Sumário

- Lema da Relaxação de Caminho
- Algoritmo de Bellman Ford
- DAG Shortest Paths

Aula 11

Relaxação de Caminho

- $\langle v_0, \dots, v_n \rangle$ é um caminho mais curto em G:

t_0: $v_0 \to v_1 \to v_2 \to \cdots \to v_{n-1} \to v_n$

$\delta(s, v_0)$

t_1: $v_0 \to v_1 \to v_2 \to \cdots \to v_{n-1} \to v_n$

$\delta(s, v_0)$

t_2: $v_0 \to v_1 \to v_2 \to \cdots \to v_{n-1} \to v_n$

$\delta(s, v_0)$

$\vdots$

t_{n-1}: $v_0 \to v_1 \to v_2 \to \cdots \to v_{n-1} \to v_n$

$\delta(s, v_0)$

t_n: $v_0 \to v_1 \to v_2 \to \cdots \to v_{n-1} \to v_n$

$\delta(s, v_0)$

Relaxação de Caminho

- $\langle v_0, \dots, v_n \rangle$ é um caminho mais curto em G:

t_0: $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{n-1} \rightarrow v_n$

$\delta(s, v_0)$?   ?   ?   ?

t_1: $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{n-1} \rightarrow v_n$

$\delta(s, v_0)$ $\delta(s, v_1)$?   ?   ?

t_2: $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{n-1} \rightarrow v_n$

$\delta(s, v_0)$ $\delta(s, v_1)$ $\delta(s, v_2)$?   ?

$\vdots$

t_{n-1}: $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{n-1} \rightarrow v_n$

$\delta(s, v_0)$ $\delta(s, v_1)$ $\delta(s, v_2)$ $\delta(s, v_{n-1})$?

t_n: $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{n-1} \rightarrow v_n$

$\delta(s, v_0)$ $\delta(s, v_1)$ $\delta(s, v_2)$ $\delta(s, v_{n-1})$ $\delta(s, v_n)$

Relaxação de Caminho

Lema [Relaxação de Caminho]

Seja $G=(V,E,w)$ um grafo pesado e $p=\langle v_0, \dots, v_n \rangle$ um caminho mais curto entre v_0 e v_n. Se os arcos $(v_0, v_1), \dots, (v_{n-1}, v_n)$ forem relaxados por ordem. Então, depois das n operações de relaxação, concluímos que $v_i.d = \delta(s, v_i)$ para todo $0 \leq i \leq n$.

Prova:

- Provamos o resultado por indução em $\underline{\underline{n}}$.

- Base $n=0$ $\quad p=\langle v_0 \rangle$ e $v_0.d = \delta(s, v_0) \longrightarrow \checkmark$

- Passo $\underline{n+1}$
  - $p' = \langle v_0, \dots, v_n, v_{n+1} \rangle$ é um caminho mais curto entre v_0 e v_{n+1}
  - $p = \langle v_0, \dots, v_n \rangle$ é um caminho mais curto entre v_0 e v_n (sub-estrutura óptima)
  - Aplicando a hipótese de indução, concluímos que depois das n primeiras operações de relaxação $v_i.d = \delta(s, v_i)$ para $0 \leq i \leq n$. Resta provar que depois da última operação de relaxação $v_{n+1}.d = \delta(s, v_{n+1})$.

    Observamos que, depois da última operação de relaxação
    $$v_{n+1}.d \leq v_n.d + w(v_n, v_{n+1})$$
    $$v_{n+1}.d \leq \delta(s, v_n) + w(v_n, v_{n+1})$$
    $$v_{n+1}.d \leq \delta(s, v_{n+1})$$

De onde segue que:
$$v_{n+1}.d = \delta(s, v_{n+1})$$

Algoritmo de Bellman-Ford

Bellman-Ford (G, s)

InitializeSingleSource (G, s) $\}$ $O(V)$

for $i := 1$ to $|G.V| - 1$

for each $(u, v) \in G.E$

Relax $(G.w, u, v)$

$\}$ $O(V \cdot E)$

for each $(u, v) \in E$

if $v.d > u.v + G.w(u, v)$

return false

return true

$\}$ $O(E)$

Análise de Complexidade

Total: $O(V \cdot E)$

Algoritmo de Bellman-Ford - Correcção

(1) Se o grafo não tem ciclos negativos, o algoritmo de Bellman-Ford calcula as distâncias correctas:

$$\forall v \in V.\ v.d = \delta(s, v)$$

Prova:

- Considere-se um vértice qualquer $v \in V$. Seja p o caminho mais curto que liga s a v ($s \overset{p}{\leadsto} v$).

- Temos de provar que no fim do algoritmo de BF, $v.d = \delta(s, v)$.

- Sabemos que o caminho p tem no máximo $|V|-1$ arcos.

- Depois de $|V|-1$ etapas de relaxação nas quais relaxamos todos os arcos do grafo, existe uma subsequência de relaxações correspondentes ao caminho mais curto entre s e v.

- O resultado segue pelo Lema da Relaxação de Caminho.

Algoritmo de Bellman-Ford - Correcção

(2) O algoritmo de Bellman-Ford retorna false sse o grafo contém um ciclo negativo

(2.1) $BF(G,s) = false \implies G$ contém um ciclo negativo

$\iff G$ não contém um ciclo negativo $\implies BF(G,s) = true$

$\Updownarrow$

$\forall (u,v) \in E.\ \underline{v.d \leq u.d + w}(u,v)$

- G não contém um ciclo negativo (hyp)
- $\forall v \in V.\ v.d = \delta(s,v)$
- $\forall (u,v) \in E.\ \delta(s,v) \leq \delta(s,u) + w(u,v)$ (desigualdade triangular)
- $\forall (u,v) \in E.\ v.d \leq u.d + w(u,v)$ (correcção: 1)

Algoritmo de Bellman-Ford - Correcção

(2) O algoritmo de Bellman-Ford retorna false sse o grafo contém um ciclo negativo

(2.1) G contém um ciclo negativo $\Rightarrow BF(G, s) = \text{false}$

- Suponhamos, por contradição, q̄ G contém um ciclo negativo e $BF(G,s) = \text{true}$

- Seja $p = \langle v_0, v_1, \dots, v_{n-1}, v_n \rangle$, com $v_n = v_0$, o ciclo negativo.

$$\forall 0 \leq i \leq n-1.\ (v_i, v_{i+1}) \in E$$

$$\Rightarrow \forall 0 \leq i \leq n-1.\ v_{i+1}.d \leq v_i.d + w(v_i, v_{i+1})$$

$$\Rightarrow \sum_{i=0}^{n-1} v_{i+1}.d \leq \sum_{i=0}^{n-1} v_i.d + w(v_i, v_{i+1})$$

$$\Rightarrow \underbrace{\sum_{i=0}^{n-1} v_{i+1}.d \leq \sum_{i=0}^{n-1} v_i.d}_{=} + \underbrace{\sum_{i=0}^{n-1} w(v_i, v_{i+1})}_{\text{peso do caminho}}$$

$$\Rightarrow w(p) \geq 0$$

(contradição)

$\hookrightarrow p$ não é ciclo negativo!!

Caminhos Mais Curtos em Grafos Acíclicos

Lema [Caminho Mais Curto em DAG]

Seja $G=(V,E)$ um grafo acíclico e $\langle v_0, \dots, v_n \rangle$ uma ordenação topológica de G; então qualquer caminho em G é uma subsequência de $\langle v_0, \dots, v_n \rangle$.

Prova:

- Suponhamos q existe um caminho em G, $\langle u_1, \dots, u_k \rangle$ tal que $\langle u_1, \dots, u_k \rangle$ não é uma subsequência de $\langle v_1, \dots, v_n \rangle$. Concluímos q existe um índice i tal que u_i ocorre depois de u_{i+1} na ordenação topológica.
- $(u_i, u_{i+1}) \in E \Rightarrow f(u_i) > f(u_{i+1})$

Observação Chave:

- Todos os caminhos mais curtos são subsequências da Ordenação Topológica

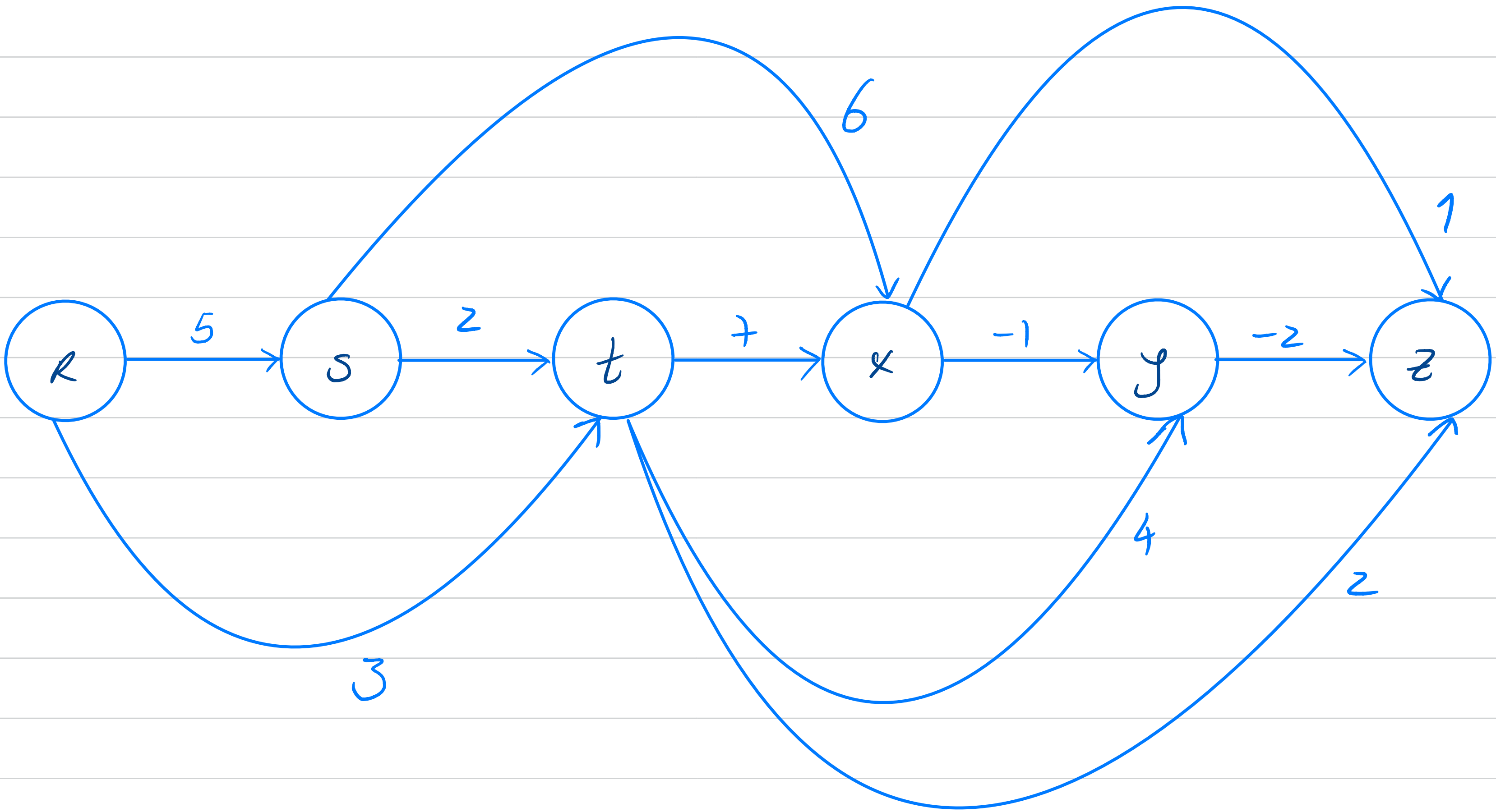
+

- Lema da Relaxação de Caminho

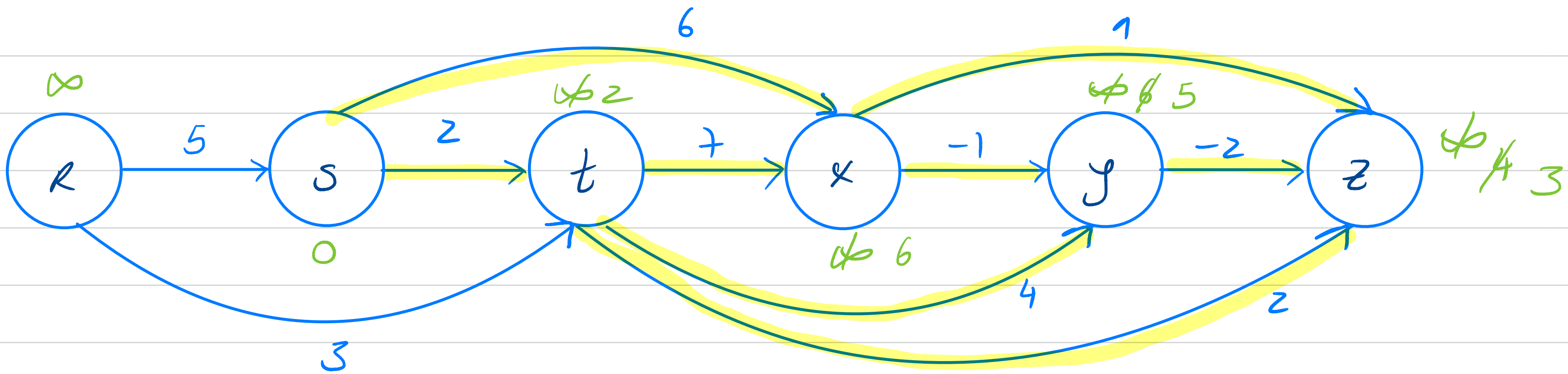
$\Downarrow$

- Em DAGs podemos calcular caminhos mais curtos, relaxando os arcos do grafo por ordem topológica!

Caminhos Mais Curtos em Grafos Acíclicos



Caminhos Mais Curtos em Grafos Acíclicos



Caminhos Mais Curtos em Grafos Acíclicos

DAG-Shortest-Paths(G, s)

1 Initialize-Single-Source(G, s)

2 for each $u \in G.V$ in topological order 4

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

Relax(G.w, u, v)

3

Análise de Complexidade

1 $O(V)$

2 $O(V)$

3 $O(E)$

4 $O(V+E)$

$O(V+E)$

Algoritmo de Johnson

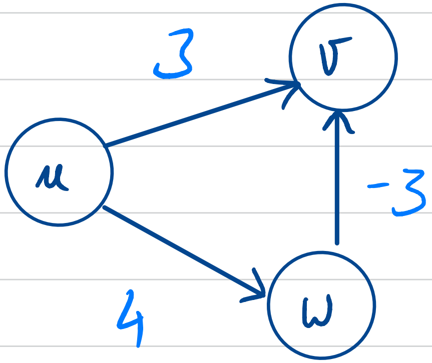
Ideia

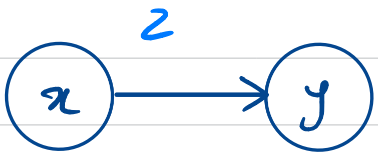
- Dado um grafo G com pesos negativos, calcular um grafo G' cujos caminhos mais curtos coincidem com os de G mas **sem** arcos com pesos negativos

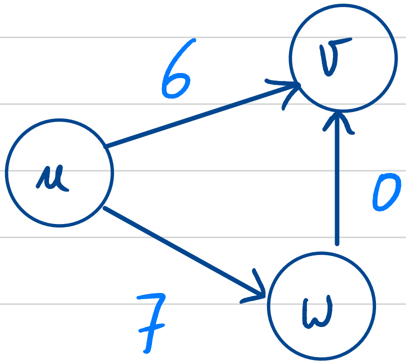
Repesagem dos Arcos

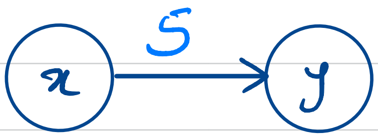
- Aplicar o algoritmo de Dijkstra a todos os vértices de G'

Repesagem dos Arcos - 1ª Ideia









Ideia Naïf:

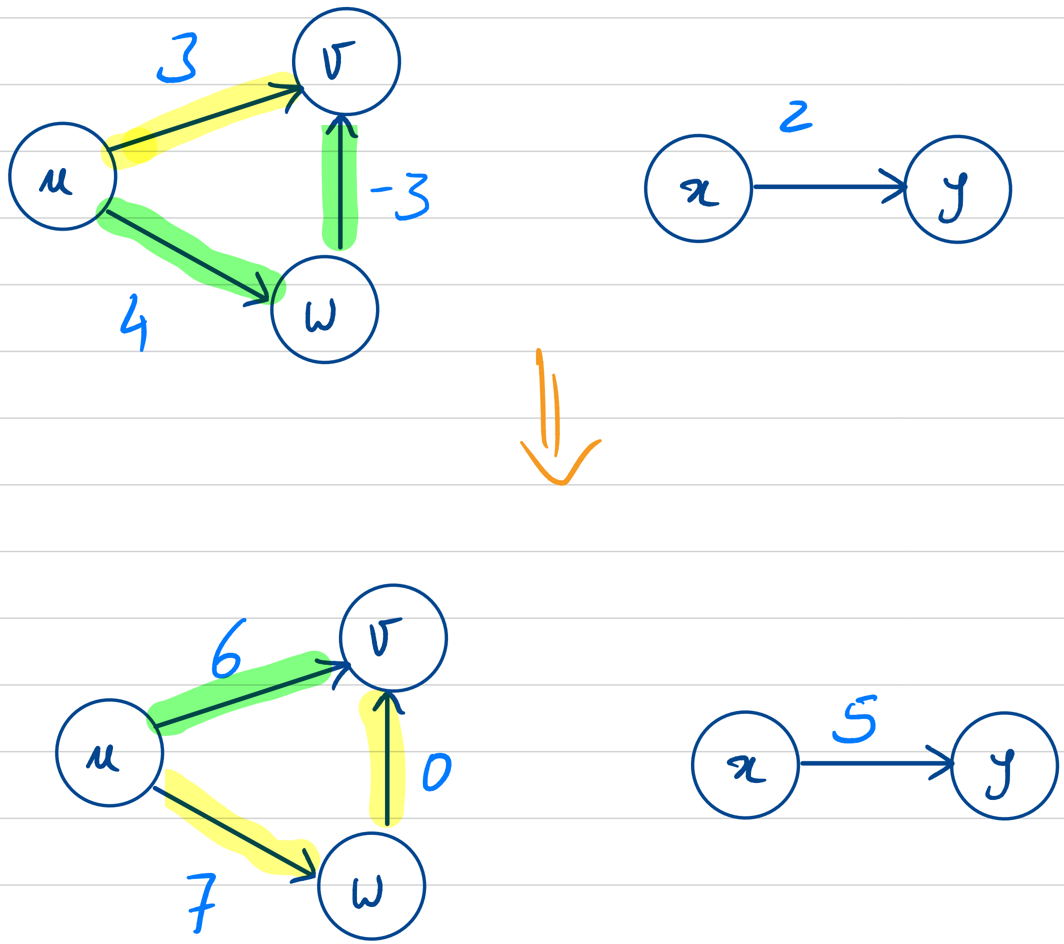
- Somar o módulo do maior dos pesos negativos a todos os arcos

Funcionou?

Repesagem dos Arcos - 1ª Ideia

Ideia Naif:

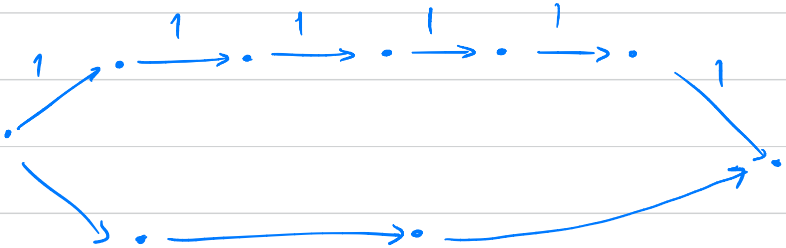
- Somar o módulo do maior dos pesos negativos a todos os arcos



Funcionou?

Não! Este método penaliza caminhos maiores!

Repesagem dos Arcos - 1ª Ideia



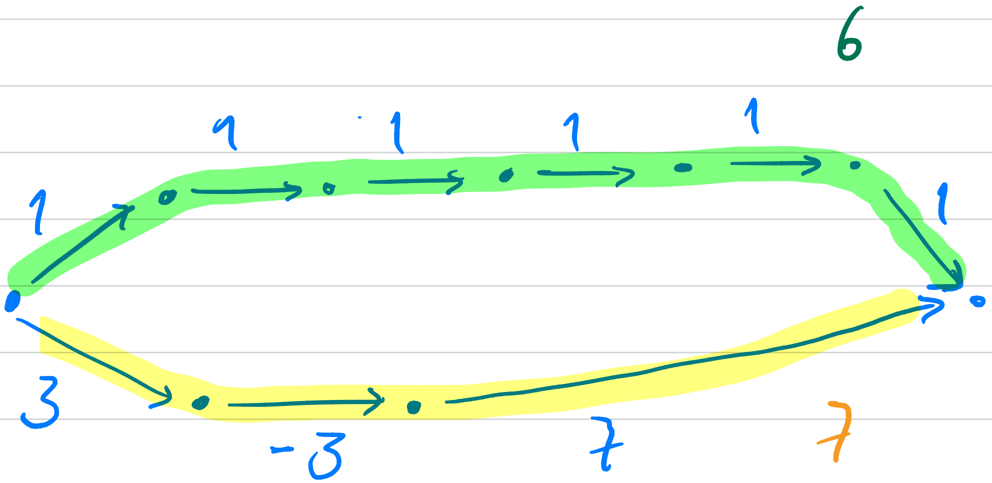
Ideia Naif:

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

Funcionou?

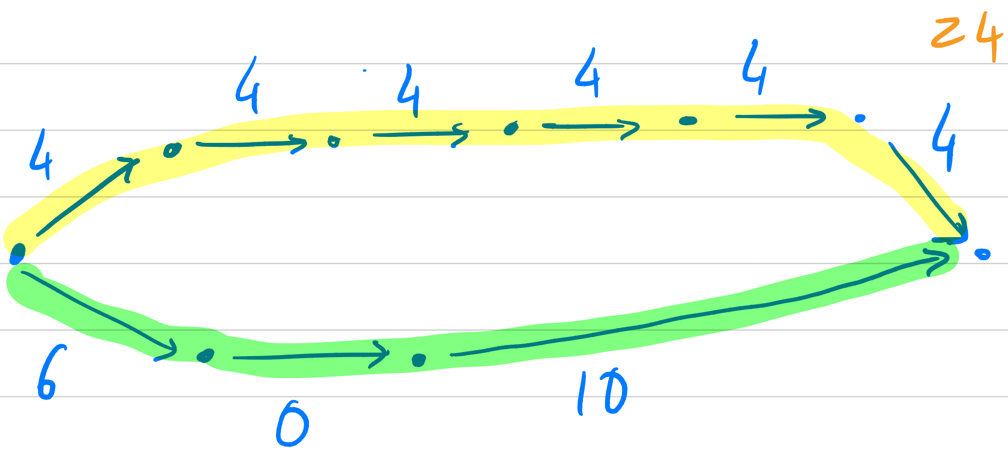
Não! Este método penaliza caminhos maiores!

Repesagem dos Arcos - 1ª Ideia



Ideia Naif:

- 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: $\overline{G} = (V \cup \{s\}, \overline{E})$

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

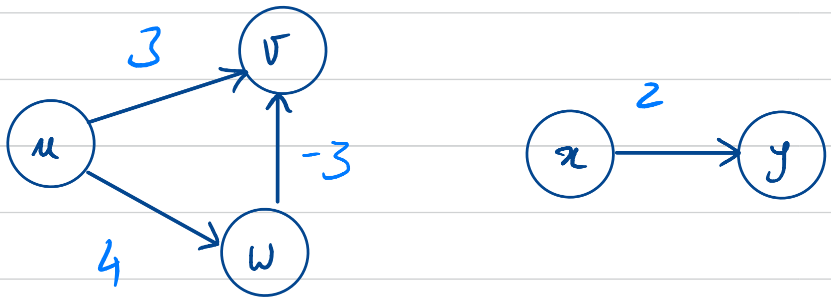
- $\overline{w}(u,v) = \begin{cases} w(u,v) & \text{se } (u,v) \in E \\ 0 & \text{se } u = s \wedge v \neq s \end{cases}$

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

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

Repesagem de Johnson

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

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

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

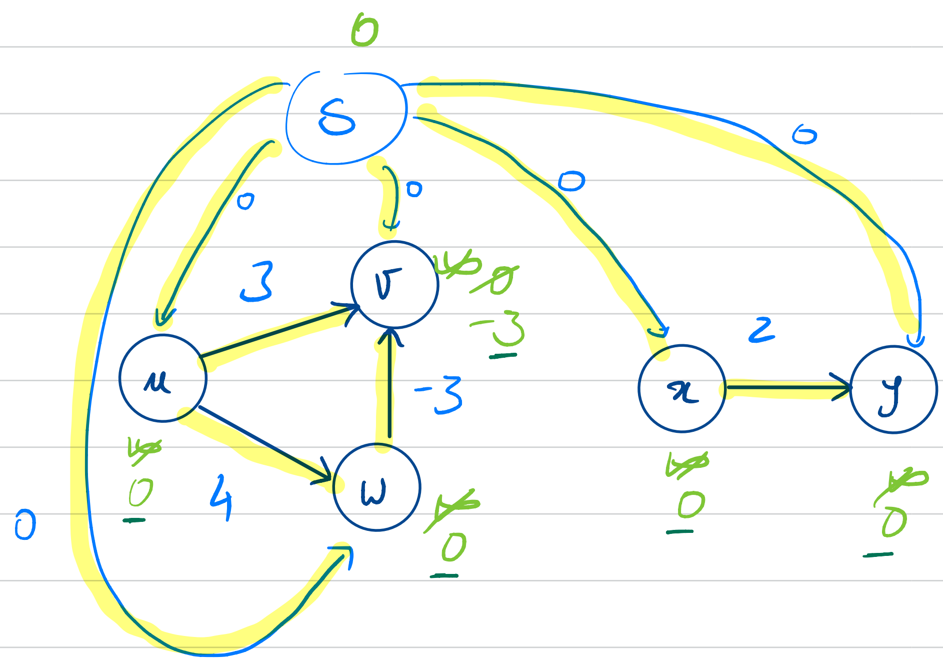
- Quem é h?

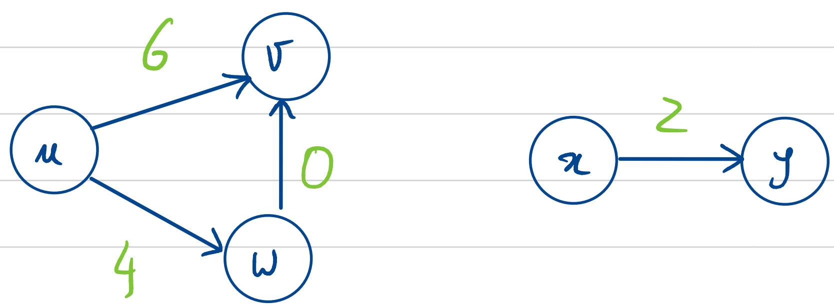
  Grafo estendido: $\bar{G} = (V \cup \{s\}, \bar{E})$

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

- Algoritmo 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 – 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

① $\hat{w}(u,v) = w(u,v) + h(u) - h(v)$
$= w(u,v) + \delta(s,u) - \delta(s,v)$ | Desigualdade Triangular
≥ 0

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.

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 $p = \langle v_1, \dots, v_n \rangle$ 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 $G = (V, E, w)$

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

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]

$O(VE \lg V)$

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.