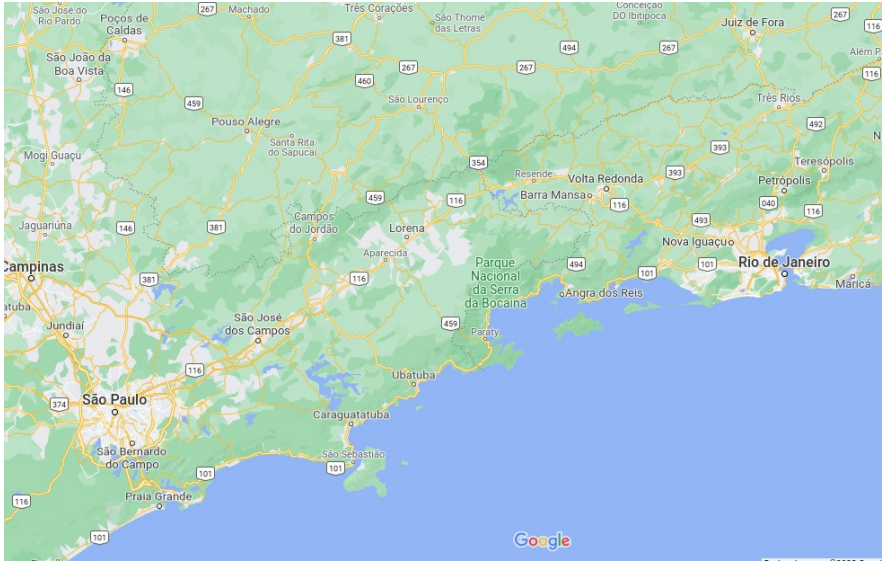
Problemas de caminho mais curto

Introdução

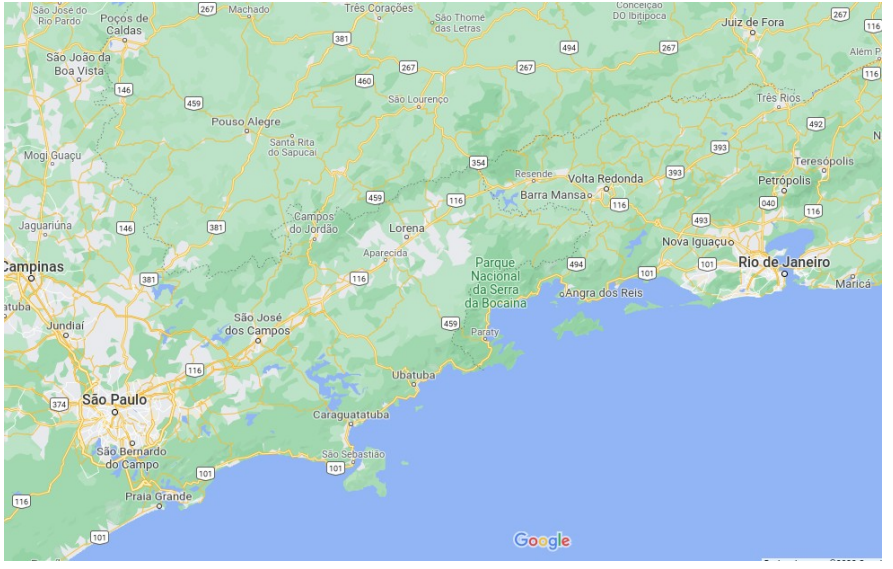
Um motorista deseja encontrar a rota mais curta possível do Rio de Janeiro a São Paulo. Dado um mapa rodoviário do Brasil no qual a distância entre cada par de interseções adjacentes esteja marcada, como ele pode determinar essa rota mais curta?



Introdução

SOLUÇÃO

- Enumerar todos os caminhos possíveis;
- Somas as distâncias de cada rota
- Selecionar a mais curta



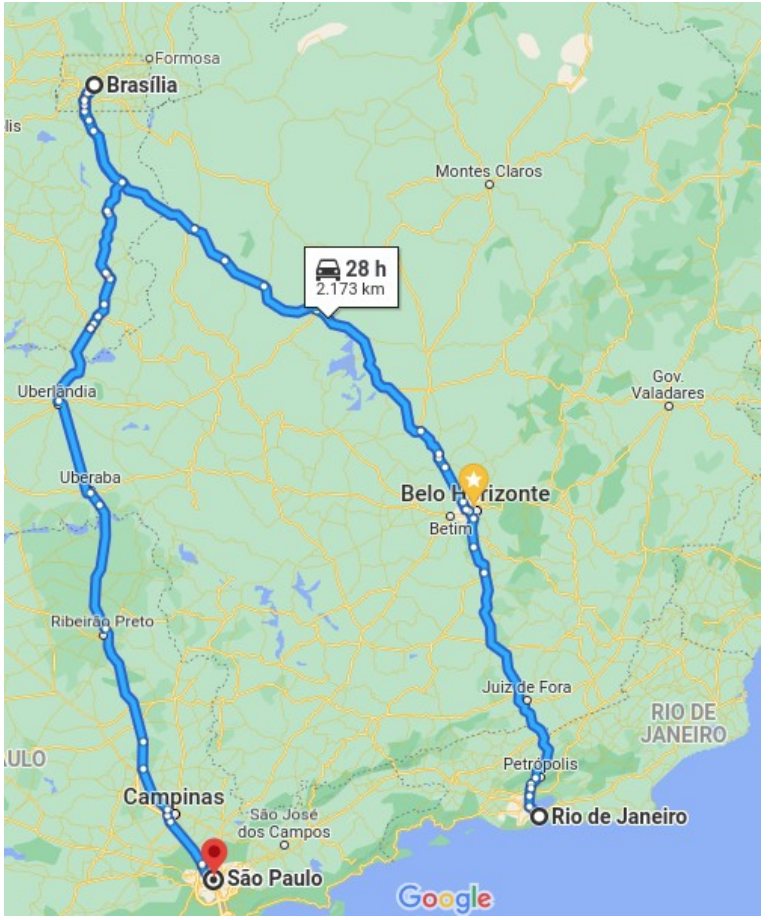
se deixarmos de lado as rotas que contêm ciclos

número enorme de possibilidades

Introdução

Exemplo de solução ineficiente

mostraremos como resolver tais problemas eficientemente



Problema de caminhos mínimos

Temos

Um grafo $G:(V,E)$ com função peso $w:E \rightarrow R$

- O peso do caminho $p:\langle v_0, v_{1,\ldots}, v_k \rangle$ é dado por

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

Problema de caminhos mínimos

Definimos o peso do caminho mínimo de u a v

$$\delta(u,v) = \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{Se há um caminho de } u \text{ para } v, \\ \infty & \text{caso contrário} \end{cases}$$

Então, um caminho mínimo do vértice u ao vértice v é definido como qualquer caminho p com peso w(p) = d(u,v).

Variantes

Focalizaremos o problema de caminhos mínimos de fonte única: dado um grafo $G = (V, E)$, queremos encontrar um caminho mínimo de determinado vértice de origem $s \in V$ a todo vértice $v \in V$.

Problemas resolvidos

- Problema de caminhos mínimos com um só destino
- Problema do caminho mínimo para um par
- Problema de caminhos mínimos para todos os pares

Arestas de peso negativo

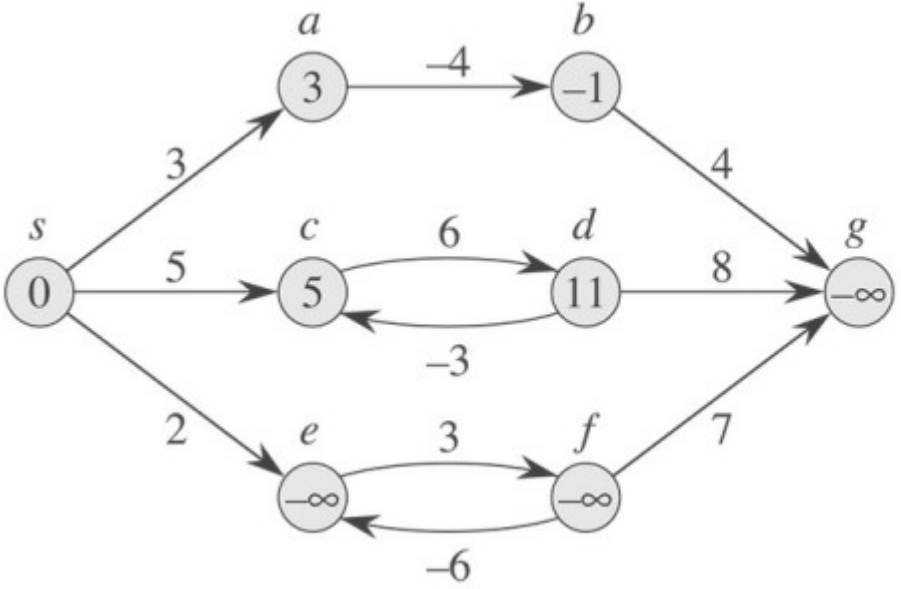
Algumas instâncias do problema de caminhos mínimos de fonte única podem incluir arestas cujos pesos são negativos.

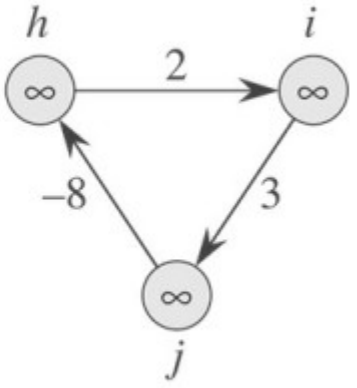
Se o grafo G = (V, E) **não contém** nenhum ciclo de peso negativo que possa ser alcançado da fonte s, então para todo v ∈ V, o peso do caminho mínimo d(s, v) permanece **bem definido**

Se o grafo G = (V, E) **contém** um ciclo de peso negativo que possa ser alcançado a partir de s, os pesos de caminhos mínimos não são bem definidos.

Arestas de peso negativo

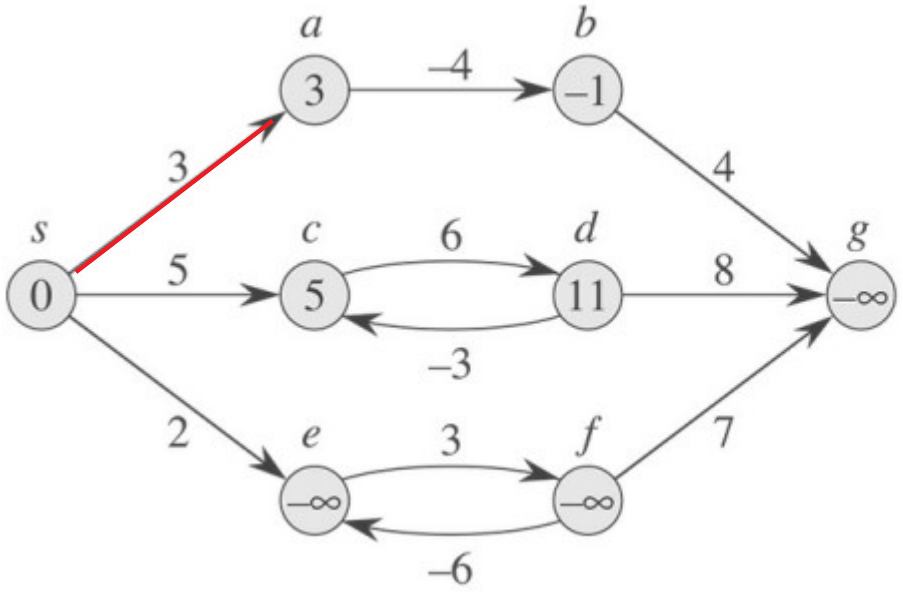
Se o grafo G = (V, E) **não contém** nenhum ciclo de peso negativo que possa ser alcançado da fonte s, então para todo v ∈ V, o peso do caminho mínimo d(s, v) permanece **<u>bem definido</u>**

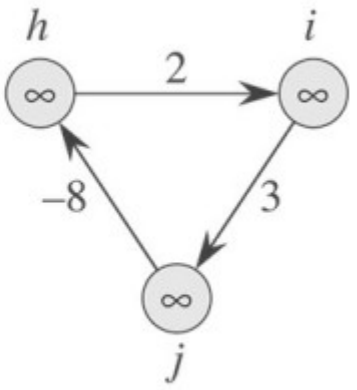




Arestas de peso negativo

Se o grafo G = (V, E) **não contém** nenhum ciclo de peso negativo que possa ser alcançado da fonte s, então para todo v ∈ V, o peso do caminho mínimo d(s, v) permanece **<u>bem definido</u>**

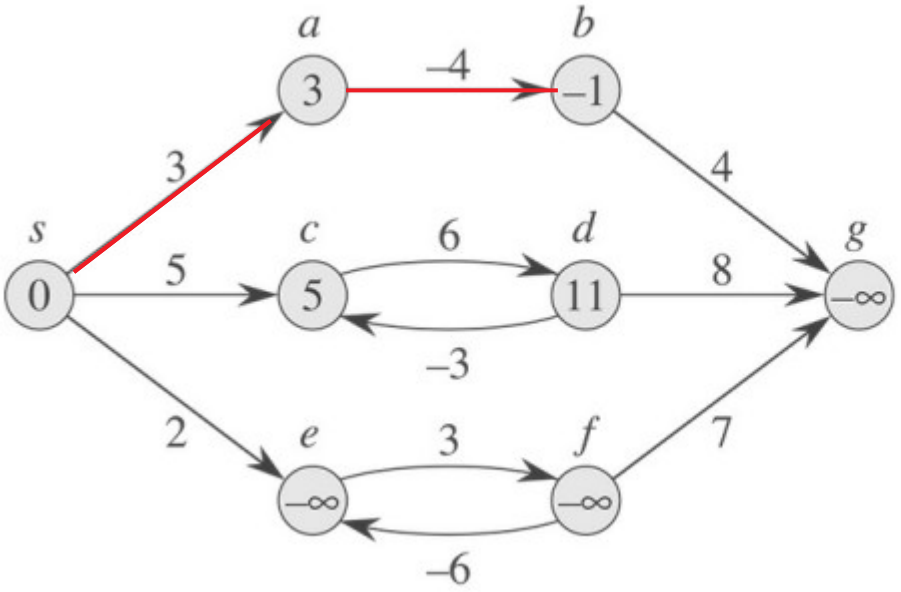


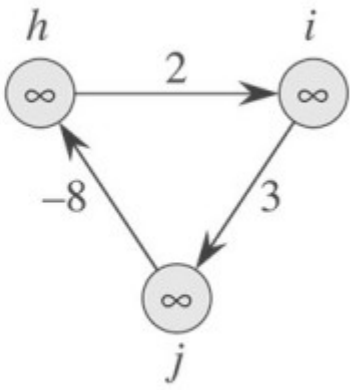


há somente um caminho de s a a (o caminho 〈 s, a 〉), temos d(s, a) = w(s, a) = 3

Arestas de peso negativo

Se o grafo G = (V, E) **não contém** nenhum ciclo de peso negativo que possa ser alcançado da fonte s, então para todo v ∈ V, o peso do caminho mínimo d(s, v) permanece **<u>bem definido</u>**

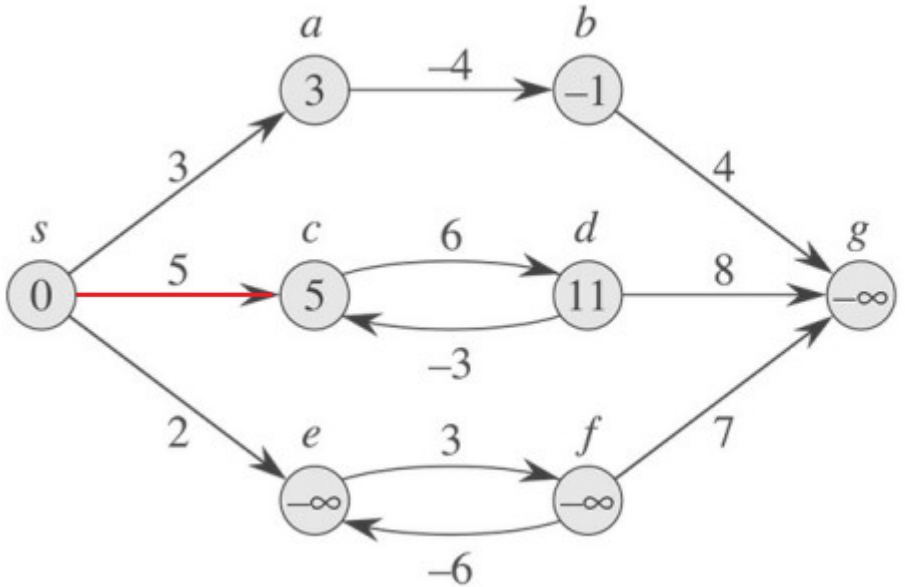


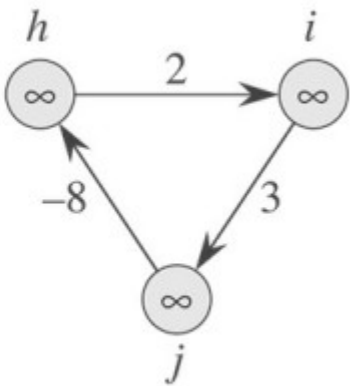


há somente um caminho de s a b (o caminho 〈 s, b 〉), temos d(s, b) = w(s, a) + w(a, b) = 3 + (–4) = –1

Arestas de peso negativo

Se o grafo G = (V, E) **não contém** nenhum ciclo de peso negativo que possa ser alcançado da fonte s, então para todo v ∈ V, o peso do caminho mínimo d(s, v) permanece **<u>bem definido</u>**



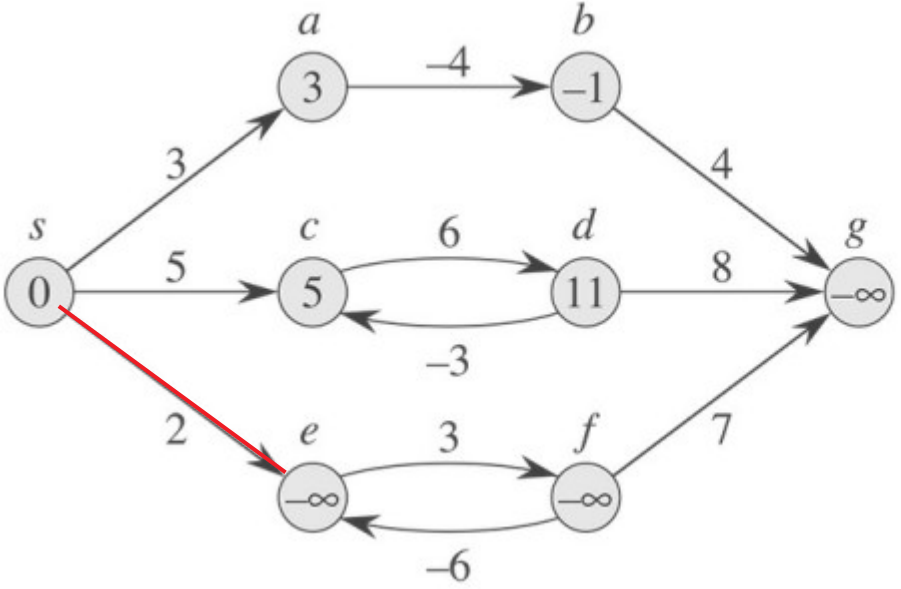


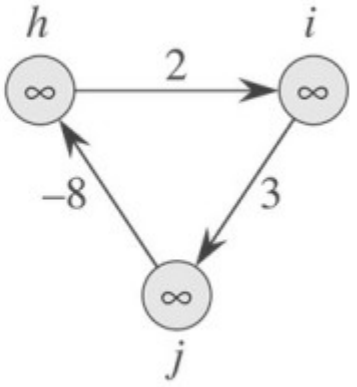
Há um número infinito de caminhos de s a c: 〈 s, c 〉 , 〈 s, c, d, c 〉 , 〈 s, c, d, c, d, c 〉 , e assim por diante

Como o ciclo 〈 c, d, c 〉 , tem peso 6 + (–3) = 3 > 0, o caminho mínimo de s a c é 〈 s, c 〉 , com peso d(s, c) = w (s, c) = 5

Arestas de peso negativo

Se o grafo G = (V, E) **contém** um ciclo de peso negativo que possa ser alcançado a partir de s, os pesos de caminhos mínimos não são bem definidos.





Há um número infinito de caminhos de s a e: 〈 s, e 〉 , 〈 s, e, f, e 〉 , 〈 s, e, f, e, f, e 〉 , e assim por diante. Porém, visto que o ciclo 〈 e, f, e 〉 , tem peso 3 + (–6) = –3 < 0, não há nenhum caminho mínimo de s a e

$$d(s, e) = -\infty. \qquad d(s, f) = -\infty$$

Arestas de peso negativo

Algoritmos de caminhos mínimos:

- Dijkstra → consideram que todos os pesos de arestas no grafo de entrada são não negativos.

- Bellman–Ford → permitem arestas de peso negativo no grafo de entrada e produzem uma resposta correta desde que nenhum ciclo de peso negativo possa ser alcançado da fonte.

Ciclos

Um caminho mínimo pode conter um ciclo?

Não pode conter um ciclo de peso negativo

Também, **não pode** conter um ciclo de peso positivo. (Pq?)

Ciclos

Um caminho mínimo pode conter um ciclo?

Não pode conter um ciclo de peso negativo

Também, **não pode** conter um ciclo de peso positivo. (Pq?)

Remover o ciclo do caminho produz um caminho com os mesmos vértices de fonte e destino, e um **peso de caminho mais baixo**.

Isso deixa apenas ciclos de peso 0. Podemos remover um ciclo de peso 0 de qualquer caminho para produzir um outro caminho cujo peso é o mesmo.

Representação de caminhos mínimos

Dado um grafo G = (V, E), mantemos para cada vértice v $\in$ V um **predecessor** π[v] que é um outro vértice ou NIL .

Subgrafo dos predecessores $G_\pi = (V_\pi, E_\pi)$ induzido pelos valores π.

$$V_\pi = \{v \in V : \pi[v] \neq \text{NIL}\} \cup \{s\} .$$

$$E_\pi = \{(\pi[v], v) \in E : v \in V_\pi - \{s\}\} .$$

Representação de caminhos mínimos

Uma árvore de caminhos mínimos com raiz em s é um subgrafo dirigido $G' = (V', E')$, onde $V' \subseteq V$ e $E' \subseteq E$, tal que

1. V’ é o conjunto de vértices que podem ser alcançados de s em G,
2. G’ forma uma árvore enraizada com raiz s e
3. para todo $v \in V'$, o único caminho simples de s a v em G’ é um caminho mínimo de s a v em G.

Caminhos mínimos não são necessariamente únicos nem são necessariamente únicas as árvores de caminhos mínimos.

Relaxamento

Para cada vértice v ∈ V, mantemos um atributo d[v], que é um limite superior para o peso de um caminho mínimo da fonte s a v. Denominamos d[v] uma estimativa de caminho mínimo.

Denominamos d[v] uma estimativa de caminho mínimo

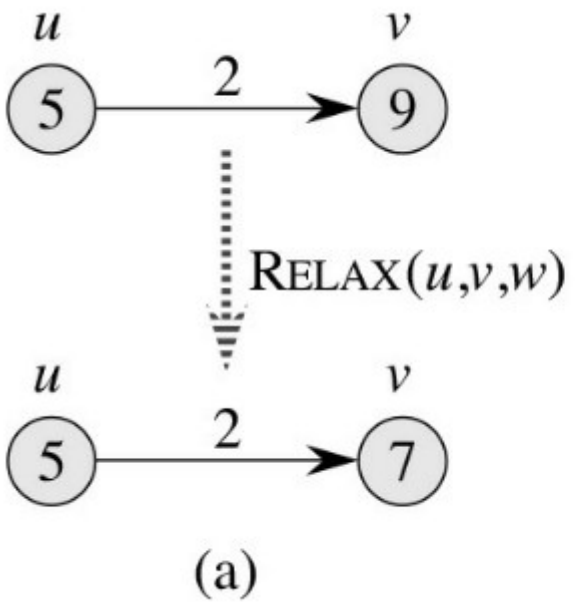
```
INITIALIZE-SINGLE-SOURCE(G, s)
1 for cada vértice v ∈ V[G]
2     do d[v] ← ∞
3        π[v] ← NIL
4 d[s] ← 0
```

Após a inicialização, temos:

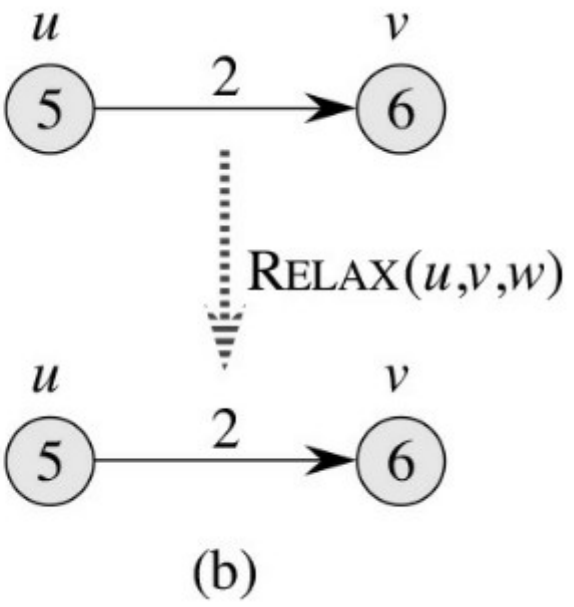
- π[v] = NIL para todo v ∈ V,
- d[s] = 0 e d[v] = ∞ para v ∈ V – {s}.

Relaxamento

O processo de relaxar uma aresta (u, v) consiste em testar se podemos **melhorar o caminho mínimo** até v que encontramos até agora passando por u e, em caso positivo.



(a)



(b)

Relaxamento

O processo de relaxar uma aresta (u, v) consiste em testar se podemos **melhorar o caminho mínimo** até v que encontramos até agora passando por u e, em caso positivo.

```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2     then d[v] ← d[u] + w(u, v)
3          π[v] ← u
```

Propriedades de caminhos mínimos e relaxamento

- **Desigualdade triangular (Lema 24.10)**

Para qualquer aresta $(u, v) \in E$, temos $d(s, v) \leq d(s, u) + w(u, v)$.

- **Propriedade do limite superior (Lema 24.11)**

Sempre temos $d[v] \geq d(s, v)$ para todos os vértices $v \in V$ e, tão logo $d[v]$ alcança o valor $d(s, v)$, nunca mais muda.

- **Propriedade de inexistência de caminho (Corolário 24.12)**

Se não existe nenhum caminho de s a v, então sempre temos $d[v] = d(s, v) = \infty$.

Propriedades de caminhos mínimos e relaxamento

Propriedade de convergência (Lema 24.14)

Se $s \overset{u}{\rightarrow} v$ é um caminho mínimo em G para algum $u, v \in V$ e se $d[u] = d(s, u)$ em qualquer instante antes de relaxar a aresta (u, v), então $d[v] = d(s, v)$ em todos os instantes posteriores.

- **Propriedade de relaxamento de caminho (Lema 24.15)**

Se $p = \langle v_0, v_1, ..., v_k \rangle$ é um caminho mínimo de $s = v_0$ a v_k e relaxamos as arestas de p na ordem $(v_0, v_1), (v_1, v_2), ..., (v_{k-1}, v_k)$, então $v_k.d = d(s, v_k)$. Essa propriedade é válida independentemente de quaisquer outras etapas de relaxamento que ocorram, ainda que elas estejam misturadas com relaxamentos das arestas de p.

- **Propriedade do subgrafo dos predecessores (Lema 24.17)**

Assim que $d[v] = d(s, v)$ para todo $v \in V$, o subgrafo dos predecessores é uma árvore de caminhos mínimos com raiz em s

O ALGORITMO DE BELLMAN- FORD

ALGORITMO DE BELLMAN- FORD

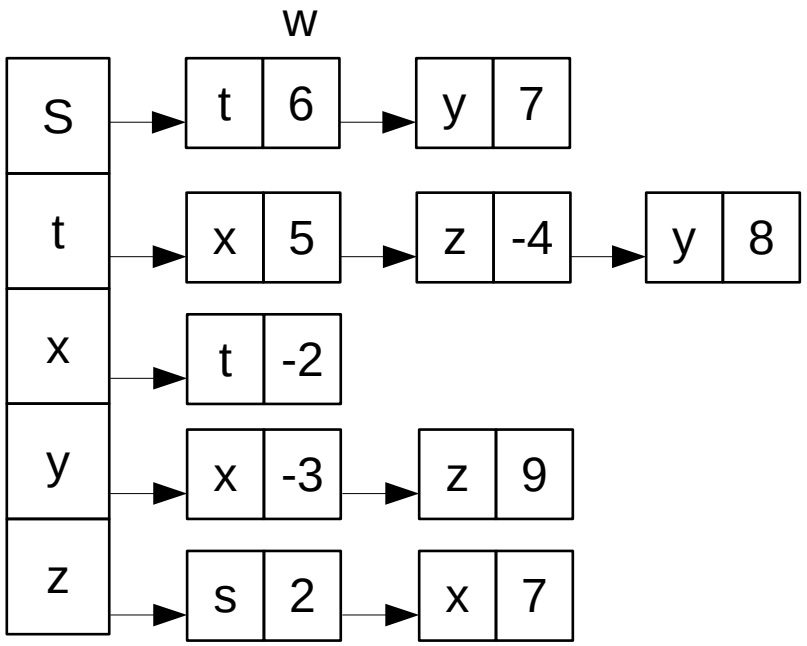
Resolve o problema de caminhos mínimos de fonte única no caso geral no qual os pesos das arestas podem ser negativos.

- Retorna se existe um ciclo de peso negativo (True).
  - Se tal ciclo existe, o algoritmo indica que não há nenhuma solução.
  - Se tal ciclo não existe, o algoritmo produz os caminhos mínimos e seus pesos

ALGORITMO DE BELLMAN- FORD

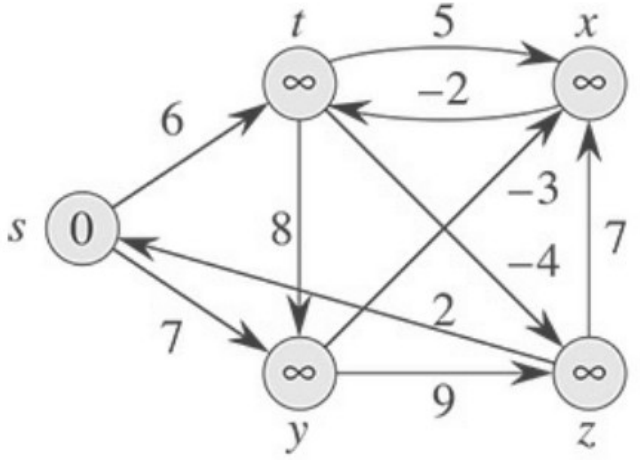
```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i ← 1 to |V[G]| – 1
3       do for cada aresta (u, v) ∈ E[G]
4             do RELAX(u, v, w)
5  for cada aresta (u, v) ∈ E[G]
6       do if d[v] > d[u] + w(u, v)
7             then return FALSE
8  return TRUE
```

ALGORITMO DE BELLMAN- FORD



	s	t	x	y	z
d					
π					

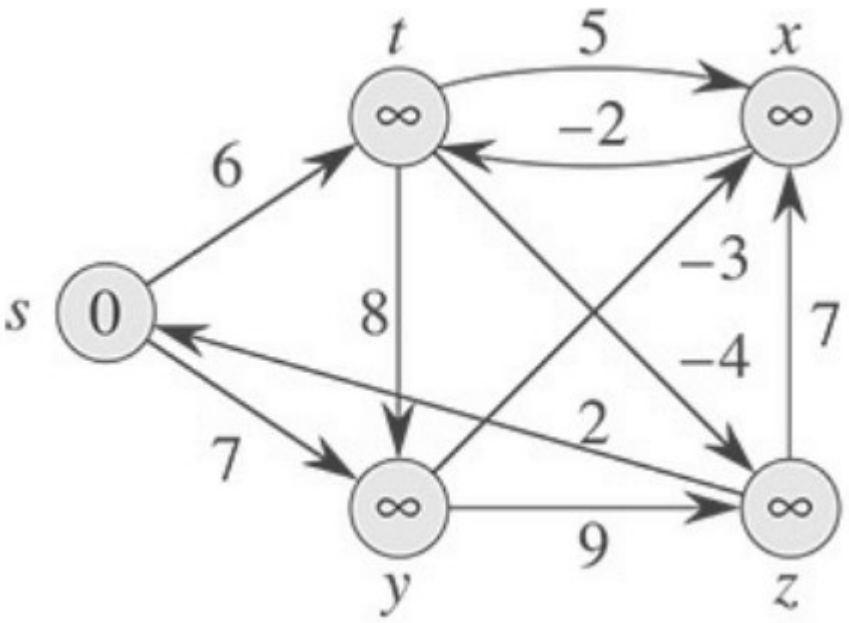
```
BELLMAN-FORD(G, w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 for i ← 1 to |V[G]| − 1
3       do for cada aresta (u, v) ∈ E[G]
4             do RELAX(u, v, w)
5 for cada aresta (u, v) ∈ E[G]
6       do if d[v] > d[u] + w(u, v)
7             then return FALSE
8 return TRUE
```



ALGORITMO DE BELLMAN- FORD

w

S	t	6	y	7
t	x	5	z	-4	y	8
x	t	-2			
y	x	-3	z	9	
z	s	2	x	7	



	s	t	x	y	z
d	0	∞	∞	∞	∞
π	nill	nill	nill	nill	nill

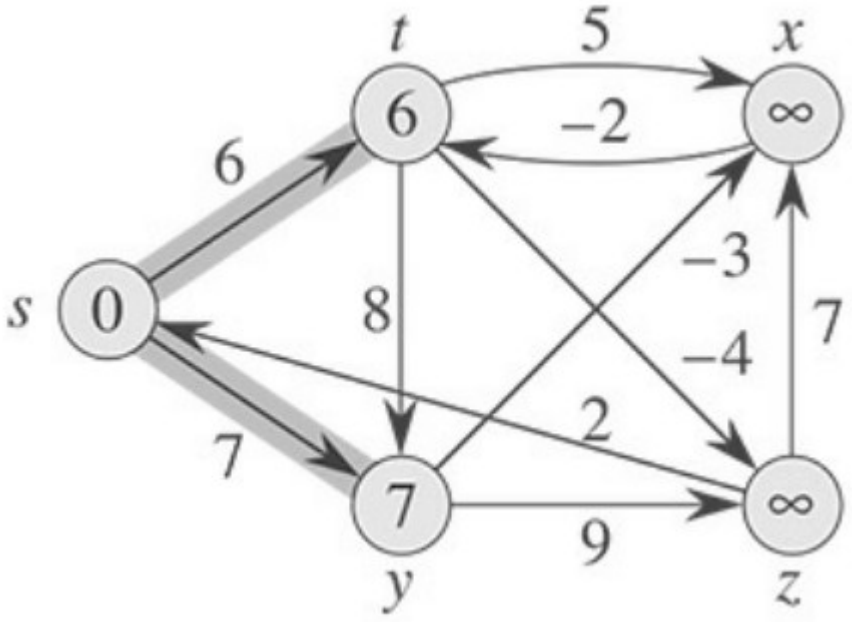
Inicializa

ALGORITMO DE BELLMAN- FORD

w

S	t	6	y	7	
t	x	5	z	-4	y 8
x	t	-2			
y	x	-3	z	9	
z	s	2	x	7	

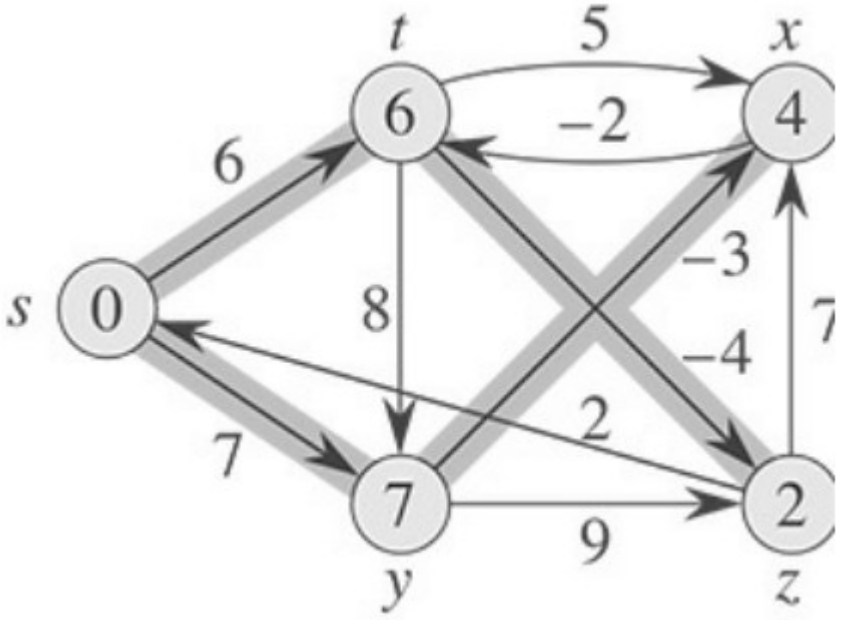
	s	t	x	y	z
d	0	6	∞	7	∞
π	nill	s	nill	s	nill



ALGORITMO DE BELLMAN- FORD

w

S	→	t	6	→	y	7
t	→	x	5	→	z	-4 → y 8
x	→	t	-2			
y	→	x	-3	→	z	9
z	→	s	2	→	x	7



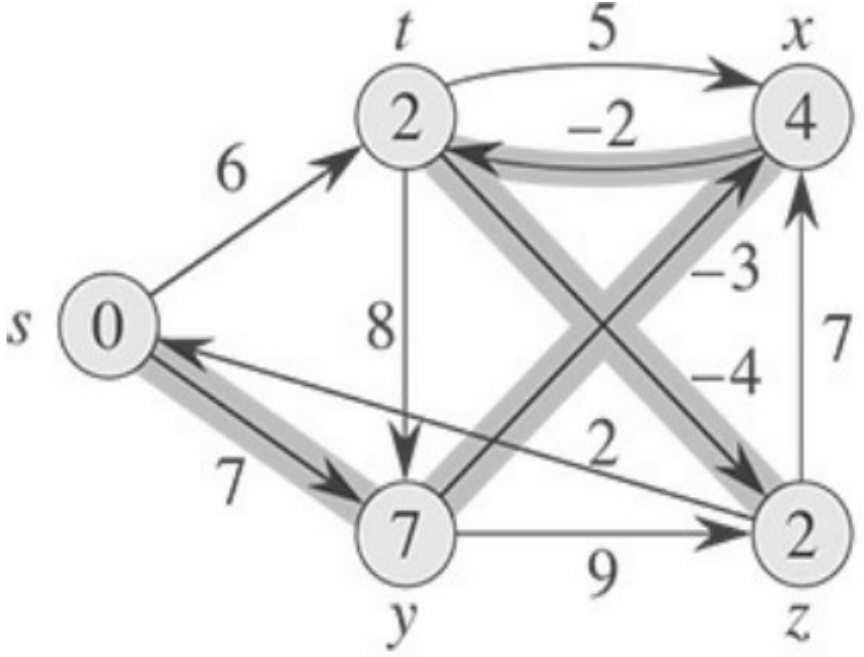
	s	t	x	y	z
d	0	6	4	7	2
π	nill	s	y	s	t

ALGORITMO DE BELLMAN- FORD

w

s	t	6	y	7	
t	x	5	z	-4	y 8
x	t	-2			
y	x	-3	z	9	
z	s	2	x	7	

	s	t	x	y	z
d	0	2	4	7	2
π	nill	x	y	s	t

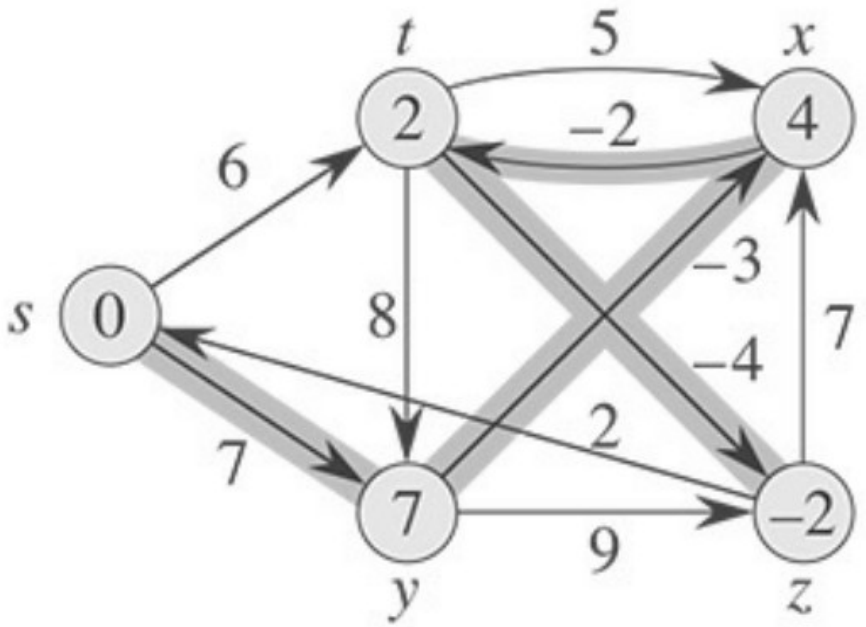


ALGORITMO DE BELLMAN- FORD

w

s	→	t	6	→	y 7
t	→	x	5	→	z -4 → y 8
x	→	t	-2		
y	→	x	-3	→	z 9
z	→	s	2	→	x 7

	s	t	x	y	z
d	0	2	4	7	-2
π	nill	x	y	s	t



ALGORITMO DE BELLMAN- FORD

O algoritmo de BellmanFord é executado no tempo O(V E)

ALGORITMO DE DIJKSTRA

ALGORITMO DE DIJKSTRA

O algoritmo de Dijkstra resolve o problema de caminhos mínimos de fonte única em um grafo dirigido ponderado G = (V, E) para o caso no qual todos os **pesos de arestas são não negativos**.

Então, supomos que:

$$w(u,v) \geq 0 \text{ para cada aresta } (u, v) \in E$$

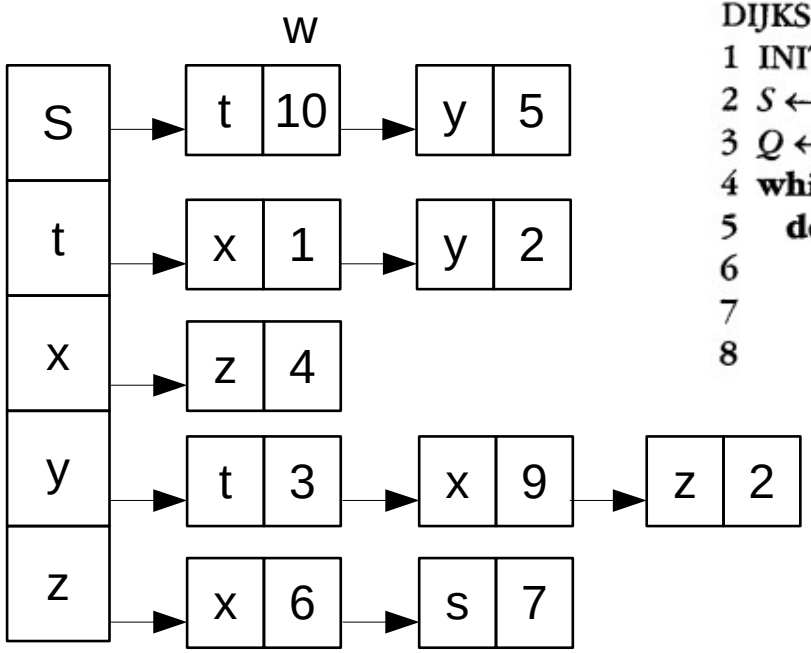
o tempo de execução do algoritmo de Dijkstra é inferior ao do algoritmo de BellmanFord

ALGORITMO DE DIJKSTRA

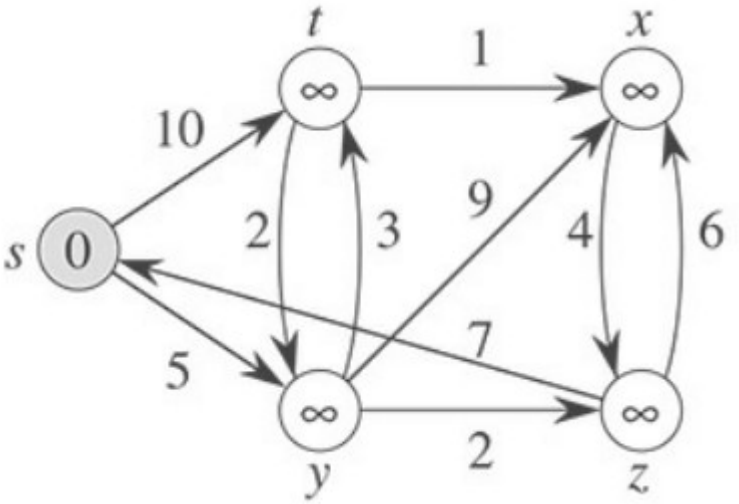
```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S ← ∅
3  Q ← V[G]
4  while Q ≠ ∅
5     do u ← EXTRACT-MIN(Q)
6        S ← S ∪ {u}
7        for cada vértice v ∈ Adj[u]
8           do RELAX(u, v, w)
```

- mantém um conjunto S de vértices cujos pesos finais de caminhos mínimos que partem da fonte s já foram determinados
- Usa-se uma fila de prioridades mínimas Q de vértices cujas chaves são os valores de d

ALGORITMO DE DIJKSTRA



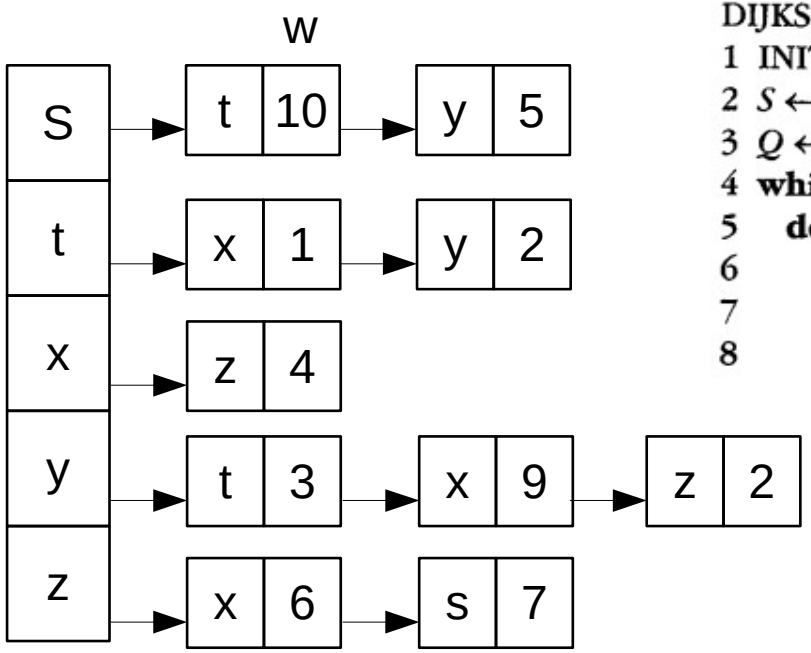

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S ← ∅
3  Q ← V[G]
4  while Q ≠ ∅
5     do u ← EXTRACT-MIN(Q)
6        S ← S ∪ {u}
7        for cada vértice v ∈ Adj[u]
8           do RELAX(u, v, w)
```



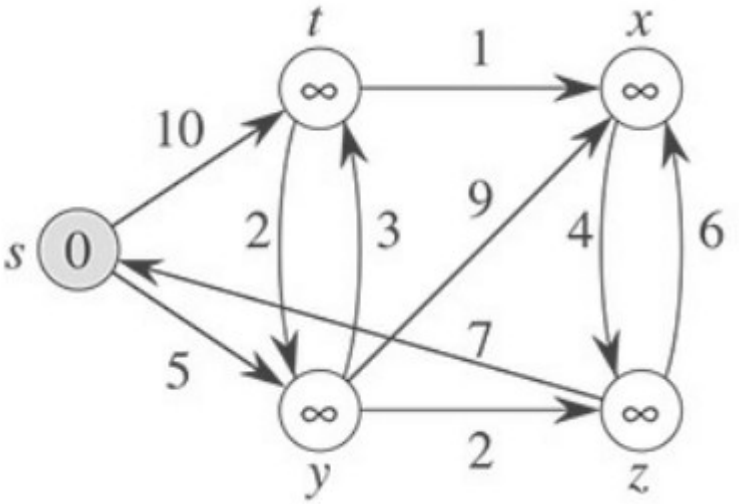
	s	t	x	y	z
d	0				
π					

S

ALGORITMO DE DIJKSTRA



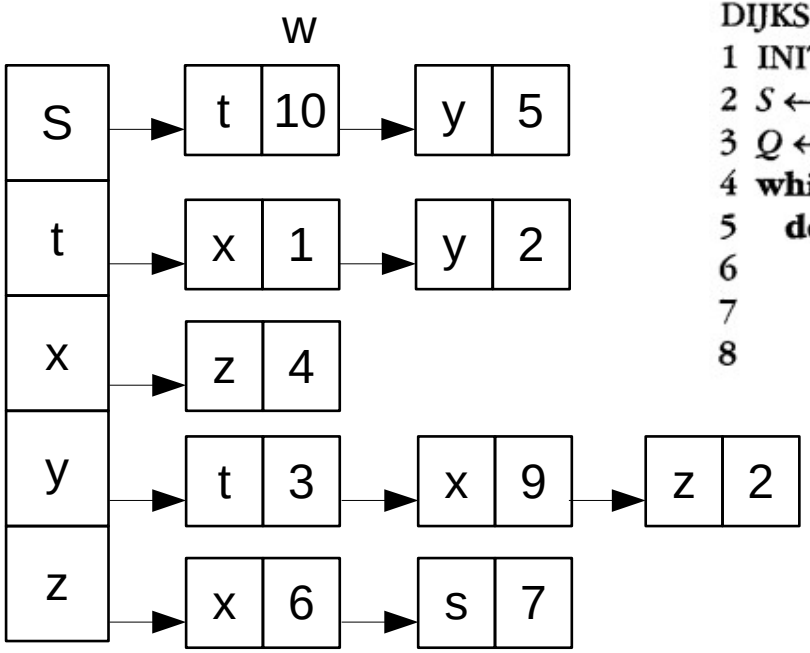

```
DIJKSTRA(G, w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 S ← ∅
3 Q ← V[G]
4 while Q ≠ ∅
5   do u ← EXTRACT-MIN(Q)
6      S ← S ∪ {u}
7      for cada vértice v ∈ Adj[u]
8          do RELAX(u, v, w)
```



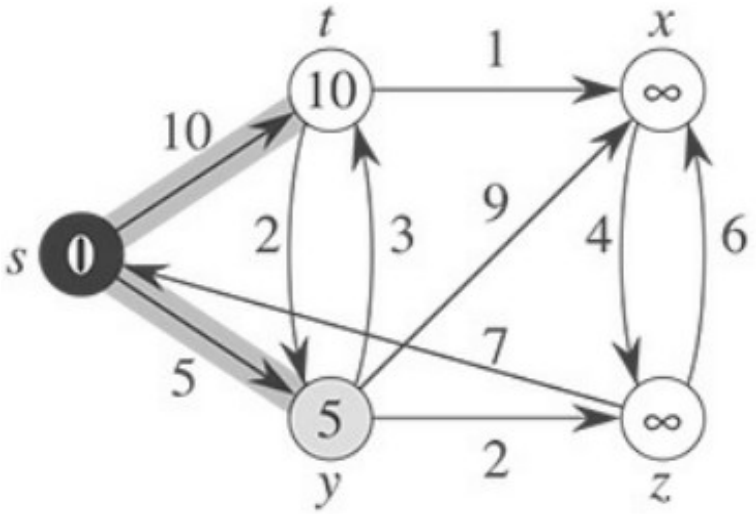
	s	t	x	y	z
d	0	∞	∞	∞	∞
π	nill	nill	nill	nill	nill

S

ALGORITMO DE DIJKSTRA




```
DIJKSTRA(G, w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 S ← ∅
3 Q ← V[G]
4 while Q ≠ ∅
5   do u ← EXTRACT-MIN(Q)
6      S ← S ∪ {u}
7      for cada vértice v ∈ Adj[u]
8          do RELAX(u, v, w)
```

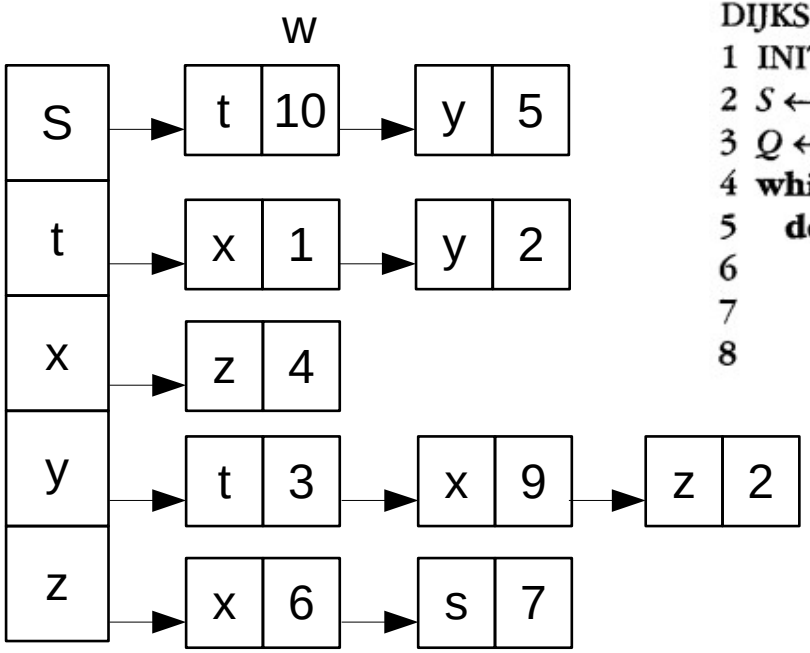


	s	t	x	y	z
d	0	10	∞	5	∞
π	nill	s	nill	s	nill

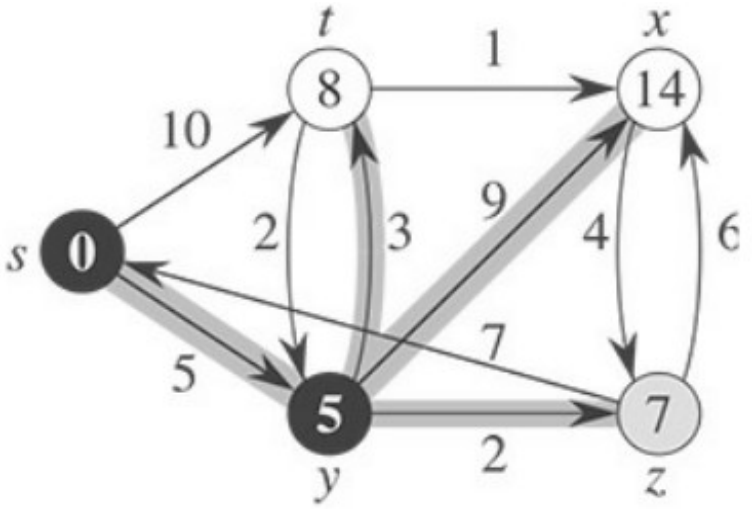
s

s,

ALGORITMO DE DIJKSTRA




```
DIJKSTRA(G, w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 S ← ∅
3 Q ← V[G]
4 while Q ≠ ∅
5    do u ← EXTRACT-MIN(Q)
6       S ← S ∪ {u}
7       for cada vértice v ∈ Adj[u]
8            do RELAX(u, v, w)
```

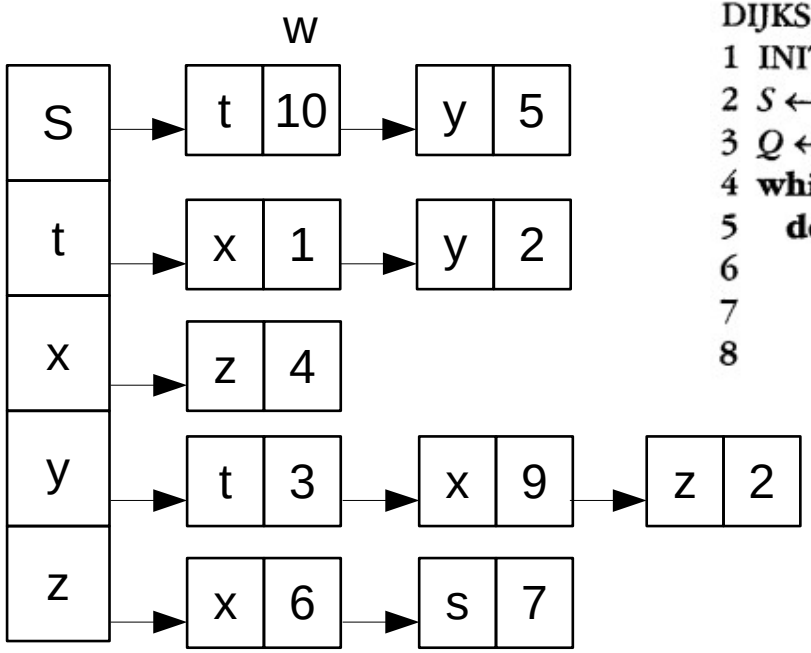


	s	t	x	y	z
d	0	8	14	5	7
π	nill	y	y	s	y

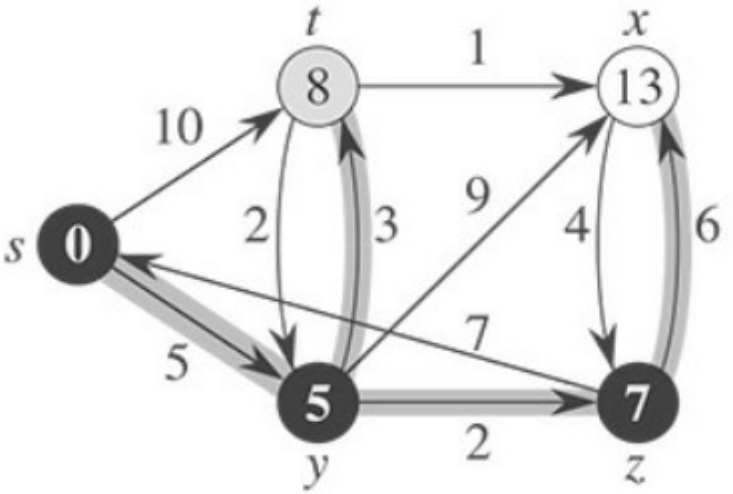
s

s,y

ALGORITMO DE DIJKSTRA




```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S ← ∅
3  Q ← V[G]
4  while Q ≠ ∅
5     do u ← EXTRACT-MIN(Q)
6        S ← S ∪ {u}
7        for cada vértice v ∈ Adj[u]
8           do RELAX(u, v, w)
```

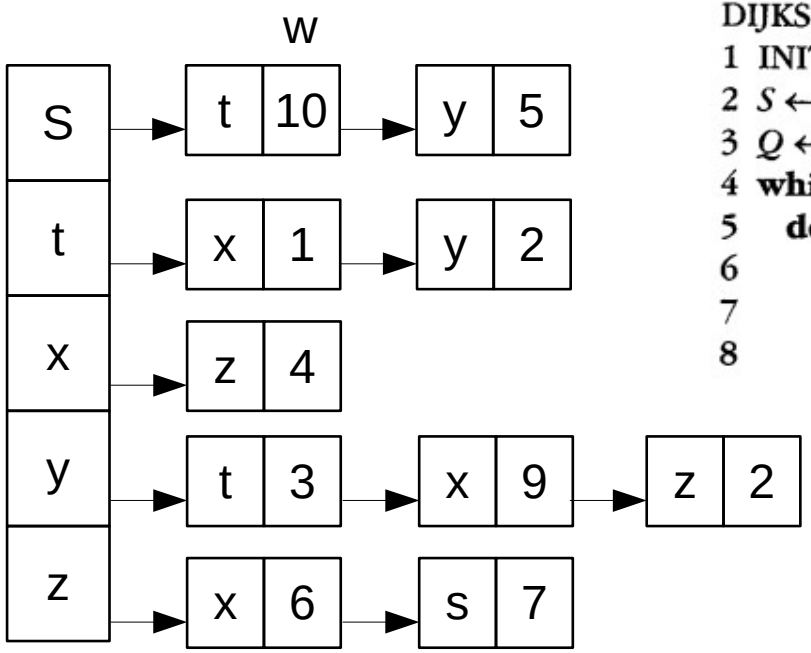


	s	t	x	y	z
d	0	8	13	5	7
π	nill	y	z	s	y

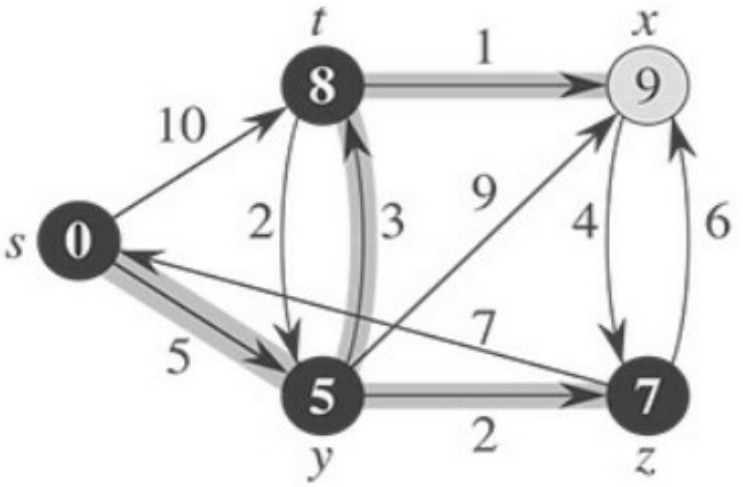
s

s,y,z,t

ALGORITMO DE DIJKSTRA




```
DIJKSTRA(G, w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 S ← ∅
3 Q ← V[G]
4 while Q ≠ ∅
5   do u ← EXTRACT-MIN(Q)
6      S ← S ∪ {u}
7      for cada vértice v ∈ Adj[u]
8          do RELAX(u, v, w)
```

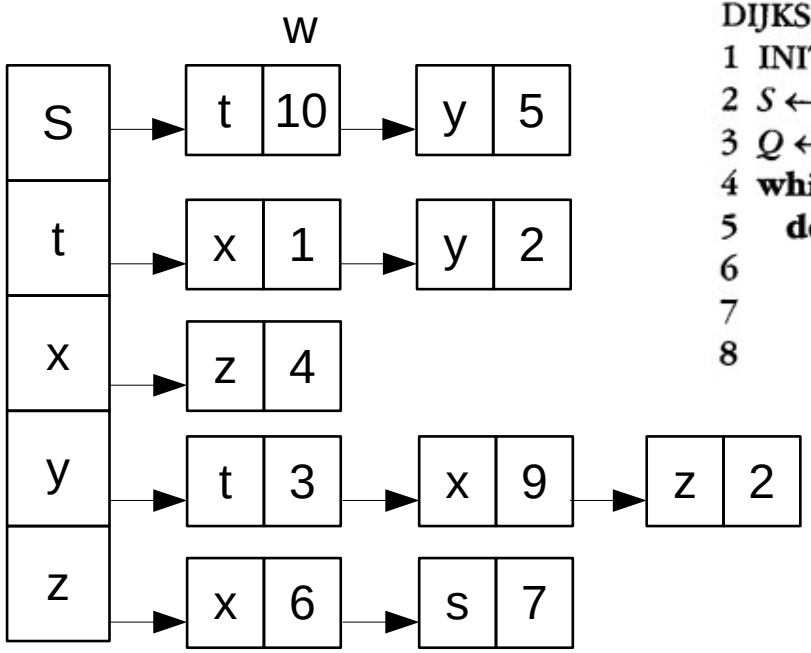


	s	t	x	y	z
d	0	8	9	5	7
π	nill	y	t	s	y

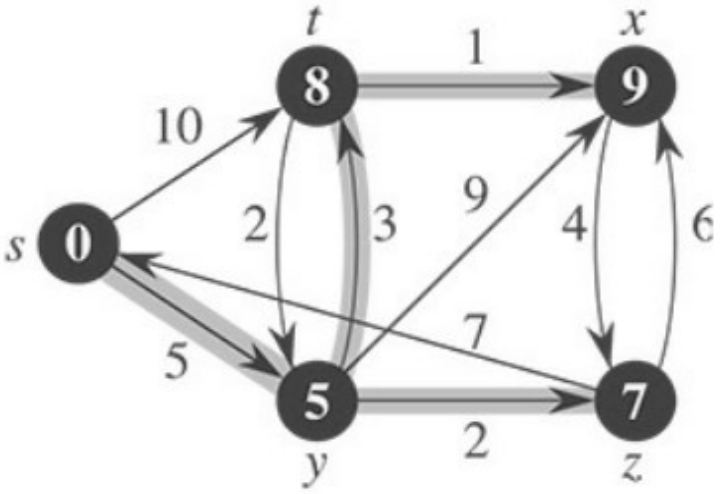
s

s,y,z,t,x

ALGORITMO DE DIJKSTRA




```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S ← ∅
3  Q ← V[G]
4  while Q ≠ ∅
5     do u ← EXTRACT-MIN(Q)
6        S ← S ∪ {u}
7        for cada vértice v ∈ Adj[u]
8           do RELAX(u, v, w)
```



	s	t	x	y	z
d	0	8	9	5	7
π	nill	y	t	s	y

s

s,y,z,t,x