

ALGORITMOS DE CAMINHO MAIS CURTO

DCE529 - Algoritmos e Estruturas de Dados III

Atualizado em: 3 de abril de 2024

Iago Carvalho

Departamento de Ciência da Computação



CAMINHO EM GRAFOS

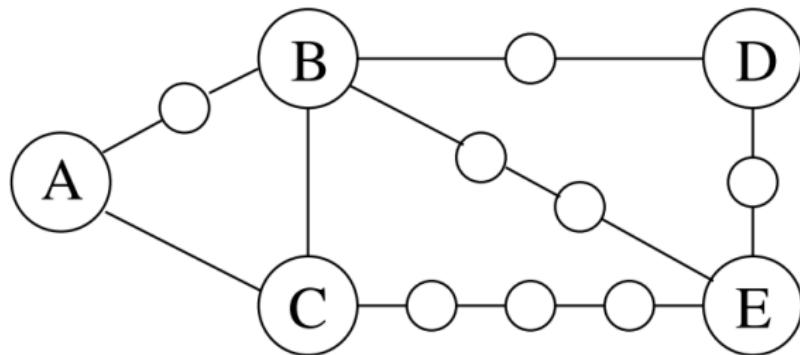
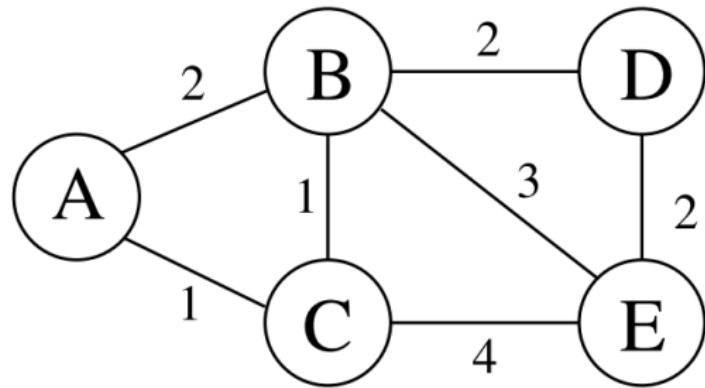
Estamos interessados em encontrar o caminho entre dois vértices de um grafo

- Diversas aplicações práticas

Em grafos não ponderados, podemos utilizar os algoritmos de busca em largura ou de busca em profundidade

Entretanto, estes dois algoritmos não lidam com arestas (ou arcos) ponderados

GRAFO NÃO-PONDERADO VS PONDERADO



DEFINIÇÃO DO PROBLEMA DO CAMINHO MÍNIMO

O problema do caminho mínimo é definido sobre um grafo não-orientado ponderado $G = (V, E)$

- Ou sobre um grafo orientado e ponderado $G = (V, A)$

Existe uma função w que mapeia cada aresta (ou arco) a um valor real que simboliza o peso da aresta (ou arco)

$$w : E \mapsto \mathbb{R} \quad \text{ou} \quad w : A \mapsto \mathbb{R}$$

DEFINIÇÃO DO PROBLEMA DO CAMINHO MÍNIMO

O peso do caminho p

$$p = \langle v_0, v_1, \dots, v_k \rangle$$

é igual ao somatório dos pesos de suas arestas (ou arcos)

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

DEFINIÇÃO DO PROBLEMA DO CAMINHO MÍNIMO

O caminho mais curto entre dois vértices $u, v \in V$ pode ser definido como

$$\delta(u, v) = \begin{cases} \min w(p) : u \xrightarrow{p} v, & \text{se existe caminho entre } u \text{ e } v \\ \infty, & \text{caso contrario} \end{cases}$$

Ou seja, o caminho mais curto entre u e v é

$$w(p) = \delta(u, v)$$

SUBESTRUTURA ÓTIMA DO CAMINHO MÍNIMO

Seja $p = \langle v_0, v_1, \dots, v_k \rangle$ o caminho mínimo entre os vértices v_0 e v_k

- $w(p) = \delta(v_0, v_k)$

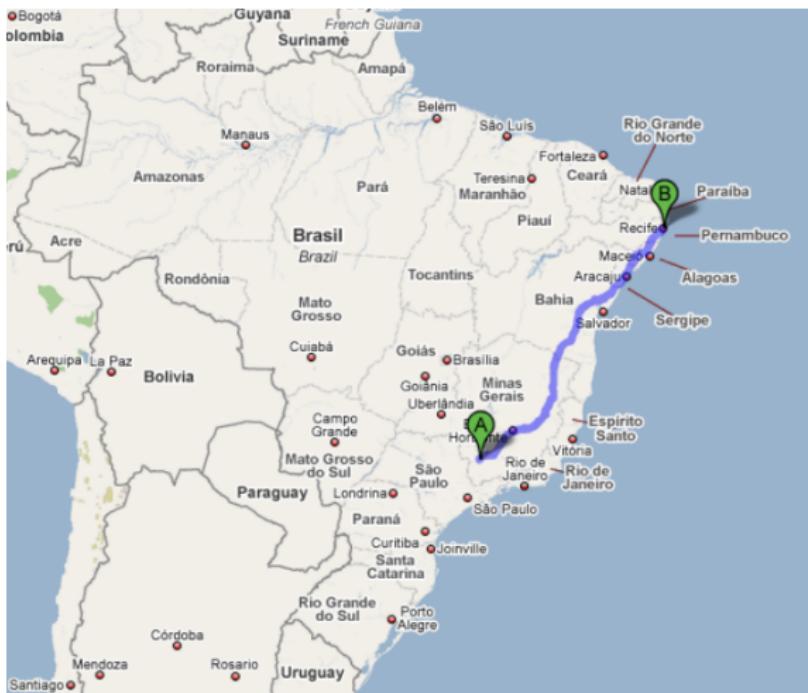
Para quaisquer i e j tais que $1 \leq i < j \leq k$

- Seja $p_{ij} = \langle v_i, v_{i+1}, \dots, v_j \rangle$ o subcaminho entre i e j em p

Temos que $p_{ij} = \delta(v_i, v_j)$

- Isto é, o caminho mínimo entre v_i e v_j é um subcaminho de p

PROBLEMA DO CAMINHO MÍNIMO



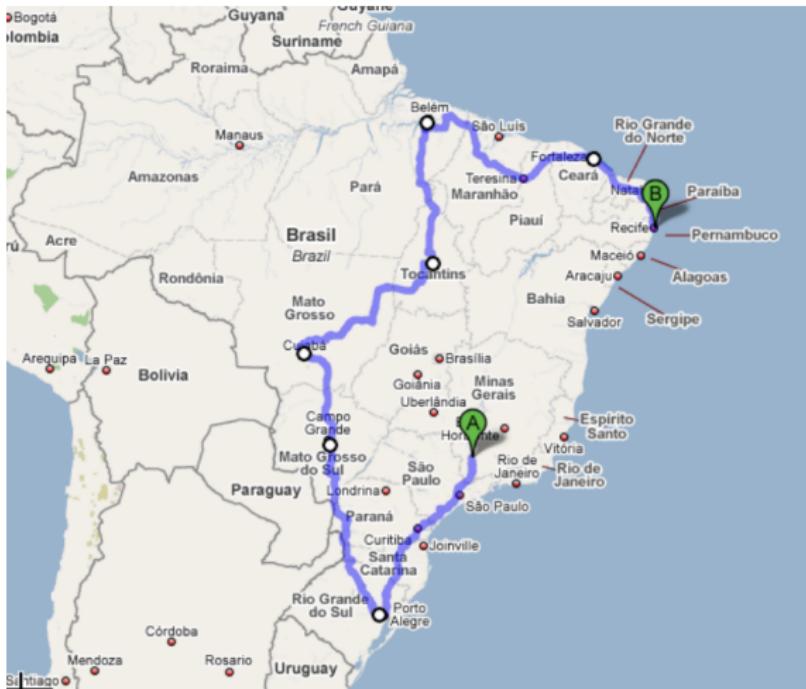
PROBLEMA DO CAMINHO MÍNIMO

Podemos montar um algoritmo de força bruta que enumera **todos** os caminhos entre as duas cidades

- Tempo exponencial
- $\#p$ -Completo

Este algoritmo considera todas as rotas existentes, mesmo aquelas que não fazem muito sentido

PROBLEMA DO CAMINHO MÍNIMO



PROBLEMA DO CAMINHO MÍNIMO

Podemos melhorar este algoritmo de força bruta utilizando enumeração *implícita*

Esquema de corte da enumeração implícita

Eu conheço uma rota completa com custo total t . Se minha rota parcial possui um custo $t' \geq t$, então não preciso continuar o processo de busca a partir desta solução parcial

ALGORITMO DE DIJKSTRA

É o algoritmo mais eficiente para encontrar o caminho mínimo entre um vértice raiz $r \in V$ e todos os outros vértices do grafo

- Ele também pode ser utilizado para encontrar o caminho mínimo entre dois vértices $u, v \in V$

Este é um algoritmo **guloso**

- Apesar de ser guloso, ele é um algoritmo exato
- Entretanto, algumas pessoas também o consideram como um algoritmo de programação dinâmica *bottom-up*

ALGORITMO DE DIJKSTRA

Ele se baseia em dois tipos de vértices

1. Aqueles com o caminho mínimo já conhecido até a raiz
2. Aqueles cujo caminho mínimo ainda é provisório
 - o Sem garantia de otimalidade

Ele utiliza quatro vetores auxiliares

$\pi[u]$	\rightarrow	pai do vértice u
$d[u]$	\rightarrow	distância da raiz até u
Q	\rightarrow	vértices com o caminho mínimo provisório
S	\rightarrow	vértices com o caminho mínimo garantido

ALGORITMO DE DIJKSTRA

Utiliza também duas funções auxiliares

INICIALIZA($G = (V, A), s)$

para cada $v \in V$

$d[v] = \infty$

$\pi[v] = \text{NULL}$

fim para

$d[s] = 0$

fim

RELAXA(u, v, w)

se $d[v] > (d[u] + w(u, v))$ então

$d[v] \leftarrow d[u] + w(u, v)$

$\pi[v] = u$

fim se

fim

PSEUDOCÓDIGO

DIJSKSTRA($G = (V, A)$, w, s)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair M inino(Q)

$S \leftarrow S \cup \{u\}$

para cada $v \in Adj[u]$

relaxa(u, v, w)

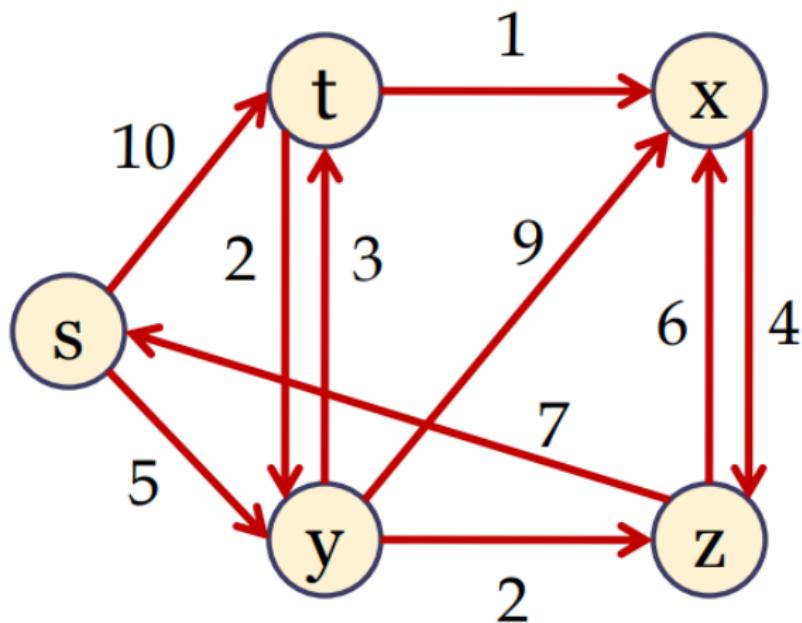
fim para

fim enquanto

fim

EXEMPLO

Vamos considerar o seguinte grafo



EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$S \leftarrow S \cup \{u\}$

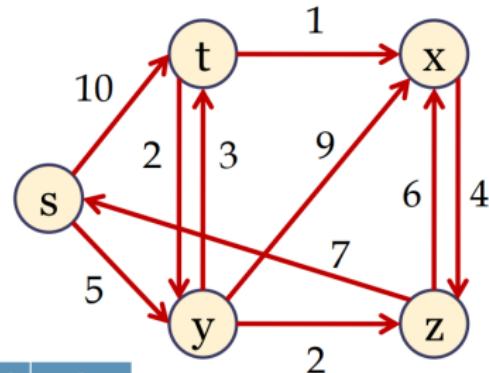
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	

vértice	s	t	x	y	z
d					
π					
Q					
S					

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

→ INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$S \leftarrow S \cup \{u\}$

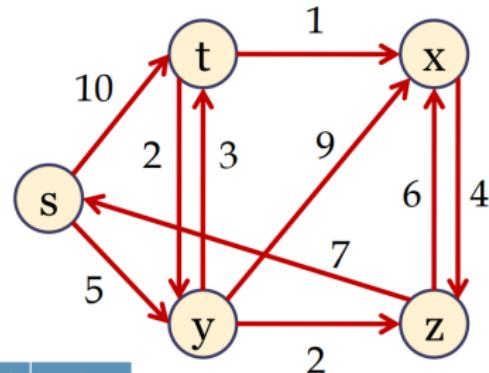
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	

vértice	s	t	x	y	z
d	0	∞	∞	∞	∞
π	NULL	NULL	NULL	NULL	NULL
Q					
S					

Algoritmo de Dijkstra

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$\rightarrow S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow \text{extrair Mínino}(Q)$

$S \leftarrow S \cup \{u\}$

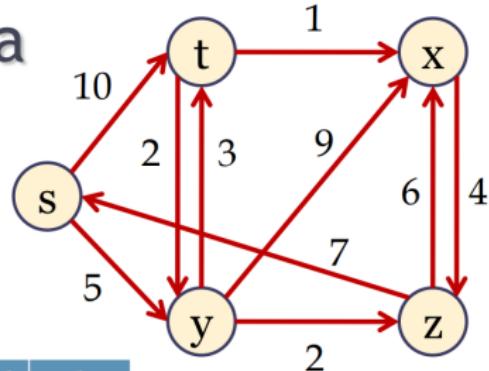
para cada $v \in \text{Adj}[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	

vértice	s	t	x	y	z
d	0	∞	∞	∞	∞
π	NULL	NULL	NULL	NULL	NULL
Q					
S	---	---	---	---	---

Algoritmo de Dijkstra

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$\rightarrow Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow \text{extrair Mínino}(Q)$

$S \leftarrow S \cup \{u\}$

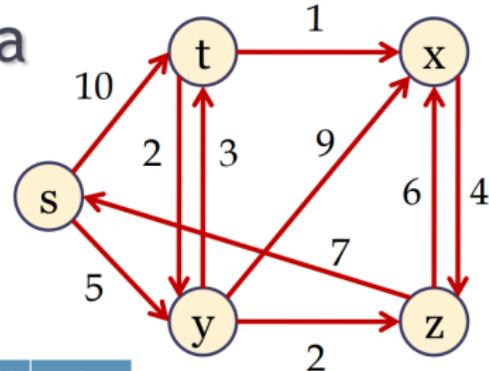
para cada $v \in \text{Adj}[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	

vértice	s	t	x	y	z
d	0	∞	∞	∞	∞
π	NULL	NULL	NULL	NULL	NULL
Q	X	X	X	X	X
S	---	---	---	---	---

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

→ Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$S \leftarrow S \cup \{u\}$

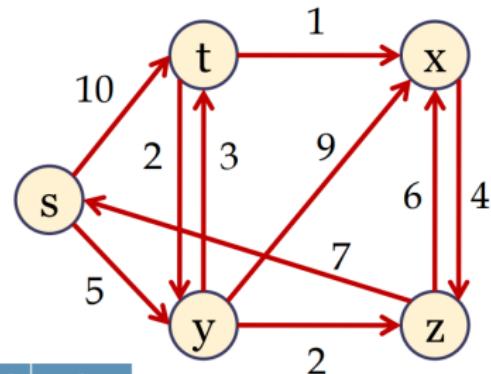
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	

vértice	s	t	x	y	z
d	0	∞	∞	∞	∞
π	NULL	NULL	NULL	NULL	NULL
Q	X	X	X	X	X
S	---	---	---	---	---

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$\rightarrow u \leftarrow \text{extrair Mínino}(Q)$

$S \leftarrow S \cup \{u\}$

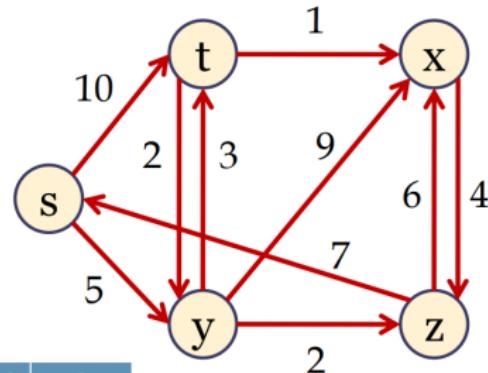
 para cada $v \in \text{Adj}[u]$

 relaxa(u, v, w)

 fim para

 fim enquanto

fim



variável	valor
u	

vértice	s	t	x	y	z
d	0	∞	∞	∞	∞
π	NULL	NULL	NULL	NULL	NULL
Q	X	X	X	X	X
S	---	---	---	---	---

Algoritmo de Dijkstra

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{\}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$\rightarrow u \leftarrow \text{extrair Mínino}(Q)$

$S \leftarrow S \cup \{u\}$

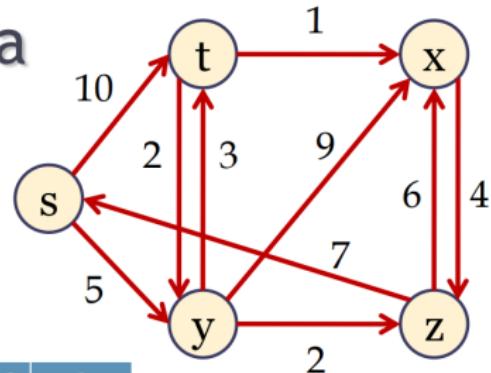
 para cada $v \in \text{Adj}[u]$

 relaxa(u, v, w)

 fim para

 fim enquanto

fim



variável	valor
u	s

vértice	s	t	x	y	z
d	0	∞	∞	∞	∞
π	NULL	NULL	NULL	NULL	NULL
Q	---	X	X	X	X
S	---	---	---	---	---

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Minino(Q)

→ $S \leftarrow S \cup \{u\}$

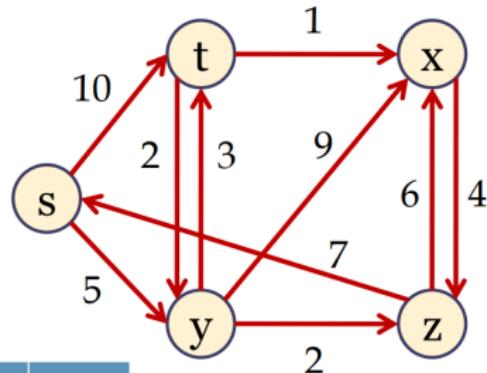
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	s

vértice	s	t	x	y	z
d	0	∞	∞	∞	∞
π	NULL	NULL	NULL	NULL	NULL
Q	---	X	X	X	X
S	X	---	---	---	---

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Minino(Q)

$S \leftarrow S \cup \{u\}$

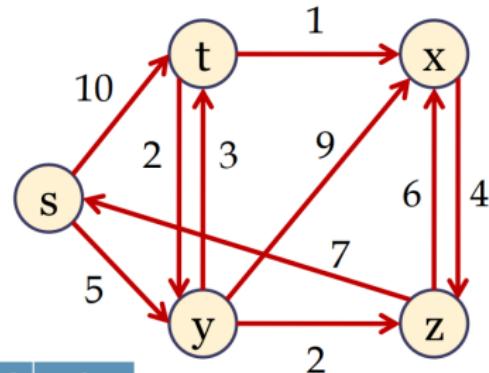
 → para cada $v \in Adj[u]$

 → relaxa(u, v, w)

 fim para

 fim enquanto

fim



variável	valor
u	s

vértice	s	t	x	y	z
d	0	10	∞	5	∞
π	NULL	s	NULL	s	NULL
Q	---	X	X	X	X
S	X	---	---	---	---

EXEMPLO

DIJSKSTRA($G = (V, A)$, w, s)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

→ Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$S \leftarrow S \cup \{u\}$

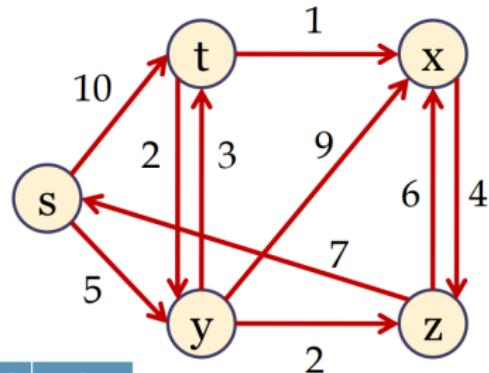
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	s

vértice	s	t	x	y	z
d	0	10	∞	5	∞
π	NULL	s	NULL	s	NULL
Q	---	X	X	X	X
S	X	---	---	---	---

Algoritmo de Dijkstra

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{\}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$\rightarrow u \leftarrow \text{extrair Mínino}(Q)$

$S \leftarrow S \cup \{u\}$

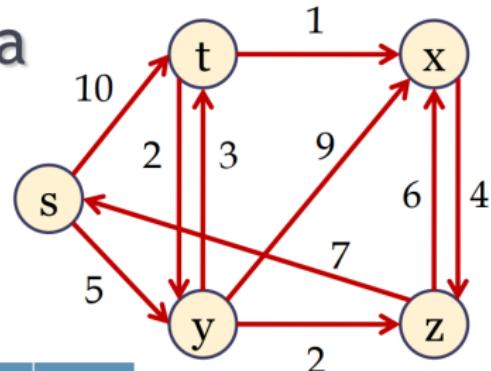
 para cada $v \in \text{Adj}[u]$

 relaxa(u, v, w)

 fim para

 fim enquanto

fim



variável	valor
u	s

vértice	s	t	x	y	z
d	0	10	∞	5	∞
π	NULL	s	NULL	s	NULL
Q	---	X	X	X	X
S	X	---	---	---	---

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$\rightarrow u \leftarrow \text{extrair Mínino}(Q)$

$S \leftarrow S \cup \{u\}$

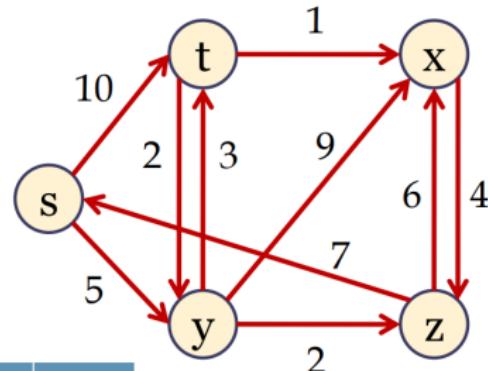
 para cada $v \in \text{Adj}[u]$

 relaxa(u, v, w)

 fim para

 fim enquanto

fim



variável	valor
u	y

vértice	s	t	x	y	z
d	0	10	∞	5	∞
π	NULL	s	NULL	s	NULL
Q	---	X	X	---	X
S	X	---	---	---	---

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Minino(Q)

$\rightarrow S \leftarrow S \cup \{u\}$

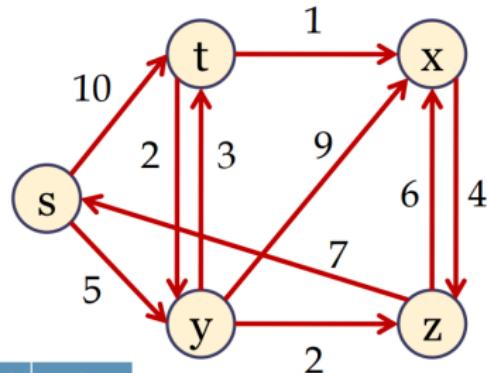
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	y

vértice	s	t	x	y	z
d	0	10	∞	5	∞
π	NULL	s	NULL	s	NULL
Q	---	X	X	---	X
S	X	---	---	X	---

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Minino(Q)

$S \leftarrow S \cup \{u\}$

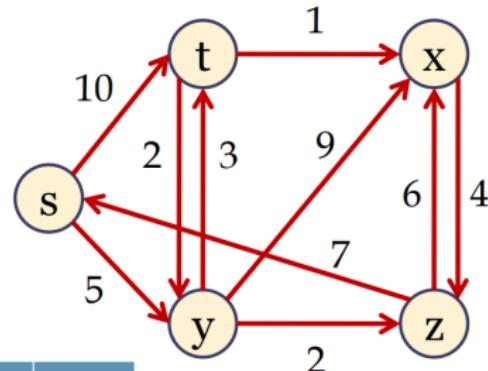
→ para cada $v \in Adj[u]$

→ relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	y

vértice	s	t	x	y	z
d	0	10	∞	5	∞
π	NULL	s	NULL	s	NULL
Q	---	X	X	---	X
S	X	---	---	X	---

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$S \leftarrow S \cup \{u\}$

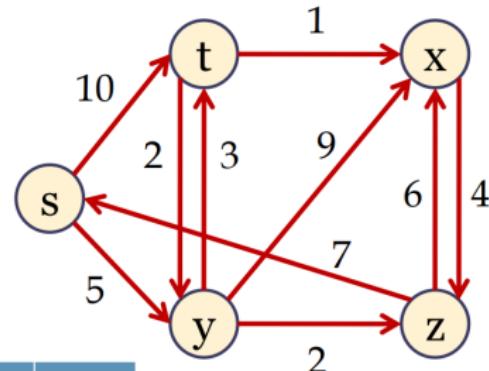
 → para cada $v \in Adj[u]$

 → relaxa(u, v, w)

 fim para

 fim enquanto

fim



variável	valor
u	y

vértice	s	t	x	y	z
d	0	8	14	5	7
π	NULL	y	y	s	y
Q	---	X	X	---	X
S	X	---	---	X	---

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

→ Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$S \leftarrow S \cup \{u\}$

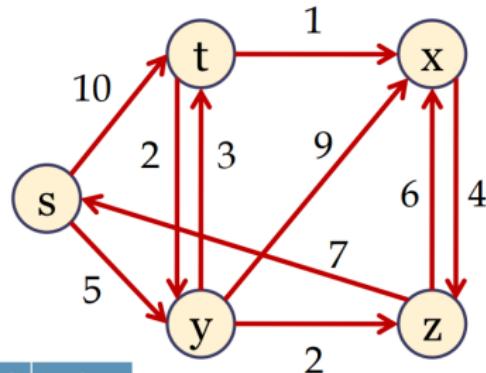
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	y

vértice	s	t	x	y	z
d	0	8	14	5	7
π	NULL	y	y	s	y
Q	---	X	X	---	X
S	X	---	---	X	---

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

 → $u \leftarrow$ extrair Minino(Q)

$S \leftarrow S \cup \{u\}$

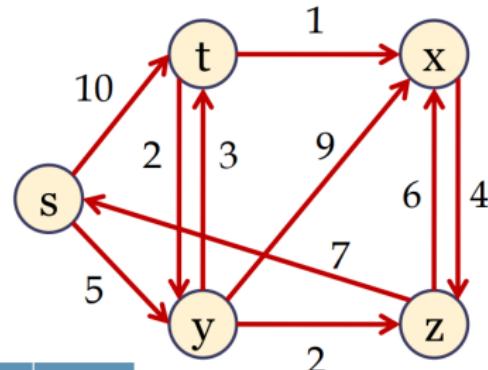
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	y

vértice	s	t	x	y	z
d	0	8	14	5	7
π	NULL	y	y	s	y
Q	---	X	X	---	X
S	X	---	---	X	---

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

 → $u \leftarrow \text{extrair Mínino}(Q)$

$S \leftarrow S \cup \{u\}$

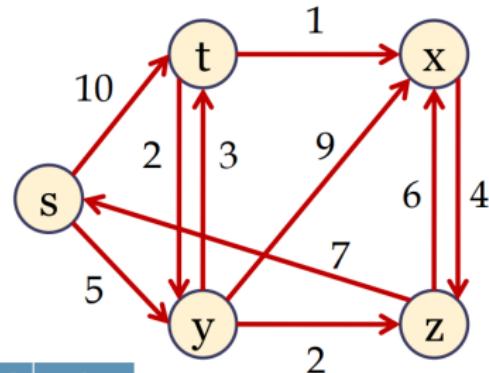
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	z

vértice	s	t	x	y	z
d	0	8	14	5	7
π	NULL	y	y	s	y
Q	---	X	X	---	---
S	X	---	---	X	---

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

→ $S \leftarrow S \cup \{u\}$

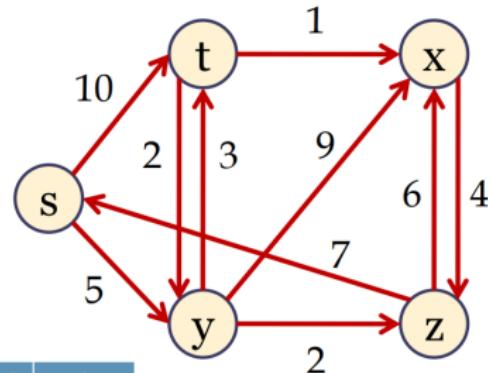
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	z

vértice	s	t	x	y	z
d	0	8	14	5	7
π	NULL	y	y	s	y
Q	---	X	X	---	---
S	X	---	---	X	X

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$S \leftarrow S \cup \{u\}$

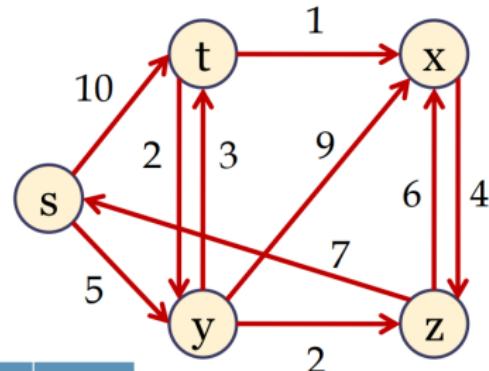
 → para cada $v \in Adj[u]$

 → relaxa(u, v, w)

 fim para

 fim enquanto

fim



variável	valor
u	z

vértice	s	t	x	y	z
d	0	8	14	5	7
π	NULL	y	y	s	y
Q	---	X	X	---	---
S	X	---	---	X	X

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$S \leftarrow S \cup \{u\}$

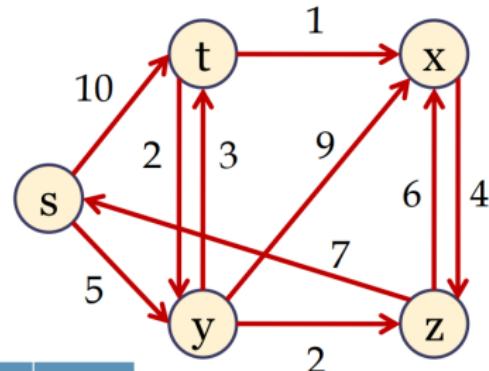
 → para cada $v \in Adj[u]$

 → relaxa(u, v, w)

 fim para

 fim enquanto

fim



variável	valor
u	z

vértice	s	t	x	y	z
d	0	8	13	5	7
π	NULL	y	z	s	y
Q	---	X	X	---	---
S	X	---	---	X	X

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

→ Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$S \leftarrow S \cup \{u\}$

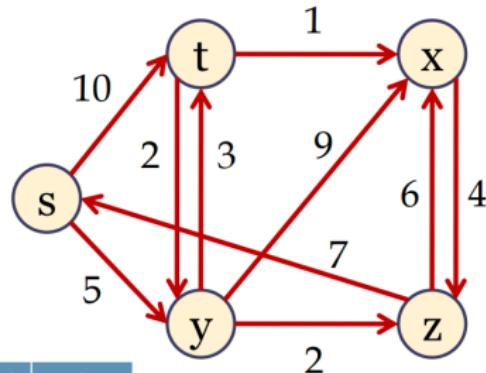
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	z

vértice	s	t	x	y	z
d	0	8	13	5	7
π	NULL	y	z	s	y
Q	---	X	X	---	---
S	X	---	---	X	X

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

 → $u \leftarrow$ extrair Minino(Q)

$S \leftarrow S \cup \{u\}$

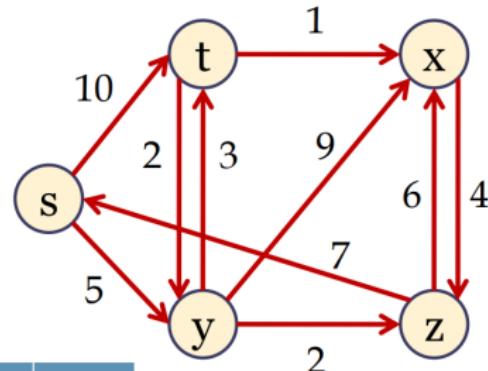
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	z

vértice	s	t	x	y	z
d	0	8	13	5	7
π	NULL	y	z	s	y
Q	---	X	X	---	---
S	X	---	---	X	X

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

 → $u \leftarrow$ extrair Minino(Q)

$S \leftarrow S \cup \{u\}$

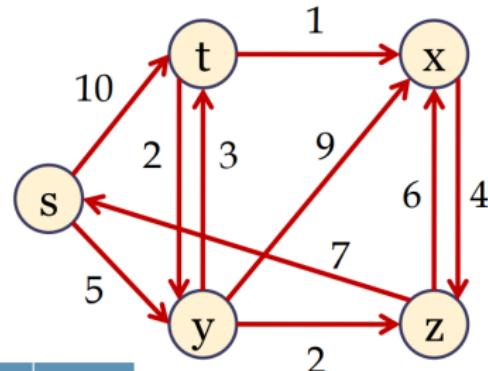
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	t

vértice	s	t	x	y	z
d	0	8	13	5	7
π	NULL	y	z	s	y
Q	---	---	X	---	---
S	X	---	---	X	X

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Minino(Q)

$\rightarrow S \leftarrow S \cup \{u\}$

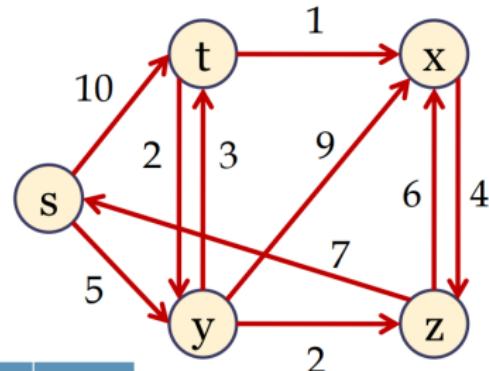
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	t

vértice	s	t	x	y	z
d	0	8	13	5	7
π	NULL	y	z	s	y
Q	---	---	X	---	---
S	X	X	---	X	X

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$S \leftarrow S \cup \{u\}$

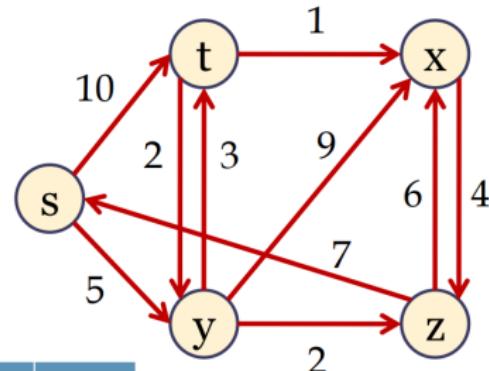
→ para cada $v \in Adj[u]$

→ relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	t

vértice	s	t	x	y	z
d	0	8	13	5	7
π	NULL	y	z	s	y
Q	---	---	X	---	---
S	X	X	---	X	X

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$S \leftarrow S \cup \{u\}$

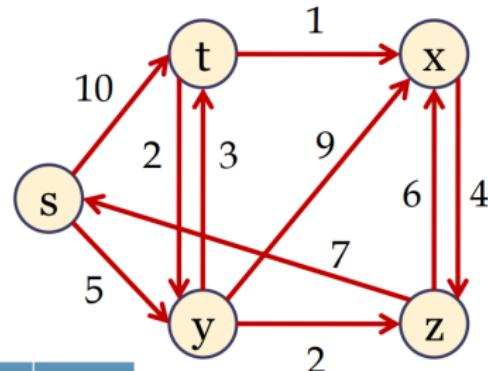
→ para cada $v \in Adj[u]$

→ relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	t

vértice	s	t	x	y	z
d	0	8	9	5	7
π	NULL	y	t	s	y
Q	---	---	X	---	---
S	X	X	---	X	X

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$\rightarrow u \leftarrow \text{extrair Mínino}(Q)$

$S \leftarrow S \cup \{u\}$

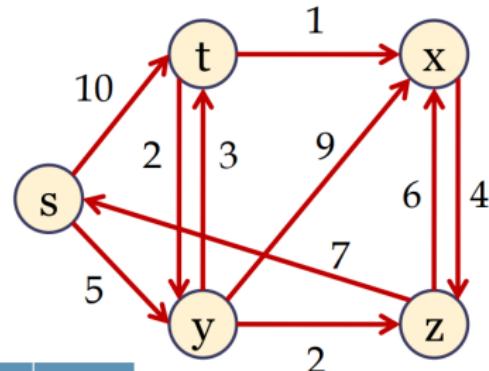
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	x

vértice	s	t	x	y	z
d	0	8	9	5	7
π	NULL	y	t	s	y
Q	---	---	---	---	---
S	X	X	---	X	X

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$\rightarrow S \leftarrow S \cup \{u\}$

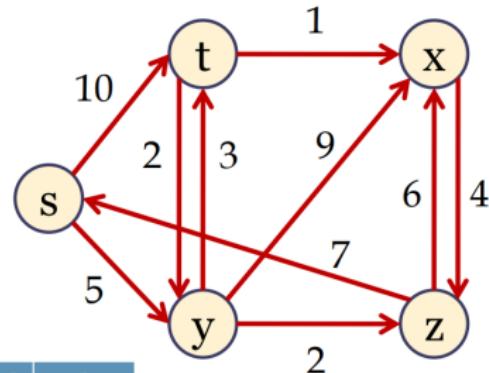
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

fim



variável	valor
u	x

vértice	s	t	x	y	z
d	0	8	9	5	7
π	NULL	y	t	s	y
Q	---	---	---	---	---
S	X	X	X	X	X

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$S \leftarrow S \cup \{u\}$

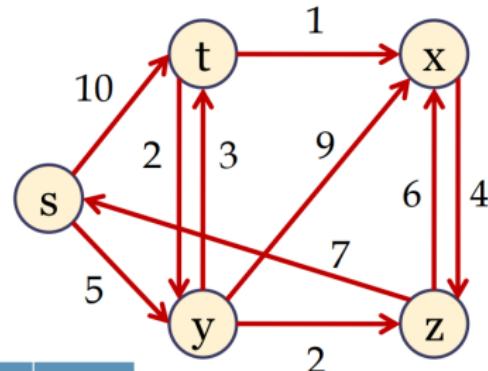
 → para cada $v \in Adj[u]$

 → relaxa(u, v, w)

 fim para

 fim enquanto

fim



variável	valor
u	x

vértice	s	t	x	y	z
d	0	8	9	5	7
π	NULL	y	t	s	y
Q	---	---	---	---	---
S	X	X	X	X	X

EXEMPLO

DIJSKSTRA($G = (V, A), w, s$)

INICIALIZA(G, s)

$S \leftarrow \{ \}$

$Q \leftarrow V$

Enquanto $|Q| \neq 0$

$u \leftarrow$ extrair Mínino(Q)

$S \leftarrow S \cup \{u\}$

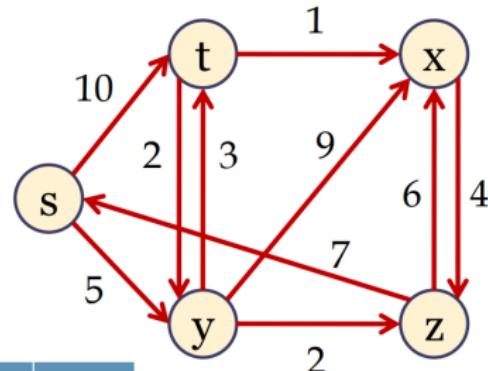
para cada $v \in Adj[u]$

relaxa(u, v, w)

fim para

fim enquanto

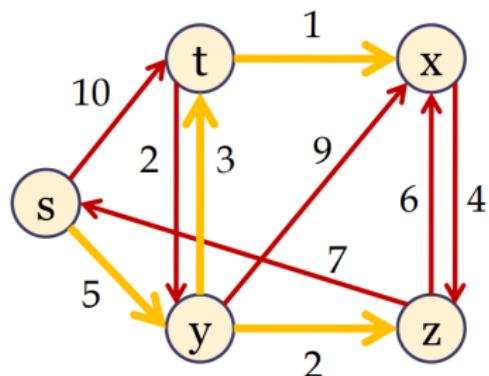
fim 



variável	valor
u	x

vértice	s	t	x	y	z
d	0	8	9	5	7
π	NULL	y	t	s	y
Q	---	---	---	---	---
S	X	X	X	X	X

EXEMPLO

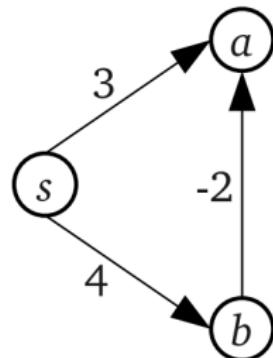


vértice	s	t	x	y	z
d	o	8	9	5	7
π	NULL	y	t	s	y

O algoritmo de Dijkstra tem complexidade de $\Theta(|V|^2)$

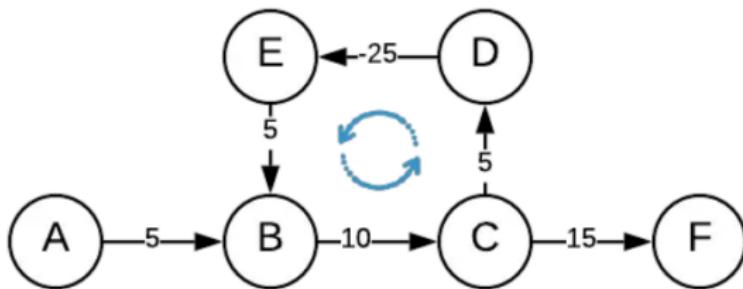
- Pode ser melhorado para $\Theta(|E| + |V|\log|V|)$ utilizando uma heap de Fibonacci na função extrairMínimo

LIMITAÇÕES



- O grafo tem que obedecer a desigualdade triangular
- Isto implica que ele não consegue lidar muito bem com pesos negativos
 - Pesos negativos podem induzir ciclos negativos, o que faria o algoritmo entrar em loop infinito

Qual é o custo do menor caminho de a até f ?



ALGORITMO DE BELLMAN-FORD

ALGORITMO DE BELLMAN-FORD

O algoritmo de Bellman-Ford pode ser executado em grafos com pesos negativos em suas arestas

- Entretanto, ele ainda não é capaz de lidar com ciclos negativos

Encontrar o caminho mais curto entre dois vértices quando o grafo possui ciclos negativos é um problema NP-Completo

PESOS NEGATIVOS

Pesos negativos são estruturas úteis na modelagem de problemas

- Por isso, devemos saber como lidar com eles

Um peso negativo pode representar

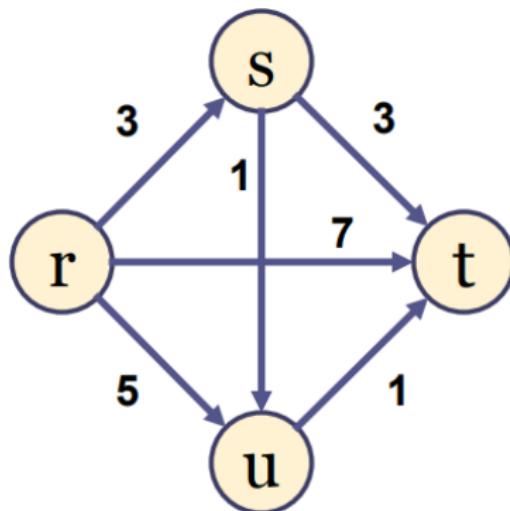
- Gasto de recursos
- Movimentações financeiras, sendo possível ter lucro ou prejuízo
- Um entregador que necessita atravessar um pedágio e pode acabar pagando mais do que recebe para entregar encomendas
- A energia gerada e consumida durante uma reação química
- Um taxista que recebe mais dinheiro do que gasta com combustível a cada viagem: se o táxi roda vazio, ele gasta mais do que recebe

PSEUDOCÓDIGO

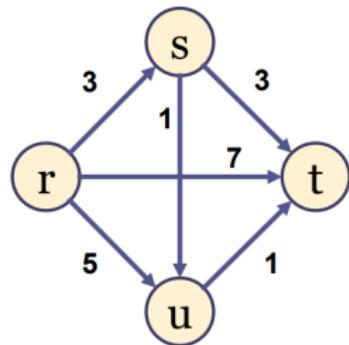
```
BELLMAN – FORD( $G = (V, A)$ ,  $w, s$ )  
    INICIALIZA( $G, s$ )  
    para  $i \leftarrow 1$  até  $|V| - 1$   
        para cada aresta  $(u, v) \in A$   
            RELAXA( $u, v, w$ )  
        fim para  
    fim para  
    para cada aresta  $(u, v) \in A$   
        se  $d[v] > d[u] + w(u, v)$   
            retorna falso  
        fim se  
    fim para  
    retorna verdadeiro  
fim
```

EXEMPLO

Vamos obter o caminho mínimo de r para todos os outros vértices do seguinte grafo



EXEMPLO



INICIALIZA($G = (V, A), s$)

para cada $v \in V$

$d[v] = \infty$

$\pi[v] = \text{NULL}$

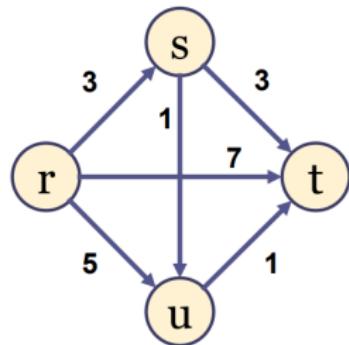
fim para

$d[s] = 0$

fim

vértice	r	s	t	u
d	0	∞	∞	∞
π	NULL	NULL	NULL	NULL

EXEMPLO



variável	valor
i	1
vértice	r s t u
d	0 ∞ ∞ ∞
π	NULL NULL NULL NULL

$BELLMAN - FORD(G = (V, A), w, s)$

$INICIALIZA(G, s)$

→ para $i \leftarrow 1$ até $|V| - 1$

para cada aresta $(u, v) \in A$

$RELAXA(u, v, w)$

fim para

fim para

para cada aresta $(u, v) \in A$

se $d[v] > d[u] + w(u, v)$

retorna falso

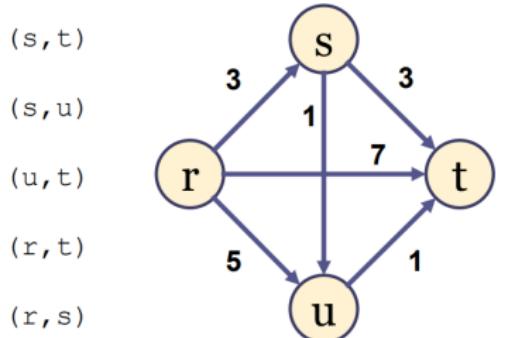
fim se

fim para

retorna verdadeiro

fim

EXEMPLO



variável	valor
i	1

vértice	r	s	t	u
d	0	∞	∞	∞
π	NULL	NULL	NULL	NULL

$BELLMAN - FORD(G = (V, A), w, s)$

$INICIALIZA(G, s)$

para $i \leftarrow 1$ até $|V| - 1$

→ para cada aresta $(u, v) \in A$
 $RELAXA(u, v, w)$

fim para

fim para

para cada aresta $(u, v) \in A$

se $d[v] > d[u] + w(u, v)$

retorna falso

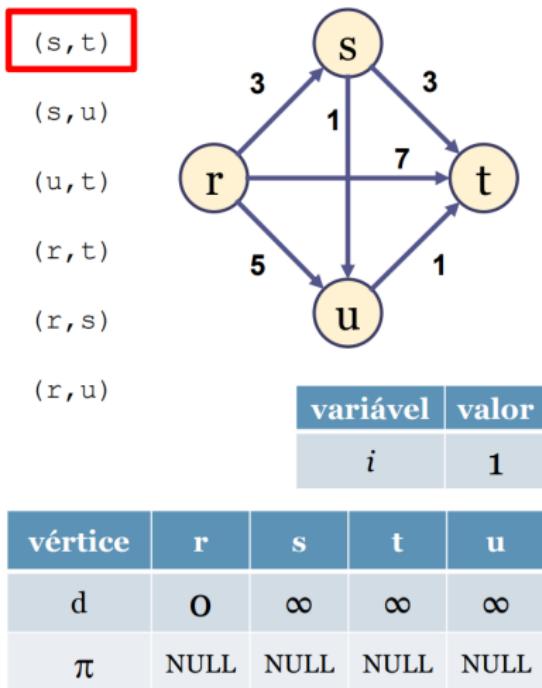
fim se

fim para

retorna verdadeiro

fim

EXEMPLO



$BELLMAN - FORD(G = (V, A), w, s)$

$INICIALIZA(G, s)$

para $i \leftarrow 1$ até $|V| - 1$

→ para cada aresta $(u, v) \in A$

→ $RELAXA(u, v, w)$

fim para

fim para

para cada aresta $(u, v) \in A$

se $d[v] > d[u] + w(u, v)$

retorna falso

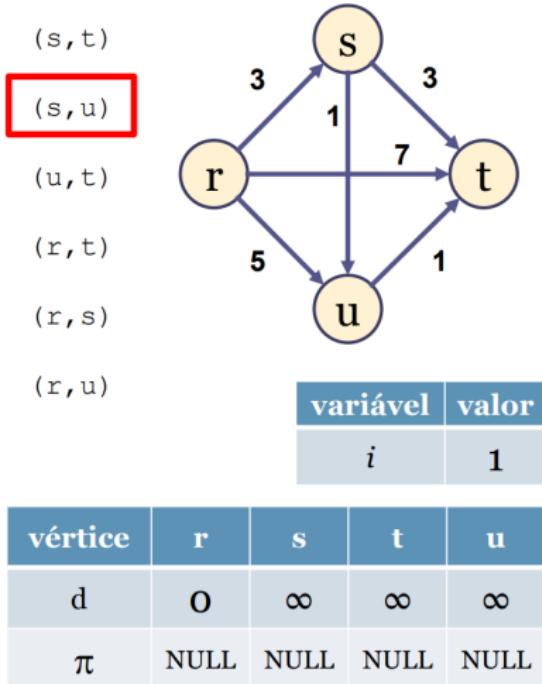
fim se

fim para

retorna verdadeiro

fim

EXEMPLO



$BELLMAN - FORD(G = (V, A), w, s)$

$INICIALIZA(G, s)$

para $i \leftarrow 1$ até $|V| - 1$

→ para cada aresta $(u, v) \in A$

→ $RELAXA(u, v, w)$

fim para

fim para

para cada aresta $(u, v) \in A$

se $d[v] > d[u] + w(u, v)$

retorna falso

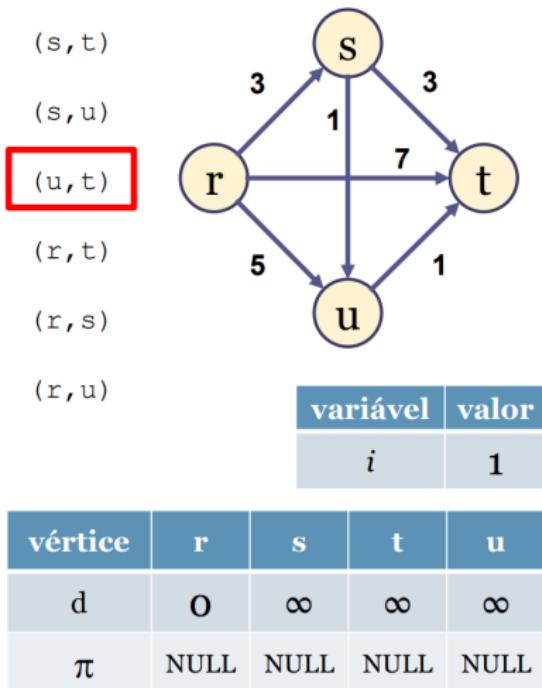
fim se

fim para

retorna verdadeiro

fim

EXEMPLO



$BELLMAN - FORD(G = (V, A), w, s)$

$INICIALIZA(G, s)$

para $i \leftarrow 1$ até $|V| - 1$

 → para cada aresta $(u, v) \in A$

 → $RELAXA(u, v, w)$

 fim para

 fim para

 para cada aresta $(u, v) \in A$

 se $d[v] > d[u] + w(u, v)$

 retorna falso

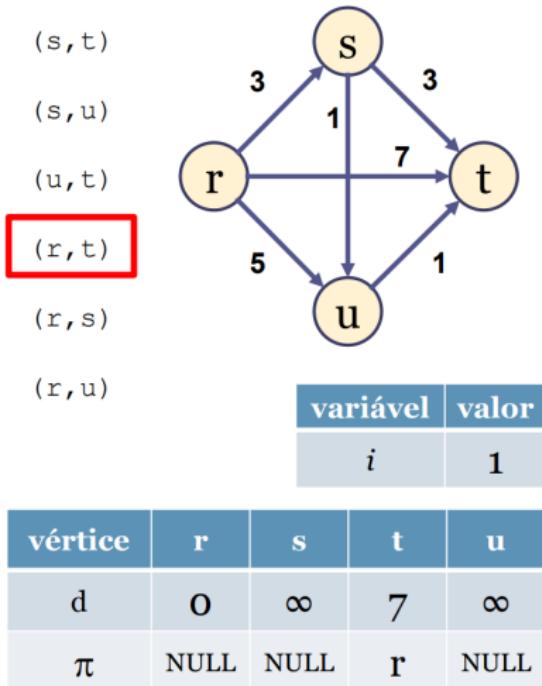
 fim se

 fim para

 retorna verdadeiro

fim

EXEMPLO



$BELLMAN - FORD(G = (V, A), w, s)$

$INICIALIZA(G, s)$

para $i \leftarrow 1$ até $|V| - 1$

 → para cada aresta $(u, v) \in A$

 → $RELAXA(u, v, w)$

 fim para

 fim para

 para cada aresta $(u, v) \in A$

 se $d[v] > d[u] + w(u, v)$

 retorna falso

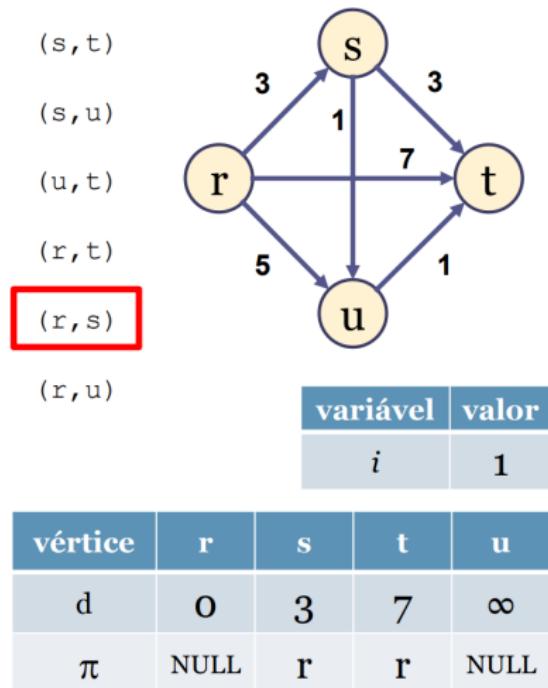
 fim se

 fim para

 retorna verdadeiro

fim

EXEMPLO



$BELLMAN - FORD(G = (V, A), w, s)$

$INICIALIZA(G, s)$

para $i \leftarrow 1$ até $|V| - 1$

 → para cada aresta $(u, v) \in A$

 → $RELAXA(u, v, w)$

 fim para

 fim para

 para cada aresta $(u, v) \in A$

 se $d[v] > d[u] + w(u, v)$

 retorna falso

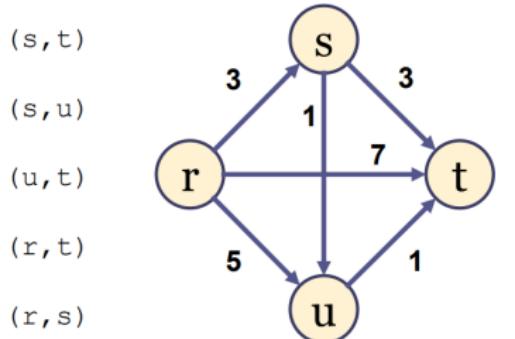
 fim se

 fim para

 retorna verdadeiro

fim

EXEMPLO



(r, u)

variável	valor
i	1

vértice	r	s	t	u
d	0	3	7	5
π	NULL	r	r	r

$BELLMAN - FORD(G = (V, A), w, s)$

$INICIALIZA(G, s)$

para $i \leftarrow 1$ até $|V| - 1$

 → para cada aresta $(u, v) \in A$

 → $RELAXA(u, v, w)$

 fim para

 fim para

 para cada aresta $(u, v) \in A$

 se $d[v] > d[u] + w(u, v)$

 retorna falso

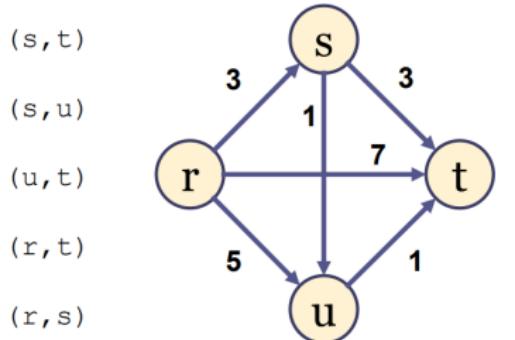
 fim se

 fim para

 retorna verdadeiro

fim

EXEMPLO



vértice	r	s	t	u
d	0	3	5	4
π	NULL	r	u	s

$BELLMAN - FORD(G = (V, A), w, s)$

$INICIALIZA(G, s)$

para $i \leftarrow 1$ até $|V| - 1$

→ para cada aresta $(u, v) \in A$

→ $RELAXA(u, v, w)$

fim para

fim para

para cada aresta $(u, v) \in A$

se $d[v] > d[u] + w(u, v)$

retorna falso

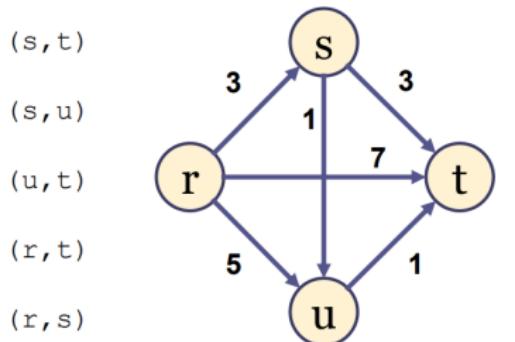
fim se

fim para

retorna verdadeiro

fim

EXEMPLO



variável	valor
i	3

vértice	r	s	t	u
d	0	3	5	4
π	NULL	r	u	s

$BELLMAN - FORD(G = (V, A), w, s)$

$INICIALIZA(G, s)$

para $i \leftarrow 1$ até $|V| - 1$

 → para cada aresta $(u, v) \in A$

 → $RELAXA(u, v, w)$

 fim para

 fim para

 para cada aresta $(u, v) \in A$

 se $d[v] > d[u] + w(u, v)$

 retorna falso

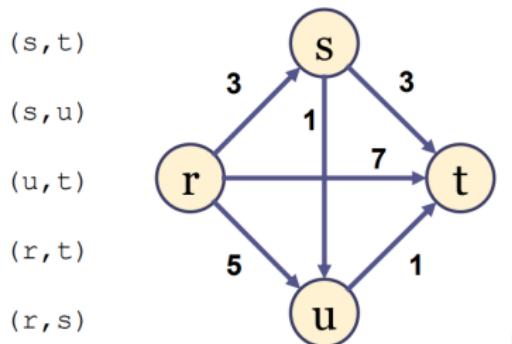
 fim se

 fim para

 retorna verdadeiro

fim

EXEMPLO



vértice	r	s	t	u
d	0	3	5	4
π	NULL	r	u	s

$BELLMAN - FORD(G = (V, A), w, s)$

$INICIALIZA(G, s)$

para $i \leftarrow 1$ até $|V| - 1$

para cada aresta $(u, v) \in A$
 $RELAXA(u, v, w)$

fim para

fim para

para cada aresta $(u, v) \in A$

se $d[v] > d[u] + w(u, v)$
 retorna falso

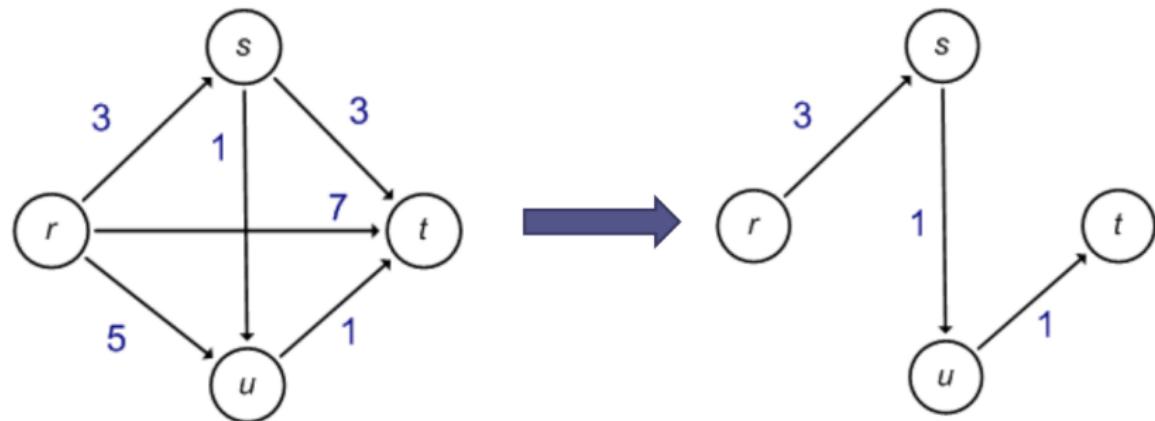
fim se

fim para

retorna verdadeiro

fim

EXEMPLO



vértice	r	s	t	u
d	0	3	5	4
π	NULL	r	u	s

O algoritmo de Bellman-Ford tem complexidade de $\Theta(|V| \times |E|)$

- Como $|E| = \mathcal{O}(|V|^2)$, então podemos dizer que a complexidade do algoritmo de Bellman-Ford é de $\Theta(|V|^3)$

OUTROS COMENTÁRIOS

O algoritmo pode ser encerrado prematuramente caso não haja modificação nos pesos em uma das iterações

Este algoritmo é utilizado em roteamento em redes

- Algoritmo de vetor de distâncias
- Camada de transporte
- Implementação distribuída
 - Cada vértice calcula sua distância para os demais
 - Tabela com as distâncias e caminhos é compartilhada
- Relativamente custoso, pois é necessário um grande número de mensagens de rede
- Alterações na topologia da rede demoram a ser refletidas no algoritmo
 - A alteração é gradual e ocorre conforme os vértices vão recalculando suas tabelas de roteamento