

ALGORITMOS DE ÁRVORE GERADORA MÍNIMA

DCE529 - Algoritmos e Estruturas de Dados III

Atualizado em: 16 de maio de 2023

Iago Carvalho

Departamento de Ciência da Computação

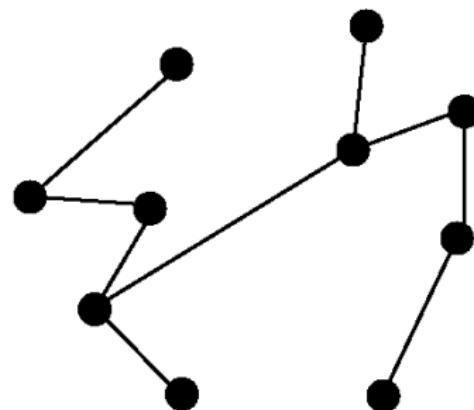
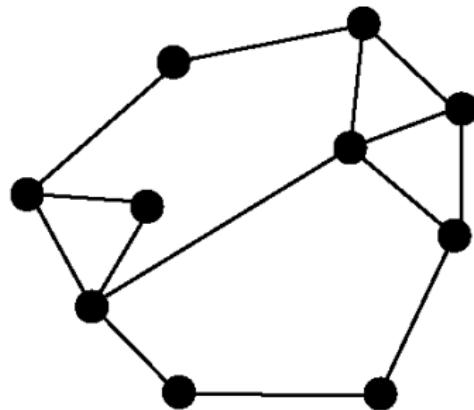


ÁRVORE

Sega $G = (V, E)$ um grafo não-orientado e ponderado

Uma árvore $T = (V, E')$ é um subgrafo induzido de G tal que

- T é conexo
- $|E'| = |V| - 1$



ÁRVORE GERADORA MÍNIMA

Uma árvore geradora mínima (AGM) é a árvore de menor custo total

Assim como no problema do caminho mínimo, considere que w que mapeia cada aresta a um valor real que simboliza o peso da aresta

$$w : E \mapsto \mathbb{R}$$

O peso de uma árvore geradora T é definido como $\sum_{e \in E'} w_e$

ÁRVORE GERADORA MÍNIMA

Existe um número exponencial de diferentes árvores geradoras em um grafo

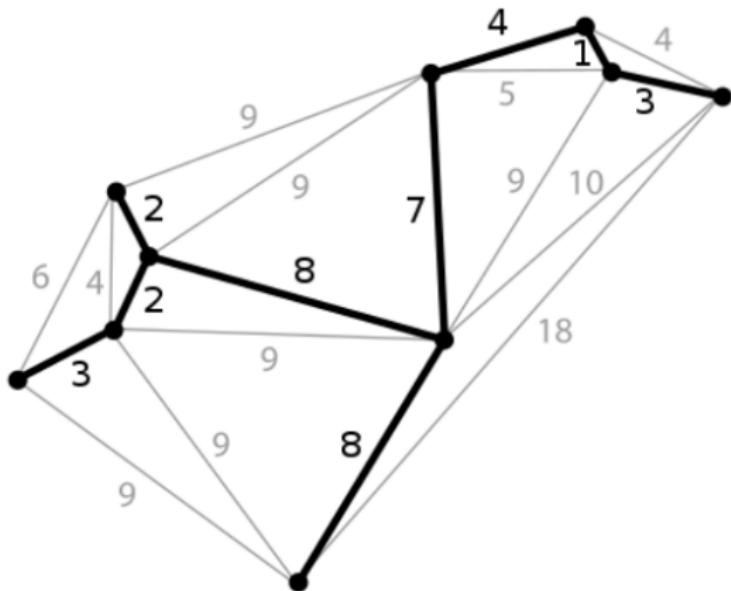
- Para ser mais preciso, existem n^{n-2} diferentes AGMs
- Enumerar todas elas é um problema $\#p$ -Completo

O problema da árvore geradora mínima consiste em encontrar a AGM de menor peso dentre todas as possíveis árvores

- Problema pode ser resolvido em tempo polinomial
- Subestrutura ótima

Em outras palavras, consiste em encontrar a combinação de arestas cuja AGM resultante tenha o menor peso possível

ÁRVORE GERADORA MÍNIMA



APLICAÇÕES

Diversos problemas práticos são modelados como AGMs

- Tabela de roteamento em redes de computadores
- Modelar uma rede de comunicações
- Análise de clusters
- Desenho de circuitos VLSI (microprocessadores)
- Taxonomia
- Problema base para diversos outros algoritmos e problemas em grafos

ALGORITMOS PARA ÁRVORE GERADORA MÍNIMA

Existem diversos algoritmos para encontrar a AGM de um grafo

- Boruvka
- Prim
- Kruskal
- Algoritmo *reverse-delete*
- Programação linear
- Algoritmo de Karger, Klein e Tarjan

O mais rápido deles é o de Karger, Klein e Tarjan

- Algoritmo randomizado
- Baseado no algoritmo de Boruvka e no *reverse-delete*
- Complexidade linear é esperada

ALGORITMO DE KRUSKAL

ALGORITMO DE KRUSKAL

Algoritmo desenvolvido por Joseph Kruskal em 1956

- Algoritmo guloso
- Obtem a AGM em tempo polinomial



ALGORITMO DE KRUSKAL

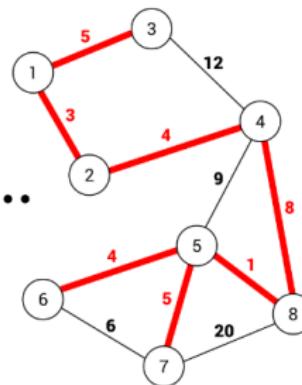
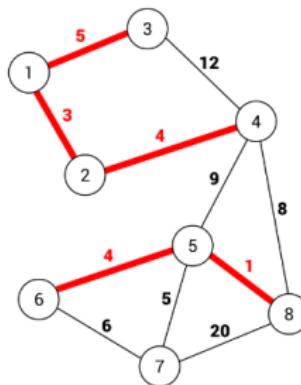
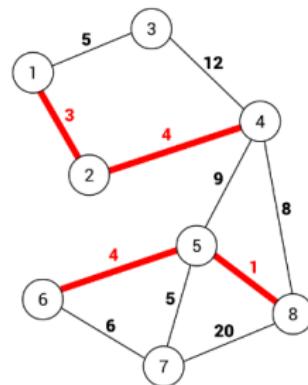
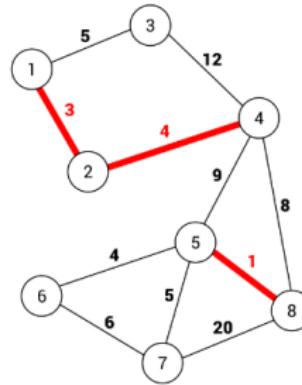
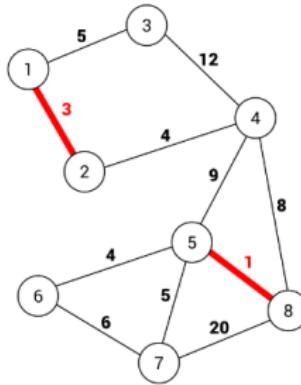
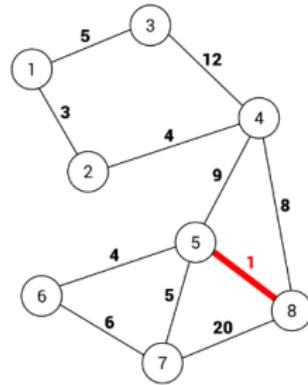
A árvore é construída de forma gulosa

Inicialmente, cada vértice é definido como um componente separado

A cada iteração do algoritmo, seleciona-se uma aresta para ser inserida

- Aresta de menor peso
- Resulta em uma estrutura acíclica
- Solução parcial é uma floresta

ALGORITMO DE KRUSKAL

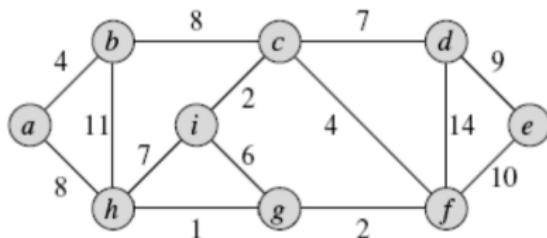


• • •

PSEUDO-CÓDIGO DO ALGORITMO DE KRUSKAL

```
AGM _ Kruskal(G(V, A), w)
    X ← { }
    para cada vértice  $v \in V$  faça
        criarConjunto( $v$ )
    fim para
     $A' \leftarrow$  ordenar as arestas de  $A$  por peso crescente
    para cada aresta  $(u, v) \in A'$  faça
        se conjuntoDe( $u$ ) ≠ conjuntoDe( $v$ ) então
             $X \leftarrow X \cup \{(u, v)\}$ 
            aplicarUnião( $u, v$ )
        fim se
    fim para
    retorno  $X$ 
fim
```

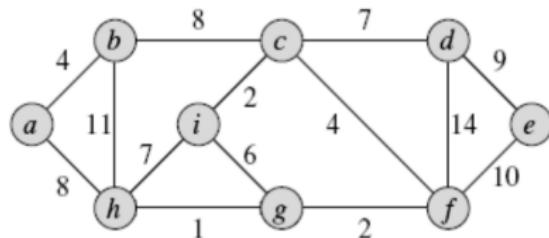
EXEMPLO



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

EXEMPLO

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

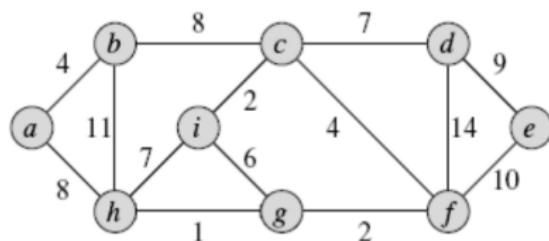


A' $(g, h); (c, i); (f, g); (a, b); (c, f); (g, i); (c, d); (h, i); (a, h); (b, c); (d, e); (e, f); (b, h); (d, f)$

EXEMPLO

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

g e h pertencem a
mesma árvore na
floresta?

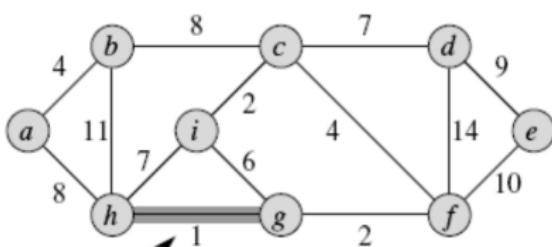


A' (g, h); (c, i); (f, g); (a, b); (c, f); (g, i); (c, d); (h, i);
(a, h); (b, c); (d, e); (e, f); (b, h); (d, f)

EXEMPLO

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

c e i pertencem a mesma árvore na floresta?

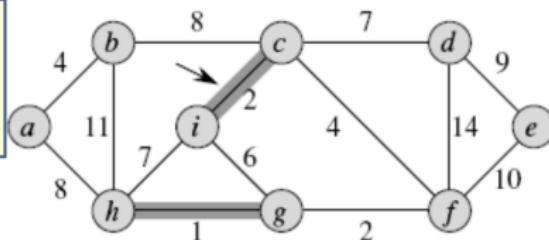


A' (g, h); (c, i); (f, g); (a, b); (c, f); (g, i); (c, d); (h, i);
(a, h); (b, c); (d, e); (e, f); (b, h); (d, f)

EXEMPLO

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

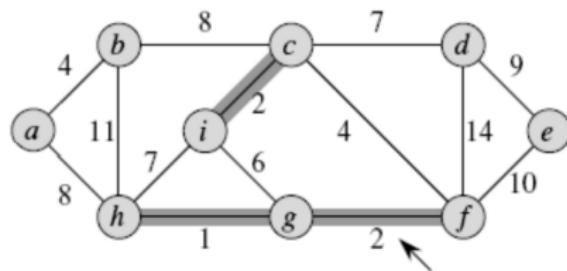
f e g pertencem a mesma árvore na floresta?



A' (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)

EXEMPLO

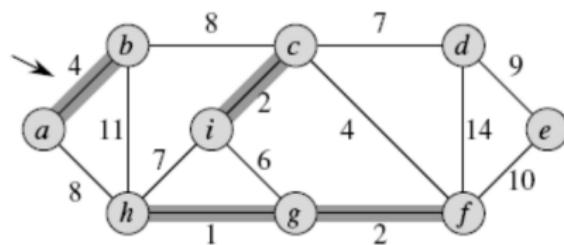
$$\{\{a\}, \{b\}, \{c, i\}, \{d\}, \{e\}, \{f, g, h\}\}$$



A' (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)

EXEMPLO

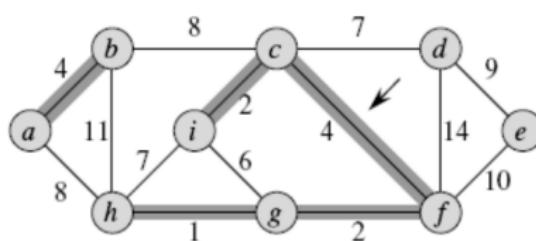
$$\{\{a,b\}, \{c,i\}, \{d\}, \{e\}, \{f,g,h\}\}$$



A' (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)

EXEMPLO

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

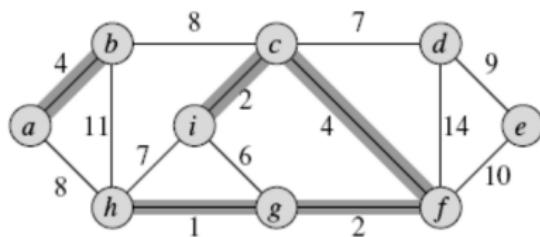


g e i pertencem a mesma árvore na floresta?

A' (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)

EXEMPLO

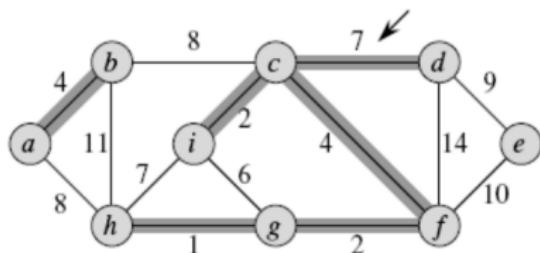
$$\{\{a,b\}, \{d\}, \{e\}, \{c, f, g, h, i\}\}$$



A' (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)

EXEMPLO

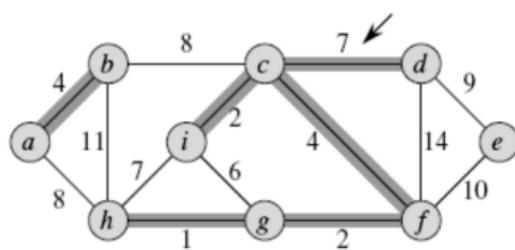
$$\{\{a,b\}, \{e\}, \{c, d, f, g, h, i\}\}$$



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

EXEMPLO

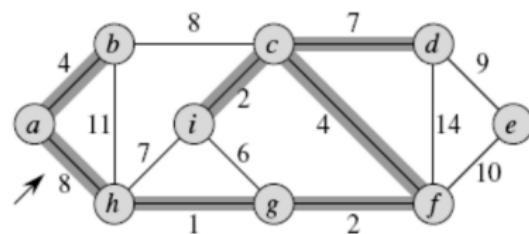
$$\{\{a,b\}, \{e\}, \{c,d,f,g,h,i\}\}$$



A' (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)

EXEMPLO

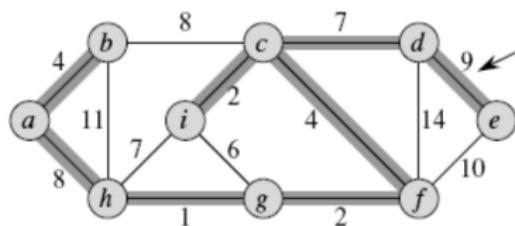
$$\{\{e\}, \{a, b, c, d, f, g, h, i\}\}$$



A' (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)~~

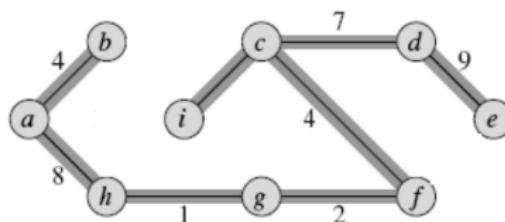
EXEMPLO

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



A' (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)

EXEMPLO



A' (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)~~)

COMPLEXIDADE DO ALGORITMO DE KRUSKAL

Neste algoritmo é necessário ordenar todas as arestas de acordo com seu peso

Desta forma, a complexidade do algoritmo de Kruskal é da ordem de complexidade do algoritmo utilizado na ordenação

- $\mathcal{O}(|E| \log |E|)$

ALGORITMO DE PRIM

ALGORITMO DE PRIM

Algoritmo desenvolvido por Robert C. Prim em 1957

- Inicialmente proposto por Vojtěch Jarník em 1930
- Algoritmo guloso
- Obtem a AGM em tempo polinomial

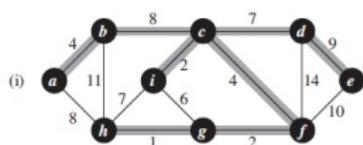
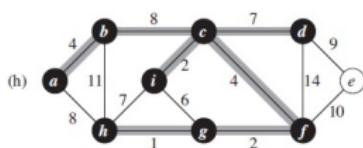
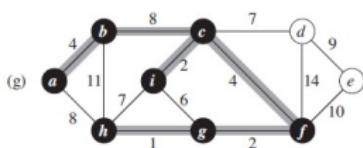
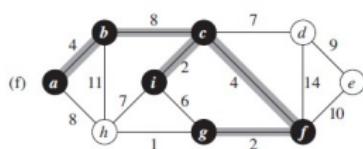
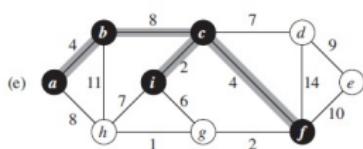
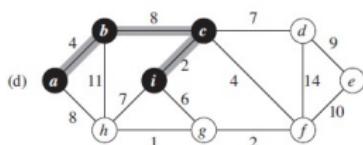
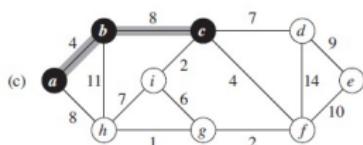
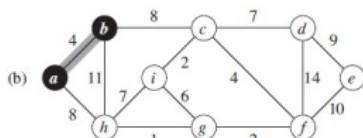
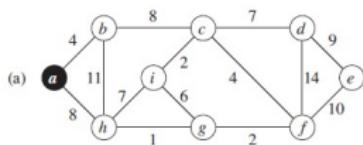


ALGORITMO DE PRIM

O algoritmo de Prim começa com um conjunto contendo um único vértice

- Incrementa este conjunto incluindo arestas de baixo peso
 - Incluindo também os vértices associados a esta aresta
- Inclui somente arestas que levam a uma estrutura acíclica
 - Arestras de um vértice deste conjunto para vértices de fora do conjunto

ALGORITMO DE PRIM



ALGORITMO DE PRIM

Utiliza três estruturas auxiliares

- **Q**: conjunto de vértices que ainda não fazem parte da AGM parcial
- **chave[u]**: peso da aresta mais leve a partir do vértice u que a conecta a AGM parcialmente construída
- $\pi[u]$: pai do vértice u

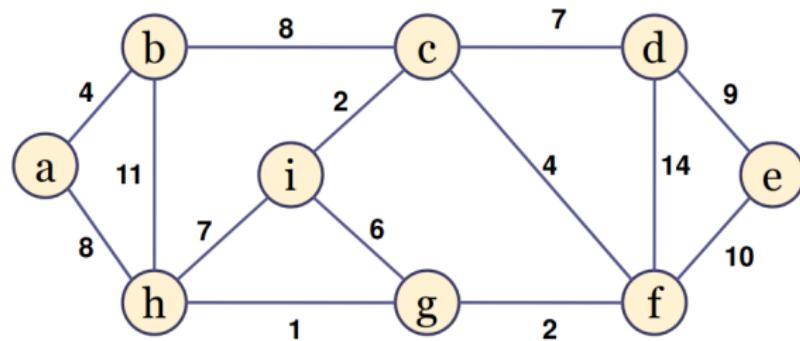
PSEUDOCÓDIGO

```
AGM_Prim(G(V,A), w, r)
  X ← { }
  para cada vértice  $u \in V$  faça
    chave[u] ←  $\infty$ 
     $\pi[u] \leftarrow \text{NULL}$ 
  fim para
  chave[r] ← 0
  Q ← V
```

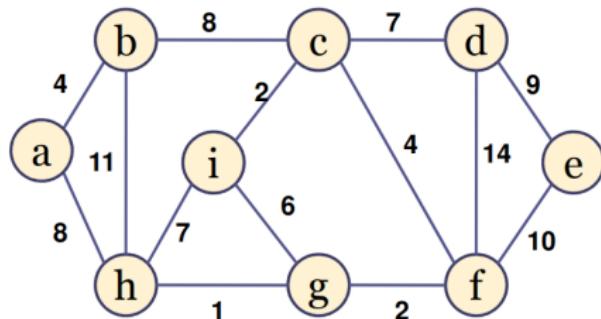
```
enquanto  $Q \neq \{ \}$ 
  u ← extrairMinimo(Q)
  X ← X ∪ {(u,  $\pi[u]$ )} // se  $u \neq r$ 
  para cada  $v \in \text{Adj}[u]$  faça
    se ( $v \in Q$ ) e ( $w(u,v) < \text{chave}[v]$ )
      chave[v] ←  $w(u,v)$ 
       $\pi[v] \leftarrow u$ 
    fim se
  fim para
  fim enquanto
  retorno X
fim
```

EXEMPLO

Vamos considerar o seguinte grafo



EXEMPLO



AGM_Prim($G(V, A)$, w , r)

$X \leftarrow \{ \}$

para cada vértice $u \in V$ faça

$chave[u] \leftarrow \infty$

$\pi[u] \leftarrow NULL$

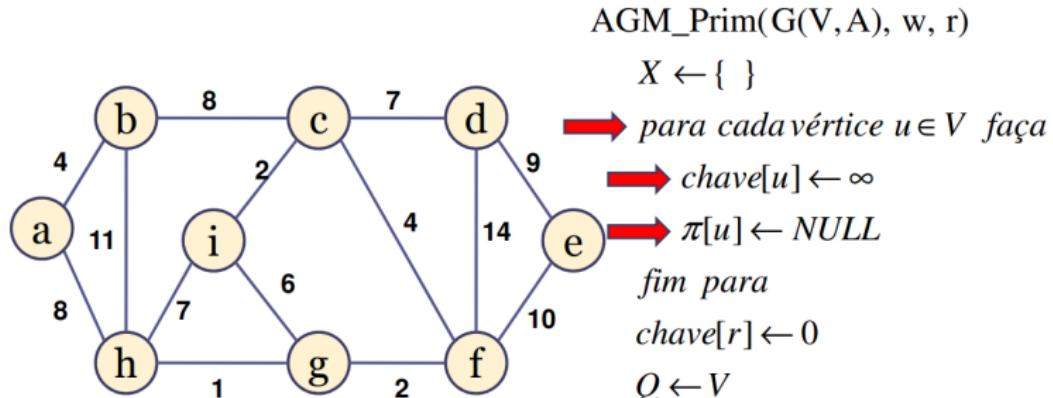
fim para

$chave[r] \leftarrow 0$

$Q \leftarrow V$

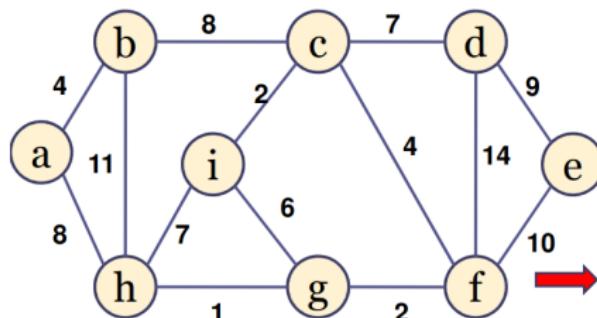
vértice	a	b	c	d	e	f	g	h	i
chave									
π									
Q									

EXEMPLO



vértice	a	b	c	d	e	f	g	h	i
chave	∞	∞	∞	∞	∞	∞	∞	∞	∞
π	NULL								
Q									

EXEMPLO



AGM_Prim($G(V, A)$, w , r)

$X \leftarrow \{ \}$

para cada vértice $u \in V$ faça

$chave[u] \leftarrow \infty$

$\pi[u] \leftarrow \text{NULL}$

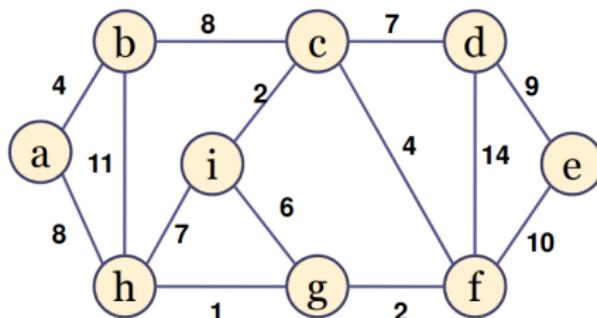
fim para

$chave[r] \leftarrow 0$

$Q \leftarrow V$

vértice	a	b	c	d	e	f	g	h	i
chave	0	∞							
π	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
Q									

EXEMPLO



AGM_Prim($G(V, A)$, w , r)

$X \leftarrow \{ \}$

para cada vértice $u \in V$ faça

$chave[u] \leftarrow \infty$

$\pi[u] \leftarrow NULL$

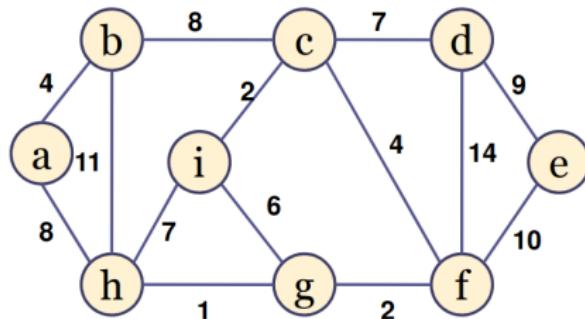
fim para

$chave[r] \leftarrow 0$

→ $Q \leftarrow V$

vértice	a	b	c	d	e	f	g	h	i
chave	0	∞							
π	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
Q	X	X	X	X	X	X	X	X	X

EXEMPLO



→ enquanto $Q \neq \{ \}$

$u \leftarrow extrairMinimo(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ faça

se $(v \in Q) \text{ e } (w(u, v) < chave[v])$

$chave[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

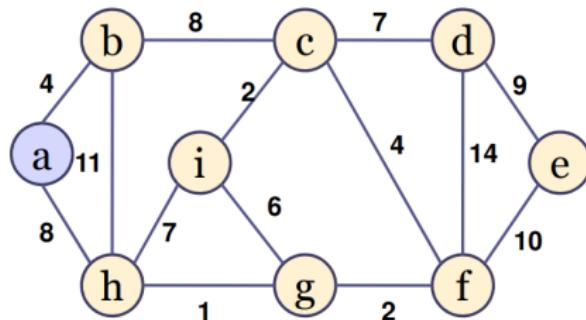
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	∞							
π	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
Q	X	X	X	X	X	X	X	X	X

EXEMPLO



enquanto $Q \neq \{ \}$

$\rightarrow u \leftarrow extrairMinimo(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ faça

se $(v \in Q) e (w(u, v) < chave[v])$

$chave[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

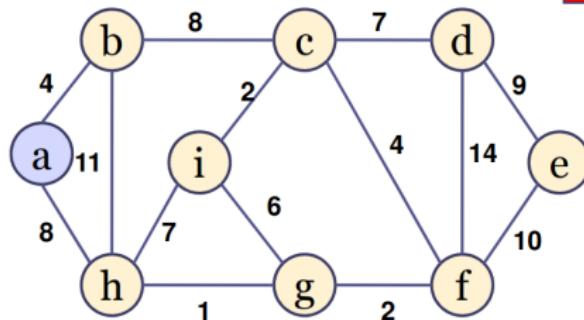
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	∞							
π	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
Q	---	X	X	X	X	X	X	X	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow extrairMinimo(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ faça

se $(v \in Q) \text{ e } (w(u, v) < chave[v])$

$chave[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

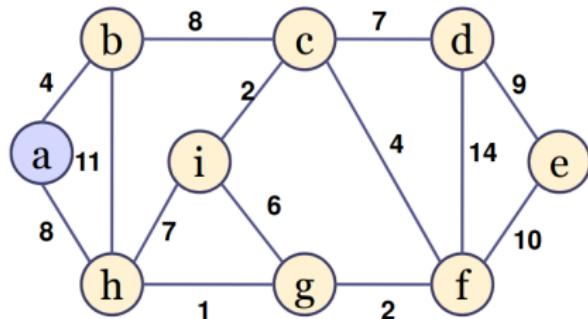
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	∞							
π	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
Q	---	X	X	X	X	X	X	X	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow \text{extrairMinimo}(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in \text{Adj}[u]$ faça

se $(v \in Q) \text{ e } (w(u, v) < \text{chave}[v])$

→ $\text{chave}[v] \leftarrow w(u, v)$

→ $\pi[v] \leftarrow u$

fim se

fim para

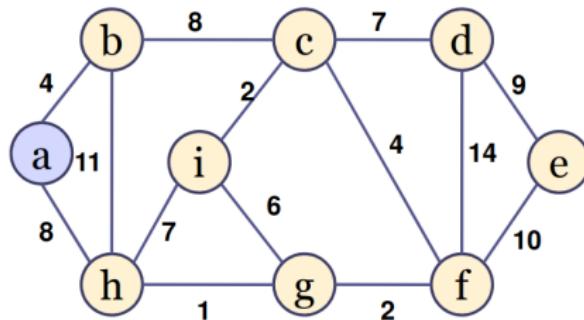
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	∞	∞	∞	∞	∞	8	∞
π	NULL	a	NULL	NULL	NULL	NULL	NULL	a	NULL
Q	---	X	X	X	X	X	X	X	X

EXEMPLO



→ enquanto $Q \neq \{ \}$

$u \leftarrow extrairMinimo(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ faça

se $(v \in Q) \text{ e } (w(u, v) < chave[v])$

$chave[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

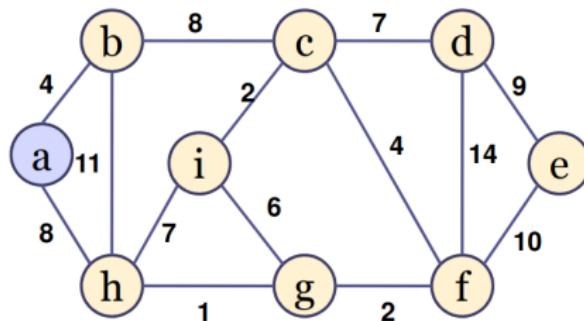
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	∞	∞	∞	∞	∞	8	∞
π	NULL	a	NULL	NULL	NULL	NULL	NULL	a	NULL
Q	---	X	X	X	X	X	X	X	X

EXEMPLO



enquanto $Q \neq \{ \}$

$\rightarrow u \leftarrow extrairMinimo(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ faça

se $(v \in Q) \text{ e } (w(u, v) < chave[v])$

$chave[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

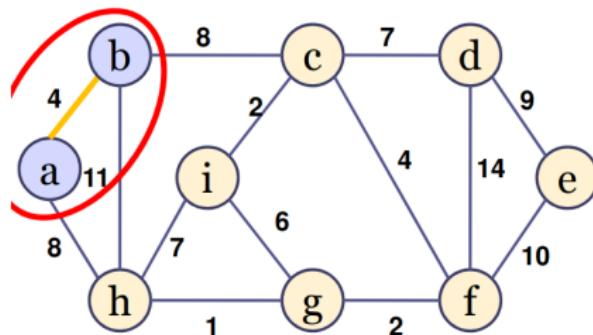
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	∞	∞	∞	∞	∞	8	∞
π	NULL	a	NULL	NULL	NULL	NULL	NULL	a	NULL
Q	---	---	X	X	X	X	X	X	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow \text{extrairMinimo}(Q)$

→ $X \leftarrow X \cup \{(u, \pi[u])\} // \text{se } u \neq r$

para cada $v \in \text{Adj}[u]$ faça

se $(v \in Q) \text{ e } (w(u, v) < \text{chave}[v])$

$\text{chave}[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

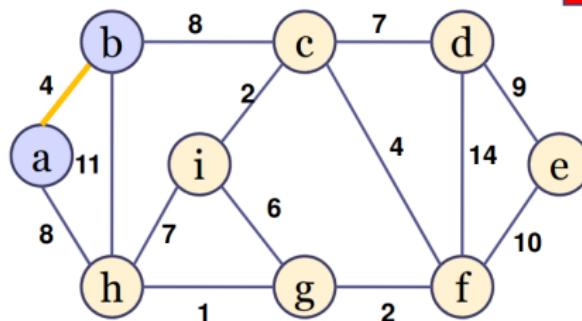
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	∞	∞	∞	∞	∞	8	∞
π	NULL	a	NULL	NULL	NULL	NULL	NULL	a	NULL
Q	---	---	X	X	X	X	X	X	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow \text{extrairMinimo}(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in \text{Adj}[u]$ faça

se $(v \in Q) e (w(u, v) < \text{chave}[v])$

$\text{chave}[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

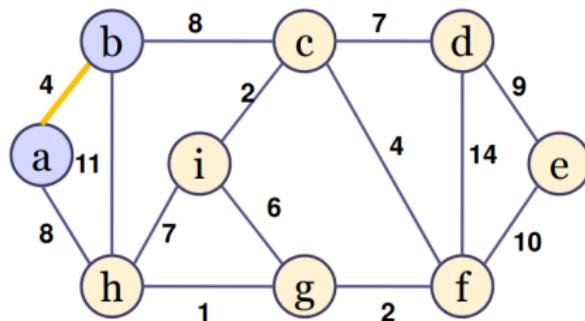
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	∞	∞	∞	∞	∞	8	∞
π	NULL	a	NULL	NULL	NULL	NULL	NULL	a	NULL
Q	---	---	X	X	X	X	X	X	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow \text{extrairMinimo}(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in \text{Adj}[u]$ faça

 se $(v \in Q) \text{ e } (w(u, v) < \text{chave}[v])$

$\text{chave}[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

 fim se

 fim para

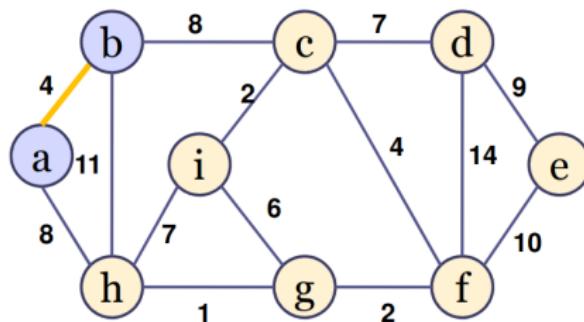
 fim enquanto

 retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	8	∞	∞	∞	∞	8	∞
π	NULL	a	b	NULL	NULL	NULL	NULL	a	NULL
Q	---	---	X	X	X	X	X	X	X

EXEMPLO



enquanto $Q \neq \{ \}$

$\rightarrow u \leftarrow extrairMinimo(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ faça

se $(v \in Q) e (w(u, v) < chave[v])$

$chave[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

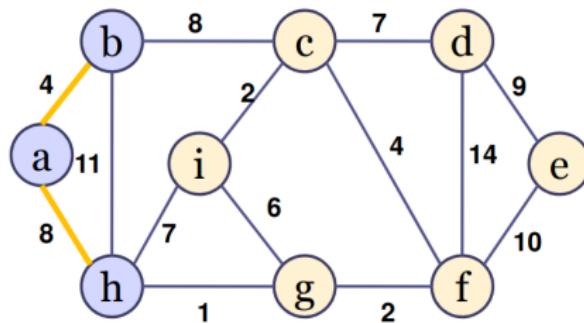
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	8	∞	∞	∞	∞	8	∞
π	NULL	a	b	NULL	NULL	NULL	NULL	a	NULL
Q	---	---	X	X	X	X	X	---	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow \text{extrairMinimo}(Q)$

→ $X \leftarrow X \cup \{(u, \pi[u])\} // \text{se } u \neq r$

para cada $v \in \text{Adj}[u]$ faça

se $(v \in Q) \text{ e } (w(u, v) < \text{chave}[v])$

$\text{chave}[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

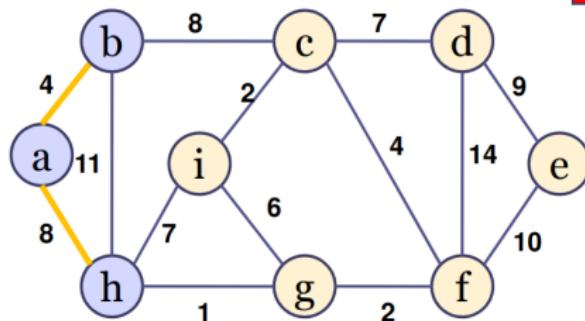
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	8	∞	∞	∞	∞	8	∞
π	NULL	a	b	NULL	NULL	NULL	NULL	a	NULL
Q	---	---	X	X	X	X	X	---	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow \text{extrairMinimo}(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // \text{se } u \neq r$

para cada $v \in \text{Adj}[u]$ faça

se $(v \in Q) \text{ e } (w(u, v) < \text{chave}[v])$

$\text{chave}[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

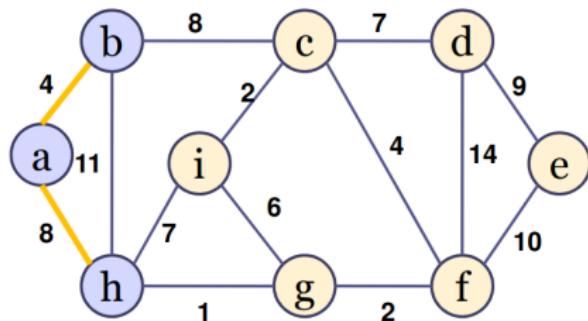
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	8	∞	∞	∞	∞	8	∞
π	NULL	a	b	NULL	NULL	NULL	NULL	a	NULL
Q	---	---	X	X	X	X	X	---	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow \text{extrairMínimo}(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in \text{Adj}[u]$ faça

 se $(v \in Q) \text{ e } (w(u, v) < \text{chave}[v])$

$\text{chave}[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

 fim se

 fim para

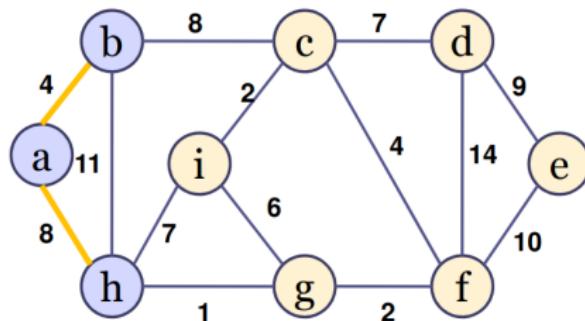
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	8	∞	∞	∞	1	8	7
π	NULL	a	b	NULL	NULL	NULL	h	a	h
Q	---	---	X	X	X	X	X	---	X

EXEMPLO



enquanto $Q \neq \{ \}$

$\rightarrow u \leftarrow extrairMinimo(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ faça

se $(v \in Q) e (w(u,v) < chave[v])$

$chave[v] \leftarrow w(u,v)$

$\pi[v] \leftarrow u$

fim se

fim para

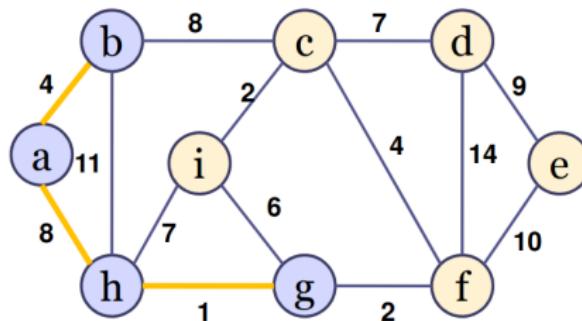
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	8	∞	∞	∞	1	8	7
π	NULL	a	b	NULL	NULL	NULL	h	a	h
Q	---	---	X	X	X	X	---	---	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow \text{extrairMinimo}(Q)$

→ $X \leftarrow X \cup \{(u, \pi[u])\} // \text{se } u \neq r$

para cada $v \in \text{Adj}[u]$ faça

se $(v \in Q) \text{ e } (w(u, v) < \text{chave}[v])$

$\text{chave}[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

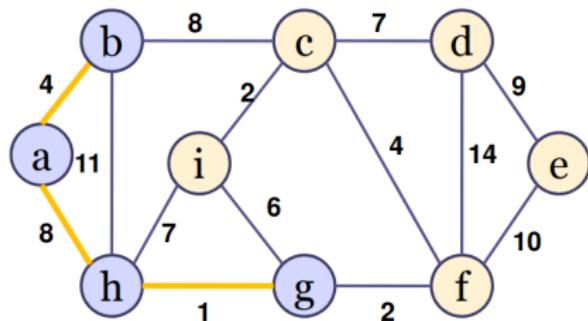
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	8	∞	∞	∞	1	8	7
π	NULL	a	b	NULL	NULL	NULL	h	a	h
Q	---	---	X	X	X	X	---	---	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow \text{extrairMínimo}(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in \text{Adj}[u]$ faça

 se $(v \in Q) \text{ e } (w(u, v) < \text{chave}[v])$

$\text{chave}[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

 fim se

 fim para

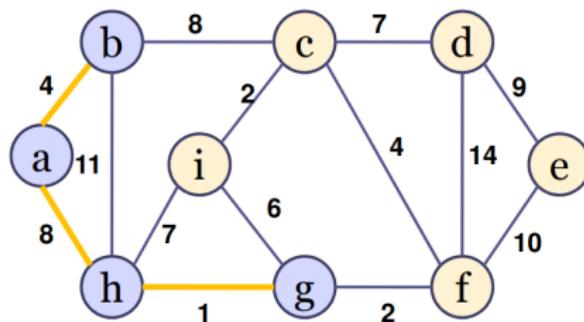
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	8	∞	∞	2	1	8	6
π	NULL	a	b	NULL	NULL	g	h	a	g
Q	---	---	X	X	X	X	---	---	X

EXEMPLO



enquanto $Q \neq \{ \}$

$\rightarrow u \leftarrow extrairMinimo(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ faça

se $(v \in Q) \text{ e } (w(u, v) < chave[v])$

$chave[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

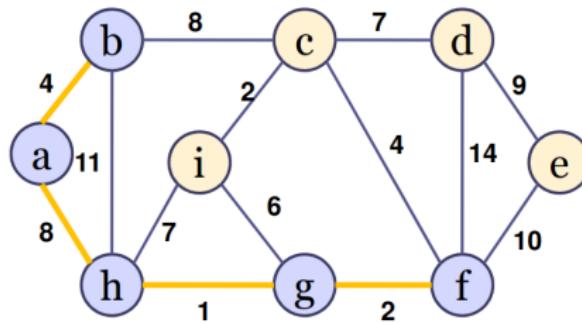
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	8	∞	∞	2	1	8	6
π	NULL	a	b	NULL	NULL	g	h	a	g
Q	---	---	X	X	X	---	---	---	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow \text{extrairMinimo}(Q)$

→ $X \leftarrow X \cup \{(u, \pi[u])\} // \text{se } u \neq r$

para cada $v \in \text{Adj}[u]$ faça

se $(v \in Q) \text{ e } (w(u, v) < \text{chave}[v])$

$\text{chave}[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

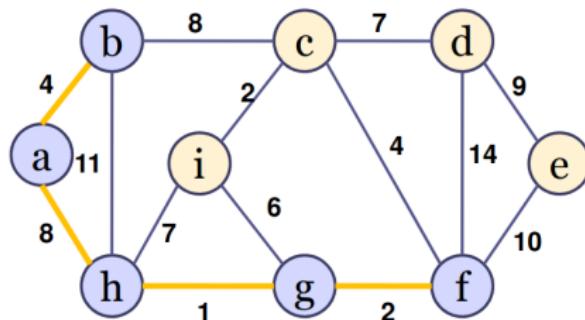
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	8	∞	∞	2	1	8	6
π	NULL	a	b	NULL	NULL	g	h	a	g
Q	---	---	X	X	X	---	---	---	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow \text{extrairMínimo}(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in \text{Adj}[u]$ faça

 se $(v \in Q) \text{ e } (w(u, v) < \text{chave}[v])$

$\text{chave}[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

 fim se

 fim para

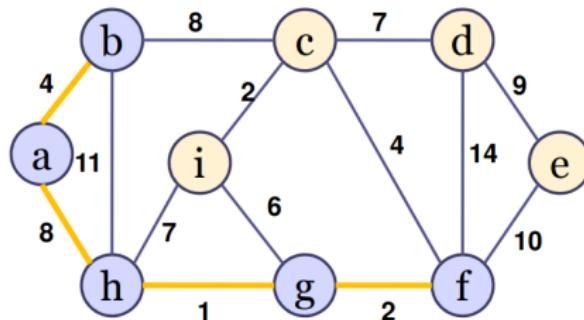
 fim enquanto

 retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	4	14	10	2	1	8	6
π	NULL	a	f	f	f	g	h	a	g
Q	---	---	X	X	X	---	---	---	X

EXEMPLO



enquanto $Q \neq \{ \}$

$\rightarrow u \leftarrow extrairMinimo(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ faça

se $(v \in Q) e (w(u, v) < chave[v])$

$chave[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

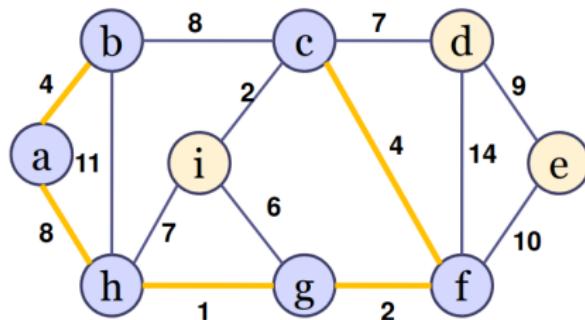
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	4	14	10	2	1	8	6
π	NULL	a	f	f	f	g	h	a	g
Q	---	---	---	X	X	---	---	---	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow extrairMinimo(Q)$

→ $X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ faça

se $(v \in Q) \text{ e } (w(u, v) < chave[v])$

$chave[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

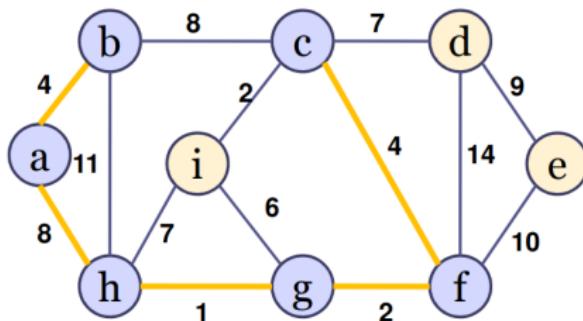
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	4	14	10	2	1	8	6
π	NULL	a	f	f	f	g	h	a	g
Q	---	---	---	X	X	---	---	---	X

Algoritmo de Prim



enquanto $Q \neq \{ \}$

$u \leftarrow \text{extrairMinimo}(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in \text{Adj}[u]$ faça

 se ($v \in Q$) e ($w(u, v) < \text{chave}[v]$)

$\text{chave}[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

 fim se

fim para

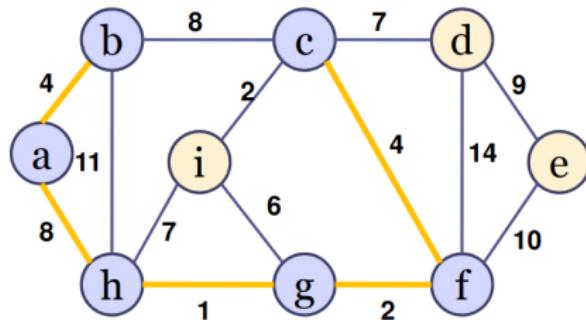
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	4	7	10	2	1	8	2
π	NULL	a	f	c	f	g	h	a	c
Q	---	---	---	X	X	---	---	---	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow extrairMinimo(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ *faça*

se ($v \in Q$) *e* ($w(u, v) < chave[v]$)

$chave[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

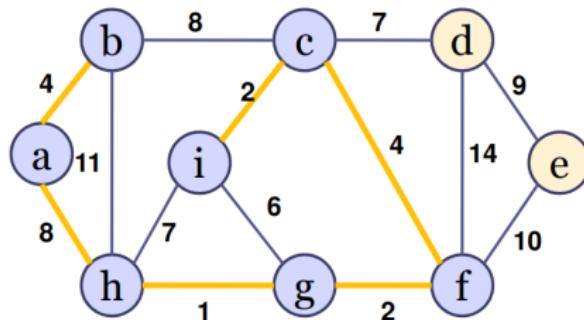
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	4	7	10	2	1	8	2
π	NULL	a	f	c	f	g	h	a	c
Q	---	---	---	X	X	---	---	---	X

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow \text{extrairMinimo}(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // \text{se } u \neq r$

para cada $v \in \text{Adj}[u]$ *faça*

se $(v \in Q) \text{ e } (w(u, v) < \text{chave}[v])$

$\text{chave}[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

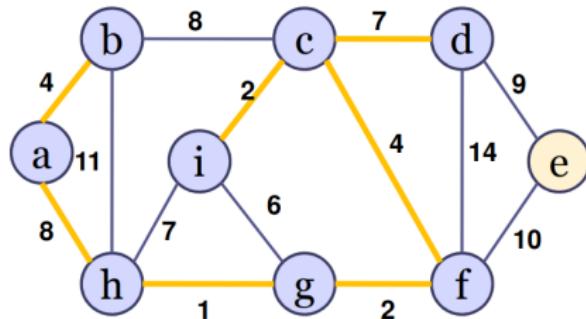
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	4	7	10	2	1	8	2
π	NULL	a	f	c	f	g	h	a	c
Q	---	---	---	X	X	---	---	---	---

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow extrairMinimo(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ faça

se $(v \in Q) \text{ e } (w(u,v) < chave[v])$

$chave[v] \leftarrow w(u,v)$

$\pi[v] \leftarrow u$

fim se

fim para

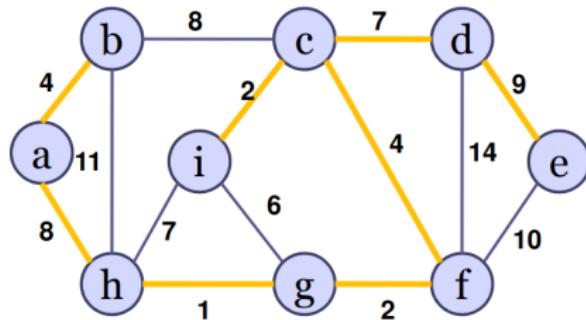
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	4	7	9	2	1	8	2
π	NULL	a	f	c	d	g	h	a	c
Q	---	---	---	---	X	---	---	---	---

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow extrairMinimo(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ *faça*

se ($v \in Q$) *e* ($w(u, v) < chave[v]$)

$chave[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

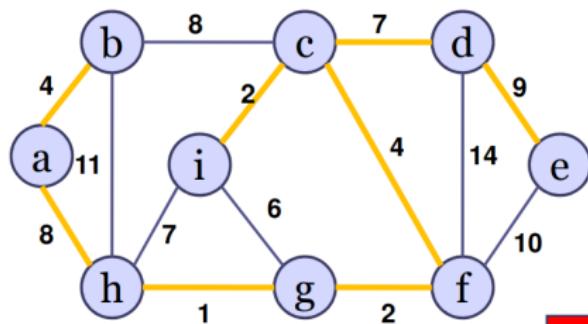
fim enquanto

retorne X

fim.

vértice	a	b	c	d	e	f	g	h	i
chave	0	4	4	7	9	2	1	8	2
π	NULL	a	f	c	d	g	h	a	c
Q	---	---	---	---	---	---	---	---	---

EXEMPLO



enquanto $Q \neq \{ \}$

$u \leftarrow extrairMinimo(Q)$

$X \leftarrow X \cup \{(u, \pi[u])\} // se u \neq r$

para cada $v \in Adj[u]$ faça

se $(v \in Q) \text{ e } (w(u, v) < chave[v])$

$chave[v] \leftarrow w(u, v)$

$\pi[v] \leftarrow u$

fim se

fim para

fim enquanto

retorne X

fim.



vértice	a	b	c	d	e	f	g	h	i
chave	0	4	4	7	9	2	1	8	2
π	NULL	a	f	c	d	g	h	a	c
Q	---	---	---	---	---	---	---	---	---

COMPLEXIDADE DO ALGORITMO DE PRIM

O algoritmo de Prim tem complexidade de $\Theta(|V + E| \log |V|)$

- Esta complexidade é atingida utilizando uma heap de prioridades
 - Heap de Fibonacci
- $\Theta(|E| \log |V|)$ usando heap e matriz de adjacências