Sumário

- Algoritmo de Johnson
  - Repesagem de Johnson
- MSTs (Árvores Abrangentes de Menor Custo)
  - Propriedades Elementares
  - Algoritmo de Prim

Aula 12

Algoritmo de Johnson

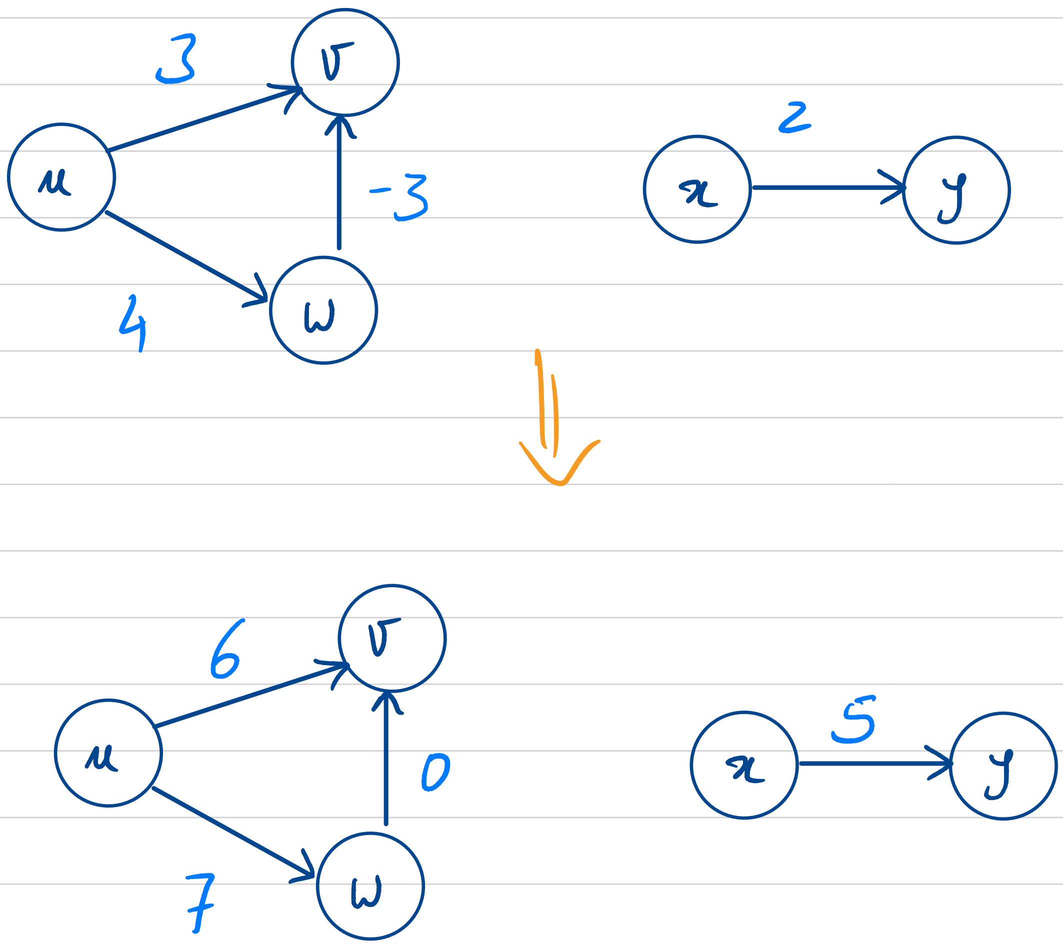
Ideia

- Dado um grafo G com pesos negativos, calcular um grafo G' cujos caminhos mais curtos coincidem com os de G mas **sem** arcos com pesos negativos

  Repesagem dos Arcos

- Aplicar o algoritmo de Dijkstra a todos os vértices de G'

Repesagem dos Arcos - 1ª Ideia

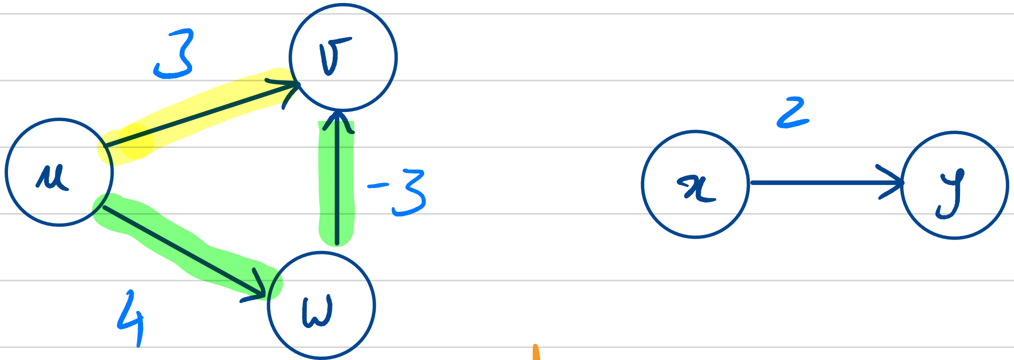


Ideia Naif:

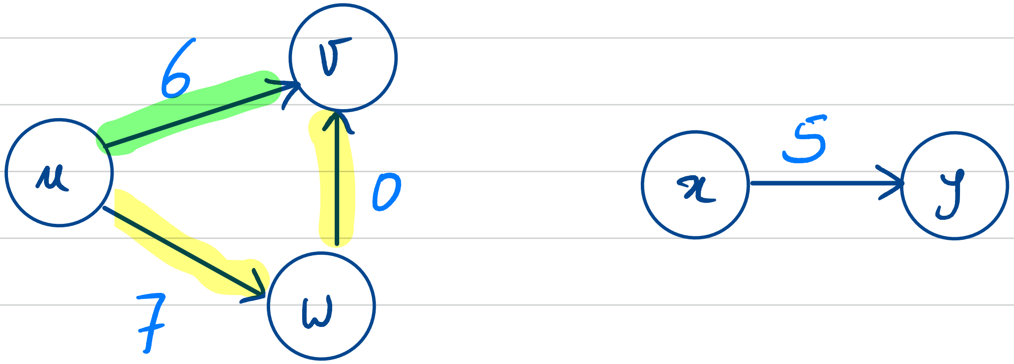
- Somar o módulo do maior dos pesos negativos a todos os arcos

Funcionou?

Repesagem dos Arcos - 1ª Ideia



⇓



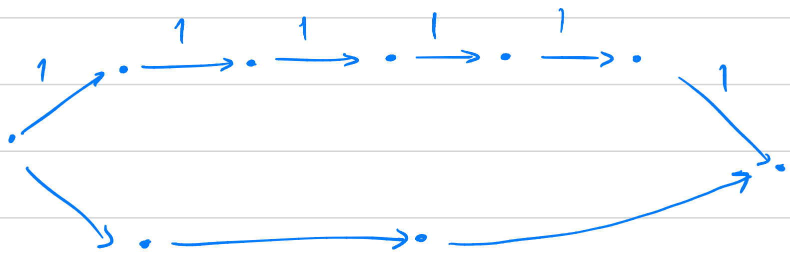
Ideia Naïf:

- Somar o módulo do maior dos pesos negativos a todos os arcos

Funcionou?

Não! Este método penaliza caminhos maiores!

Repesagem dos Arcos - 1ª Ideia



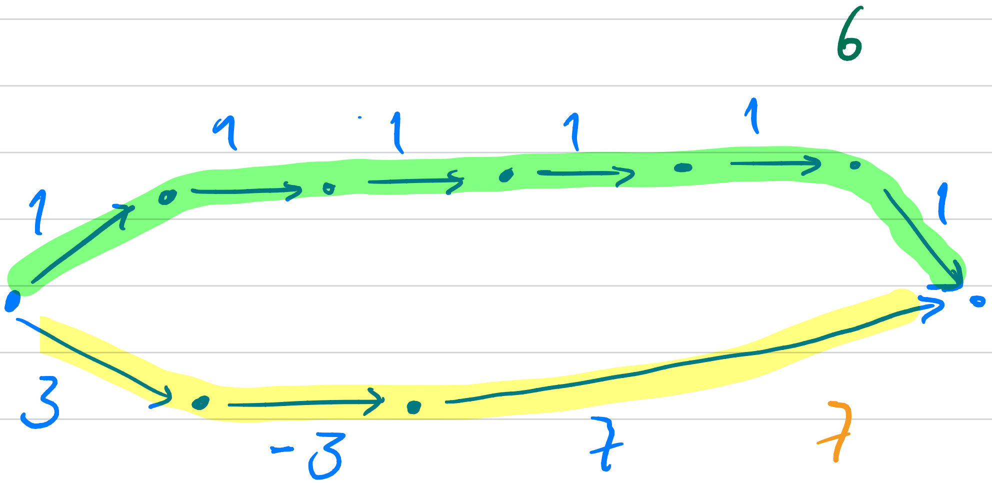
Ideia Naif:

- Somar o módulo do maior dos pesos negativos a todos os arcos

Funcionou?

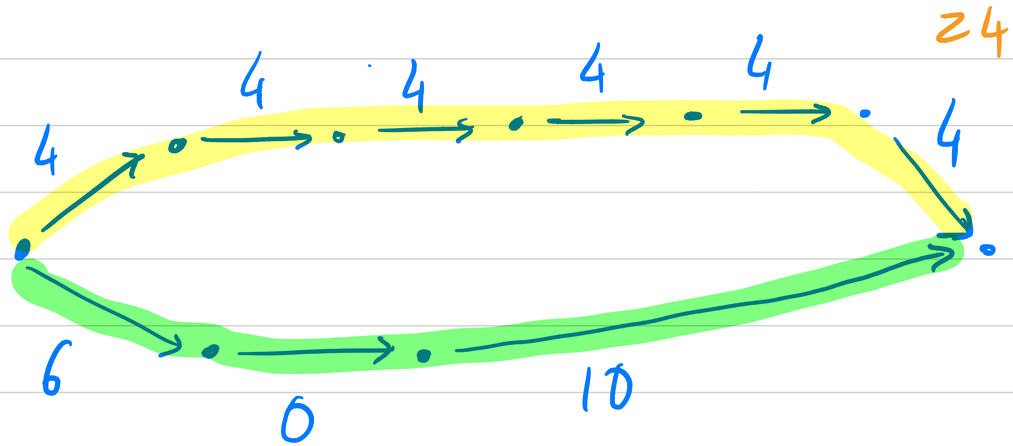
Não! Este método penaliza caminhos maiores!

Repesagem dos Arcos - 1ª Ideia



Ideia Naïf:

- Somar o módulo do maior dos pesos negativos a todos os arcos



Funcionou?

Não! Este método penaliza caminhos maiores!

Repesagem de Johnson

- Encontrar uma função de alturas $h: V \to \mathbb{R}$:

$G = (V, E, w)$

$\Downarrow$

$G = (V, E, \hat{w})$ onde: $\hat{w}(u,v) = w(u,v) + h(u) - h(v)$

- Quem é h?

Grafo estendido: $\bar{G} = (V \cup \{s\}, \bar{E})$

$\cdot\ \bar{E} = E \cup \{(s,v) \mid v \in V\}$ $\qquad \underline{\underline{h(v)}} = \delta(s,v)$

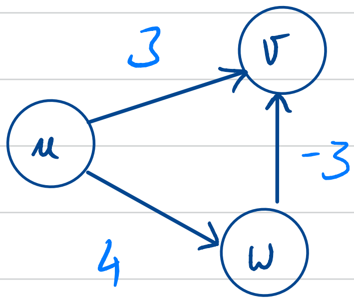
$$\cdot\ \bar{w}(u,v) = \begin{cases} w(u,v) & \text{se } (u,v) \in E \\ 0 & \text{se } u = s \wedge v \neq s \end{cases}$$

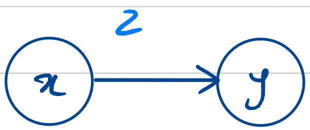
Repesagem de Johnson

- Encontrar uma função de alturas $h: V \to \mathbb{R}$:

$G=(V,E,w) \Rightarrow G=(V,E,\hat{w})$

onde: $\hat{w}(u,v) = w(u,v) + h(u) - h(v)$





- Quem é h?

Grafo estendido: $\bar{G} = (V \cup \{s\}, \bar{E})$

· $\bar{E} = E \cup \{(s,v) \mid v \in V\}$ $\underline{h(v) = \delta(s,v)}$

Repesagem de Johnson

- Encontrar uma função de alturas $h : V \to \mathbb{R}$:

  $G = (V, E, w) \Rightarrow G = (V, E, \hat{w})$

  onde: $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$

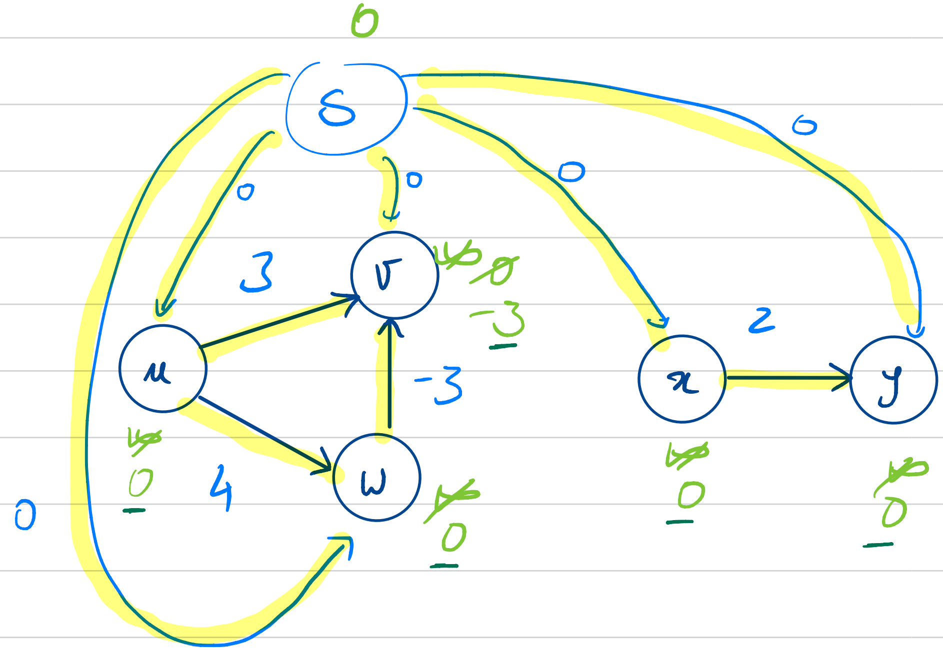
- Quem é h?

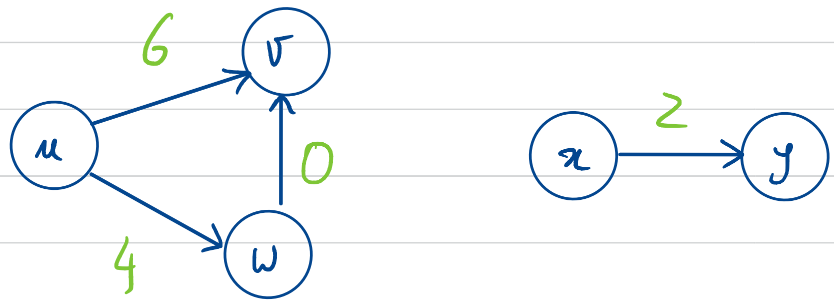
  Grafo estendido: $\bar{G} = (V \cup \{s\}, \bar{E})$

  $\cdot\ \bar{E} = E \cup \{(s, v) \mid v \in V\}$

- Altura de Johnson:

  $h(v) = \delta_{\bar{G}}(s, v)$





Repesagem de Johnson - Corretude

1. Se G não contém ciclos negativos, $\hat{G}$ não contém arcos com pesos negativos.

2. Se p é um caminho mais curto em G então p é um caminho mais curto em $\hat{G}$

3. Se G contém um ciclo negativo, então $\hat{G}$ tb contém um ciclo negativo

1.

Repesagem de Johnson – Correção

1. Se G não contém ciclos negativos, $\hat{G}$ não contém arcos com pesos negativos.

2. Se p é um caminho mais curto em G então p é um caminho mais curto em $\hat{G}$

3. Se G contém um ciclo negativo, então $\hat{G}$ tb contém um ciclo negativo

1.
$$\begin{aligned}\hat{w}(u,v) &= w(u,v) + h(u) - h(v)\\ &= w(u,v) + \delta(s,u) - \delta(s,v)\\ &\geq 0\end{aligned}$$

| Desigualdade Triangular

Repesagem de Johnson - Corretude

1. Se G não contém ciclos negativos, $\hat{G}$ não contém arcos com pesos negativos.

2. Se p é um caminho mais curto em G
   então p é um caminho mais curto em $\hat{G}$

3. Se G contém um ciclo negativo, então $\hat{G}$
   tb contém um ciclo negativo

2.

Repesagem de Johnson – Correcção

1. Se G não contém ciclos negativos, $\hat{G}$ não contém arcos com pesos negativos.

2. Se p é um caminho mais curto em G então p é um caminho mais curto em $\hat{G}$

3. Se G contém um ciclo negativo, então $\hat{G}$ tb contém um ciclo negativo

2. Seja $p = \langle v_1, \ldots, v_n \rangle$ um caminho mais curto em G.

$$\hat{w}(p) = \sum_{i=1}^{n-1} \hat{w}(v_i, v_{i+1})$$

$$= \sum_{i=1}^{n-1} w(v_i, v_{i+1}) + \sum_{i=1}^{n-1} \left(h(v_i) - h(v_{i+1}) \right)$$

$$= w(p) + h(v_1) - h(v_n)$$

Suponhamos, por contradição, que p é o caminho mais curto em G, mas existe p' mais curto que p em $\hat{G}$:

$$\hat{w}(p') < \hat{w}(p)$$

Sabemos que:

$$\hat{w}(p) = w(p) + h(v_0) - h(v_n)$$

$$\hat{w}(p') = w(p') + h(v_0) - h(v_n)$$

De onde concluímos que:

$$w(p') < w(p) \quad \text{⚡}$$

Repesagem de Johnson – Correcção

1. Se G não contém ciclos negativos, $\hat{G}$ não contém arcos com pesos negativos.

2. Se p é um caminho mais curto em G então p é um caminho mais curto em $\hat{G}$

3. Se G contém um ciclo negativo, então $\hat{G}$ tb contém um ciclo negativo

Seja $p = \langle v_0, \ldots, v_n \rangle$ com $v_n = v_0$ um ciclo em G

$$\hat{w}(p) = w(p) + h(v_0) - h(v_n)$$
$$= w(p)$$

Algoritmo de Johnson

- Calcular os caminhos curtos entre todos os pares em $G = (V, E, w)$

[Complexidade]

① Calcular o grafo estendido $\bar{G}_s = (\bar{V}, \bar{E}, \bar{w})$ com $s \notin V$

② Usar o algoritmo de Bellman-Ford p/ determinar a função de altura h
Se o algoritmo de Bellman-Ford retornar falso, o algoritmo de Johnson
tb retorna falso.

③ Calcular o grafo repesado $\hat{G} = (V, \hat{E}, \hat{w})$

④ Para cada vértice $u \in V$, usar o algoritmo de Dijkstra
D_{uv} e Π_{uv} para todo $v \in V$

⑤ Retornar D e Π.

Algoritmo de Johnson

- Calcular os caminhos curtos entre todos os pares em $G = (V, E, w)$

[Complexidade]

$O(V E \lg V)$

① Calcular o grafo estendido $\bar{G}_s = (\bar{V}, \bar{E}, \bar{w})$ com $s \notin V$ $O(V+E)$

② Usar o algoritmo de Bellman-Ford p/ determinar a função de altura h $\| O(E.V)$

Se o algoritmo de Bellman-Ford retornar falso, o algoritmo de Johnson tb retorna falso.

③ Calcular o grafo repesado $\hat{G} = (V, \hat{E}, \hat{w})$ $\| O(V+E)$

④ Para cada vértice $u \in V$, usar o algoritmo de Dijkstra D_{uv} e Π_{uv} para todo $v \in V$ $\| O(E \lg V)$ $\| O(V.E.\lg V)$

⑤ Retornar D e Π.

Árvores Abrangentes de Menor Custo
(Minimum Spanning Trees (MSTs))

Definição [Árvore Abrangente de Menor Custo]

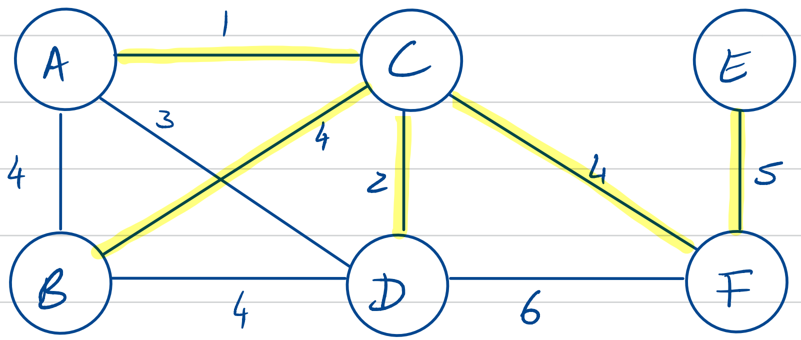
- Seja $G = (V, E, w)$ um grafo dirigido pesado, uma árvore abrangente de G é um subconjunto dos arcos de G, $T \subseteq E$, tal que:
  - T não contém ciclos
  - T "toca" em todos os vértices de G

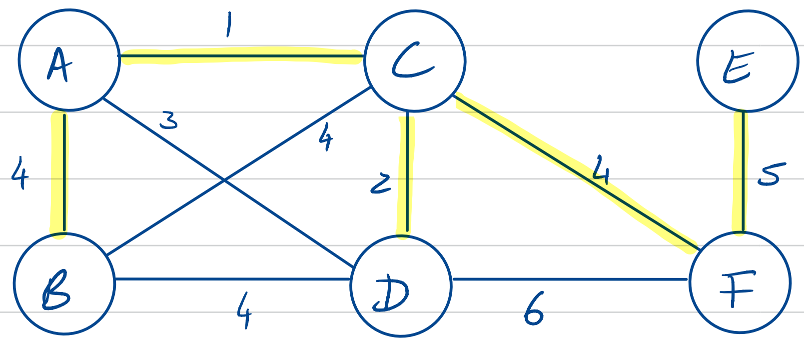
- Dado uma árvore abrangente T, o peso de T é definido como a soma dos pesos de T:

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

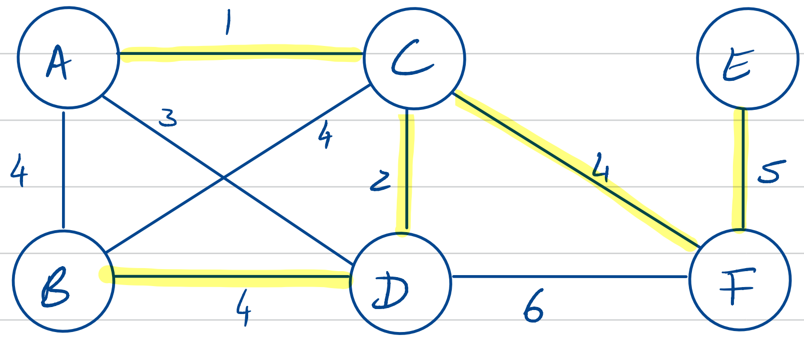
- Uma árvore abrangente de menor custo (MST) é uma árvore abrangente de peso mínimo.

Árvores Abrangentes de Menor Custo
(Minimum Spanning Trees (MSTs))





- $w(T) = 16$



Algoritmo de Prim

Prim(G, w, r)

  for each $v \in G.V$

    $v.key = \infty$; $v.\pi = Nil$

  $r.key := 0$;

  let Q be a min-priority queue with content G.V

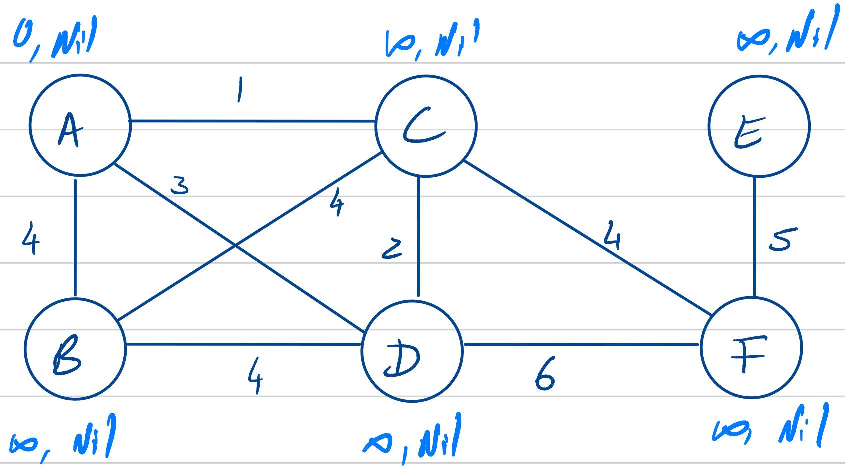
  while $Q \neq [\,]$

    let u = ExtractMin(Q)

    for each $v \in G.Adj[u]$

      if $(v.key > w(u,v))$ && $v \in Q$

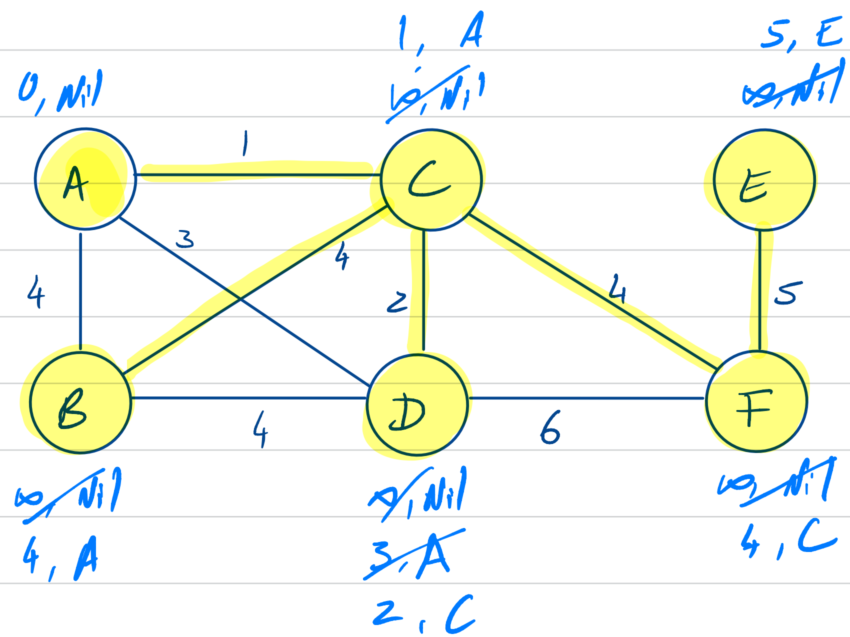
        $v.key := w(u,v)$; $v.\pi := u$



Algoritmo de Prim

Prim(G, w, r)

```
for each v ∈ G.V
    v.key = ∞; v.π = Nil
r.key := 0;
let Q be a min-priority queue with content G.V
while Q ≠ []
    let u = ExtractMin(Q)
    for each v ∈ G.Adj[u]
        if (v.key > w(u,v)) && v ∈ Q
            v.key := w(u,v); v.π := u
```



Algoritmo de Prim

Prim (G, w, r)

for each $v \in G.V$

  $v.key = \infty$; $v.\pi = Nil$

$r.key := 0$;

let Q be a min-priority queue with content G.V

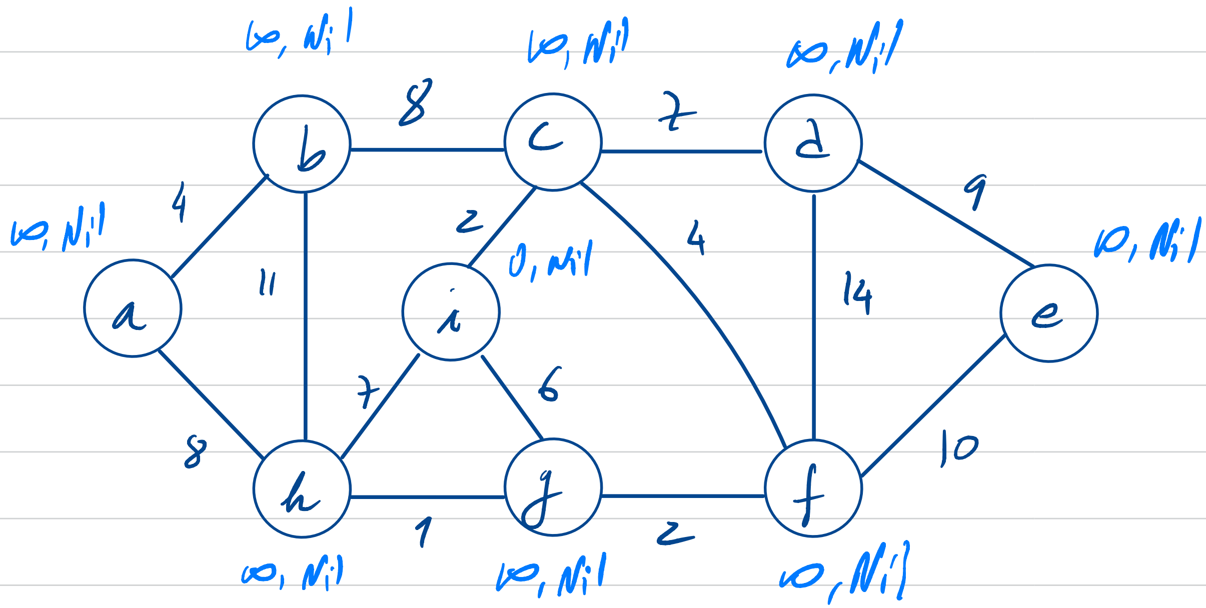
while $Q \neq [\,]$

  let u = ExtractMin (Q)

  for each $v \in G.Adj[u]$

    if ($v.key > w(u,v)$) && $v \in Q$

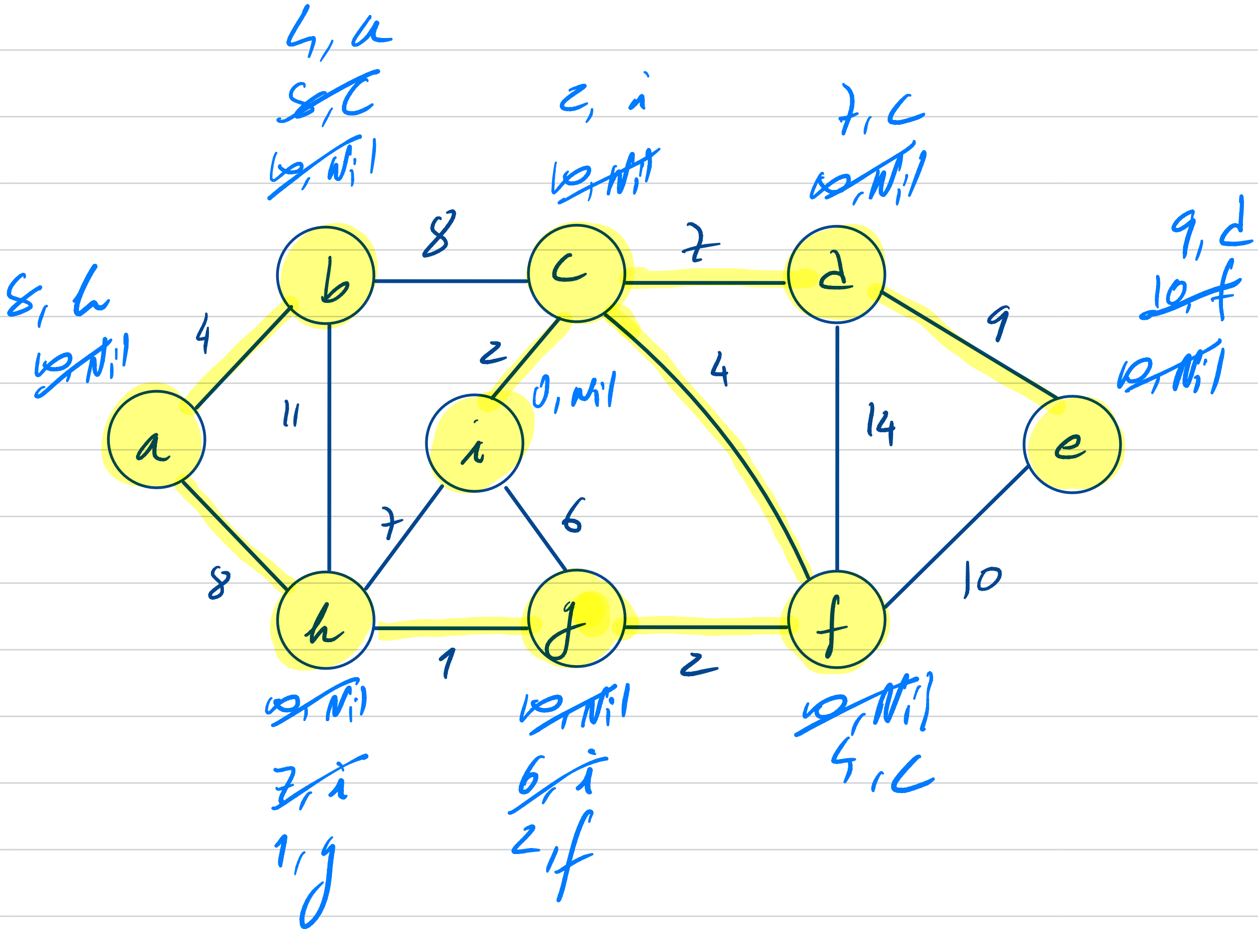
      $v.key := w(u,v)$; $v.\pi := u$



Algoritmo de Prim

Prim(G, w, r)

```
for each v ∈ G.V
    v.key = ∞; v.π = Nil
r.key := 0;
let Q be a min-priority queue with content G.V
while Q ≠ []
    let u = ExtractMin(Q)
    for each v ∈ G.Adj[u]
        if (v.key > w(u,v)) && v ∈ Q
            v.key := w(u,v); v.π := u
```



Algoritmo de Prim

Análise de Complexidade

```
Prim(G, w, r)
  for each v ∈ G.V
    v.key = ∞; v.π = Nil
  r.key := 0;
  let Q be a min-priority queue with content G.V
  while Q ≠ [ ]
    let u = ExtractMin(Q)
    for each v ∈ G.Adj[u]
      if (v.key > w(u,v)) && v ∈ Q
        v.key := w(u,v); v.π := u
```

Algoritmo de Prim

Prim(G, w, r)

for each $v \in G.V$ — $O(|V|)$

  $v.key = \infty$; $v.\pi = Nil$

$r.key := 0$;

let Q be a min-priority queue with content $G.V$ — $O(|V|)$

while $Q \neq [\,]$ $\longrightarrow$ nº de iterações $O(|V|)$

  let u = ExtractMin(Q)

  for each $v \in G.Adj[u]$ $\rightarrow$ nº TOTAL de iterações: $O(|E|)$

    if $(v.key > w(u,v))$ && $v \in Q$

      $v.key := w(u,v)$; $v.\pi := u$

      $\hookrightarrow O(\lg |V|)$

Análise de Complexidade

Custo total do ciclo for: $O(|E| \cdot \lg |V|)$

Custo total: $O(|E| \cdot \lg |V|)$

Algoritmo de Prim

Prim(G, w, r)

for each $v \in G.V$

$v.key = \infty$; $v.\pi = Nil$

$r.key := 0$;

$A := \emptyset$

let Q be a min-priority queue with content $G.V$

while $Q \neq [\,]$

let u = ExtractMin(Q)

if$(u \neq r)$ $A := A \cup \{(u.\pi, u)\}$

for each $v \in G.Adj[u]$

if $(v.key > w(u,v))$ && $v \in Q$

$v.key := w(u,v)$; $v.\pi := u$

Análise da Correcção

(I1) $A = \{(v.\pi, v) \mid v \notin Q \wedge v.\pi \neq Nil\}$ é um subconjunto de uma MST

(I2) $\forall u \in Q.$

$u.key = \min\{w(u,v) \mid v \in V \setminus Q\}$

(I3) $\forall v \in V.$

$v.\pi \neq Nil \Rightarrow w(v.\pi, v) = v.key$

Invariante do Algoritmo de Prim

I_1 $A = \{ (v.\pi, v) \mid v \notin Q \wedge v.\pi \neq Nil \}$
é um subconjunto de uma MST

I_2 $\forall u \in Q.$
$u.key = \min \{ w(u,v) \mid v \in V \setminus Q \}$

I_3 $\forall v \in V.$
$v.\pi \neq Nil \Rightarrow w(v.\pi, v) = v.key$

Inicialização (fim da primeira iteração)

I_1 $A = \emptyset$ é subconjunto de uma MST ✓

I_2 $V \setminus Q = \{r\}$

$\forall v \in N(r).\ v.key = w(r,v)$
$\forall v \notin N(r).\ v.key = \infty$ ✓

I_3

$v.\pi \neq Nil \Leftrightarrow v.\pi = r$
$\Leftrightarrow v.key = w(r,v)$ ✓

Invariante do Algoritmo de Prim

I1 $A = \{ (v.\pi, v) \mid v \notin Q \wedge v.\pi \neq Nil \}$
é um subconjunto de uma MST

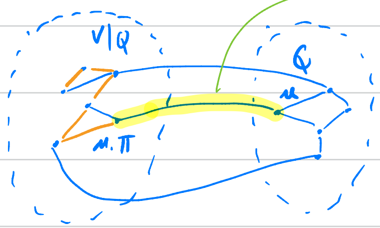
I2 $\forall u \in Q.$
$u.key = \min \{ w(u,v) \mid v \in V \setminus Q \}$

I3 $\forall v \in V.$
$v.\pi \neq Nil \Rightarrow w(v.\pi, v) = v.key$

Manutenção

I1 $A' = A \cup \{ (u.\pi, u) \}$

- Há que provar que $(u.\pi, u)$ é seguro para A
- Temos de encontrar um corte $(S, V \setminus S)$ que respeita A e para o qual $(u.\pi, u)$ seja <u>leve</u>.
- $(V \setminus Q, Q)$ respeita A



$(u.\pi, u)$ é arco leve q cruza $(V \setminus Q, Q)$
↳ <u>Invariante 2</u>