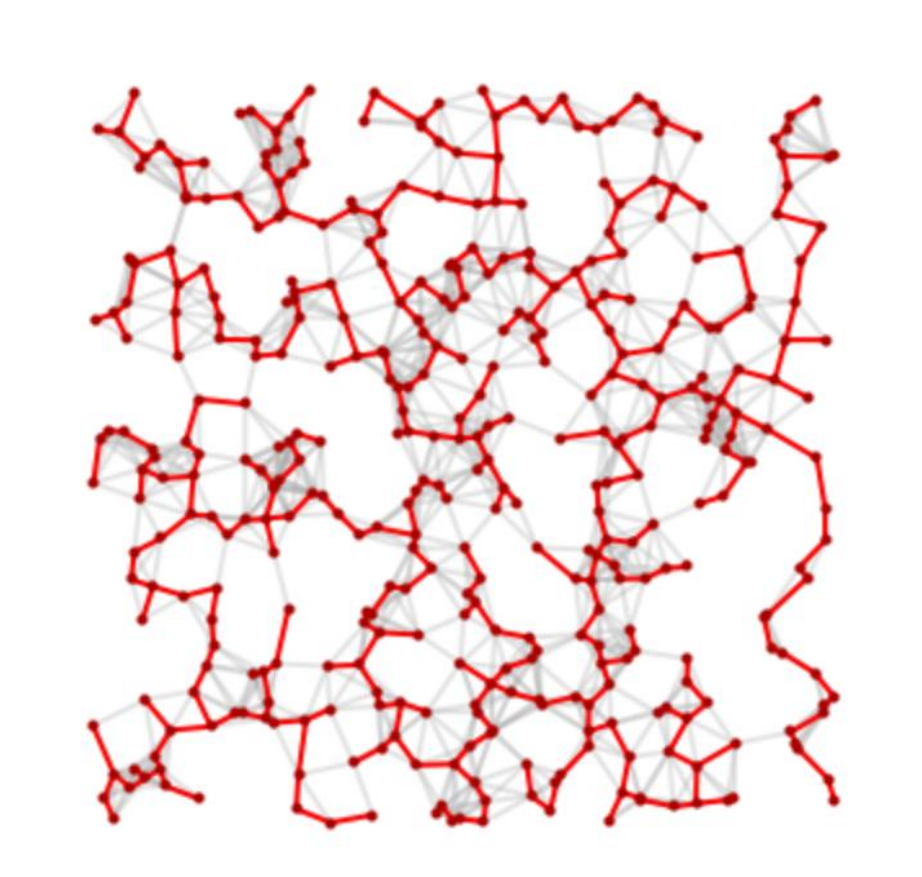
C.E.S.A.R school

TEORIA DOS GRAFOS

Profª Laura Pacifico

2025 | SETEMBRO

Árvore Geradora (Spanning Tree)

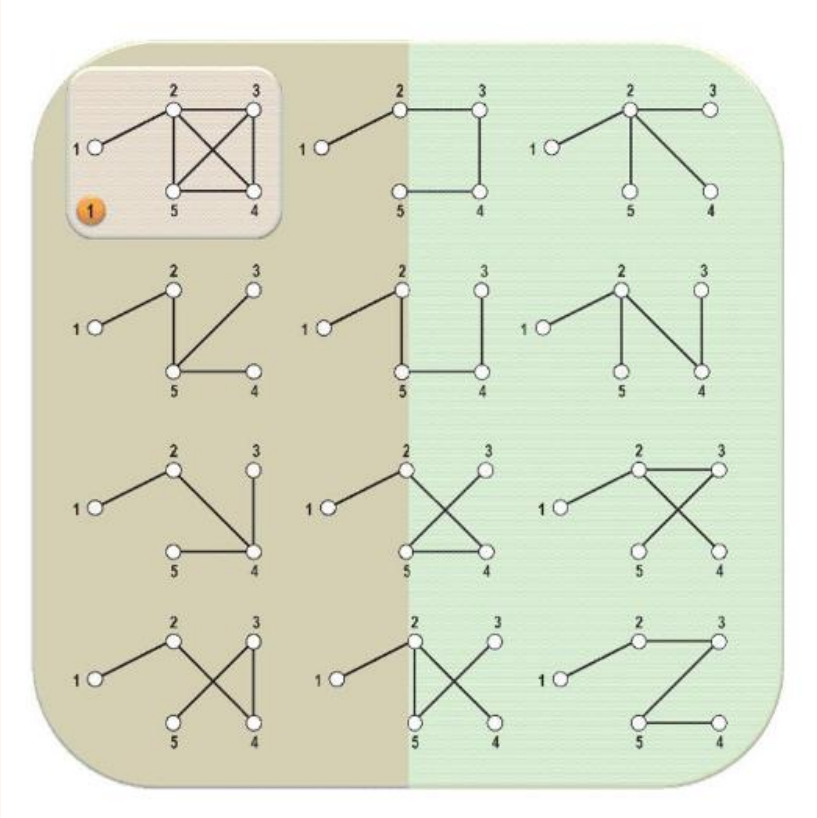


Árvore Geradora (Spanning Tree)

Uma árvore geradora de um grafo G é um subgrafo gerador **conexo** e **acícliclo** (sem ciclos).

A árvore geradora de um Grafo G contém exatamente os mesmo vértices de G, e |N|-1 arestas (onde |N| é a Ordem de G).

A figura ao lado representa 11 diferentes árvores geradoras do grafo em 1.



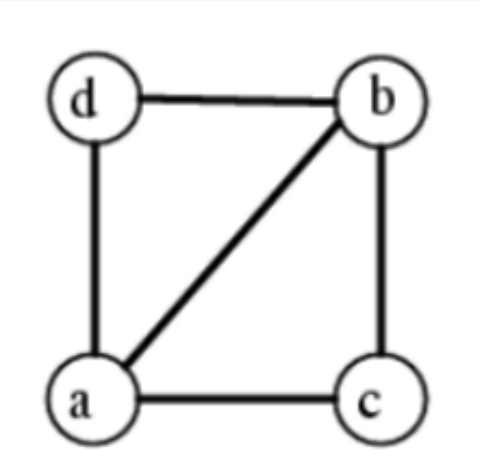
Como obter uma árvore geradora de G?

Procedimento:

1. Se G não possui circuitos, G é sua própria árvore geradora.
2. Se G possui circuitos, retire uma aresta do circuito. O subgrafo resultante é conexo.
3. Se existirem mais circuitos, repita a operação até retirar uma aresta do último circuito do grafo.
4. O subgrafo resultante é conexo, sem circuitos e possui todos os vértices de G. Portanto é uma árvore geradora de G.

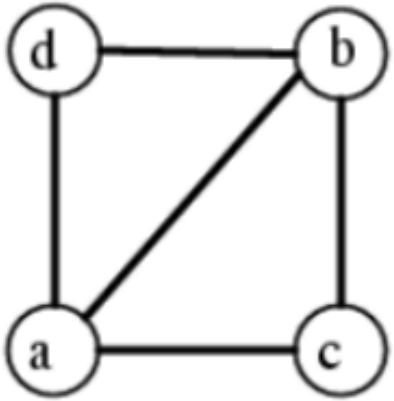
Exemplo

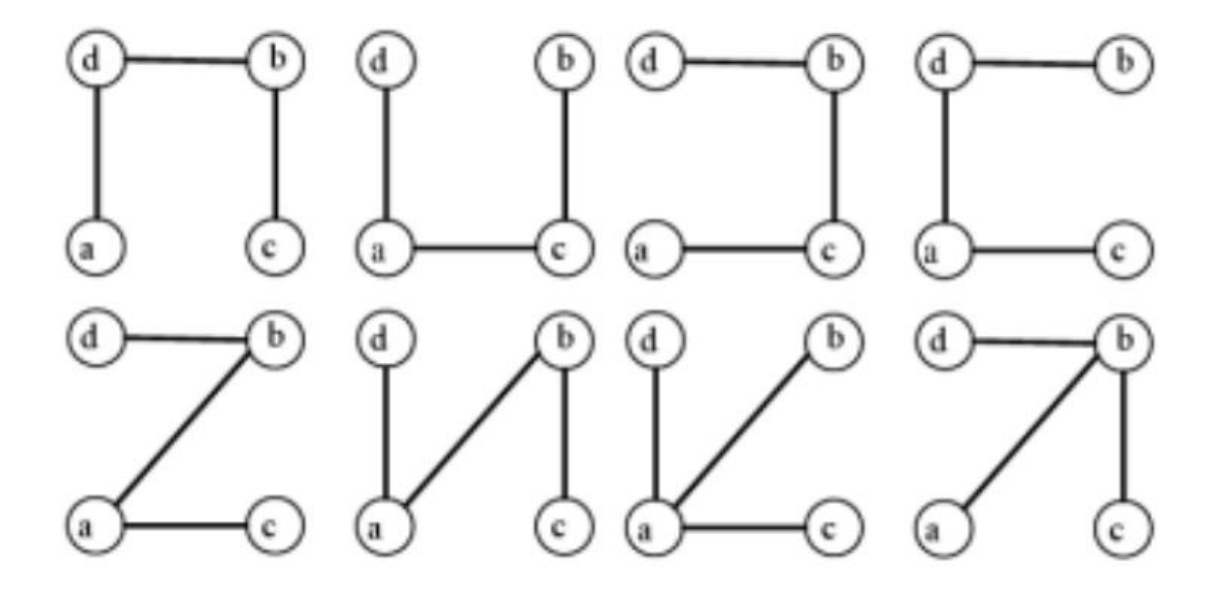
Liste 8 árvores geradoras do grafo abaixo



Exemplo

Liste todas as 8 árvores geradoras do grafo abaixo





Árvore Geradora Mínima (AGM)

A árvore geradora mínima (T_{min}) é a árvore geradora de menor custo, dentre todas as possíveis em G.

O custo de uma árvore geradora T de um grafo ponderado G é dado pelo somatório dos custos das arestas de T.

T_{max} é a árvore geradora de maior custo em G.

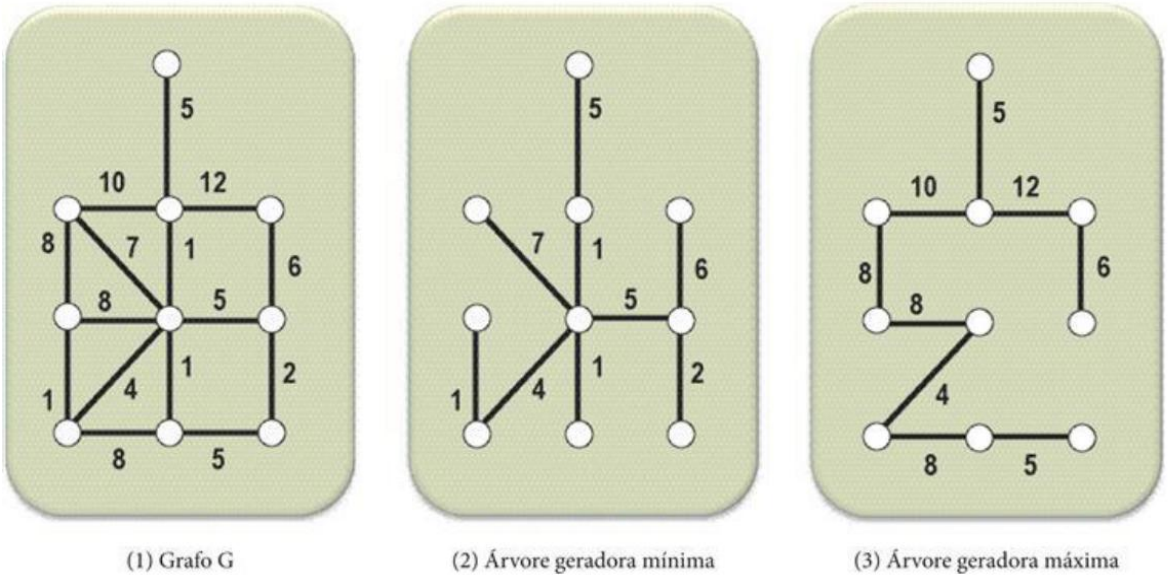
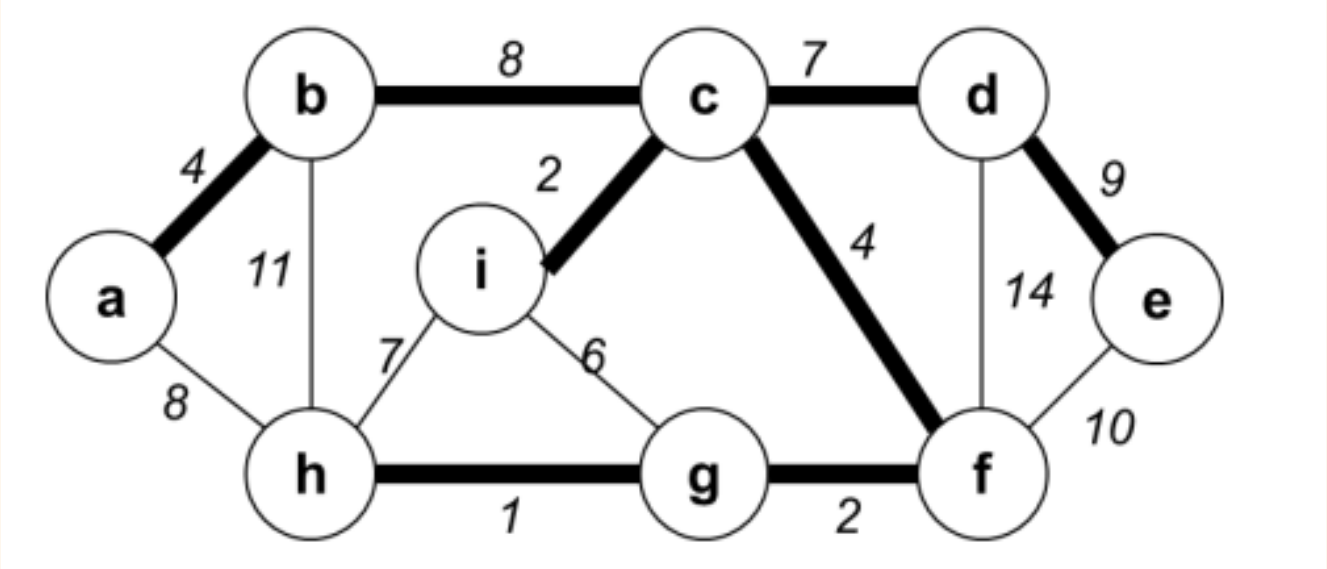
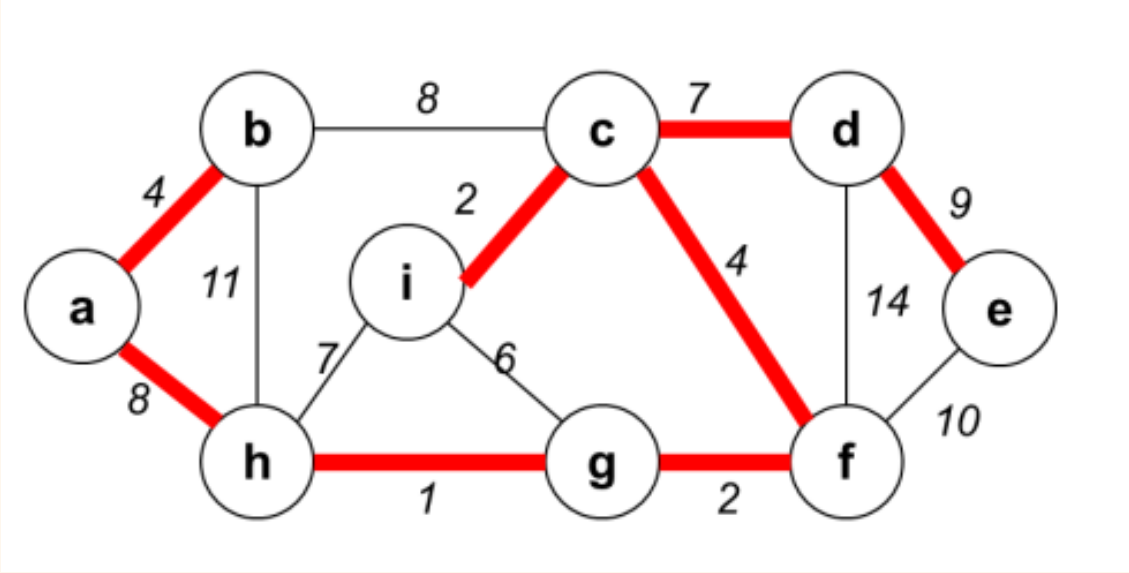


FIGURA 3.10 Árvore geradora mínima e máxima

Só existe uma única árvore geradora mínima para um grafo conexo?



Só existe uma única árvore geradora mínima para um grafo conexo?



O Teorema da Matriz-Árvore.

Seja G um grafo e L(G) a sua matriz laplaciana. Seja u um vértice arbitrário de G. Seja L(G)u o menor principal de L(G) retirando-se a linha e a coluna correspondente ao vértice u, então det(L(G)u) é igual ao número de árvores geradoras de G.

Onde o Laplaciano de um grafo, é uma representação matricial calculada a partir da matriz de adjacência (A) e da matriz de graus (D) através da fórmula L = D - A

Algoritmos de Solução de Árvore Geradora Mínima

- Principais trabalhos publicados na tabela ao lado
- Veremos em detalhe:
  - Kruskal (1956)
  - Prim (1957)

Trabalho	Técnica Utilizada	Complexidade
Borůvka (1926)	União de conjuntos disjuntos	$O(m \log n)$
Kruskal (1956)	União de conjuntos disjuntos e heapsort	$O(m \log n)$
Prim (1957)	Seleção de arestas	$O(n^2)$
Johnson (1975)	Algoritmo de Prim implementado com d—Heap	$O(nd \log_d n + m \log_d n)$
Yao (1975)	Algoritmo de Borůvka implementado utilizando heaps e seleção	$O(m \log \log n)$
Cheriton & Tarjan (1976)	Algoritmo de Borůvka com fila duplamente ligada, heap e união de conjuntos disjuntos	$O(m \log \log n)$
Gabow et al. (1986)	Heap de Fibonacci com pacotes e união de conjuntos disjuntos	$O(m \log \beta(m, n))$ (a)
Fredman & Tarjan (1987)	Algoritmo de Prim implementado com heap de Fibonacci	$O(n \log n + m)$
Fredman & Tarjan (1987)	Algoritmo de Borůvka implementado com heap de Fibonacci	$O(m \beta(m, n))$
Karger (1993)	Aleatorização	$O(n \log n + m)$
Karger et al. (1995)	Aleatorização, recursão e verificação de tempo linear	$O(m)$ (b)
Chazelle (1997)	Heap com particularidades de movimentação (soft heap)	$O(m\alpha(m,n) \log\alpha(m,n))$ (c)
Pettie (1999)	Divisão do problema em subproblemas e procedimentos de retração e extensão	$O(m\alpha(m,n))$
Chazelle (2000a)	Aperfeiçoamento da técnica em soft heap	$O(m\alpha(m,n)\log\alpha(m,n))$
Pettie (1999)	Divisão do problema em subproblemas e procedimentos de retração e extensão	$O(m\alpha(m,n))$
Chazelle (2000a)	Aperfeiçoamento da técnica em soft heap	$O(m\alpha(m,n)\log\alpha(m,n))$
Chazelle (2000b)	Aperfeiçoamento da técnica em soft heap	$O(m\alpha(m, n))$
Pettie & Ramachandran (2002)	Aperfeiçoamento do trabalho Pettie (1999)	Entre $\Omega(m)$ e $O(m\alpha(m,n))$ (d)
Pettie & Ramachandran (2008)	Aleatorização	$O(m)$ (b)

Algoritmo de Prim

- Proposto por Robert Clay Prim em 1957.
- *Algoritmo:*
- *Escolha um vértice **S** para iniciar o subgrafo*
  - ***enquanto** houver vértices que não estão no subgrafo*
    - *selecione uma aresta segura (aresta conectada ao subgrafo e com menor custo dentre todas as arestas conectadas so subgrafo)*
    - *insira a aresta segura e seu vértice no subgrafo*

Algoritmo de Prim

A

Prim

Ler $G = (N, M)$ **e** $D = [d_{ij}]$ a matriz de pesos de G

Escolha qualquer vértice $i \in N$

$T \leftarrow \{i\}$

$V \leftarrow N \setminus \{i\}$

$T_{min} \leftarrow \emptyset$

Enquanto $T \neq N$ **Faça**

Encontrar a aresta $(j, k) \in M$ tal que $j \in T, k \in V$ e d_{jk} é mínimo

$T \leftarrow T \cup \{k\}$

$V \leftarrow V \setminus \{k\}$

$T_{Min} \leftarrow T_{Min} \cup (j,k)$

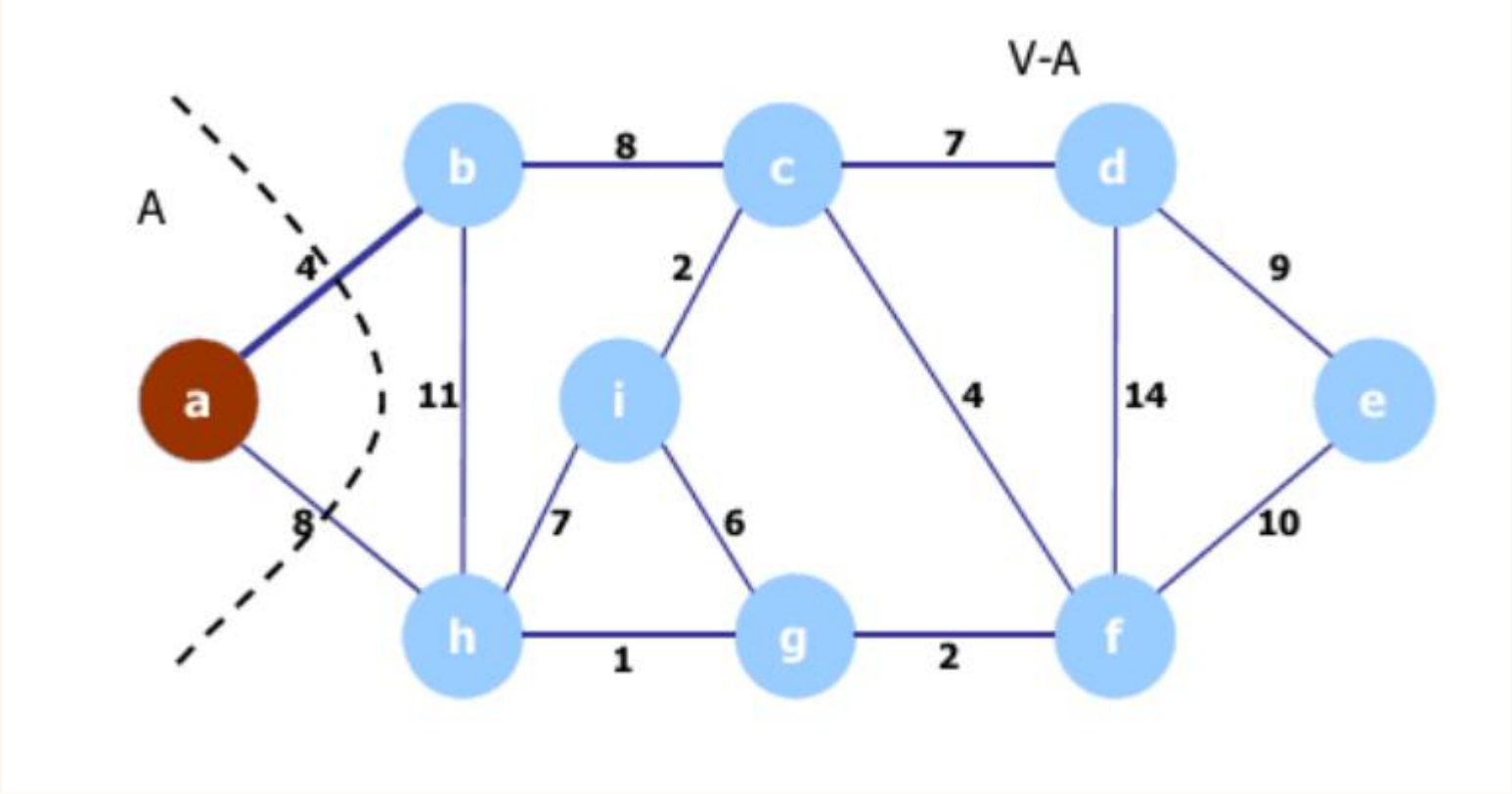
Fim_Enquanto

Escrever T_{Min} {o conjunto das arestas da árvore geradora mínima}

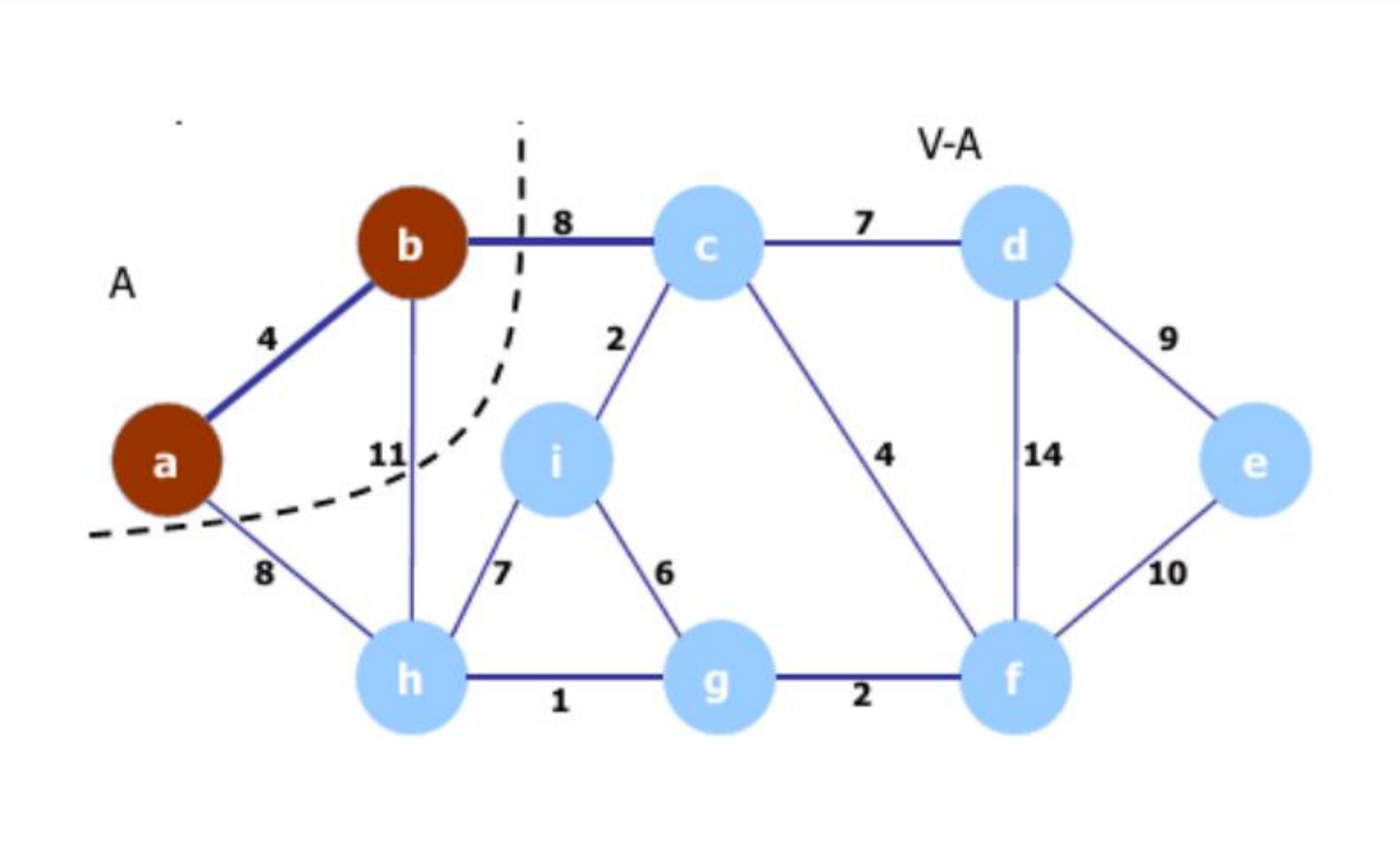
Algoritmo de Prim

1. Escolha qualquer vértice para começar (pode ser aleatório).
2. A partir dele, sempre procure a **aresta mais barata** que conecta um vértice já escolhido com um vértice que ainda não foi escolhido.
3. Inclua esse novo vértice e a aresta no conjunto.
4. Repita até que todos os vértices estejam conectados.

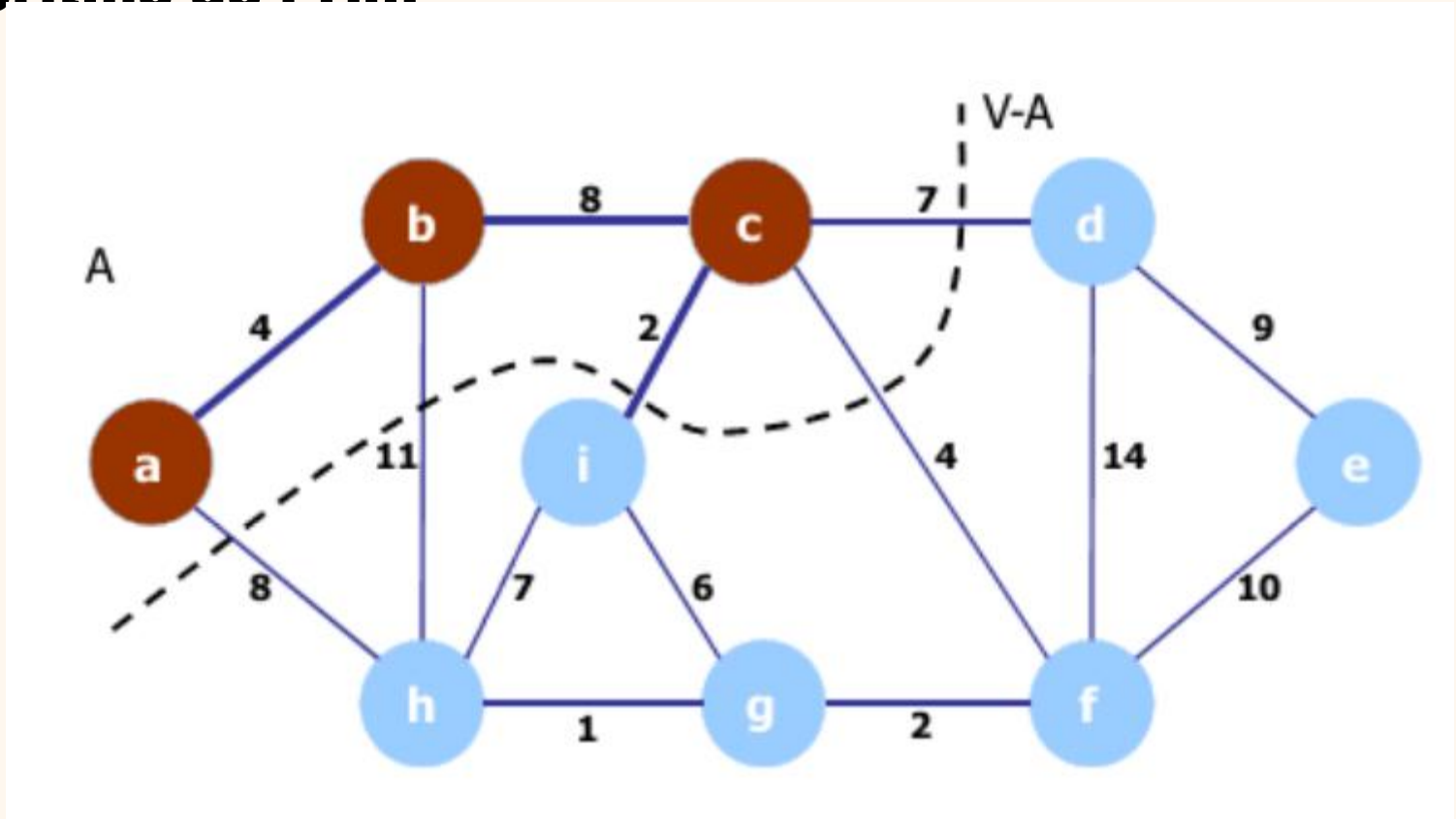
Algoritmo de Prim



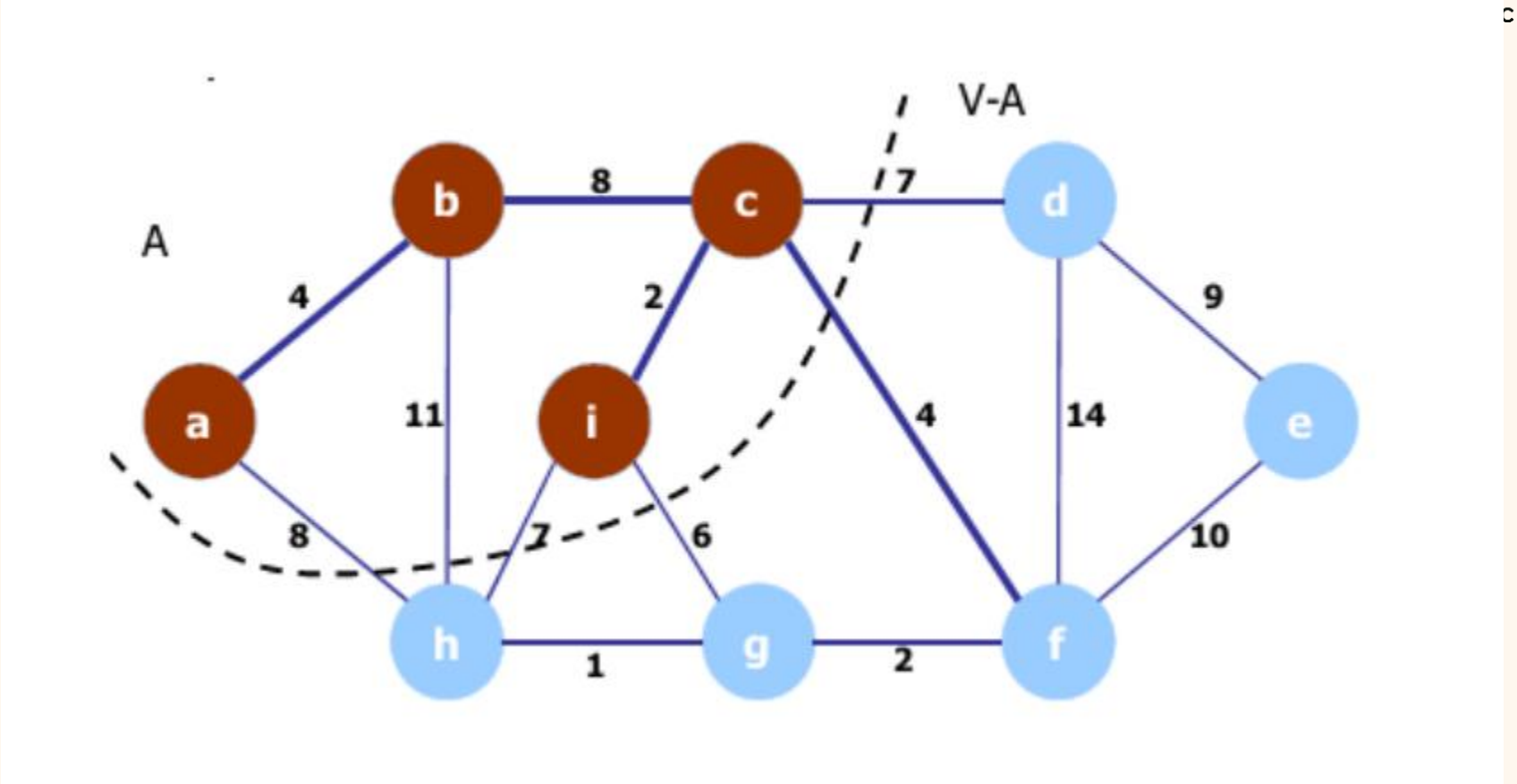
Alg



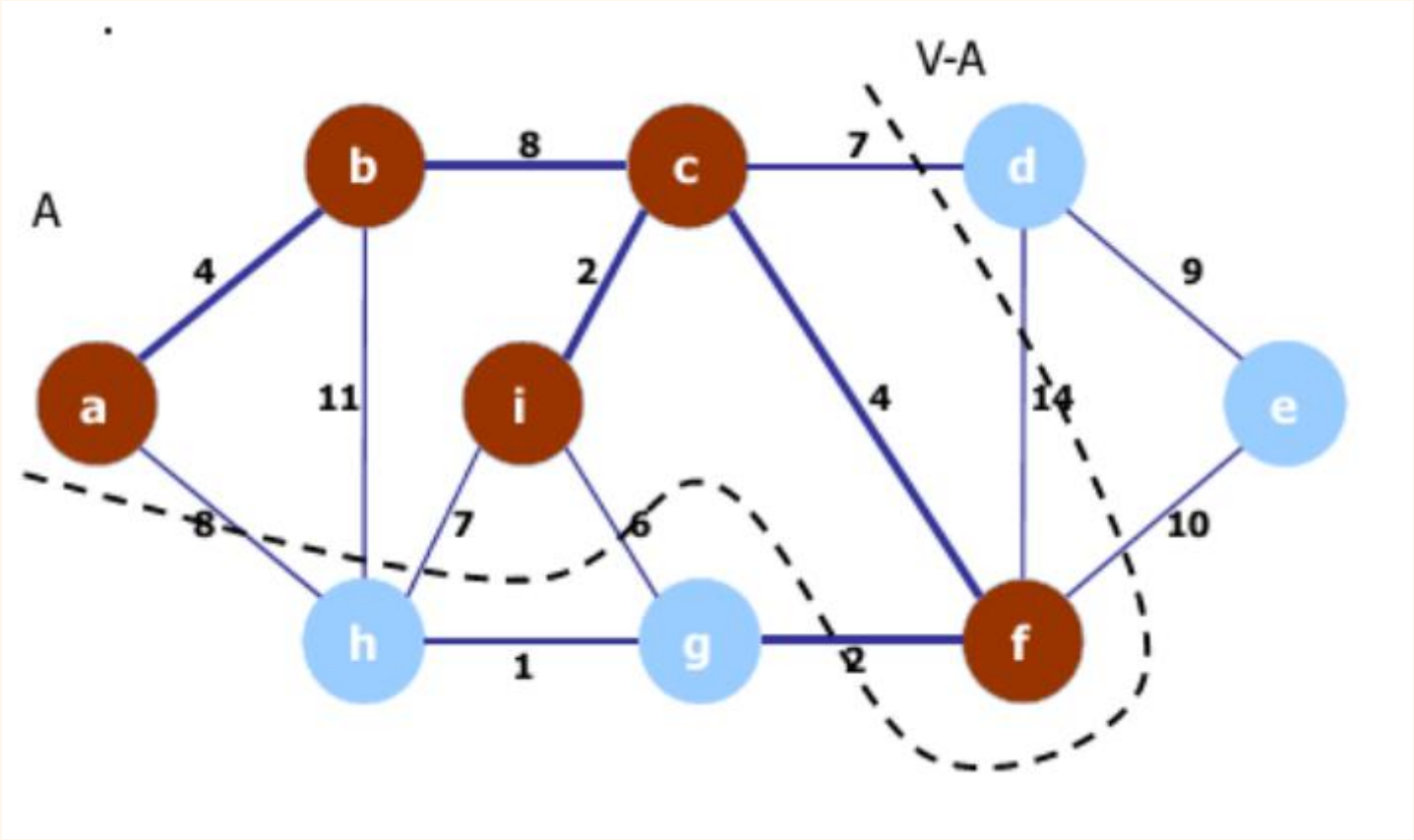
Algoritmo de Prim



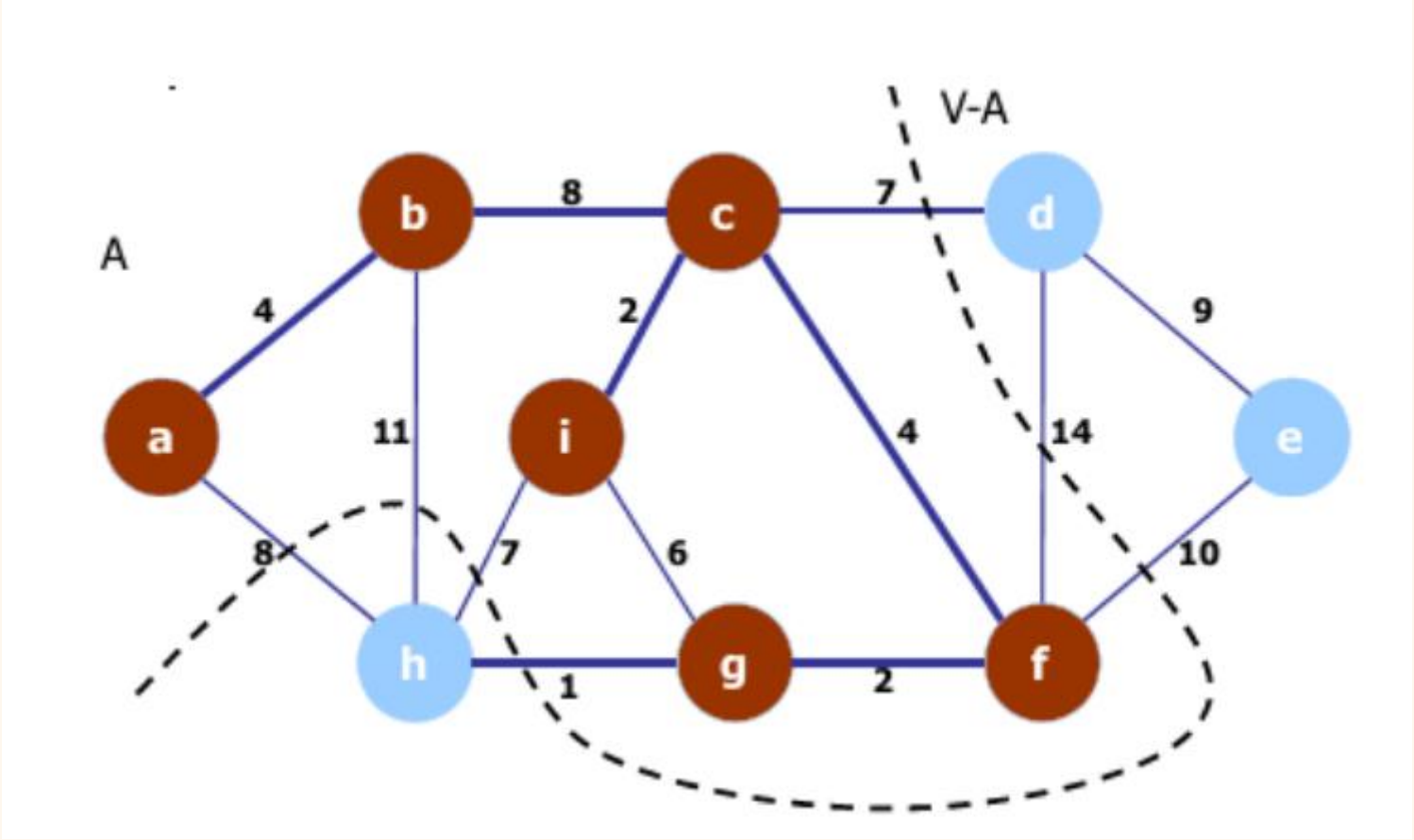
Algoritmo de Prim



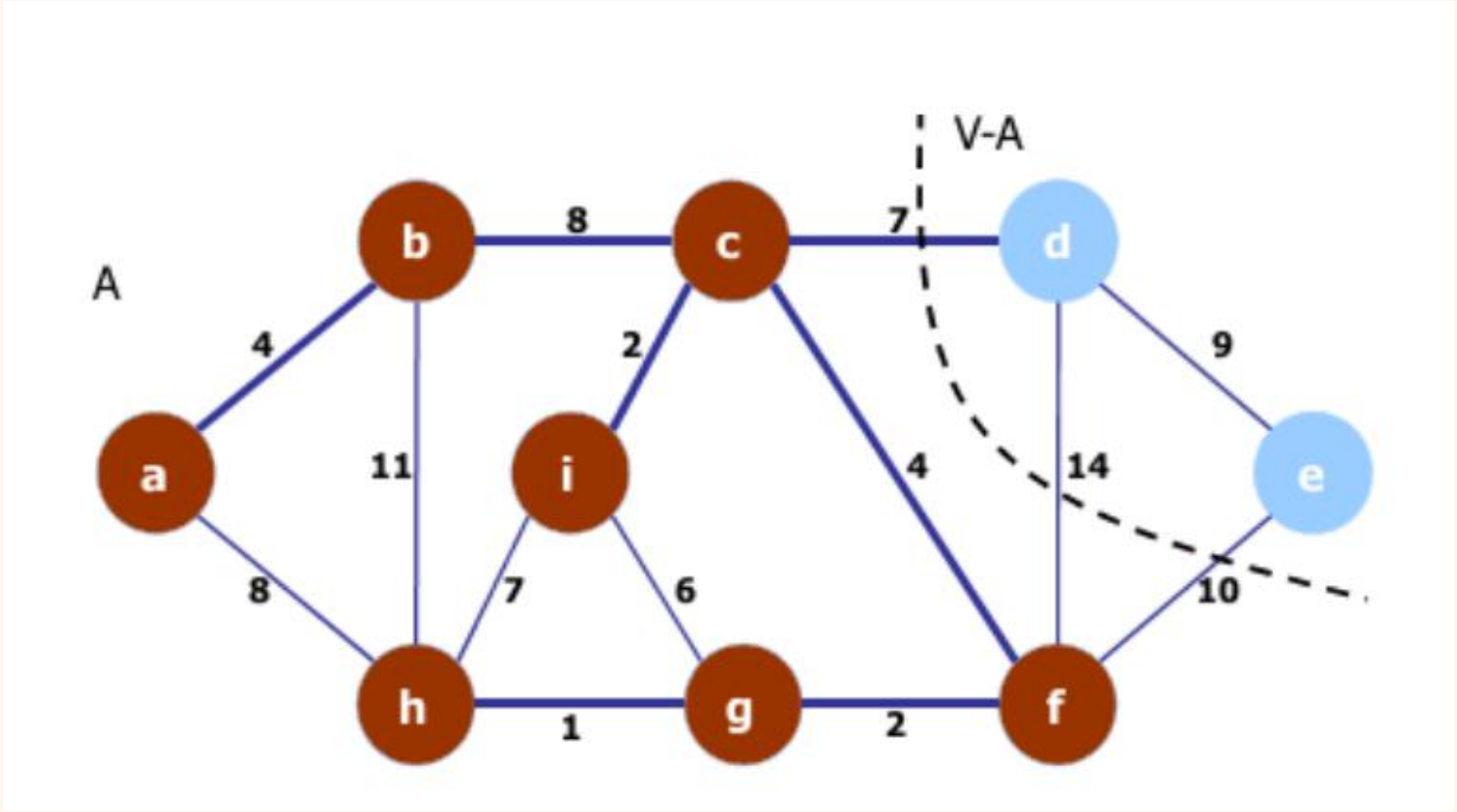
Algoritmo de Prim



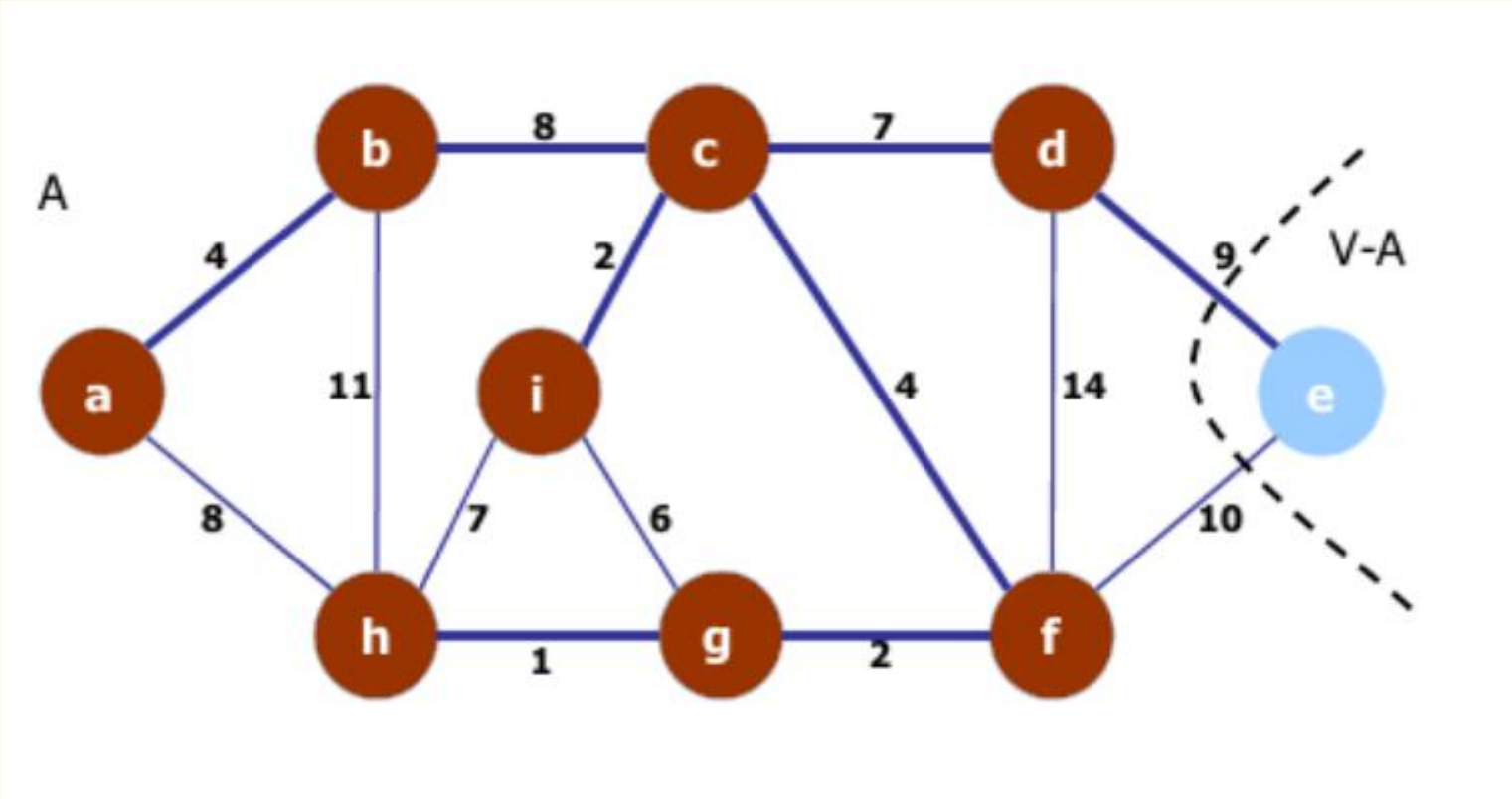
Algoritmo de Prim



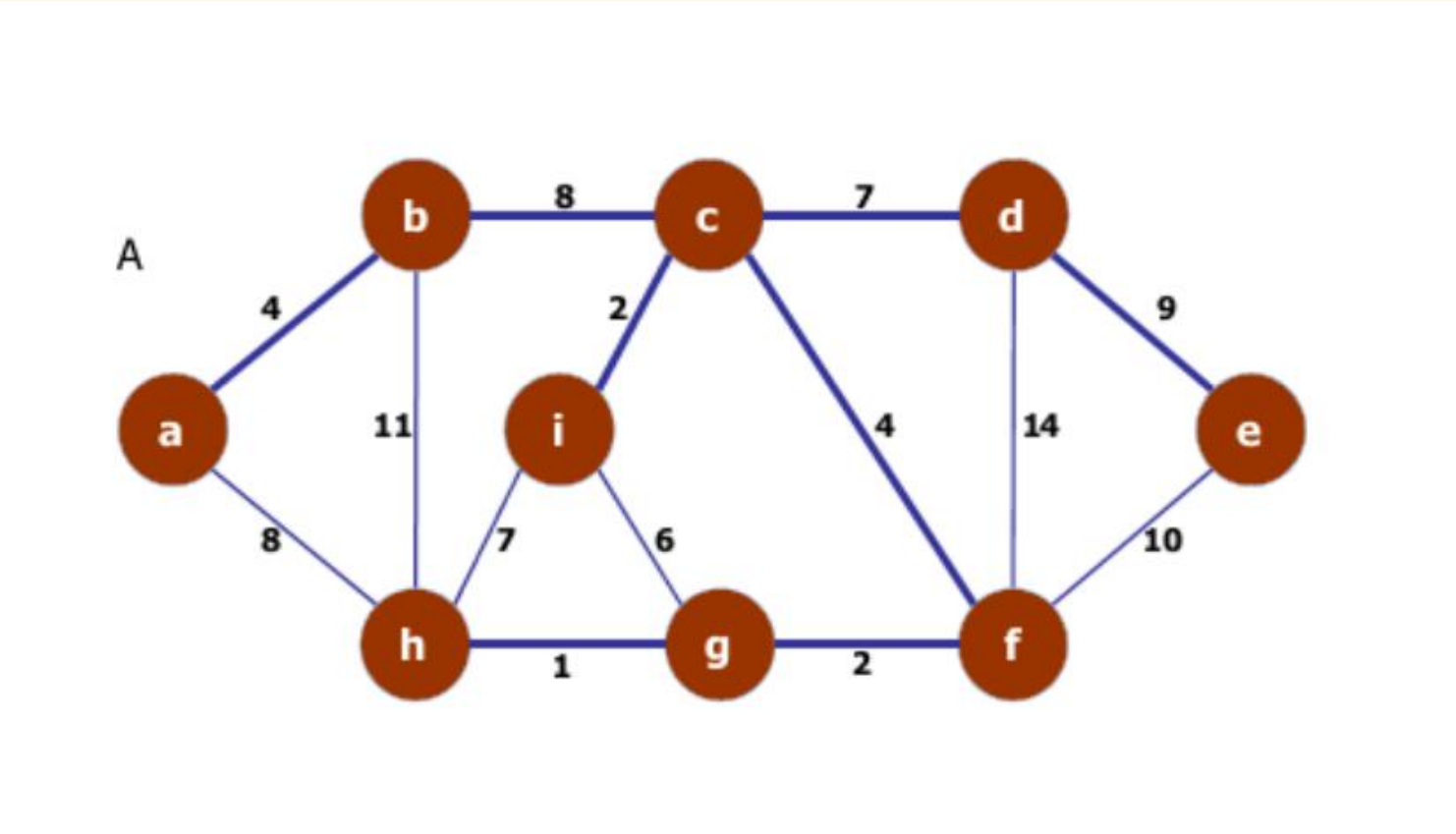
Algoritmo de Prim



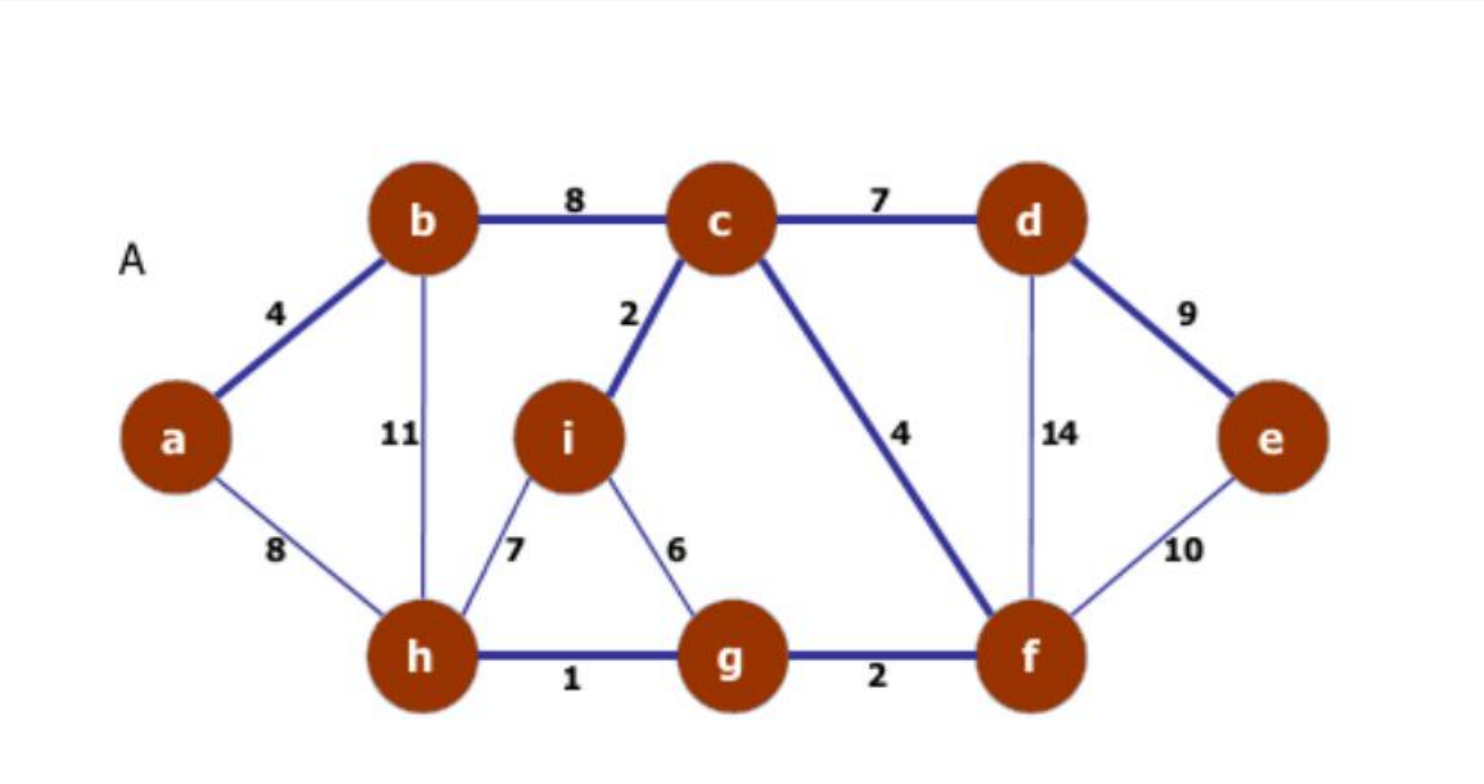
Algoritmo de Prim



Algoritmo de Prim



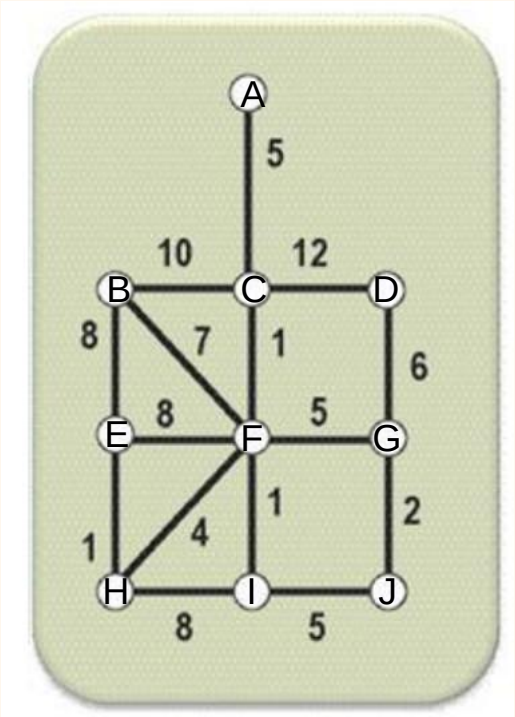
Algoritmo de Prim



Custo total = 4 + 8 + 1 + 2 + 4 + 2 + 7 + 9 = 37.

Algoritmos de Prim

- Encontre a árvore geradora mínima do grafo ao lado usando Prim a partir do vértice A.
- Qual o vértice incluído na quarta iteração do algoritmo?
- Qual o valor do custo da AGM ?



Dúvidas?

Laura Alves Pacifico
laps@cesar.school
Slack: Laura Pacifico