

# Aula 27: Grafos: caminho de custo mínimo

## Árvores de extensão mínima

David Déharbe

Programa de Pós-graduação em Sistemas e Computação

Universidade Federal do Rio Grande do Norte

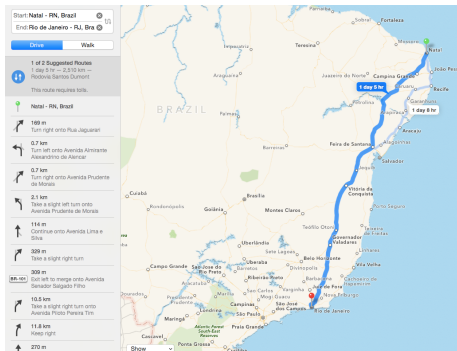
Centro de Ciências Exatas e da Terra

Departamento de Informática e Matemática Aplicada

Download me from <http://DavidDeharbe.github.io>



# Plano



## Introdução

Algumas propriedades de caminhos de custo mínimo

Relaxação

O algoritmo de Dijkstra

O algoritmo de Bellman-Ford

Referência: Cormen, cap 25.

# Introdução

- ▶ qual o menor caminho para ir de Natal até o Rio de Janeiro?
- ▶ problema de grafo?
  - ▶ vértices: cruzamentos
  - ▶ arestas: há uma estrada/rua entre esses dois cruzamentos
  - ▶ peso: distância entre dois cruzamentos
- ▶ muitas possibilidades!
- ▶ soluções não tem ciclos
- ▶ pesos podem ser outras medidas (tempo, custo financeiro, pontos turísticos)
- ▶ **otimização combinatória**: caminho de custo mínimo  
*single source shortest path, all pairs shortest path*



# Definições

$G = (V, E, W)$  : grafo dirigido com pesos

## Definição (Custo de um caminho)

Seja um caminho  $\pi = (v_0, v_1, \dots, v_k)$ . Então, temos que cada  $(v_i, v_{i+1}) \in E$ . O **peso do caminho**  $\pi$ , denotado  $W(\pi)$  é tal que

$$W(\pi) = \sum_{i=1}^k W(v_{i-1}, v_i).$$

## Definição (Caminho de custo mínimo)

O **custo mínimo** entre dois vértices  $u$  e  $v$ , denotado  $\delta(u, v)$  é tal que:

$$\delta(u, v) = \begin{cases} \min\{W(p) : u \overset{p}{\rightsquigarrow} v\} & \text{se existe um caminho de } u \text{ até } v \\ \infty & \text{caso contrário} \end{cases}$$



# Diferentes problemas de caminho de custo mínimo

- ▶ se não há pesos: busca em largura
- ▶ vértice origem fixado
  - ▶ custo mínimo de  $s$  para todos os demais vértices
- ▶ vértice destino fixado
  - ▶ custo mínimo de cada vértice até  $s$
  - ▶ obtido do anterior com o grafo transposto
- ▶ vértices origem e destino fixados
  - ▶ não há solução assintoticamente melhor do que quando o vértice origem é fixado
- ▶ entre todos os pares de vértices
  - ▶ fixar sucessivamente a origem em todos os vértices
  - ▶ mas existe algoritmos assintoticamente melhores para este problema

# Pesos negativos

- ▶ a presença ou ausência de pesos negativos muda os algoritmos possíveis
  - ▶ sem pesos negativos: algoritmo de Dijkstra
  - ▶ com pesos negativos: algoritmo de Bellman-Ford
- ▶ se há um ciclo tal que a soma dos pesos é negativa, então o custo mínimo é  $-\infty$ !
  - ▶ Bellman-Ford detecta ciclos negativos



# Representação de caminhos de custo mínimo

- ▶ outra saída esperada: caminho de custo mínimo
- ▶ dados:  $v.up$
- ▶ o caminho de custo mínimo de  $s$  até  $v$  é obtido seguindo  $v, v.up, v.up., \dots$  até  $s$
- ▶ durante a aplicação do algoritmo, o valor de  $up$  é atualizado
  - ▶ cada  $(v, v.up)$  forma uma aresta da **árvore dos caminhos de custo mínimo**
- ▶ em um grafo pode haver várias árvores dos caminhos de custo mínimo



# Propriedades dos caminhos de custo mínimo

1. Supondo que  $\pi = (v_0, v_1, \dots, v_{k-1})$  é um caminho de custo mínimo de  $v_0$  até  $v_{k-1}$ :
  - ▶ O que podemos dizer do caminho  $(v_i, v_{i+1}, \dots, v_j)$  contido em  $\pi$ ?



# Propriedades dos caminhos de custo mínimo

1. Supondo que  $\pi = (v_0, v_1, \dots, v_{k-1})$  é um caminho de custo mínimo de  $v_0$  até  $v_{k-1}$ :
  - ▶ O que podemos dizer do caminho  $(v_i, v_{i+1}, \dots, v_j)$  contido em  $\pi$ ?
2. Supondo que:  $s \overset{p}{\rightsquigarrow} v$  é o caminho de custo mínimo de  $s$  até  $v$  e é composto por  $s \overset{p'}{\rightsquigarrow} u$  e  $(u, v)$ :
  - ▶ o que podemos dizer de  $\delta(s, u)$ ,  $\delta(s, v)$ ,  $W(u, v)$ ?

# Propriedades dos caminhos de custo mínimo

1. Supondo que  $\pi = (v_0, v_1, \dots, v_{k-1})$  é um caminho de custo mínimo de  $v_0$  até  $v_{k-1}$ :
  - ▶ O que podemos dizer do caminho  $(v_i, v_{i+1}, \dots, v_j)$  contido em  $\pi$ ?
2. Supondo que:  $s \xrightarrow{p} v$  é o caminho de custo mínimo de  $s$  até  $v$  e é composto por  $s \xrightarrow{p'} u$  e  $(u, v)$ :
  - ▶ o que podemos dizer de  $\delta(s, u)$ ,  $\delta(s, v)$ ,  $W(u, v)$ ?
3. Em geral, se  $(u, v)$  é uma aresta, qual relação podemos estabelecer entre  $\delta(s, u)$ ,  $\delta(s, v)$  e  $W(u, v)$ ?

# Trecho de caminho de custo mínimo

## Propriedade

### Lema (trecho de um caminho de custo mínimo)

*Seja  $\pi = (v_0, v_1, \dots, v_k)$  um caminho de custo mínimo de  $v_0$  até  $v_k$ , e  $0 \leq i \leq j \leq k$ , então o trecho  $(v_i, v_{i+1}, \dots, v_j)$  de  $\pi$  é um caminho de custo mínimo de  $v_i$  até  $v_j$ .*

# Trecho de caminho de custo mínimo

## Propriedade

### Lema (trecho de um caminho de custo mínimo)

Seja  $\pi = (v_0, v_1, \dots, v_k)$  um caminho de custo mínimo de  $v_0$  até  $v_k$ , e  $0 \leq i \leq j \leq k$ , então o trecho  $(v_i, v_{i+1}, \dots, v_j)$  de  $\pi$  é um caminho de custo mínimo de  $v_i$  até  $v_j$ .

### Demonstração.

- ▶  $\pi = v_0 \xrightarrow{\pi_1} v_i \xrightarrow{\pi_2} v_j \xrightarrow{\pi_3} v_k$
- ▶ (por contradição)  $\pi : v_0 \xrightarrow{\pi} v_k$  caminho de custo mínimo de  $v_0$  até  $v_k$
- ▶  $\pi_2$  é um caminho de custo não mínimo de  $v_i$  até  $v_j$
- ▶ seja  $\pi'$  um caminho de custo mínimo entre  $v_i$  e  $v_j$ .
- ▶ temos  $\pi_1, \pi', \pi_3$  um caminho de  $v_0$  até  $v_k$  de custo menor que  $\pi$
- ▶ contradizendo a hipótese inicial



# Decomposição do custo mínimo

## Propriedade

### Corolário (decomposição de custo mínimo)

*Seja  $\pi$  um caminho de custo mínimo de  $s$  até  $v$  tal que  $\pi$  pode ser decomposto em  $s \xrightarrow{\pi'} u \rightarrow v$  para algum caminho  $\pi'$  e aresta  $(u, v)$ . Então  $\delta(s, v) = \delta(s, u) + W(u, v)$ .*

# Decomposição do custo mínimo

## Propriedade

### Corolário (decomposição de custo mínimo)

*Seja  $\pi$  um caminho de custo mínimo de  $s$  até  $v$  tal que  $\pi$  pode ser decomposto em  $s \xrightarrow{\pi'} u \rightarrow v$  para algum caminho  $\pi'$  e aresta  $(u, v)$ . Então  $\delta(s, v) = \delta(s, u) + W(u, v)$ .*

### Demonstração.

- ▶ aplicação do lema do trecho de caminho de custo mínimo
- ▶  $\pi$  tem custo mínimo
- ▶  $\pi'$  é um trecho de  $\pi$
- ▶ logo,  $\pi'$  tem custo mínimo
- ▶  $\delta(s, v) = W(\pi) = W(\pi') + W(u, v) = \delta(s, u) + W(u, v)$

# Relação entre custos mínimos e pesos

## Propriedade

Lema (relação entre custos mínimos e pesos)

*Para qualquer aresta  $(u, v)$ , temos*

$$\delta(s, v) \leq \delta(s, u) + W(u, v).$$

# Relação entre custos mínimos e pesos

## Propriedade

Lema (relação entre custos mínimos e pesos)

*Para qualquer aresta  $(u, v)$ , temos*

$$\delta(s, v) \leq \delta(s, u) + W(u, v).$$

Demonstração.

?





- ▶ relaxação: reduz o limite superior (**estimativa conservadora**) do custo de um caminho
- ▶ relaxação é aplicada repetidas vezes até chegar à solução
- ▶ o que é **relaxação**?

# Algoritmo de relaxação

- ▶  $s$ : um nó origem dado
- ▶  $v.d$ : limite superior do custo de  $s$  até  $v$
- ▶  $W(u, v)$ : peso da aresta  $(u, v)$

RELAX( $u, v, W$ )

```
1  if  $v.d > u.d + W(u, v)$   
2       $v.d = u.d + W(u, v)$   
3       $v.up = u$ 
```

# Algoritmo de relaxação

- ▶  $s$ : um nó origem dado
- ▶  $v.d$ : limite superior do custo de  $s$  até  $v$
- ▶  $W(u, v)$ : peso da aresta  $(u, v)$

RELAX( $u, v, W$ )

```
1  if  $v.d > u.d + W(u, v)$ 
2       $v.d = u.d + W(u, v)$ 
3       $v.up = u$ 
```

Justificativa:

- ▶ se a estimativa do custo de  $s$  até  $v$  é maior que a estimativa do custo de  $s$  até  $u$  somado ao peso de  $(u, v)$ ;
- ▶ existe um caminho de  $s$  até  $v$  com custo menor que os caminhos já calculados;
- ▶ este caminho termina pela aresta  $(u, v)$ .

# Relaxação: ponto de partida

- ▶ procedimento para iniciar com um estado compatível com a relaxação
- ▶  $d$  é inicializado com uma estimativa conservadora:  $\infty$
- ▶  $s$ : vértice origem, logo  $s.d = 0 = \delta(s, s)$

INIT-RELAX( $G, s$ )

```
1  for  $v \in G.V$ 
2       $v.d = \infty$ 
3       $v.up = \text{NIL}$ 
4   $s.d = 0$ 
```



# Propriedades

## Algoritmo de relaxação

RELAX( $u, v, W$ )

```
1  if  $v.d > u.d + W(u, v)$   
2       $v.d = u.d + W(u, v)$   
3       $v.up = u$ 
```

Lema (Pós-condição de RELAX( $u, v, W$ ))

*Após executar RELAX( $u, v, W$ ), temos a seguinte relação:*

$$v.d \leq u.d + W(u, v)$$

# Invariante

## Algoritmo de relaxação

### Lema (Invariante)

*O invariante  $\delta(s, v) \leq v.d$  é*

- 1. satisfeito após executar INIT-RELAX( $G, s$ ).*
- 2. preservado pela execução de RELAX( $u, v, W$ )*

# Invariante

## Algoritmo de relaxação

### Lema (Invariante)

*O invariante  $\delta(s, v) \leq v.d$  é*

- 1. satisfeito após executar INIT-RELAX( $G, s$ ).*
- 2. preservado pela execução de RELAX( $u, v, W$ )*

### Demonstração.

inicialização:

- ▶  $\delta(s, s) = 0$  ou  $\delta(s, s) = -\infty$  (se  $s$  estiver em um ciclo de peso negativo)
- ▶ em todo caso  $s.d \geq \delta(s, s)$ .
- ▶ para  $v \neq s$ ,  $s.d = \infty \geq \delta(s, v)$ .



# Invariante

## Algoritmo de relaxação

### Lema (Invariante)

*A invariante  $\delta(s, v) \leq v.d$  é*

- 1. satisfeito após executar INIT-RELAX( $G, s$ ).*
- 2. preservado pela execução de RELAX( $u, v, W$ )*

### Demonstração.

preservação por RELAX( $u, v, W$ ) (**por contradição**)

- ▶ seja  $v$  o primeiro vértice tal que  $v.d < \delta(s, v)$
- ▶ então, após RELAX( $u, v, W$ ) temos

$$\begin{aligned} u.d + W(u, v) &= v.d \\ &< \delta(s, v) \leq \delta(s, u) + W(u, v) \end{aligned}$$

- ▶ logo  $u.d < \delta(s, u)$
- ▶ contradiz que  $v$  foi o 1º vértice t.q.  $v.d < \delta(s, v)$ .



# Vértices não alcançáveis

## Propriedades

### Corolário (Vértices não alcançáveis)

*Depois de executar  $\text{INIT-RELAX}(G, s)$ , para qualquer vértice  $v$  não alcançável a partir de  $s$ , temos que  $v.d = \delta(s, v)$ .*

*A execução de  $\text{RELAX}(u, v, W)$  mantém  $v.d = \delta(s, v)$ .*

### Demonstração.

Pelo lema do invariante, temos que  $v.d \geq \delta(s, v)$  sempre. Como  $\delta(s, v) = \infty$ , então teremos  $v.d$  conserva seu valor inicial  $\infty$  sempre. □



# Extensão de caminho de custo mínimo

## Lema (Extensão de caminho de custo mínimo)

*Seja  $s \rightsquigarrow u \rightarrow v$  um caminho de custo mínimo de  $s$  até os vértices  $u$  e  $v$ . Se  $u.d = \delta(s, u)$  antes de executar  $\text{RELAX}(u, v, W)$ , então  $v.d = \delta(s, v)$ .*

## Demonstração.

► lema do invariante:

1.  $\delta(s, v) \leq v.d$  sempre
2. se  $\delta(s, u) = u.d$  antes de executar  $\text{RELAX}(u, v, W)$ , também  $\delta(s, u) = u.d$  depois

► Depois executar  $\text{RELAX}(u, v, W)$ ,

$$\begin{aligned} v.d &\leq u.d + W(u, v) && \text{lema da pós-condição} \\ &= \delta(s, u) + W(u, v) && \text{hipótese vigente} \\ &= \delta(s, v) && \text{corolário decomposição de custo mínimo} \end{aligned}$$

# Algoritmo de Dijkstra

## Princípios

- ▶ Mantem um conjunto de vértices  $S = \{v \cdot v.d = \delta(s, v)\}$ .
- ▶ Escolhe um vértice  $u$  a inserir em  $S$ 
  - ▶ custo mínimo de  $S$  para  $u$
- ▶ Relaxa com as arestas adjacentes a  $u$
- ▶ abordagem gulosa



# Algoritmo de Dijkstra

SINGLE-SOURCE-SHORTEST-PATH( $G, s$ )

```
1  INIT-RELAX( $G, s$ )
2   $S = \emptyset$ 
3   $Q = G.V$ 
4  while  $Q \neq \emptyset$ 
5       $u = \text{EXTRACT-MIN}(Q)$ 
6       $S = S \cup \{u\}$ 
7      for  $(u, v) \in G.E$ 
8          RELAX( $u, v, G.W$ )
```

$Q$ : fila de prioridade com chave  $d$

# Ilustração

## Algoritmo de Dijkstra

Quadro

# Complexidade

## Algoritmo de Dijkstra

Assumindo que  $Q$  é um *heap* binário...

SINGLE-SOURCE-SHORTEST-PATH( $G, s$ )

```
1  INIT-RELAX( $G, s$ ) //  $\Theta(V)$ 
2   $S = \emptyset$ 
3   $Q = G.V$  //  $O(V)$ 
4  while  $Q \neq \emptyset$  //  $\Theta(V)$  repetições
5       $u = \text{EXTRACT-MIN}(Q)$  //  $O(\lg V)$ 
6       $S = S \cup \{u\}$ 
7      for  $(u, v) \in G.E$  // total:  $\Theta(E)$ 
8          RELAX( $u, v, G.W$ ) //  $O(\lg V)$ 
```



# Complexidade

## Algoritmo de Dijkstra

Assumindo que  $Q$  é um *heap* binário...

SINGLE-SOURCE-SHORTEST-PATH( $G, s$ )

```
1  INIT-RELAX( $G, s$ ) //  $\Theta(V)$ 
2   $S = \emptyset$ 
3   $Q = G.V$  //  $O(V)$ 
4  while  $Q \neq \emptyset$  //  $\Theta(V)$  repetições
5       $u = \text{EXTRACT-MIN}(Q)$  //  $O(\lg V)$ 
6       $S = S \cup \{u\}$ 
7      for  $(u, v) \in G.E$  // total:  $\Theta(E)$ 
8          RELAX( $u, v, G.W$ ) //  $O(\lg V)$ 
```

Total:  $O(V \lg V + E \lg V)$



Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$





Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$  (**contradição**).

- ▶  $v$ : primeiro vértice inserido em  $S$  t.q.  $v.d \neq \delta(v, s)$

Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$  (**contradição**).

- ▶  $v$ : primeiro vértice inserido em  $S$  t.q.  $v.d \neq \delta(v, s)$
- ▶ certamente  $v \neq s$ , pois
  - ▶  $s$  é o primeiro vértice inserido em  $S$
  - ▶  $s.d$  é 0 quando é inserido, e  $\delta(s, s) = 0$ .

Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$  (**contradição**).

- ▶  $v$ : primeiro vértice inserido em  $S$  t.q.  $v.d \neq \delta(v, s)$
- ▶  $v \neq s$

Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$  (**contradição**).

- ▶  $v$ : primeiro vértice inserido em  $S$  t.q.  $v.d \neq \delta(v, s)$
- ▶  $v \neq s$
- ▶ logo  $S \neq \emptyset$  quando  $v$  é inserido em  $S$ .

Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$  (**contradição**).

- ▶  $v$ : primeiro vértice inserido em  $S$  t.q.  $v.d \neq \delta(v, s)$
- ▶  $v \neq s$ ,  $S \neq \emptyset$

Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$  (**contradição**).

- ▶  $v$ : primeiro vértice inserido em  $S$  t.q.  $v.d \neq \delta(v, s)$
- ▶  $v \neq s$ ,  $S \neq \emptyset$
- ▶ necessariamente há um caminho de  $s$  até  $v$ 
  1. senão, pelo corolário dos vértices não alcançáveis,  $v.d = \delta(s, v)$ ,
  2. contradizendo hipótese vigente  $v.d \neq \delta(v, s)$ .

Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$  (**contradição**).

- ▶  $v$ : primeiro vértice inserido em  $S$  t.q.  $v.d \neq \delta(v, s)$
- ▶  $v \neq s$ ,  $S \neq \emptyset$ ,  $s \rightsquigarrow v$
- ▶ logo existe um caminho de custo mínimo entre  $s$  e  $v$ , digamos  $p$  (não necessariamente todo em  $S$ ).
- ▶ seja  $y$  o primeiro vértice de  $p$  em  $V - S$  e  $x$  o predecessor de  $y$  em  $p$

Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$  (**contradição**).

- ▶  $v$ : primeiro vértice inserido em  $S$  t.q.  $v.d \neq \delta(v, s)$
- ▶  $v \neq s$ ,  $S \neq \emptyset$ ,  $s \rightsquigarrow v$ ,  $p = p_1(x, y)p_2$ ,  $s \xrightarrow{p_1} x \rightarrow y \xrightarrow{p_2} v$ ,  
 $p_1 \subseteq S$ ,  $W(p) = \delta(s, v)$



Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$  (**contradição**).

- ▶  $v$ : primeiro vértice inserido em  $S$  t.q.  $v.d \neq \delta(v, s)$
- ▶  $v \neq s$ ,  $S \neq \emptyset$ ,  $s \rightsquigarrow v$ ,  $p = p_1(x, y)p_2$ ,  $s \xrightarrow{p_1} x \rightarrow y \xrightarrow{p_2} v$ ,  
 $p_1 \subseteq S$ ,  $W(p) = \delta(s, v)$
- ▶ necessariamente  $x.d = \delta(s, x)$ ,
- ▶  $x \in S$ , então foi aplicado  $\text{RELAX}(x, y, W)$
- ▶ pelo lema da extensão de caminho de custo mínimo,  
 $y.d = \delta(s, y)$

Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$  (**contradição**).

- ▶  $v$ : primeiro vértice inserido em  $S$  t.q.  $v.d \neq \delta(v, s)$
- ▶  $v \neq s$ ,  $S \neq \emptyset$ ,  $s \rightsquigarrow v$ ,  $p = p_1(x, y)p_2$ ,  $s \xrightarrow{p_1} x \rightarrow y \xrightarrow{p_2} v$ ,  
 $p_1 \subseteq S$ ,  $W(p) = \delta(s, v)$ ,  $y.d = \delta(s, y)$

Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$  (**contradição**).

- ▶  $v$ : primeiro vértice inserido em  $S$  t.q.  $v.d \neq \delta(v, s)$
- ▶  $v \neq s$ ,  $S \neq \emptyset$ ,  $s \rightsquigarrow v$ ,  $p = p_1(x, y)p_2$ ,  $s \xrightarrow{p_1} x \rightarrow y \xrightarrow{p_2} v$ ,  
 $p_1 \subseteq S$ ,  $W(p) = \delta(s, v)$ ,  $y.d = \delta(s, y)$
- ▶  $y$  ocorre antes de  $v$  em um caminho de custo mínimo
- ▶ os pesos não são negativos
- ▶ logo  $\delta(s, y) \leq \delta(s, v)$
- ▶ pelo lema do invariante  $\delta(s, v) \leq v.d$ ,
- ▶ por transitividade de  $\leq$ , deduzimos que  $\delta(s, y) \leq \delta(s, v)$

Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$  (**contradição**).

- ▶  $v$ : primeiro vértice inserido em  $S$  t.q.  $v.d \neq \delta(v, s)$
- ▶  $v \neq s$ ,  $S \neq \emptyset$ ,  $s \rightsquigarrow v$ ,  $p = p_1(x, y)p_2$ ,  $s \xrightarrow{p_1} x \rightarrow y \xrightarrow{p_2} v$ ,  
 $p_1 \subseteq S$ ,  $W(p) = \delta(s, v)$ ,  $y.d = \delta(s, y)$ ,  $\delta(s, y) \leq \delta(s, v)$

Cada vez que  $u$  é inserido em  $S$ ,  $u.d = \delta(s, u)$  (**contradição**).

- ▶  $v$ : primeiro vértice inserido em  $S$  t.q.  $v.d \neq \delta(v, s)$
- ▶  $v \neq s$ ,  $S \neq \emptyset$ ,  $s \rightsquigarrow v$ ,  $p = p_1(x, y)p_2$ ,  $s \xrightarrow{p_1} x \rightarrow y \xrightarrow{p_2} v$ ,  
 $p_1 \subseteq S$ ,  $W(p) = \delta(s, v)$ ,  $y.d = \delta(s, y)$ ,  $\delta(s, y) \leq \delta(s, v)$
- ▶  $v = \text{EXTRACT-MIN}(Q)$
- ▶ logo  $v.d \leq y.d$
- ▶ temos  $v.d \geq \delta(s, v) \geq \delta(s, y) = y.d \geq v.d$
- ▶ logo  $v.d = \delta(s, v)$  **entramos em contradição**

## Teorema (Correção do algoritmo de Dijkstra)

*O algoritmo SINGLE-SOURCE-SHORTEST-PATH( $G, s$ ) calcula corretamente o custo dos caminhos de custo mínimo de  $s$  até cada vértice de  $G$ .*

## Teorema (Correção do algoritmo de Dijkstra)

*O algoritmo SINGLE-SOURCE-SHORTEST-PATH( $G, s$ ) calcula corretamente o custo dos caminhos de custo mínimo de  $s$  até cada vértice de  $G$ .*

## Demonstração.

- ▶ o lema do invariante garante que  $v.d \geq \delta(s, v)$  ao longo da execução
- ▶ mostramos que quando um vértice é inserido em  $S$ ,  
 $v.d = \delta(s, v)$
- ▶ todos os vértices são inseridos em  $S$ .



## Teorema (Correção do algoritmo de Dijkstra)

*O algoritmo SINGLE-SOURCE-SHORTEST-PATH( $G, s$ ) calcula corretamente o custo dos caminhos de custo mínimo de  $s$  até cada vértice de  $G$ .*

### Demonstração.

- ▶ o lema do invariante garante que  $v.d \geq \delta(s, v)$  ao longo da execução
- ▶ mostramos que quando um vértice é inserido em  $S$ ,  
 $v.d = \delta(s, v)$
- ▶ todos os vértices são inseridos em  $S$ .



Omitimos a prova que os caminhos de custo mínimo são calculados corretamente (atributo *up*).





# Algoritmo de Bellman-Ford

## Princípios

- ▶ leva em conta arestas negativas
- ▶ detecta e reporta se existe ciclos de custo mínimo
- ▶ também baseado na relaxação



# Algoritmo de Bellman-Ford

SINGLE-SOURCE-SHORTEST-PATHS( $G, s$ )

```
1  INIT-RELAX( $G, s$ )
2  for  $i = 1$  to  $|G.V| - 1$ 
3      for  $(u, v) \in G.E$ 
4          RELAX( $u, v, G.W$ )
5  for  $(u, v) \in G.E$ 
6      if  $v.d > u.d + G.W(u, v)$ 
7          return FALSE
8  return TRUE
```



# Ilustração

## Algoritmo de Bellman-Ford

No quadro

# Complexidade

## Algoritmo de Bellman-Ford

SINGLE-SOURCE-SHORTEST-PATHS( $G, s$ )

```
1  INIT-RELAX( $G, s$ ) //  $\Theta(V)$ 
2  for  $i = 1$  to  $|G.V| - 1$  //  $\Theta(V)$  repetições
3      for  $(u, v) \in G.E$  //  $\Theta(E)$  repetições
4          RELAX( $u, v, G.W$ ) //  $\Theta(1)$ 
5  for  $(u, v) \in G.E$  //  $O(E)$  repetições
6      if  $v.d > u.d + G.W(u, v)$ 
7          return FALSE
8  return TRUE
```

# Complexidade

## Algoritmo de Bellman-Ford

SINGLE-SOURCE-SHORTEST-PATHS( $G, s$ )

```
1  INIT-RELAX( $G, s$ ) //  $\Theta(V)$ 
2  for  $i = 1$  to  $|G.V| - 1$  //  $\Theta(V)$  repetições
3      for  $(u, v) \in G.E$  //  $\Theta(E)$  repetições
4          RELAX( $u, v, G.W$ ) //  $\Theta(1)$ 
5  for  $(u, v) \in G.E$  //  $O(E)$  repetições
6      if  $v.d > u.d + G.W(u, v)$ 
7          return FALSE
8  return TRUE
```

Total:  $\Theta(V E)$



# Correção

## Algoritmo de Bellman-Ford

1. caso  $G$  não contem ciclos de custo negativo
2. caso geral

### Lema (Bellman-Ford sem ciclos de custo negativo)

*Se  $G$  não possui ciclos de custo negativo, então, após executar o algoritmo de Bellman-Ford com  $G$  e  $s$ , temos que  $v.d = \delta(s, v)$  para todas as arestas alcançáveis de  $s$ .*

### Lema (Bellman-Ford sem ciclos de custo negativo)

*Se  $G$  não possui ciclos de custo negativo, então, após executar o algoritmo de Bellman-Ford com  $G$  e  $s$ , temos que  $v.d = \delta(s, v)$  para todas as arestas alcançáveis de  $s$ .*

### Demonstração.

- ▶ Seja  $p = (v_0, v_1, \dots, v_k)$  um caminho de custo mínimo entre  $v_0 = s$  e  $v_k = v$
- ▶ Não há ciclos de custo negativo, logo  $p$  não contém ciclo e  $k \leq |V| - 1$ .
- ▶ Abordagem de prova:
  - ▶ indução
  - ▶ após a iteração  $i$ ,  $v_i.d = \delta(s, v_i)$
  - ▶ como há  $|V| - 1$  iterações, provar esta propriedade é suficiente.



### Lema (Bellman-Ford sem ciclos de custo negativo)

*Se  $G$  não possui ciclos de custo negativo, então, após executar o algoritmo de Bellman-Ford com  $G$  e  $s$ , temos que  $v.d = \delta(s, v)$  para todas as arestas alcançáveis de  $s$ .*

### Demonstração.

- ▶ Seja  $p = (v_0, v_1, \dots, v_k)$  um caminho de custo mínimo entre  $v_0 = s$  e  $v_k = v$ ,  $k \leq |V| - 1$



### Lema (Bellman-Ford sem ciclos de custo negativo)

*Se  $G$  não possui ciclos de custo negativo, então, após executar o algoritmo de Bellman-Ford com  $G$  e  $s$ , temos que  $v.d = \delta(s, v)$  para todas as arestas alcançáveis de  $s$ .*

### Demonstração.

- ▶ Seja  $p = (v_0, v_1, \dots, v_k)$  um caminho de custo mínimo entre  $v_0 = s$  e  $v_k = v$
- ▶ No **caso de base**,  $i = 0$ ,
  - ▶  $\delta(s, v_0) = v_0.d = 0$  após a inicialização;
  - ▶ mantido pelas etapas de relaxação

### Lema (Bellman-Ford sem ciclos de custo negativo)

*Se  $G$  não possui ciclos de custo negativo, então, após executar o algoritmo de Bellman-Ford com  $G$  e  $s$ , temos que  $v.d = \delta(s, v)$  para todas as arestas alcançáveis de  $s$ .*

### Demonstração.

- ▶ Seja  $p = (v_0, v_1, \dots, v_k)$  um caminho de custo mínimo entre  $v_0 = s$  e  $v_k = v$
- ▶ No **caso geral**, a hipótese de indução é  $v_{i-1}.d = \delta(s, v_{i-1})$  após a iteração  $i - 1$ .
- ▶ A relaxação é aplicada a  $(v_{i-1}, v_i)$  na iteração  $i$ ,
- ▶ pelo lema da extensão de caminhos de custo mínimo,  $v_i.d = \delta(s, v_i)$  após a iteração  $i$ .

Corolário (Alcançabilidade e Bellman-Ford sem ciclos de custo negativo)

*Se  $G$  não possui ciclos de custo negativo, então, existe um caminho de  $s$  até  $v$  se e somente se, após executar o algoritmo de Bellman-Ford a  $G$  e  $s$ , temos que  $v.d \neq \infty$ .*

# Correção

## Algoritmo de Bellman-Ford

### Teorema (Correção do algoritmo de Bellman-Ford)

*Se  $G$  não possui ciclos de custo negativo, então o algoritmo retorna `TRUE` e  $v.d = \delta(s, v)$  para todo  $v \in V$ .*

*Se  $G$  possui um ciclo de custo negativo, então o algoritmo retorna `FALSE`.*



### Demonstração.

- ▶ Se  $G$  **não possui** ciclos de custo negativo
  - ▶ Se  $v$  é alcançável, temos  $v.d = \delta(s, v)$  (aplicação do lema)
  - ▶ Se  $v$  não é alcançável, também  $v.d = \delta(s, v)$  (corolário dos vértices inalcançáveis)
- ▶ Quando o laço das relaxações termina:  
$$v.d = \delta(s, v) \leq \delta(s, u) + W(u, v) = u.d + W(u, v)$$
- ▶ o teste  $v.d > u.d + W(u, v)$  nunca é verdadeiro
- ▶ o algoritmo retorna TRUE



### Demonstração.

- ▶ Se  $G$  possui um ciclo de custo negativo  $c = (v_0, v_1, \dots, v_k)$ , com  $v_0 = v_k$ , alcançável a partir de  $s$ .
- ▶  $\sum_{i=1}^k W(v_{i-1}, v_i) < 0$
- ▶ (por contradição) hipótese: algoritmo retorna TRUE
- ▶ logo  $v_i \cdot d \leq v_{i-1} \cdot d + W(v_{i-1}, v_i)$  para todo  $i = 1, \dots, k$
- ▶  $\sum_{i=1}^k v_i \cdot d \leq \sum_{i=1}^k (v_{i-1} \cdot d + W(v_{i-1}, v_i))$
- ▶ como é um ciclo:  $\sum_{i=1}^k v_i \cdot d = \sum_{i=1}^k v_{i-1} \cdot d$
- ▶ conclusão  $0 \leq \sum_{i=1}^k W(v_{i-1}, v_i)$ , contradizendo  $\sum_{i=1}^k W(v_{i-1}, v_i) < 0$ .