# Árvores Geradoras Mínimas

em projetos eletrônicos, muitas vezes, é necessário que os pinos de vários componentes se tornem eletricamente equivalentes, o que é conseguido ligando-os uns aos outros.

- Para conectar N pinos, liga-se todos os arranjos possíveis, e mantém-se aquele que usa a menor quantidade de material

- Outro exemplo:
  - fios elétricos em um bairro
  -
  -

- Modelagem: grafo conexo não-dirigido $G(V,E)$

Problema é encontrar um subconjunto acíclico $T \subseteq E$ que conecte todos os possíveis vértices e tenha o menor peso total:

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

→ como $T$ é acíclico, deve-se gerar uma árvore, árvore geradora, já que "gera" o grafo $G$.

→ problema de árvore geradora mínima

• dois algoritmos:

- algoritmo de Kruskal
- algoritmo de Prim

} $O(E \log V)$ usando heaps binários

Os dois algoritmos são gulosos, ou seja, escolhem a melhor opção no momento.

- não garante que se encontre a solução global ótima.

## 01 Algoritmo Genérico

Input:

- grafo conexo não dirigido $G(V,E)$,
- função peso $w: E \to \mathbb{R}$

Estratégia gulosa:

- administra um conjunto de arestas A
- antes de cada iteração, A é um subconjunto de alguma árvore geradora mínima.
- em cada etapa/iteração, determinamos uma aresta $(u,v)$ que pode ser adicionada em A sem violar essa propriedade.

* aresta segura

GENERIC_MST (G, w)

1. A = ∅
2. enquanto A não formar uma árvore geradora
3. encontre uma aresta (u,v) que seja segura para A
4. A = A ∪ {(u,v)}
5. return A

Obs: a parte complicada é encontrar uma aresta segura (3).

## 02 Algoritmo de Prim

↳ funciona de modo muito semelhante ao algoritmo de Dijkstra para localizar caminhos mínimos

- o algoritmo de Prim tem a seguinte propriedade:
  - → as arestas em A sempre formam uma árvore única
  - → a árvore começa em um vértice arbitrário r e aumenta até conter todos os vértices V.
  - → cada etapa adiciona uma aresta que conecta A até um vértice isolado

- Para implementar o algoritmo de Prim, precisamos de um método eficiente para selecionar uma aresta para ser adicionada ao conjunto A.

Inputs:

- Grafo (G)
- vértice de origem (r)

durante:

- todos os vértices que não estão na árvore ficam em uma fila de prioridade mínima Q baseada em um atributo chave.
- para cada vértice v, o atributo v.chave = peso mínimo de qualquer aresta que conecta v a um vértice. Por convenção v.chave = $\infty$ se não existe tal aresta.
- v.pai: é o pai de v na árvore

$A = \{(v, v.pai) : v \in V - \{r\} - Q\}$

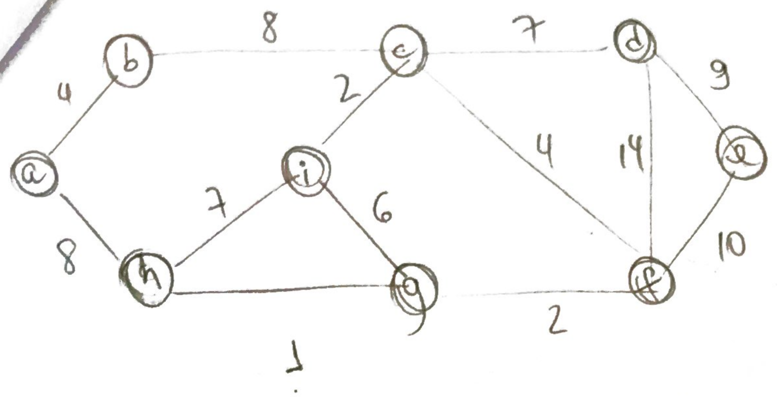
final:

- Q está vazia, e
- $A = \{(v, v.pai) : v \in V - \{r\}\}$

Prim(G,w,r)

```
para cada u ∈ V[G]
    u.chave = ∞
    u.pai = NULL
r.chave = 0
Q = V[G]
enquanto Q ≠ ∅
    u = Extrai_min(Q)
    para cada v ∈ G.Adj[u]
        se v ∈ Q e w(u,v) < v.chave
            v.pai = u
            v.chave = w(u,v)
```

mplo:




$r = a$

| Vértices | chave | pai |
|---|---|---|
| a | 0 | $\perp$ |
| b | ~~∞~~ 4 | A |
| c | ~~∞~~ 8 | B |
| d | ~~∞~~ 7 | C |
| e | ~~∞~~ ~~10~~ 9 | ~~f~~ d |
| f | ~~∞~~ 4 | c |
| g | ~~∞~~ ~~6~~ 2 | ~~i~~ f |
| h | ~~∞~~ ~~8~~ ~~7~~ 1 | ~~A~~ ~~i~~ G |
| i | ~~∞~~ 2 | c |

$Q = \{a, b, c, d, e, f, g, h, i\}$
$\quad 0 \; \infty \; \infty \; \infty \; \infty \; \infty \; \infty \; \infty \; \infty$

$A = \{\}$

- iteração 01:

$u = \text{ExtraiMin}(Q)$

$u = a$

* adjacentes de a

– b

$b \in Q$ ? e $w(a,b) < b.\text{chave}$

Sim e $4 < \infty \Rightarrow$ True

$b.\text{chave} = 4$

$b.\text{pai} = a$

– h

$h \in Q$ ? e $w(a,h) < h.\text{chave}$

Sim e $8 < \infty$

$h.\text{chave} = 8$

$h.\text{pai} = a$

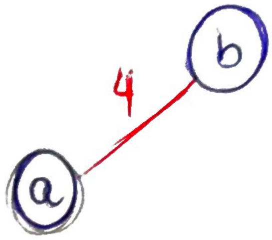
$Q = \{b, c, d, e, f, g, h, i\}$
$\quad 4 \; \infty \; \infty \; \infty \; \infty \; \infty \; 8 \; \infty$

$A = \{a\}$

iteração 02:

$u = \text{ExtractMin}()$

$b[4]$




- adjacentes de b

– b, a

$a \in Q$?

Não ✗

$A = \{a, b\}$

$Q = \{c, d, e, f, g, h, i\}$

– b, h

$h \in Q$? $w(b,h) < h.\text{chave}$

Sim $4 < 8$

Não ✗

– b, c

$c \in Q$? $w(b,c) < c$...

Sim $7 < \infty$

Sim e Sim!

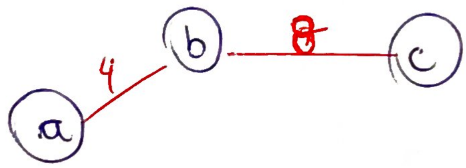
- c.chave = 7

c.pai = B

- iteração 3:

$Q = \{c, d, e, f, g, h, i\}$
$\;\;\;\;\;\;7\;\;\infty\;\;\infty\;\;\infty\;\;\infty\;\;8\;\;\infty$

min = c
$\;\;\;\;\;\;\;\;\;\;7$




–(c, i)

$i \in Q$? $w(c,i) < i.\text{chave}$

Sim $2 < \infty$?

Yes!

i.chave = 2

i.pai = c

–(c, d)

$d \in Q$? $w(c,d) < \infty$

Sim 7

Yes

d.chave = 7

d.pai = c

–(c, f)

$f \in Q$? $w(c,f) < \infty$

Sim Sim

f.chave = 4

f.pai = c

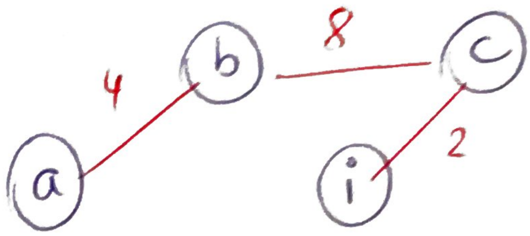
$A = \{a, b, c\}$

$Q = \{d, e, f, g, h, i\}$
$\;\;\;\;\;\;7\;\;\infty\;\;4\;\;\infty\;\;8\;\;2$

ração 04:

$u = i\,[2]$

* adjacentes de $i$




$(i, c)$

$c \in Q$?

Não X

$(i, h)$

$h \in Q$? $w(i,h) < h.chave$

Sim $7 < 8$

$h.chave = 7$

$h.pai = i$

$(i, g)$

$g \in Q$? $w(i,g) < g.chave$

Sim $6 < \infty$

$g.chave = 6$

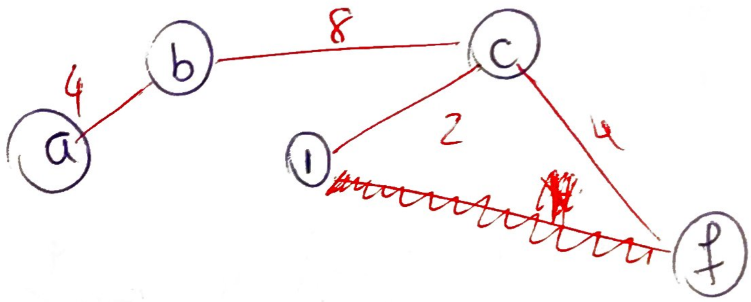
$g.pai = i$

$A = \{a, b, c, i\}$

$Q = \{d, e, f, g, h\}$
$\quad\; 7 \;\; \infty \;\; 4 \;\; 6 \;\; 7$

• iteração 05:

$u = f\,[4]$




$(f, c)$ X

$c \notin Q$!

$(f, d)$ X

$d \in Q$? $w(f,d) < d.chave$

S $14 < 7$

X

$(f, e)$

$e \in Q$? $w(f,e) < e.chave$

S $10 \quad \infty$ ok!

$e.chave = 10$

$e.pai = f$

$(f, g)$ $w(f,g) < g.chave$

$g \in Q$? $2 < 6$

S

$\rightarrow \begin{cases} g.chave = 2 \\ g.pai = f \end{cases}$

$A = \{a, b, c, i, f\}$

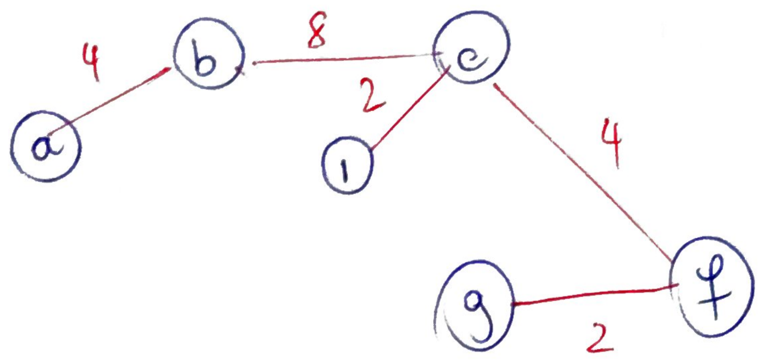
$Q = \{d, e, g, h\}$
$\quad 7, 10, 2, 7$

• iteração 06:

$u = g[2]$

$(g,f)$ ×   $f \notin Q$!

$(g,i)$ ×   $i \notin Q$




$(g,h)$

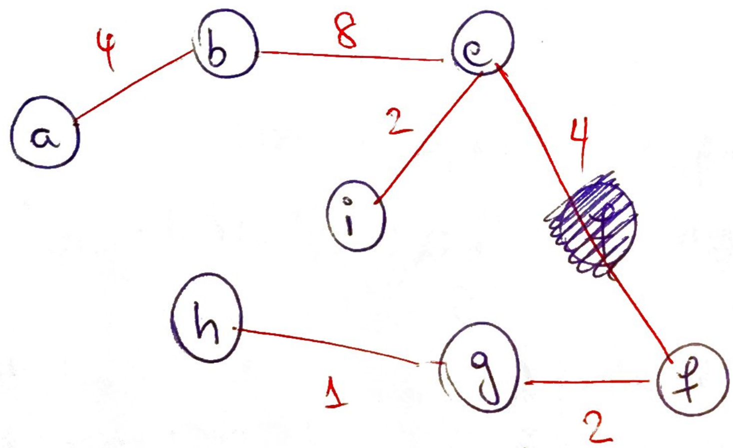
$h \in Q$?   $w(g,h) < h.chave$
S   $1 < 7$
Sim

$\Big\}$ $h.chave = 1$
$h.pai = g$

$A = \{a, b, c, i, f, g\}$

$Q = \{d, e, h\}$
7 10 1

• iteração 07: $u = h[1]$




$(h,a)$ ×   $a \notin Q$

$(h,b)$ ×   $b \notin Q$

$(h,i)$ ×   $i \notin Q$

$A = \{a, b, c, i, f, g, h\}$

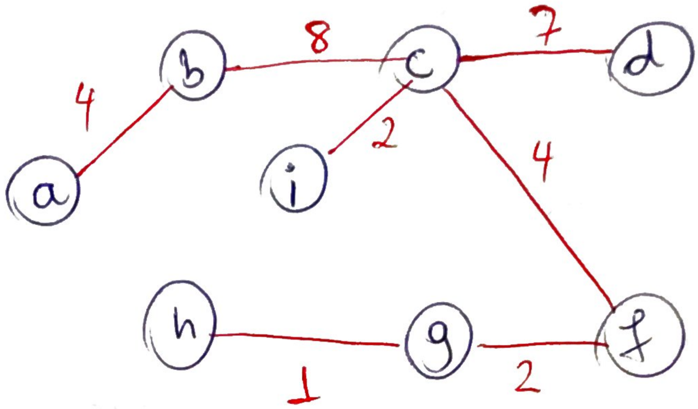
$Q = \{d, e\}$
7, 10

...ração 08: $u = d[7]$

$(d, c) \Rightarrow c \notin Q$ ×

$(d, e) \Rightarrow d \in Q$, $w(d,e) <$ e.chave? $9 < 10$? Sim

$(d, f) \Rightarrow f \notin Q$ ×




e.pai = d
e.chave = 9

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

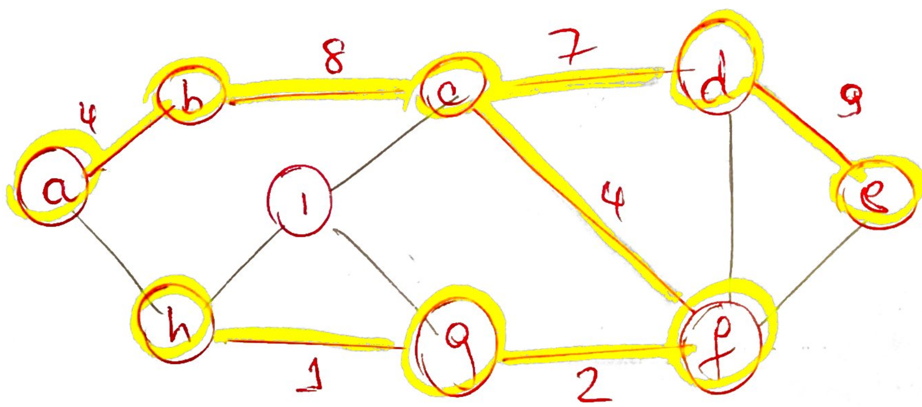
$Q = \{\underset{9}{e}\}$

- iteração 09: $u = e[9]$

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

$Q = \{\}$

solução




Exercício: iniciar o algoritmo de outros nós iniciais aleatoriamente