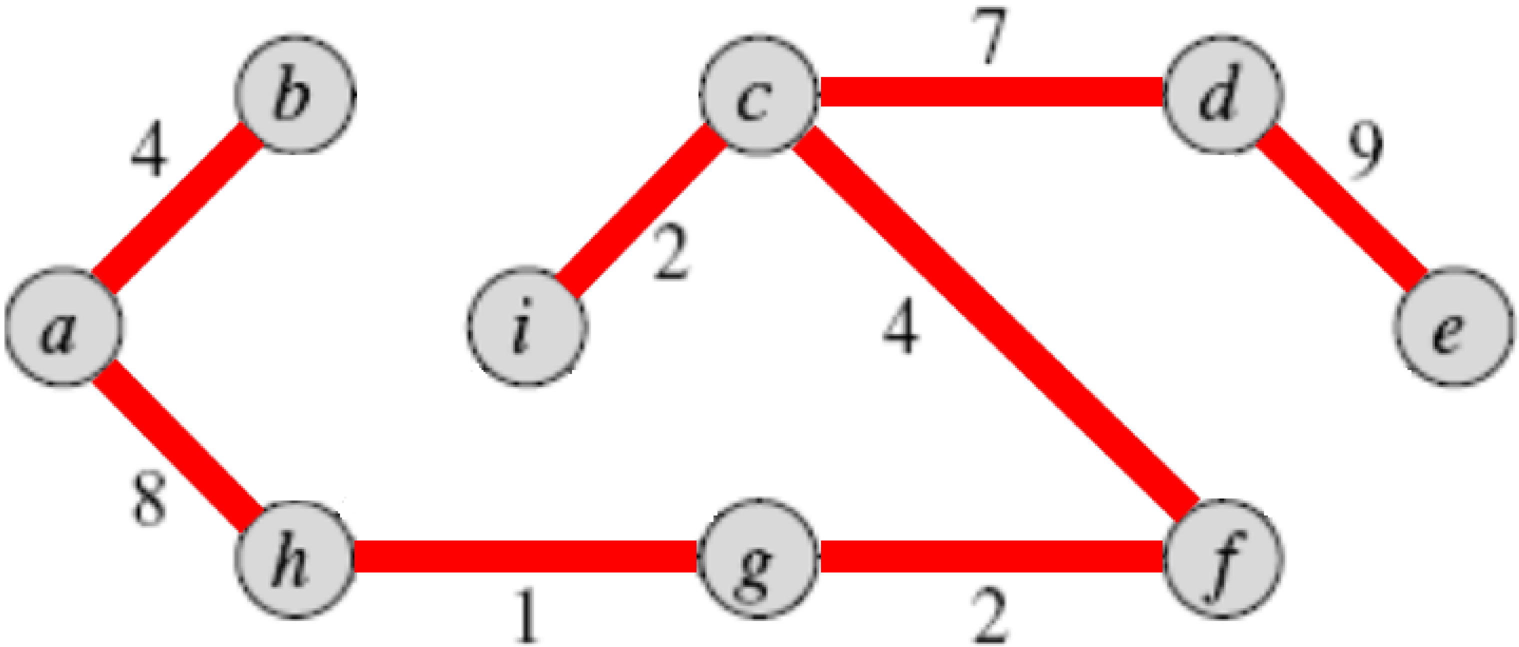
E': (g,h); (c,i); (f,g); (a,b); (c,f); $\cancel{(g,i)}$; (c,d); $\cancel{(h,i)}$; (a,h); $\cancel{(b,c)}$; (d,e); $\cancel{(e,f)}$; $\boxed{(b,h)}$; $\boxed{(d,f)}$

As demais arestas fecham ciclos, logo descartamos



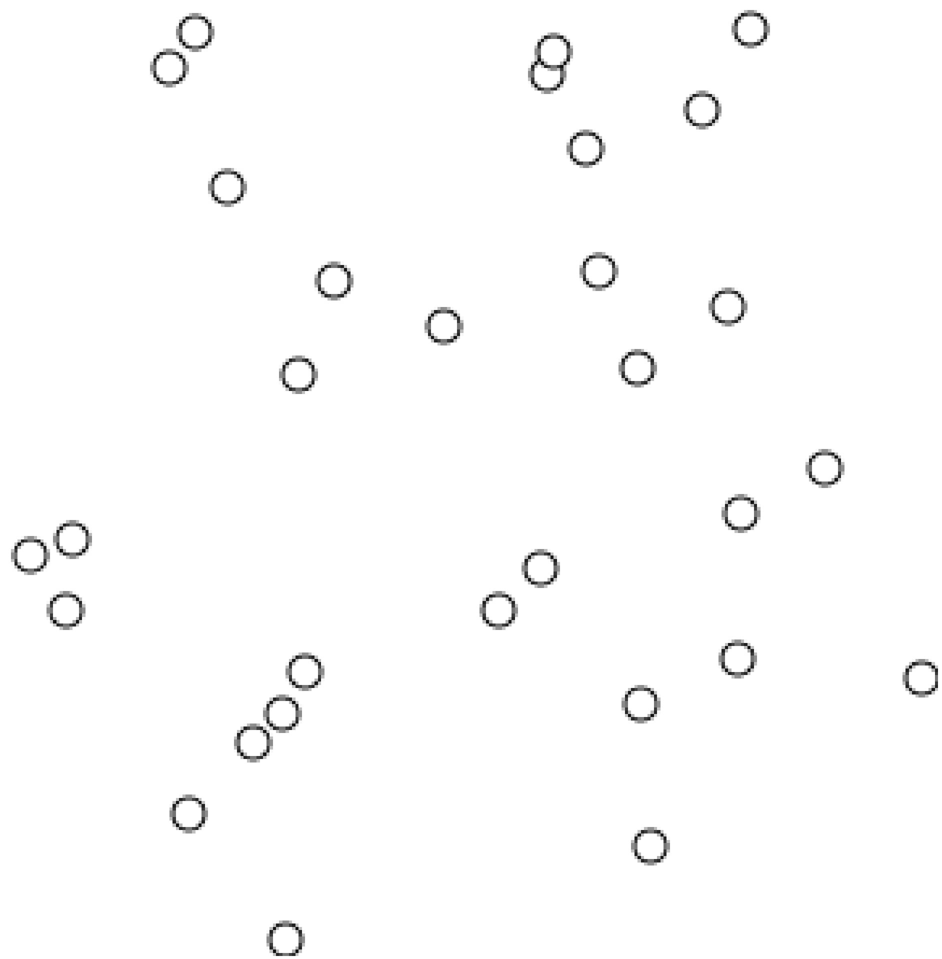
$$\{\{a,b,c,d,e,f,g,h,i\}\}$$

O conjunto X (arestas da AGM) foi gerado ao longo da execução do algoritmo, onde as arestas não descartadas de E' foram sendo adicionadas à AGM.



E': (g,h); (c,i); (f,g); (a,b); (c,f); $\cancel{(g,i)}$; (c,d); $\cancel{(h,i)}$; (a,h); $\cancel{(b,c)}$; (d,e); $\cancel{(e,f)}$; $\boxed{(b,h)}$; $\boxed{(d,f)}$

Algoritmo de Kruskal - Demo



Demonstração do algoritmo de Kruskal em um gráfico completo com pesos baseados na distância euclidiana (https://en.wikipedia.org/wiki/Kruskal%27s_algorithm#/media/File:KruskalDemo.gif)