

## Aula 7

- Teoria da Relaxação do Caminho
- Algoritmo de Bellman-Ford
  - Complexidade
  - Convergência
- Caminhos em grafos acíclicos
- Caminhos Mais Curtos entre todas as pares
  - Algoritmo Recursivo
  - Algoritmo de Floyd-Warshall

## Caminhos Mais Curtos

### Definição 3 [ Grafo Pesoado ]

- Um grafo pesoado é um grafo  $G = (V, E)$  com uma função de peso  $w: E \rightarrow \mathbb{R}$ ; para facilitar a notação escrevemos  $G = (V, E, w)$ .

- Seja  $G = (V, E, w)$  um grafo pesoado, o peso de um caminho  $p = \langle v_0, \dots, v_n \rangle$  em  $G$  é dado por:

$$w(p) = \sum_{i=0}^{n-1} w(v_i, v_{i+1})$$

- Seja  $G = (V, E, w)$ , e função  $S: V \times V \rightarrow \mathbb{R}$  impõe cada par de vértices no peso do caminho mais curto q̄ os liga:

$$S(u, v) = \begin{cases} w(p) & \text{se } u \xrightarrow{p} v \\ \infty & \text{caso contrário} \end{cases}$$

### Definição 4 [ Problemas de Caminhos Mais Curtos ]

- Caminhos mais curtos de origem única

Dado um vértice  $s$  determinam para todo o vértice  $v \in V$  o caminho  $p$  tal que  $s \xrightarrow{p} v$  e  $S(s, v) = w(p)$ .

- Caminhos mais curtos de origem única e ponto final

Dados dois vértices  $s$  e  $t$  determinam o caminho  $p$  tal que  $s \xrightarrow{p} t$  e  $S(s, t) = w(p)$ .

- Caminhos mais curtos entre todos os pares

Determinam p̄ todos os pares de vértices  $u, v \in V$ , o caminho mais curto que os une.

### Lema 1 [ Caminhos mais curtos - subestrutura óptima ]

Seja  $G = (V, E, w)$  um grafo pesoado e seja  $p = \langle v_0, \dots, v_n \rangle$  um caminho mais curto em  $G$ ,

temos q̄:

$$\forall i: 0 \leq i \leq n. \quad \forall i < j \leq n. \quad w(\langle v_i, \dots, v_j \rangle) = S(v_i, v_j) \quad \text{se } \langle v_i, \dots, v_j \rangle \text{ é um caminho mais curto entre } v_i \text{ e } v_j$$

Prova:

- Suponhamos q̄ existam  $i < j$  tais que  $\langle v_i, \dots, v_j \rangle$  não é um caminho mais curto entre  $v_i$  e  $v_j$ . Seja  $p'$  o caminho mais curto entre  $v_i$  e  $v_j$ , temos que:

$$w(p') < w(\langle v_i, \dots, v_j \rangle) \quad \text{e} \quad v_i \xrightarrow{p'} v_j$$

- Trabalhe:

$$p'' = \langle v_0, \dots, v_{i-1} \rangle + p' + \langle v_{j+1}, \dots, v_n \rangle$$

é um caminho entre  $v_0$  e  $v_n$ .

$$w(p'') = w(\langle v_0, \dots, v_{i-1} \rangle) + w(p') + w(\langle v_{j+1}, \dots, v_n \rangle)$$

$$w(p) = w(\langle v_0, \dots, v_{i-1} \rangle) + w(\langle v_i, \dots, v_j \rangle) + w(\langle v_{j+1}, \dots, v_n \rangle)$$

De onde segue que:  $w(p'') \leq w(p)$ , q̄ implica q̄  $p$  não é um caminho mais curto.

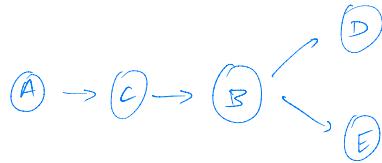
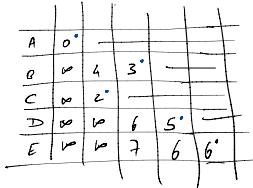
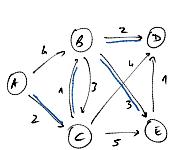
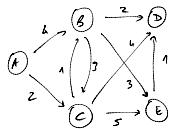
(2)

## Lema 5 [Desigualdade Triangular]

$$(u, v) \in E \Rightarrow \delta(s, v) \leq \delta(s, u) + w(u, v)$$

P prova:

## Exemplo 2 [Dijkstra]



## Algorithm 2 [Dijkstra]

Dijkstra( $G, s$ )

for each  $v \in G.V$       Initialize Single Source  
 $v.d = \infty; v.\pi = \text{Nil}$

 $s.d = 0$ let  $Q$  be a min priority queue with content  $V$  $S = \emptyset$ while ( $Q \neq \emptyset$ )   $m = \text{ExtractMin}(Q)$    $S = S \cup \{m\}$   for each  $v \in G.Adj[m]$      $\text{Relax}(m, v, G.w)$ Relax( $m, v, w$ )if ( $v.d > m.d + w(m, v)$ )   $v.d = m.d + w(m, v)$    $v.\pi = m$ 

## [Complexidade]

• Inicialização:  $O(V)$ • Outer loop:  $O(|V| \times \log |V|) = O(V \log V)$ • Inner loop:  $O(|E| \times \log |V|) = O(E \log V)$ total:  $O((V+E) \log V)$  $O(E \log V)$  porque  $|E| > |V|$ 

## [Correção]

Invariante 1:  $\forall v \in G.V. v.d \geq \delta(s, v)$ 

• Início:

-  $v = s \Rightarrow s.d = 0 = \delta(s, s)$ -  $v \neq s \Rightarrow v.d = \infty \geq \delta(s, v)$ 

• Manutenção:

- Se  $v.d$  mudar num iteração do loopsignifica que o parco  $(v, r)$  foi relaxado.-  $r.d = m.d + w(m, r)$  (Relax) $r.d \geq \delta(s, m) + w(m, r)$  (Invariante) $r.d \geq \delta(s, r)$  (Desigualdade Triangular)

- Invariante 2:  $\forall r \in S \quad r.d = \delta(s, r) \wedge S = V \setminus Q$

- Início:  $S = \emptyset \wedge Q = V \Rightarrow r$

- Manutenção:

$$S' = S \cup \{y\}, \text{ onde } y.d = \min \{r.d \mid r \in Q\}$$

Há que provar  $\underline{\delta}(s, y) = y.d$

- Suponhamos, por contradição,  $\underline{\delta}(s, y) \neq \delta(s, y)$ . Então, pelo Invariante 1, sabemos  $\underline{\delta}(s, y) > \delta(s, y)$ . Assim sendo, existe um caminho  $p$  em  $G$  tal que o  $\underline{\delta}(s, y) < w(p) = \delta(s, y) < y.d$ .

Seja  $\langle s, \dots, x, y \rangle$  esse caminho.

Há dois casos a considerar:  $x \in S \in x \notin S$ .

- $x \in S$



Todas as arestas  $\underline{\delta}$  causam o caminho já foram rebaixadas, pelo  $\underline{\delta}$ :  $y.d \leq x.d + w(x, y)$   
 $= \delta(s, x) + w(x, y) \quad \text{) porque } \langle s, \dots, x, y \rangle \text{ é um caminho mais curto.}$   
 $= \delta(s, y) \quad \therefore$

- $x \notin S$



a)  $p = \langle s, \dots, z, z', \dots, x, y \rangle$

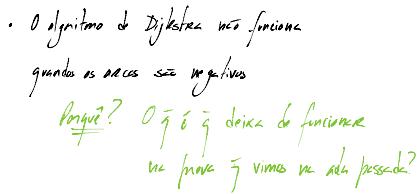
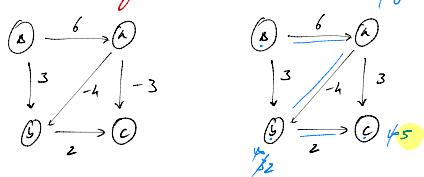
b)  $\begin{aligned} \delta(s, y) &= \delta(s, z) + w(z, z') + \delta(z', y) \\ &= z.d + w(z, z') + \delta(z', y) \quad \text{) porque a aresta } (z, z') \text{ já foi rebaixada} \\ &= z.d + \delta(z', y) \\ &\geq z'.d \end{aligned}$

c) Mas  $y$  é o nó em  $Q \setminus S$  com menor valor de  $d$ .

Por isso:  $y.d \leq z'.d$

Decorrendo que assumimos que  $\delta(s, y) < y.d$ , concluímos que:  $\delta(s, y) < z'.d$ , o que contradiz b).

## Exemplo 1 [Passos negativos]



## Lema 1 [Relaxação de caminho]

Sendo  $G = (V, E, w)$  um grafo peso e  $p = \langle v_0, \dots, v_n \rangle$  um caminho mais curto entre  $v_0 \in V_0$  e  $v_n \in V_n$ , se os arcos  $(v_0, v_1), \dots, (v_{n-1}, v_n)$  forem relaxados por ordem e  $v_i.d = \delta(s, v_i)$ , então depois de aplicadas as operações de relaxação temos  $\bar{q}$ :  $V_i \in V_i, v_i.d = \delta(s, v_i)$ .

Prova:

A prova faz-se por indução no comprimento do caminho.

-  $i=0$ :  $v_0.d = \delta(s, v_0)$  já é hipótese.

-  $nH$ :

- Após  $n$  relaxações temos  $\bar{q}$  || Hipótese de indução  
Vizinhos:  $v_i.d = \delta(s, v_i)$

- Depois de relaxar o arco  $(v_n, v_{n+1})$  temos  $\bar{q}$ :

$$d_{n+1} \leq v_n.d + w(v_n, v_{n+1})$$

$$= \delta(s, d) + w(v_n, v_{n+1})$$

$$= \delta(s, v_{n+1})$$

- Observando que  $v_{n+1}.d \geq \delta(s, v_{n+1})$ , concluímos  $\bar{q}$ :

$$v_{n+1}.d = \delta(s, v_{n+1})$$

## Algoritmo 1 [Bellman-Ford]

Bellman-Ford( $G, s$ )Initialize Single Source( $G, s$ )for  $i = 1$  to  $|G.V|-1$ for each  $(u, v) \in G.E$ Relax( $u, v, G.w$ )for each  $(u, v) \in G.E$ if  $v.d > u.d + w(u, v)$ 

return false

return true

Complexidade:

Inicialização:  $O(|V|)$ Loop Principal:  $O(|V| \cdot |E|)$ Loop Final:  $O(|E|)$  $O(|E|)$

(5)

## [Correctness]

- Temos de provar duas propriedades

[P1] Se o grafo não contém um ciclo negativo, no fim da execução temos:

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

[P2] O algoritmo retorna falso se o grafo dado como input

contém um ciclo negativo.

[P1] Qualquer que seja o vértice  $v \in V$ , o caminho mais curto entre  $s$  e  $v$  tem no máximo  $|V|-1$  arestas. Assim sendo, depois de  $|V|-1$  relaxações de todos os arcos do grafo, existe subsequência de relaxação correspondente ao caminho mais curto entre  $s$  e  $v$ , pelo que, usando o Lema 1, o resultado segue.

## [P2]

$\Rightarrow$  O algoritmo retorna falso  $\Rightarrow$  O grafo contém um ciclo negativo  
 $\Leftrightarrow$  O grafo não contém um ciclo negativo  $\Rightarrow$  O algoritmo retorna true.

- Se o grafo não contém um ciclo negativo, então concluímos a função  $d.P1$  que o algoritmo calcula os caminhos mais curtos pelo que, se qualquer arco  $(u, v) \in E$  se tem:  $u.d = \delta(s, u)$  e  $v.d = \delta(s, v)$ . Logo, da desigualdade triangular segue que:  $v.d \leq u.d + w(u, v)$ .

$\Leftarrow$  O grafo contém um ciclo negativo  $\Rightarrow$  o algoritmo retorna falso

- Separaremos que o grafo contém um ciclo negativo. Seja  $f = \langle v_1, \dots, v_k \rangle$

com  $v_1 = v_k$  esse ciclo. Dado que  $f$  é um ciclo negativo, temos:

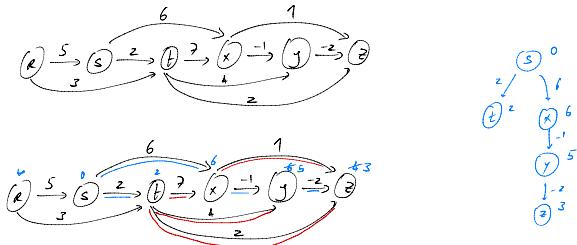
$$\sum_{i=1}^{k-1} w(v_i, v_{i+1}) < 0$$

- Suponhamos, para contradição, que o algoritmo retorna true; segue que para todos os arcos  $(u, v) \in E$ :  $v.d \leq u.d + w(u, v)$ .

- Somando as desigualdades calculadas por todos os vértices no caminho circular

$$\begin{aligned} \sum_{i=1}^{k-1} v_{i+1}.d &\leq \sum_{i=1}^{k-1} v_i.d + w(v_i, v_{i+1}) \\ &= \sum_{i=1}^{k-1} v_i.d + \sum_{i=1}^{k-1} w(v_i, v_{i+1}) \\ &= v_1 + \sum_{i=1}^{k-2} v_{i+1}.d + \sum_{i=1}^{k-1} w(v_i, v_{i+1}) \\ &= v_k + \sum_{i=1}^{k-2} v_{i+1}.d + \sum_{i=1}^{k-1} w(v_i, v_{i+1}) \\ &= \sum_{i=1}^{k-1} v_{i+1}.d + \sum_{i=1}^{k-1} w(v_i, v_{i+1}) \\ \Leftrightarrow \sum_{i=1}^{k-1} w(v_i, v_{i+1}) &\geq 0 \end{aligned}$$

### Exemplo 2 [ Caminhos Mais Curtos em Grafos Aciclicos ]



### Lema 2 [ Caminho em DAG ]

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

[Prova]

- Suponhamos que existe um caminho  $\langle v_{i_1}, \dots, v_{i_m} \rangle$  em  $G$  que não é uma subsequência de  $\langle v_0, \dots, v_n \rangle$ . Então existe um índice  $j \in \overline{q}$
- $v_{i_j} \neq v_{j_{m+1}} \neq v_{i_1}$ . Contudo, como  $(v_{i_j}, v_{j_{m+1}}) \in E$ , concluimos  $\exists v_{j_{m+1}} \neq v_{i_1}$ .

■

### Algoritmo 2 [ DAG-Shorest Paths ]

DAG-Shorest-Paths( $G, s$ )

Initialize-Single-Source( $G, s$ )

for each  $m \in G.V$  in topological order

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

Relax( $m, v, G.v$ )

Complexity:

Initialization:  $O(V+E)$

$O(V+E)$

• Help:  $O(E)$

[Correção]

- Observamos que o caminho mais longo em  $G$  tem no máximo  $|V|-1$  arestas.
- Ao relaxarmos as arestas do grafo  $G$  de acordo com a ordenação topológica, relaxamos todos os caminhos de  $G$  para ordem (primeiro, de acordo com o Lema 2, todos os caminhos no grafo são subsequências da ordenação topológica).
- Apartando o Lema de Relaxação de Caminho, concluimos que após a execução da loop principal, se todo o vértice  $v \in V$  atingível a partir de  $s$ , se temos:  $v.d = S(s, v)$ .

### Problema 1 [ Verificar os Caminhos Mais Curtos ]

Propõe-se um algoritmo para verificar se o resultado da execução de um algoritmo de caminhos mais curtos é correcto.

Solução:

- Considerar a inversa dos caminhos mais curtos

def  $f(a, T)$ : Verificar se  $f$  é o vértice da inversa:

$$- V.T \neq M \Rightarrow (V.T, V) \in E \wedge V.d = V.T.d + W(V.T, V)$$

$$- S.T = 0$$

$$- \forall (a, r) \in E. V.d \leq a.d + W(a, r)$$

⑦

• He que mover f:

O gostava colar os caminhos mais rectos se e só se