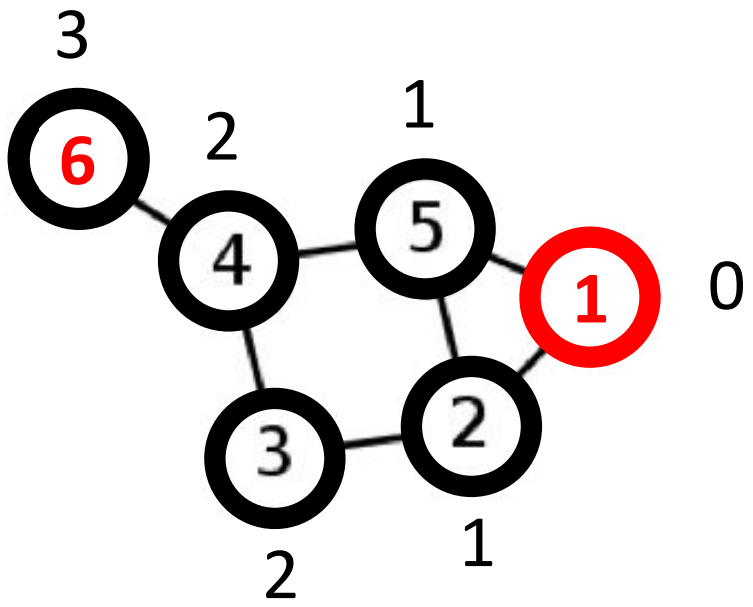
Introdução à Teoria dos Grafos

Prof. Alexandre Noma

Semana passada...

- Caminho mínimo
  - É um caminho de comprimento mínimo.

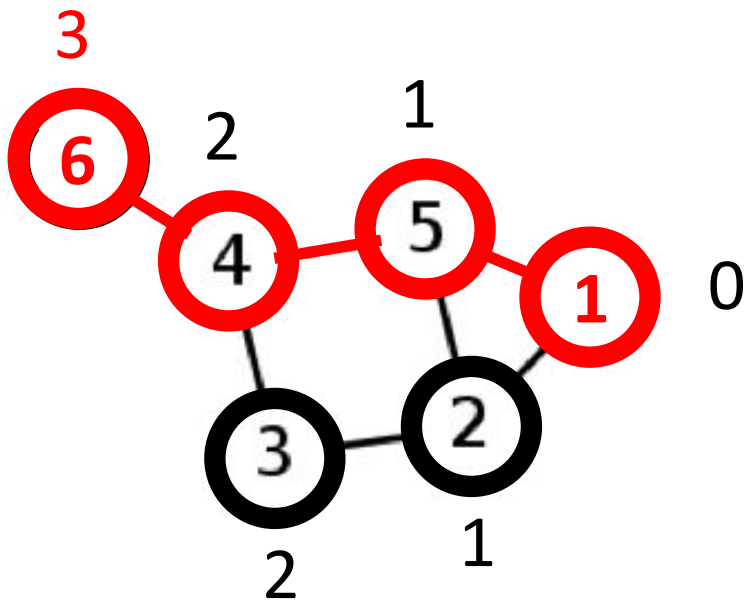
- Ex. qual seria um caminho mínimo de 1 até 6?



Semana passada...

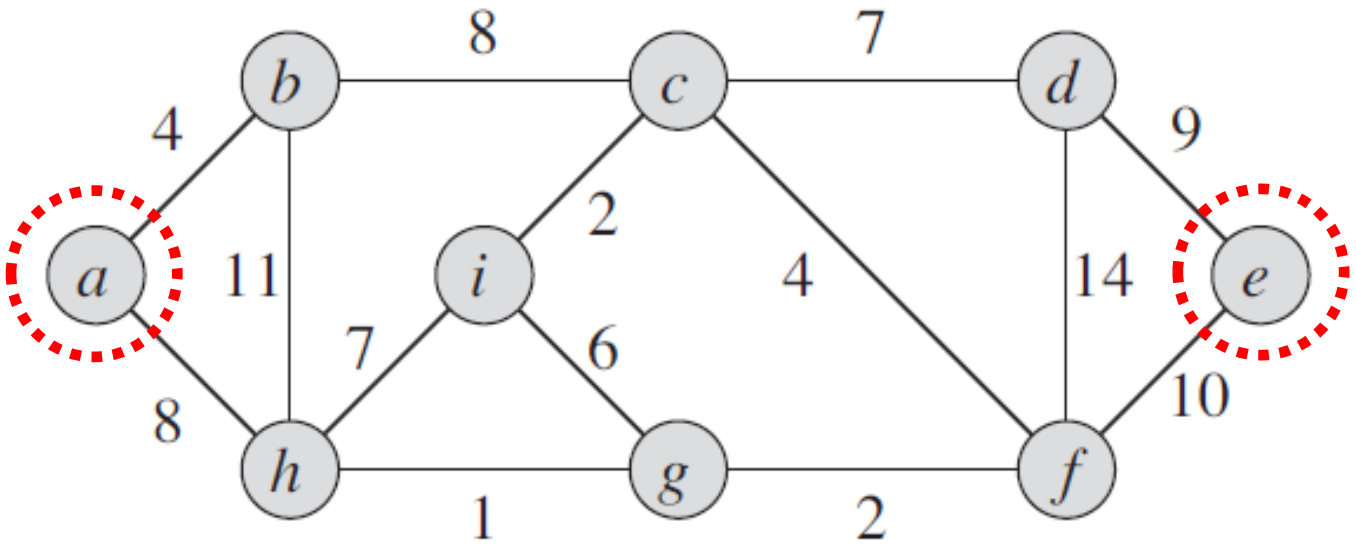
- Caminho mínimo
  - É um caminho de comprimento mínimo.

- Ex. qual seria um caminho mínimo de 1 até 6?



Hoje

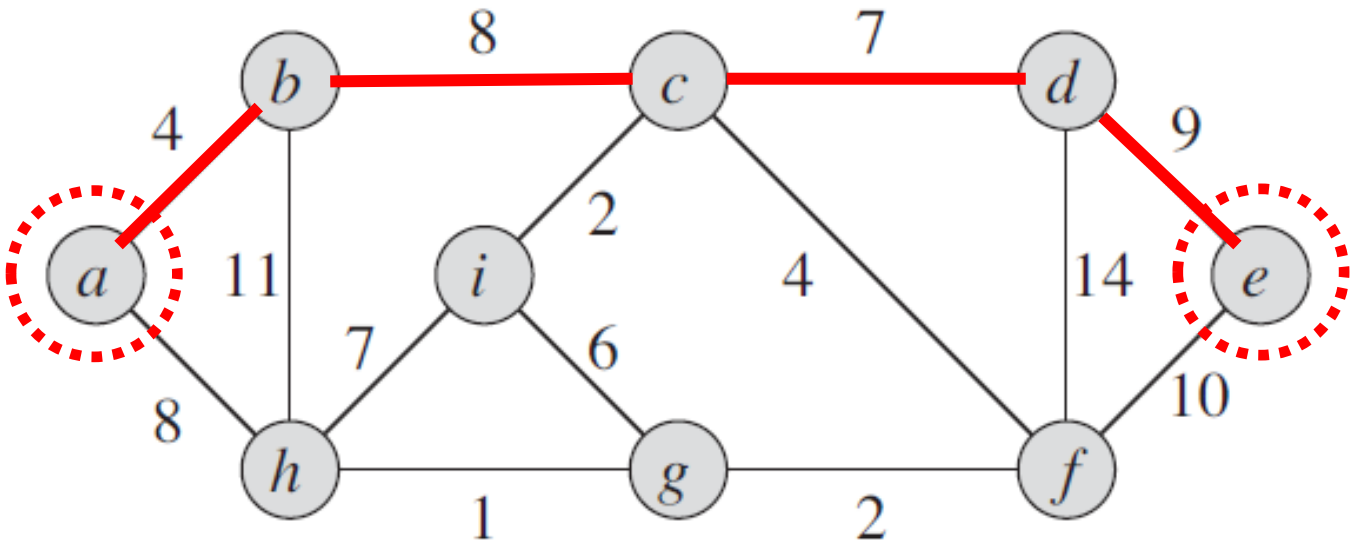
- Calcule um **caminho mínimo** de **a** até **e**?



Hoje

- Calcule um **caminho mínimo** de **a** até **e**?

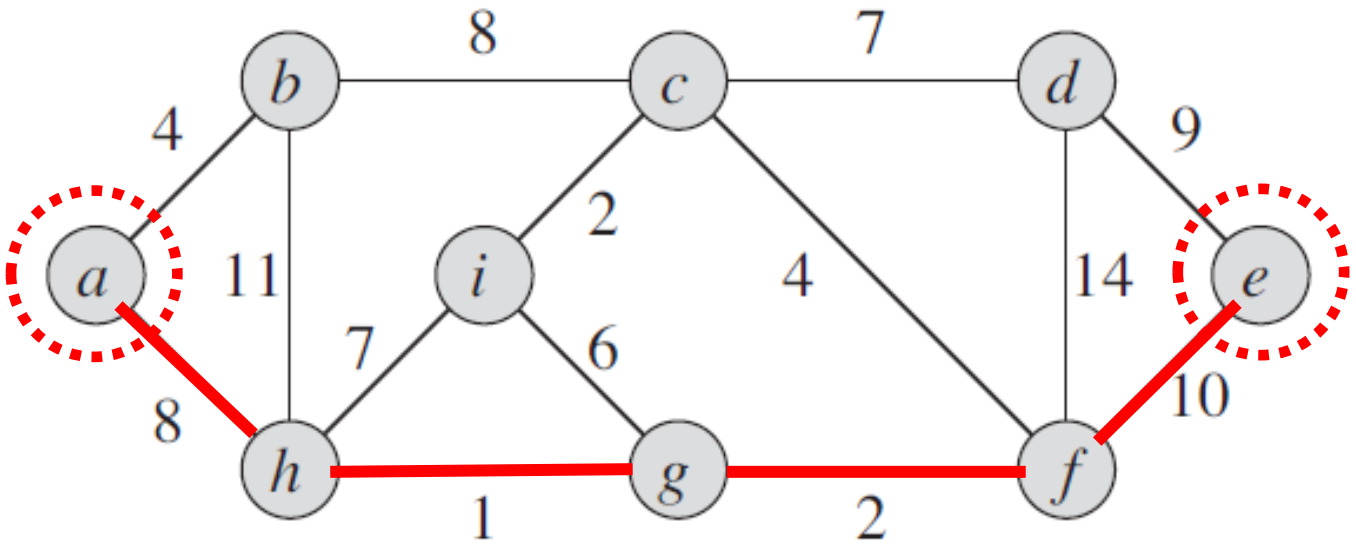
$|P_1| = 4 + 8 + 7 + 9 = 28$ é mínimo?



Hoje

- Calcule um **caminho mínimo** de **a** até **e**?

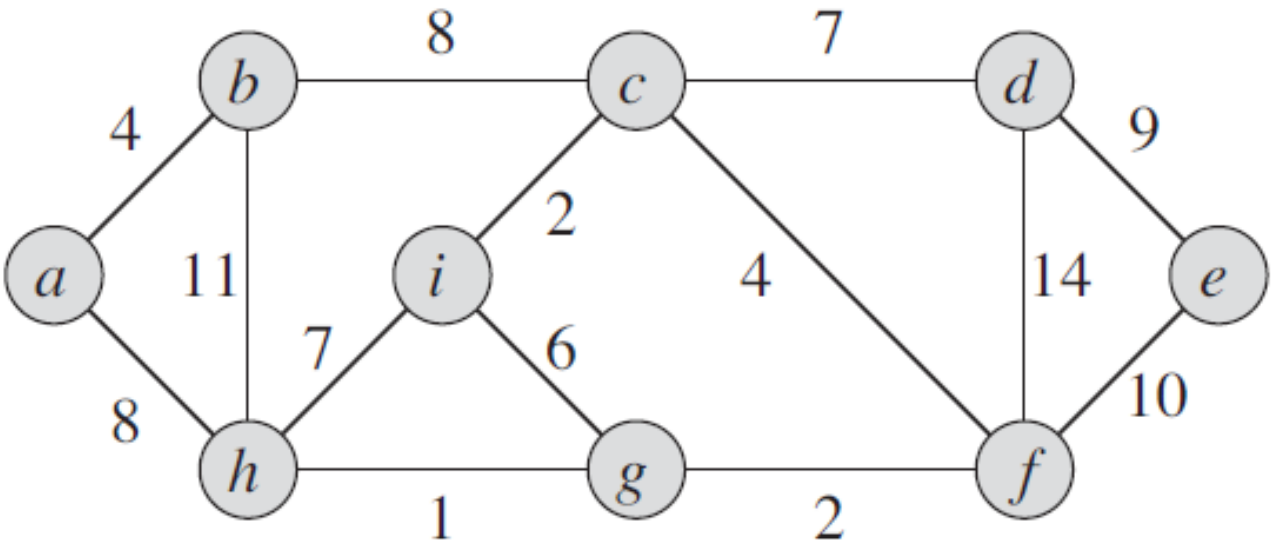
$|P_1| = 4 + 8 + 7 + 9 = 28$ não é mínimo!



$|P_2| = 8 + 1 + 2 + 10 = 21$ é mínimo?

Grafo ponderado

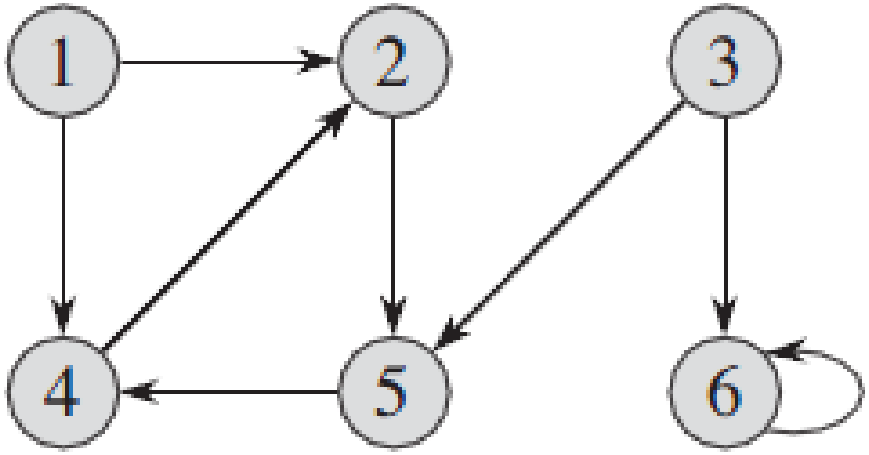
- É um grafo com pesos nas arestas.
- Como representar no computador?



Aulas passadas

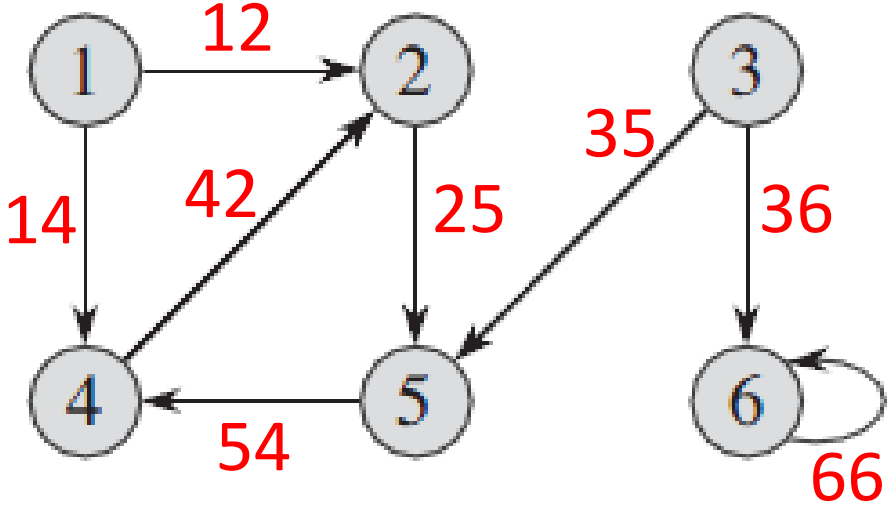
- Representar um grafo no computador:
- (1) Matriz de adjacências
- (2) Listas de adjacências

1. Matriz de Adjacências



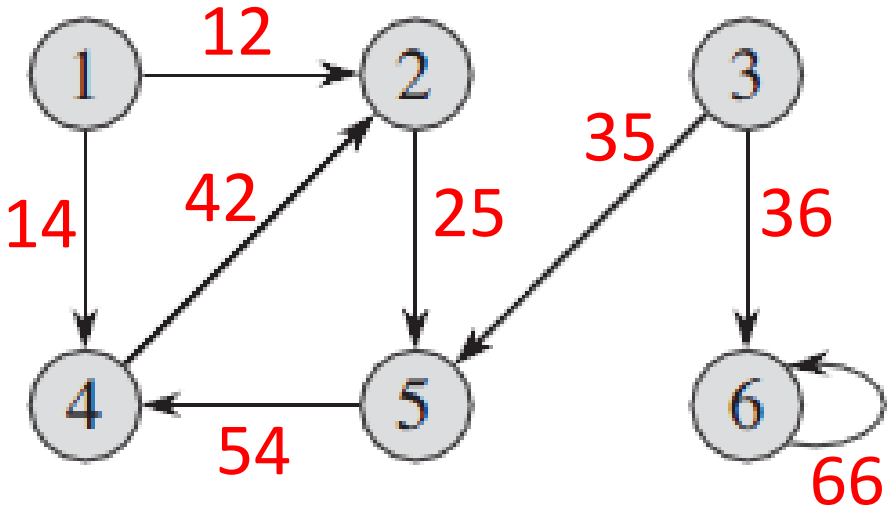
	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

1. Matriz de Adjacências?



	1	2	3	4	5	6
1	0	?	0	?	0	0
2	0	0	0	0	?	0
3	0	0	0	0	?	?
4	0	?	0	0	0	0
5	0	0	0	?	0	0
6	0	0	0	0	0	?

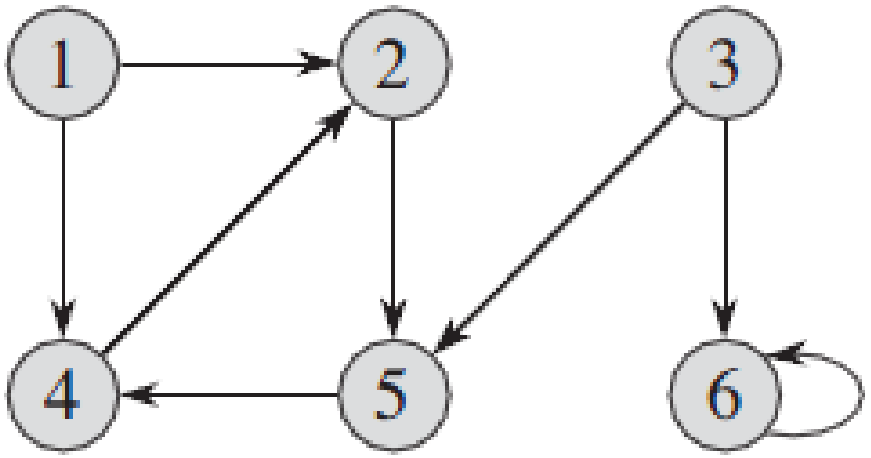
1. Matriz de Adjacências?



	1	2	3	4	5	6
1	0	12	0	14	0	0
2	0	0	0	0	25	0
3	0	0	0	0	35	36
4	0	42	0	0	0	0
5	0	0	0	54	0	0
6	0	0	0	0	0	66

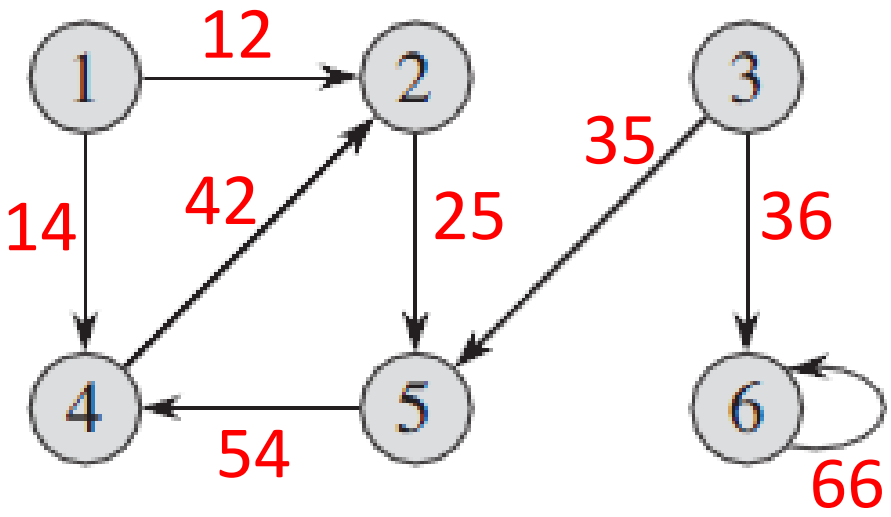
2. Listas de Adjacências

(2) Listas de adjacências



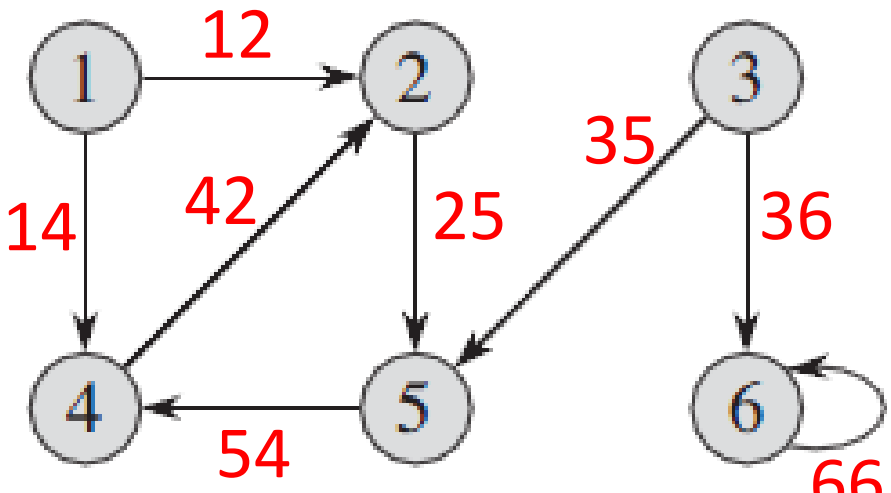
1: 2, 4
2: 5
3: 5, 6
4: 2
5: 4
6: 6

2. Listas de Adjacências?



1: ?, ?
2: ?
3: ?, ?
4: ?
5: ?
6: ?

2. Listas de Adjacências?



1: 2 (12), 4 (14)
2: 5 (25)
3: 5 (35), 6 (36)
4: 2 (42)
5: 4 (54)
6: 6 (66)

Exercício Programa

- 08-leGrafoPonderado.py

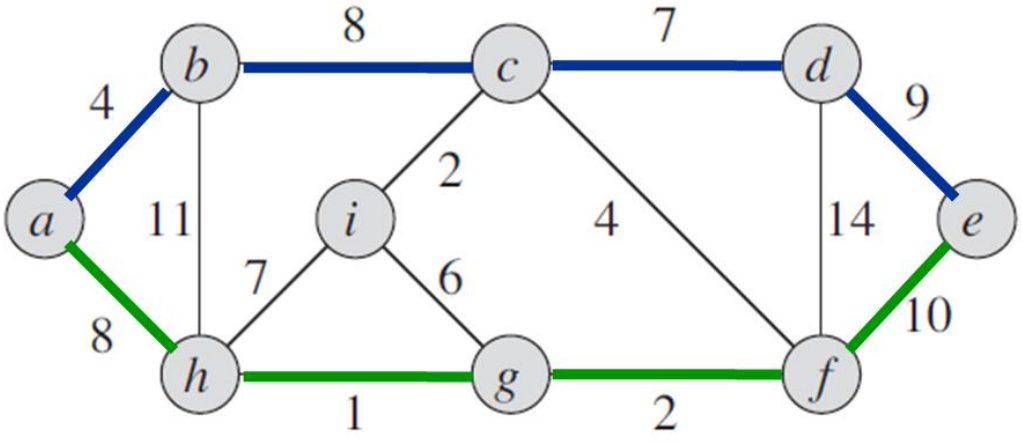
Hoje

- Algoritmo de Dijkstra
  (caminhos mínimos para grafos **ponderados**)

- Pré-requisitos:
  - **Definições**: caminho, comprimento, caminho mínimo, distância
  - Fila de prioridade

Definições

- Caminho
- Comprimento
- Caminho mínimo
- Distância



Fila de prioridade?

Fila de prioridade

- Insere(Q, x)
  - insere elemento x no conjunto Q
- Mínimo(Q)
  - devolve o elemento de Q com a menor chave
- ExtraiMínimo(Q)
  - remove e devolve o elemento de Q com a menor chave
- Vazio(Q)
  - devolve verdadeiro se fila vazia, falso caso contrário

Fila de prioridade

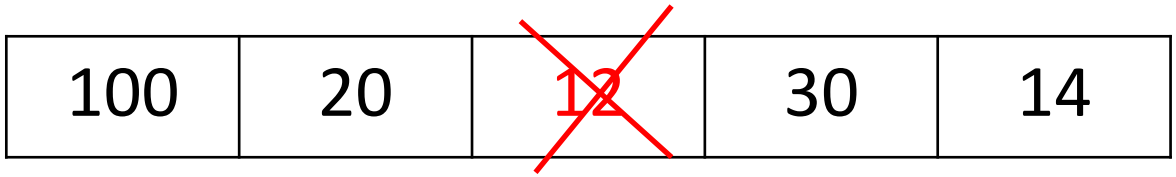
- Exemplo

100	20	12	30	14

ExtraiMínimo: ?

Fila de prioridade

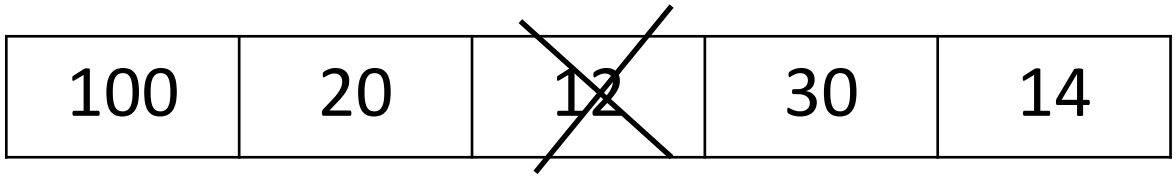
- Exemplo



ExtraiMínimo: 12

Fila de prioridade

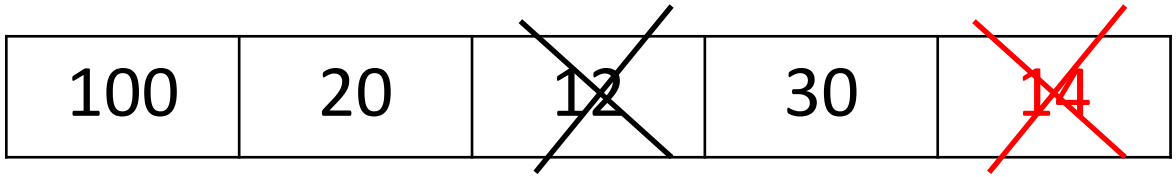
- Exemplo



ExtraiMínimo: ?

Fila de prioridade

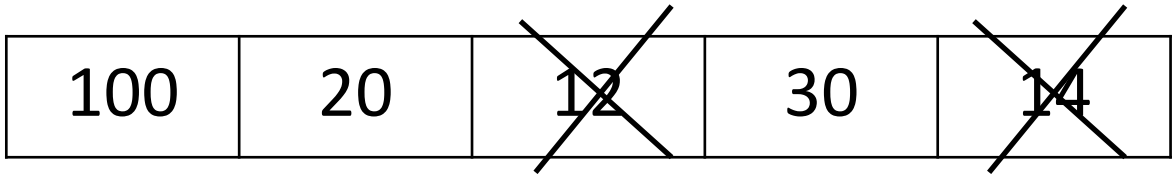
- Exemplo



ExtraiMínimo: 14

Fila de prioridade

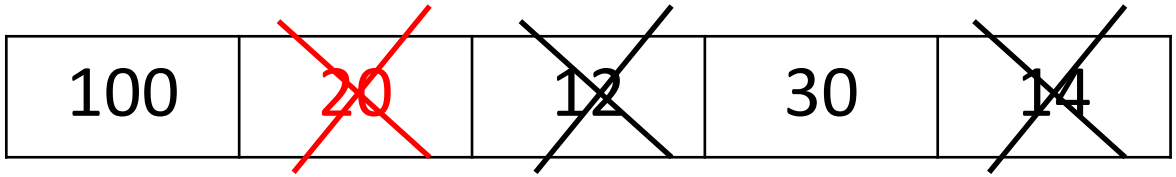
- Exemplo



ExtraiMínimo: ?

Fila de prioridade

- Exemplo



ExtraiMínimo: 20

Exercício Programa

- 09-filaDePrioridade.py
  (implementação simples e ineficiente com **vetor de índices**)

- Ex. Implementação eficiente com HEAP

Algoritmo de Dijkstra

- Dijkstra(G, w, s)
  - Entrada: um grafo **G**, ponderado com pesos **w**, e um vértice inicial **s**
  - Saída: distâncias em relação ao vértice inicial (e caminhos mínimos)
- Atributos
  - v.**d**: distância
  - v.$\prod$ (v.**pai**)
    - ("árvore de caminhos mínimos")

Dijkstra

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```

Dijkstra

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```

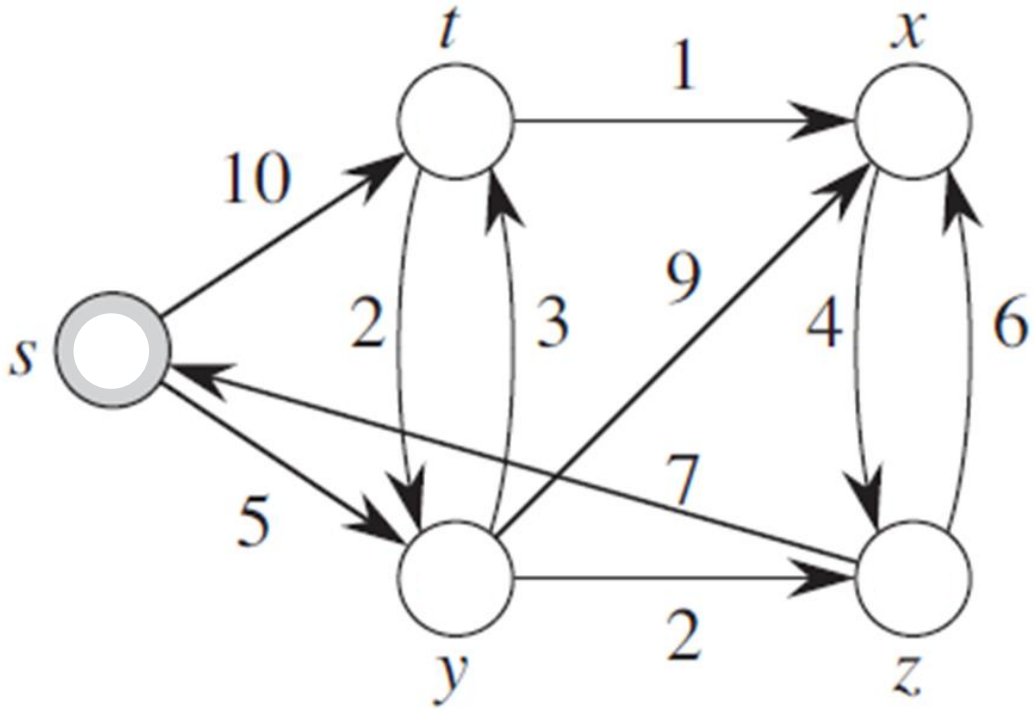
Dijkstra

```
INITIALIZE-SINGLE-SOURCE(G, s)
1  for each vertex v ∈ G.V
2      v.d = ∞
3      v.π = NIL
4  s.d = 0
```

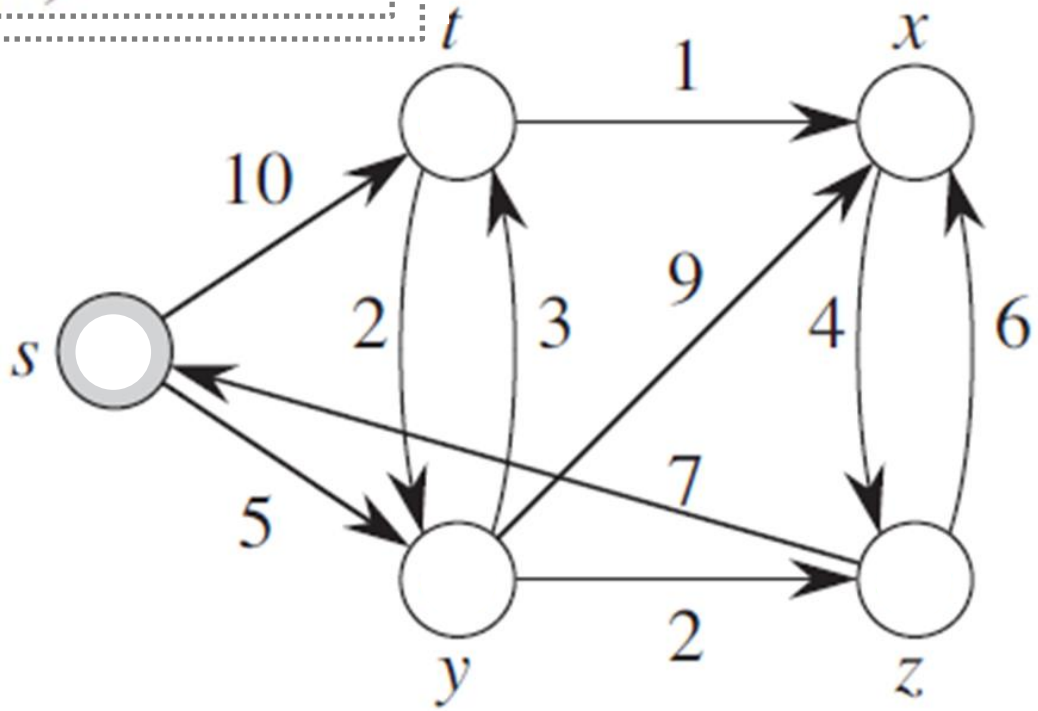
```
RELAX(u, v, w)
1  if u.d + w(u, v) < v.d
2      v.d = u.d + w(u, v)
3      v.π = u
```

Dijkstra

- Exemplo

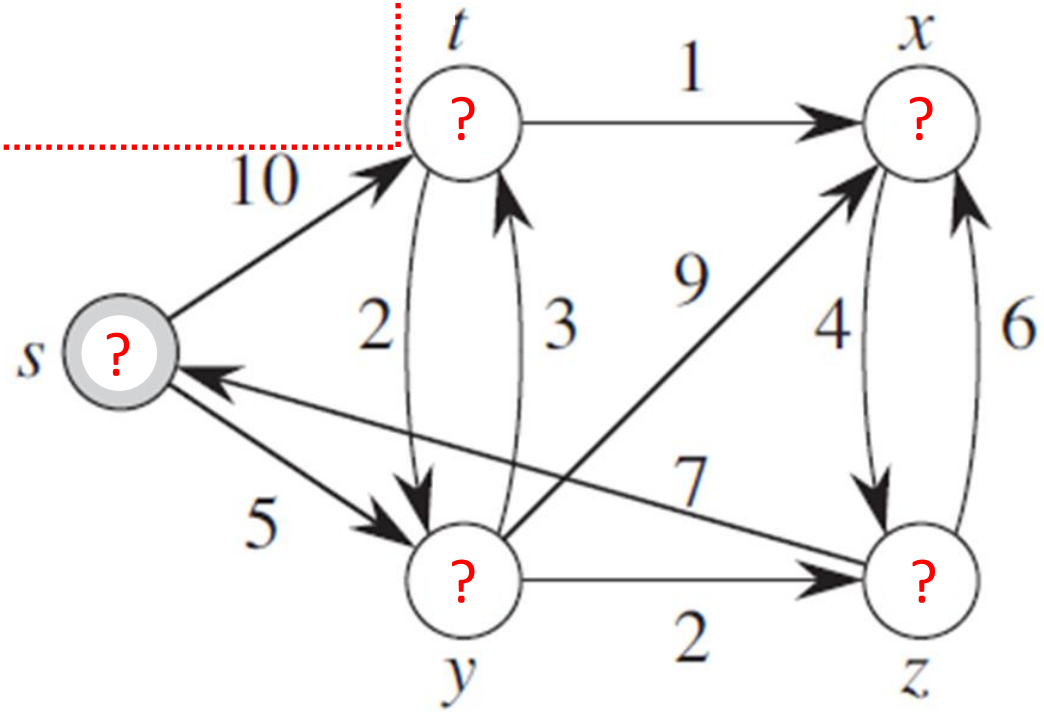


```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```



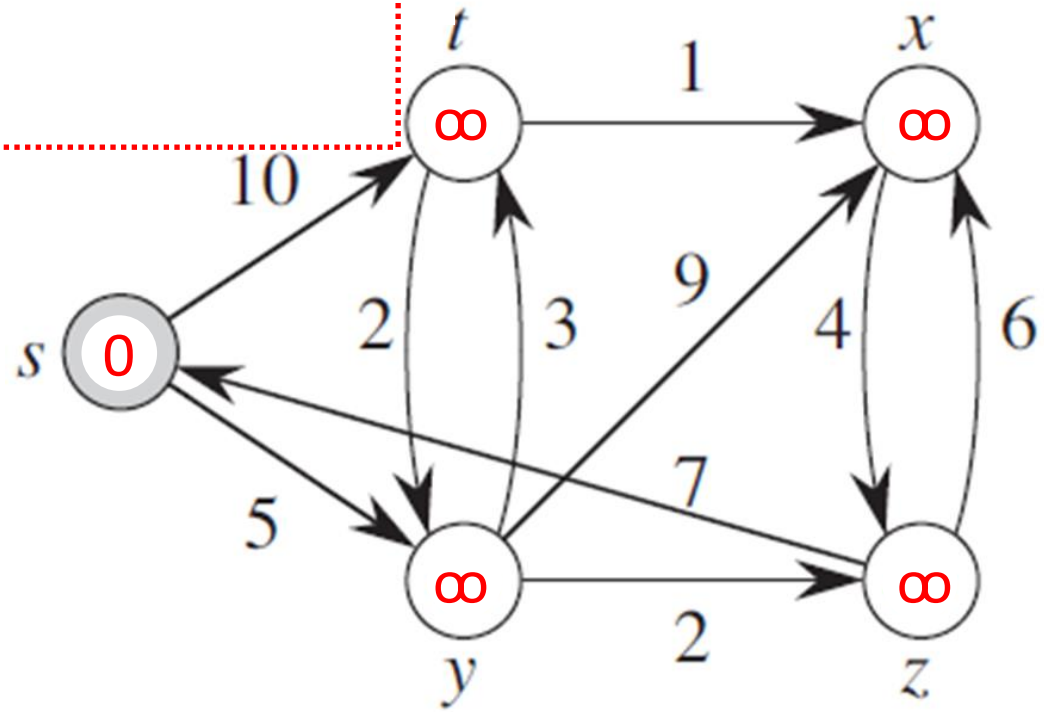
```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
```

```
INITIALIZE-SINGLE-SOURCE(G, s)
1  for each vertex v ∈ G.V
2      v.d = ∞
3      v.π = NIL
4  s.d = 0
```

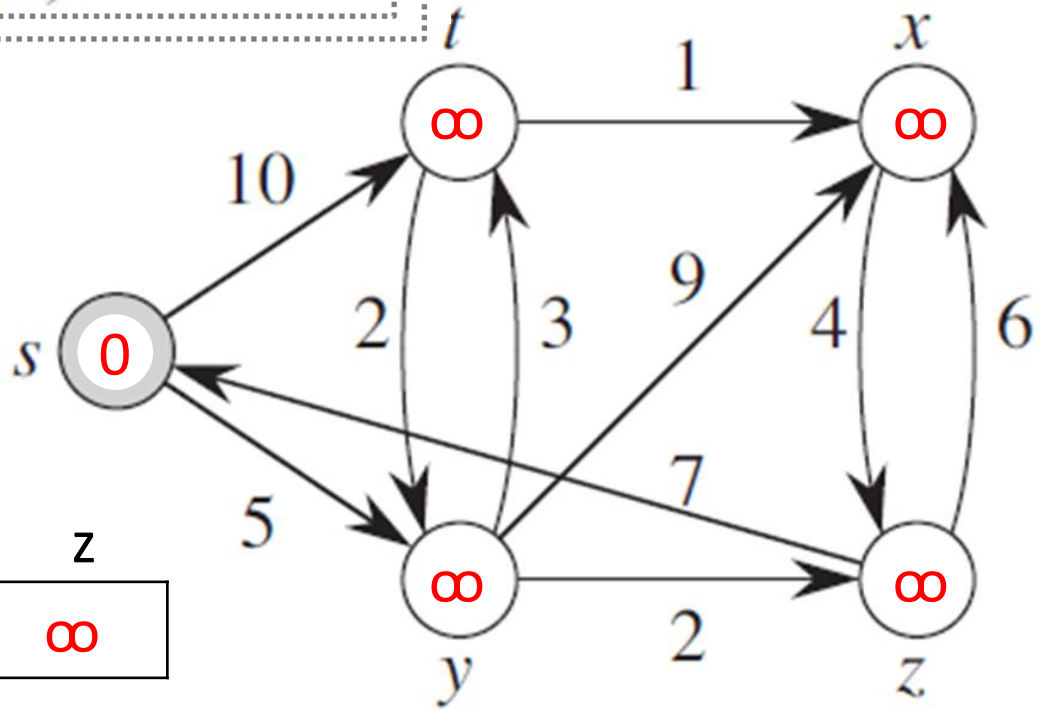


```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
```

```
INITIALIZE-SINGLE-SOURCE(G, s)
1  for each vertex v ∈ G.V
2      v.d = ∞
3      v.π = NIL
4  s.d = 0
```

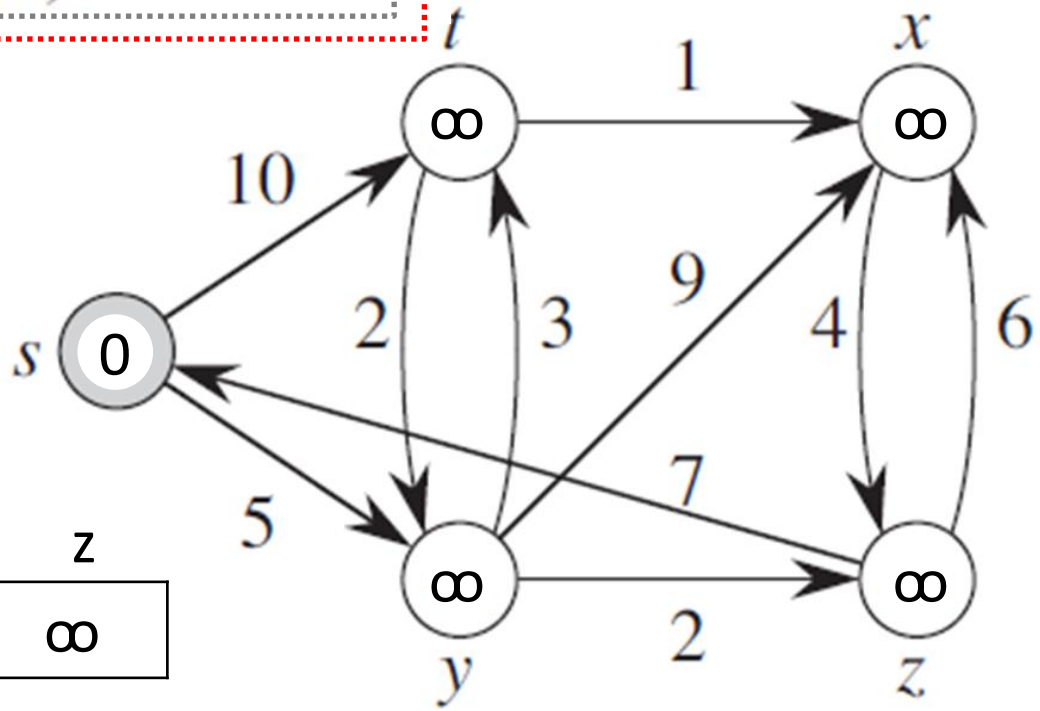


```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```



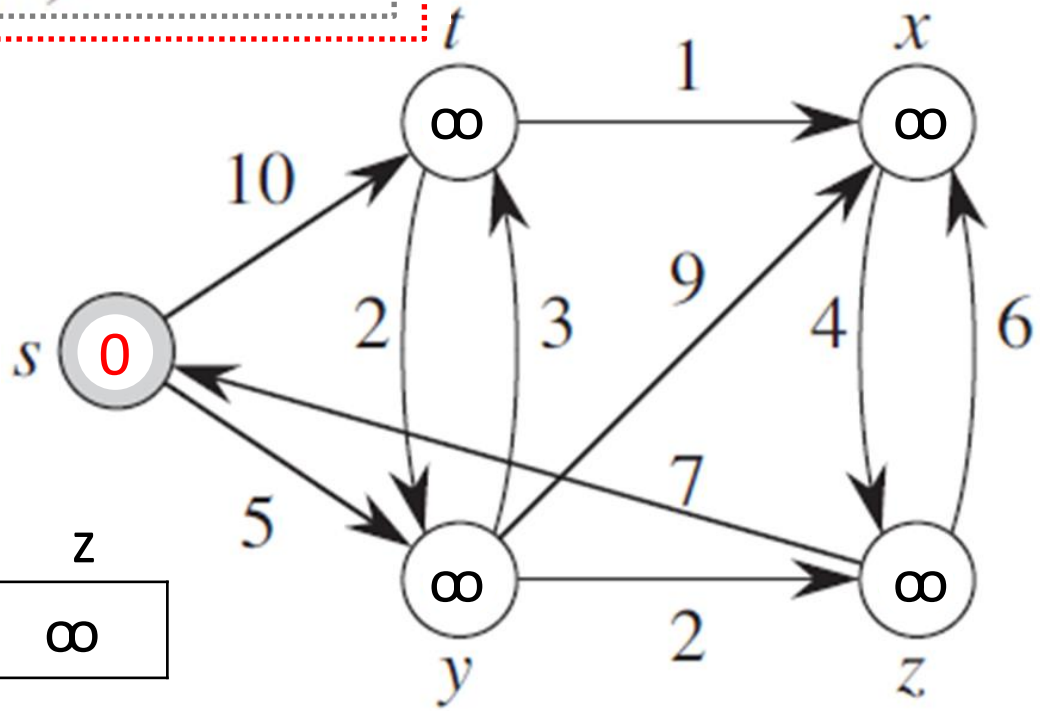
s	t	x	y	z
0	∞	∞	∞	∞

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```



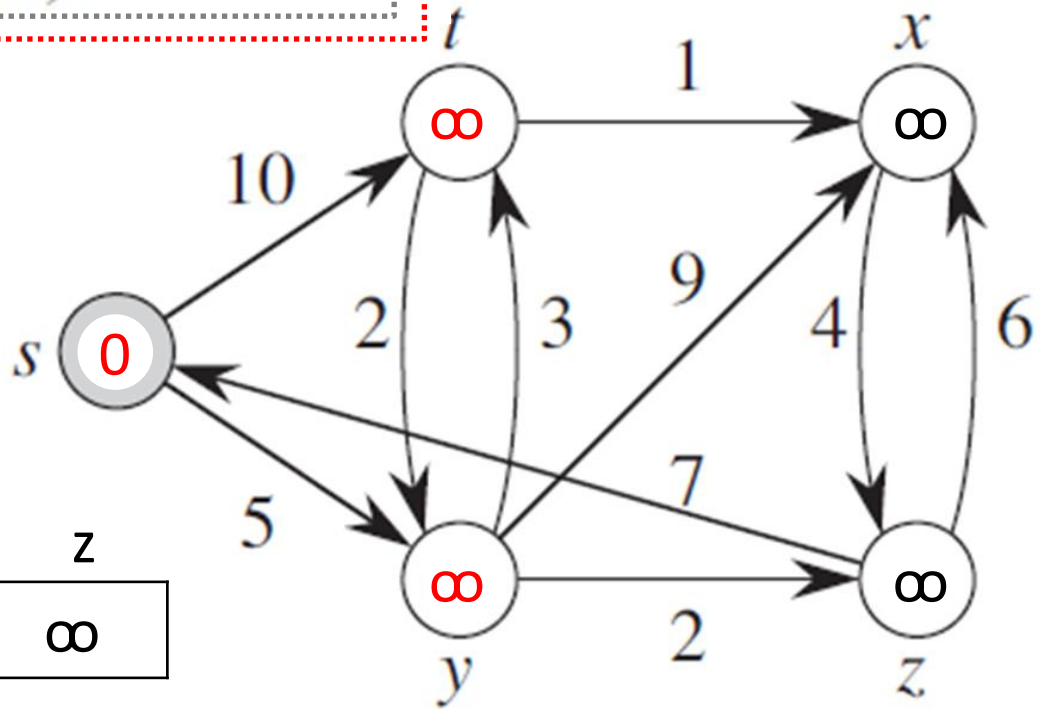
s	t	x	y	z
0	∞	∞	∞	∞

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	∞	∞	∞	∞

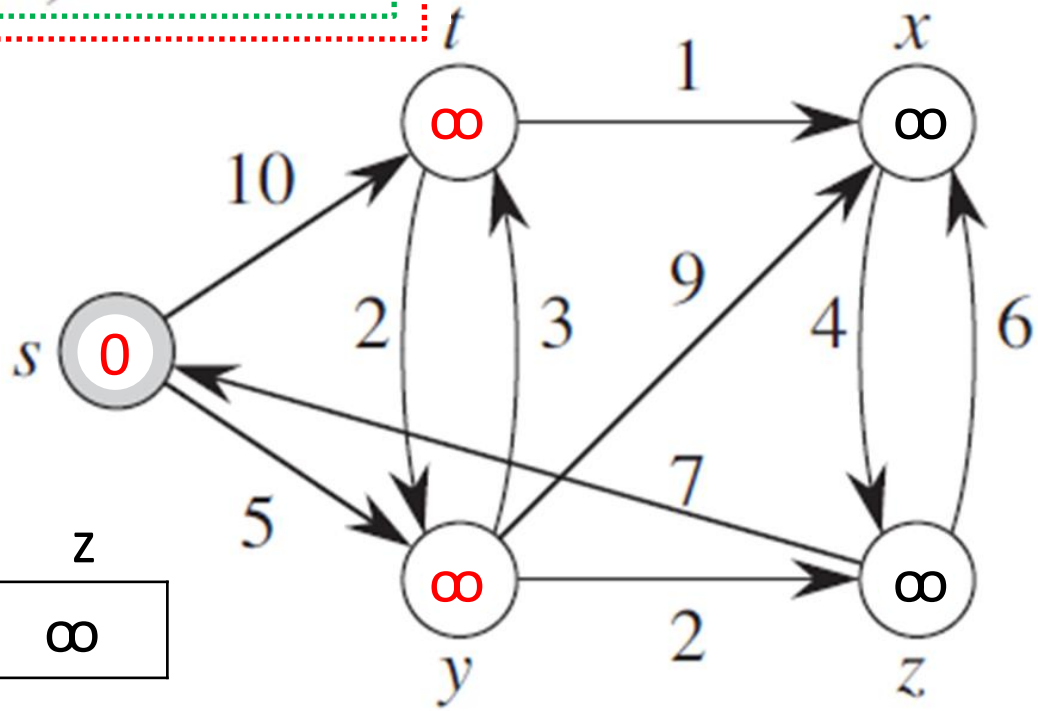
```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	∞	∞	∞	∞

```
RELAX(u, v, w)
1  if u.d + w(u, v) < v.d
2      v.d = u.d + w(u, v)
3      v.π = u
```

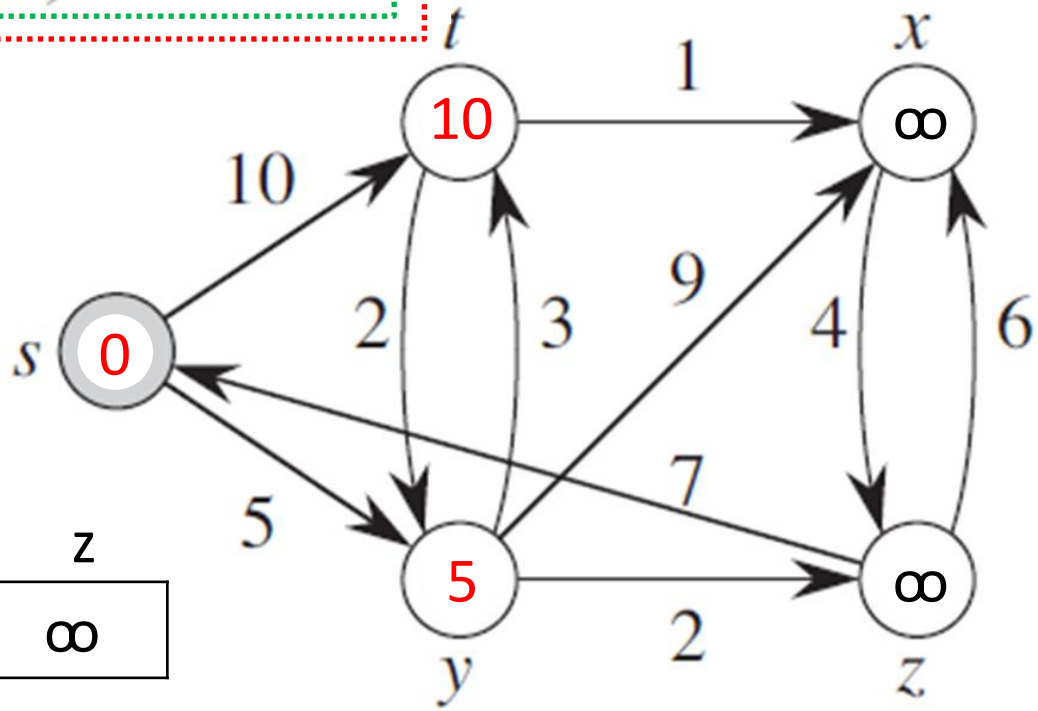
```
5  for each vertex v ∈ G.Adj[u]
6      RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	∞	∞	∞	∞

```
RELAX(u, v, w)
1  if u.d + w(u, v) < v.d
2      v.d = u.d + w(u, v)
3      v.π = u
```

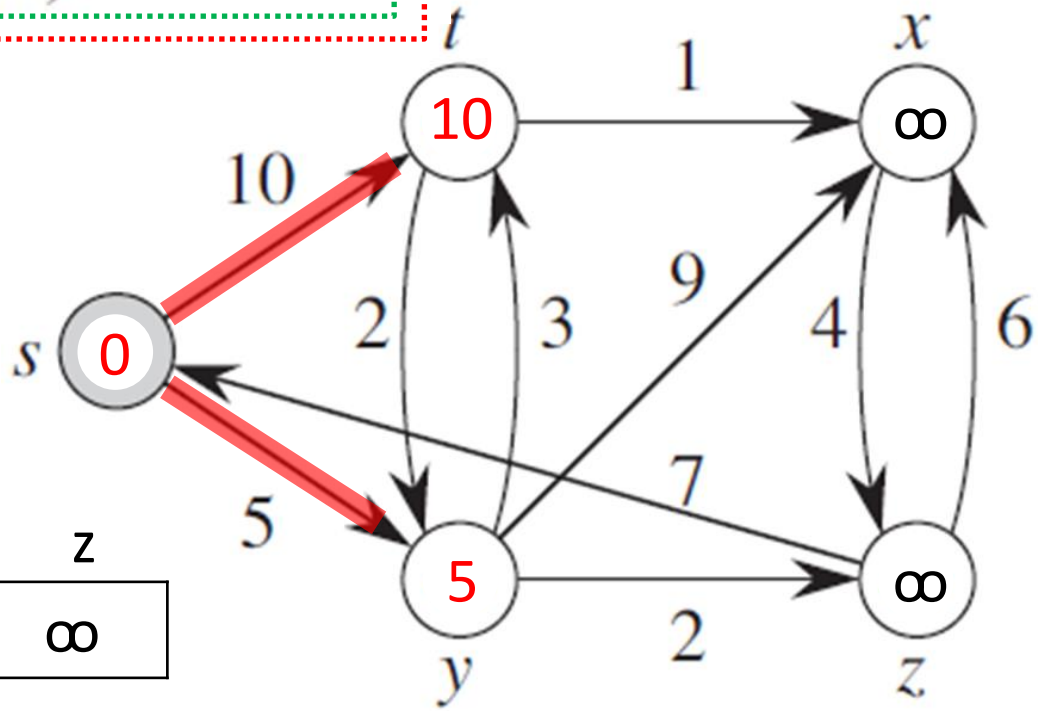
```
5  for each vertex v ∈ G.Adj[u]
6      RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	10	∞	5	∞

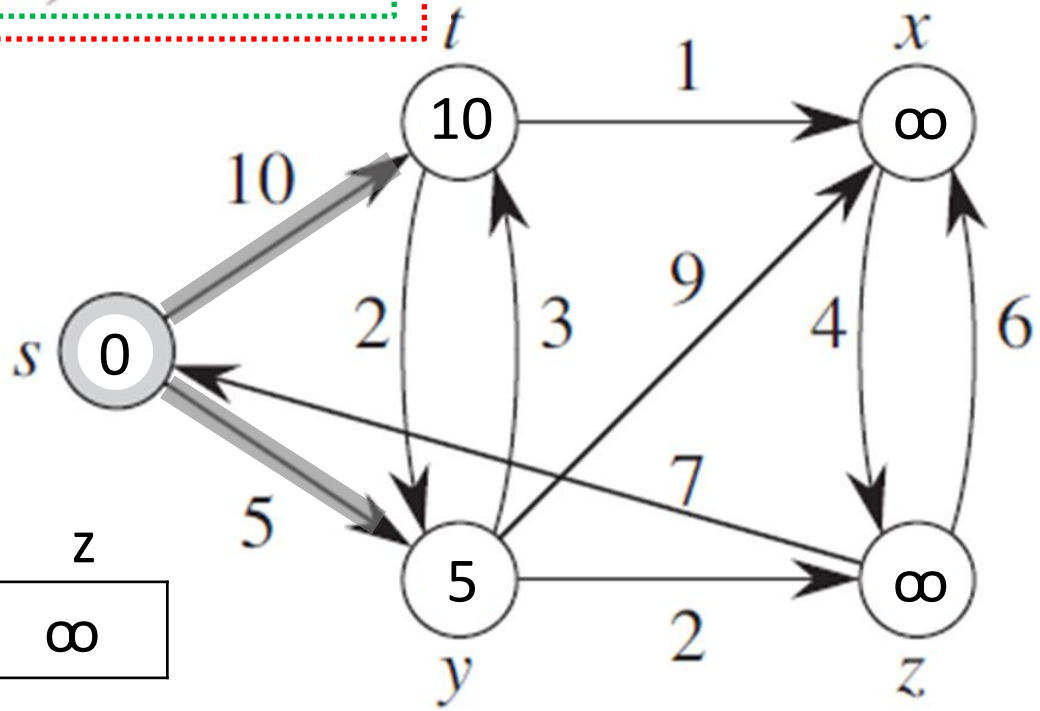
```
RELAX(u, v, w)
1   if u.d + w(u, v) < v.d
2       v.d = u.d + w(u, v)
3       v.π = u
```

```
5   for each vertex v ∈ G.Adj[u]
6       RELAX(u, v, w)
```



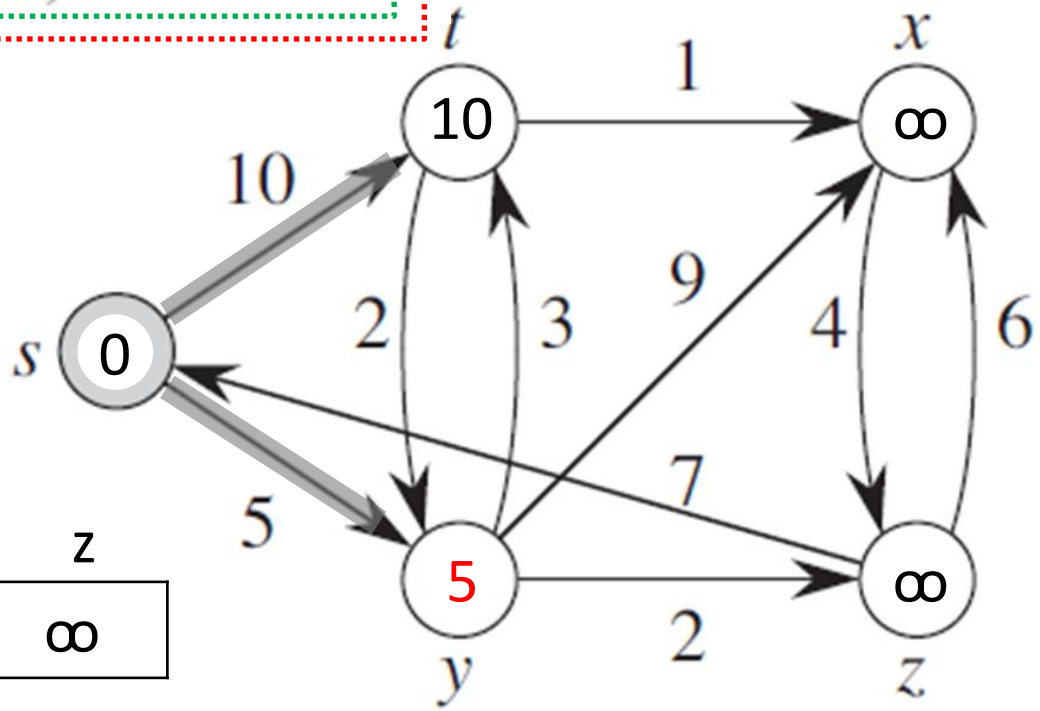
s	t	x	y	z
~~0~~	10	∞	5	∞

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4       u = EXTRACT-MIN(Q)
5       for each vertex v ∈ G.Adj[u]
6            RELAX(u, v, w)
```



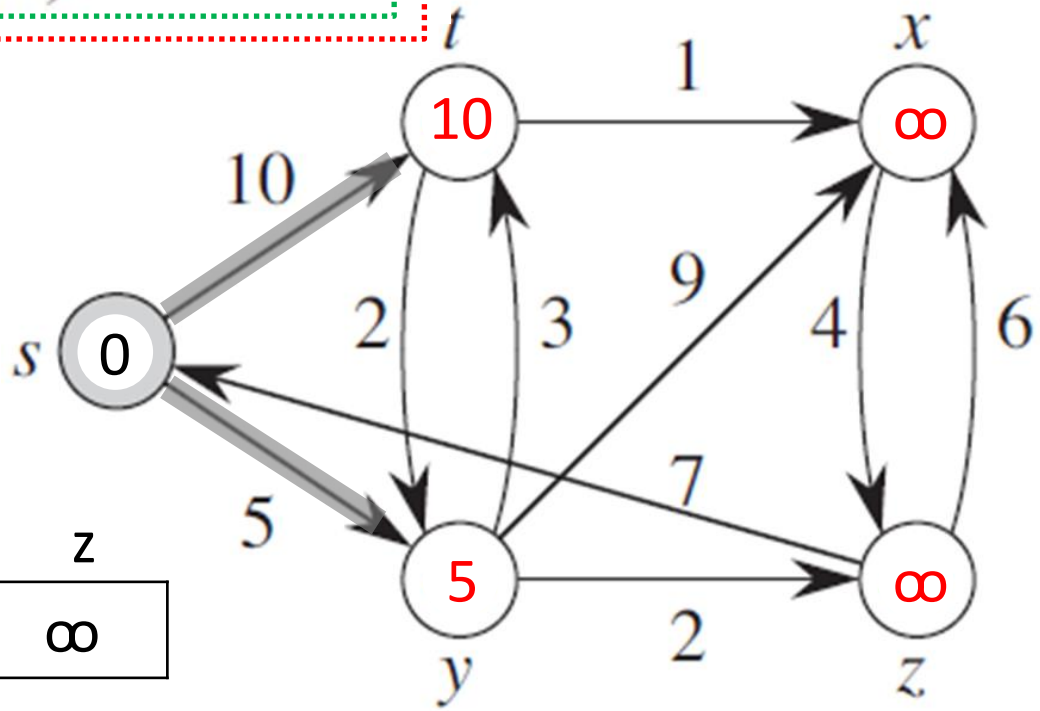
s	t	x	y	z
~~0~~	10	∞	5	∞

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	10	∞	~~5~~	∞

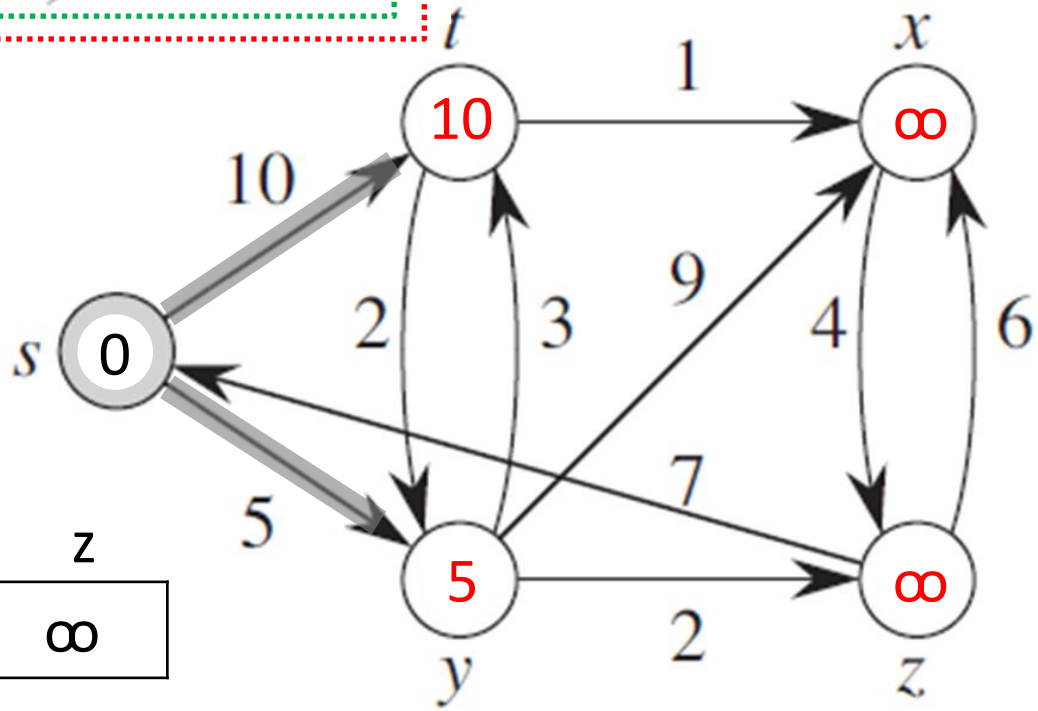
```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	10	∞	~~5~~	∞

```
RELAX(u, v, w)
1  if u.d + w(u, v) < v.d
2      v.d = u.d + w(u, v)
3      v.π = u
```

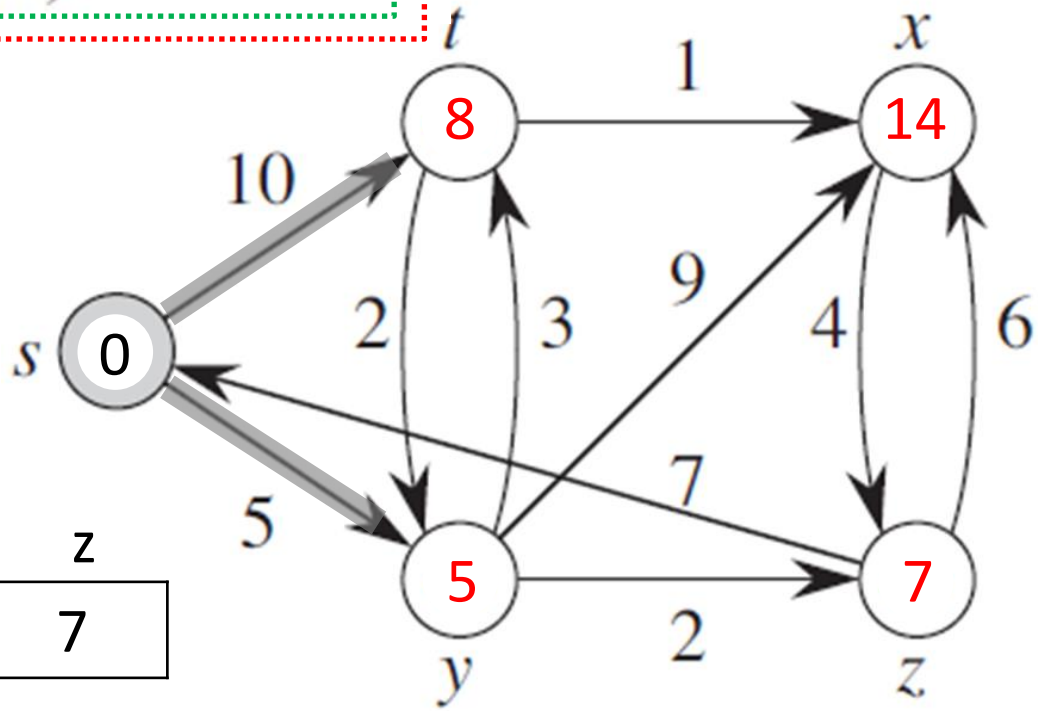
```
5  for each vertex v ∈ G.Adj[u]
6      RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	10	∞	~~5~~	∞

```
RELAX(u, v, w)
1  if u.d + w(u, v) < v.d
2      v.d = u.d + w(u, v)
3      v.π = u
```

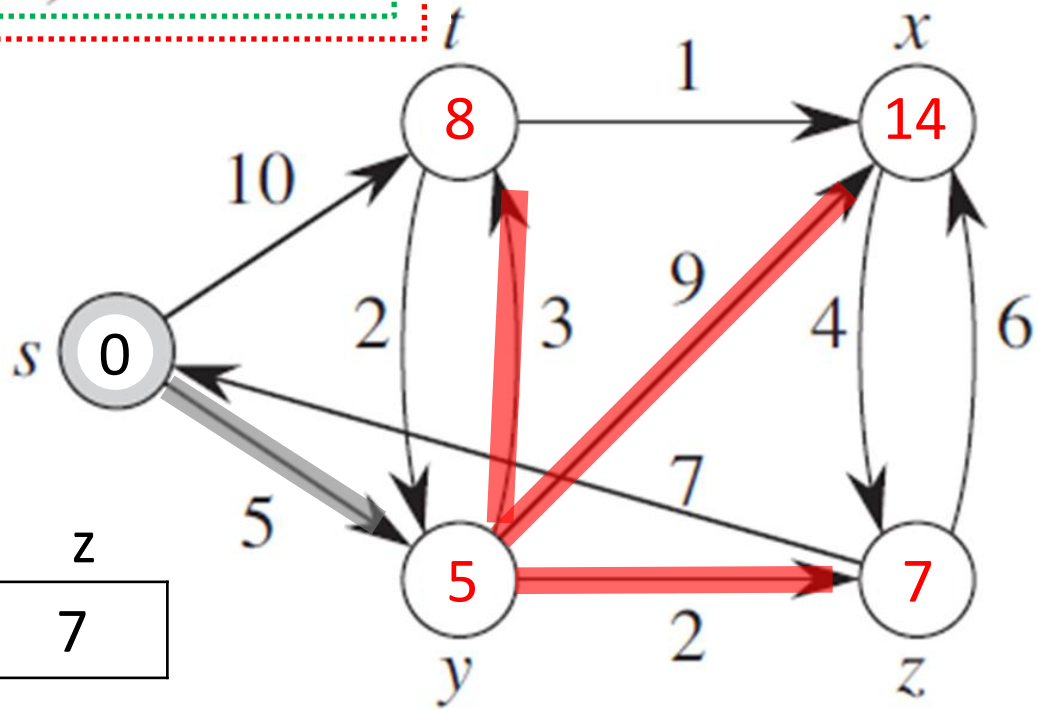
```
5  for each vertex v ∈ G.Adj[u]
6      RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	8	14	~~5~~	7

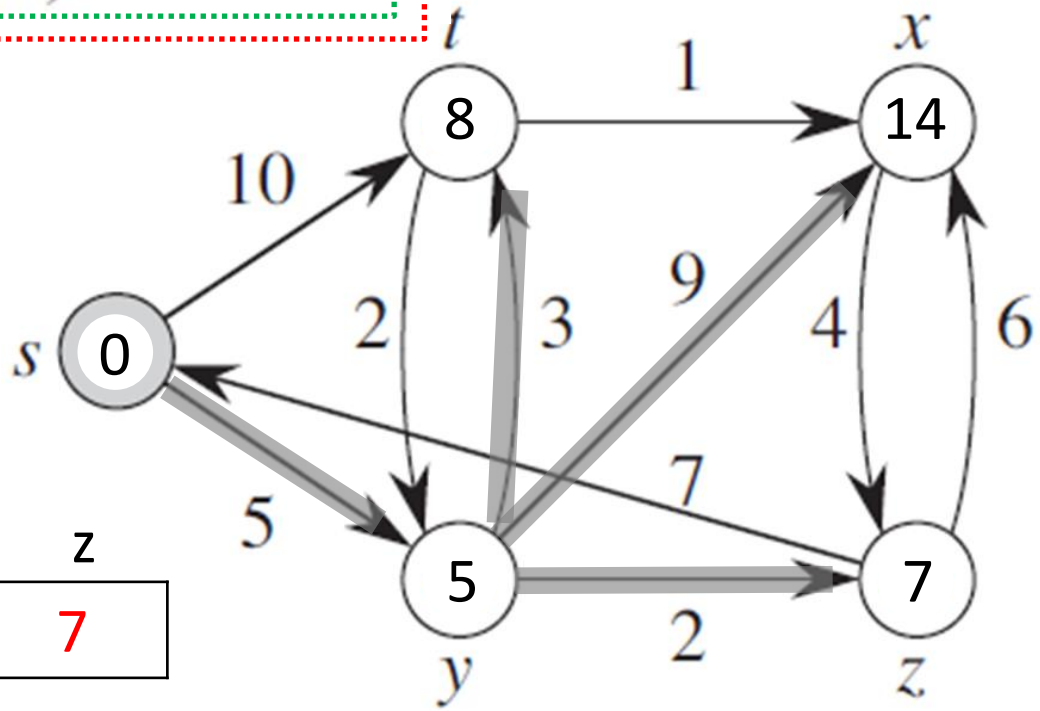
```
RELAX(u, v, w)
1   if u.d + w(u, v) < v.d
2       v.d = u.d + w(u, v)
3       v.π = u
```

```
5   for each vertex v ∈ G.Adj[u]
6       RELAX(u, v, w)
```



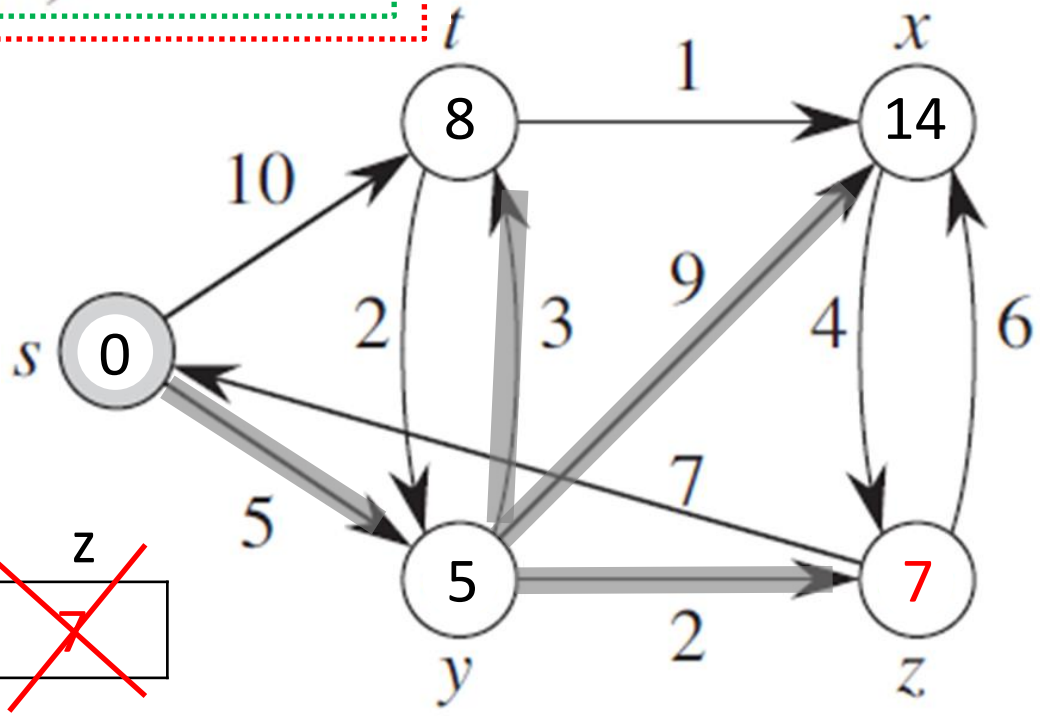
s	t	x	y	z
~~0~~	8	14	~~5~~	7

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4       u = EXTRACT-MIN(Q)
5       for each vertex v ∈ G.Adj[u]
6            RELAX(u, v, w)
```



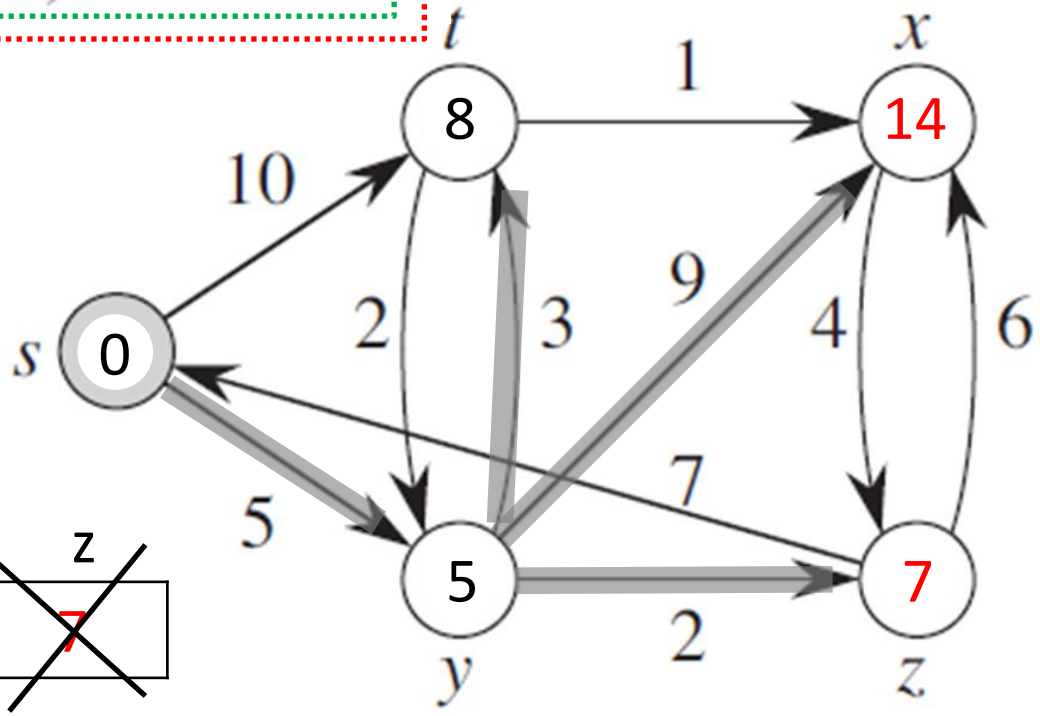
s	t	x	y	z
~~0~~	8	14	~~5~~	7

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	8	14	~~5~~	~~7~~

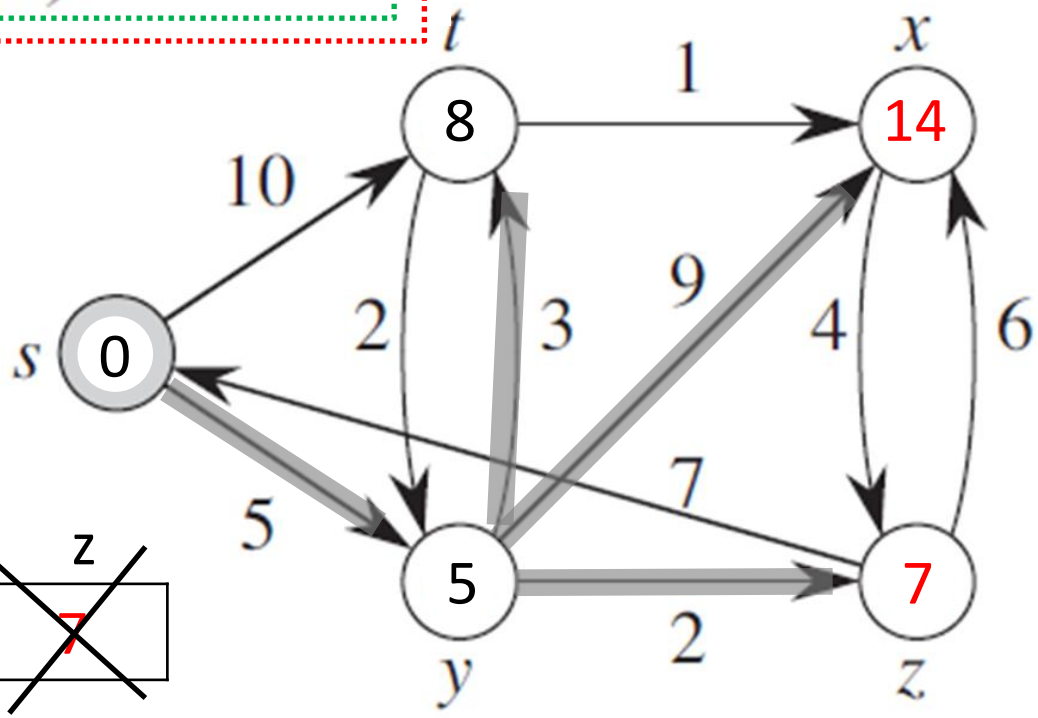
```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4       u = EXTRACT-MIN(Q)
5       for each vertex v ∈ G.Adj[u]
6            RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	8	14	~~5~~	~~7~~

```
RELAX(u, v, w)
1   if u.d + w(u, v) < v.d
2       v.d = u.d + w(u, v)
3       v.π = u
```

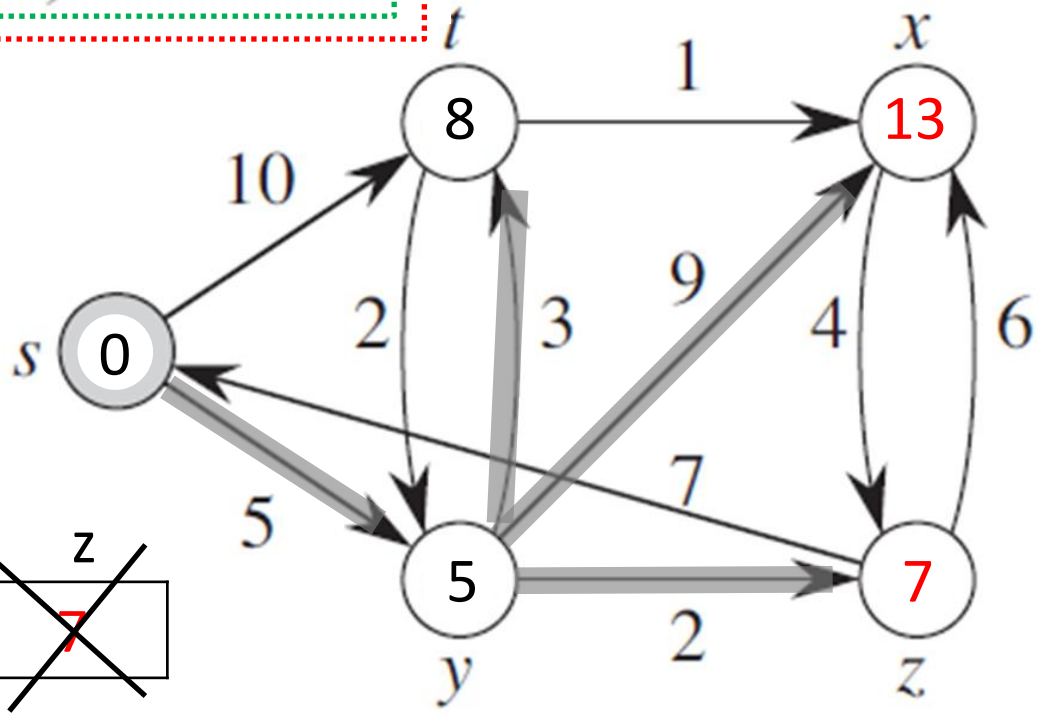
```
5   for each vertex v ∈ G.Adj[u]
6       RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	8	14	~~5~~	~~7~~

```
RELAX(u, v, w)
1  if u.d + w(u, v) < v.d
2      v.d = u.d + w(u, v)
3      v.π = u
```

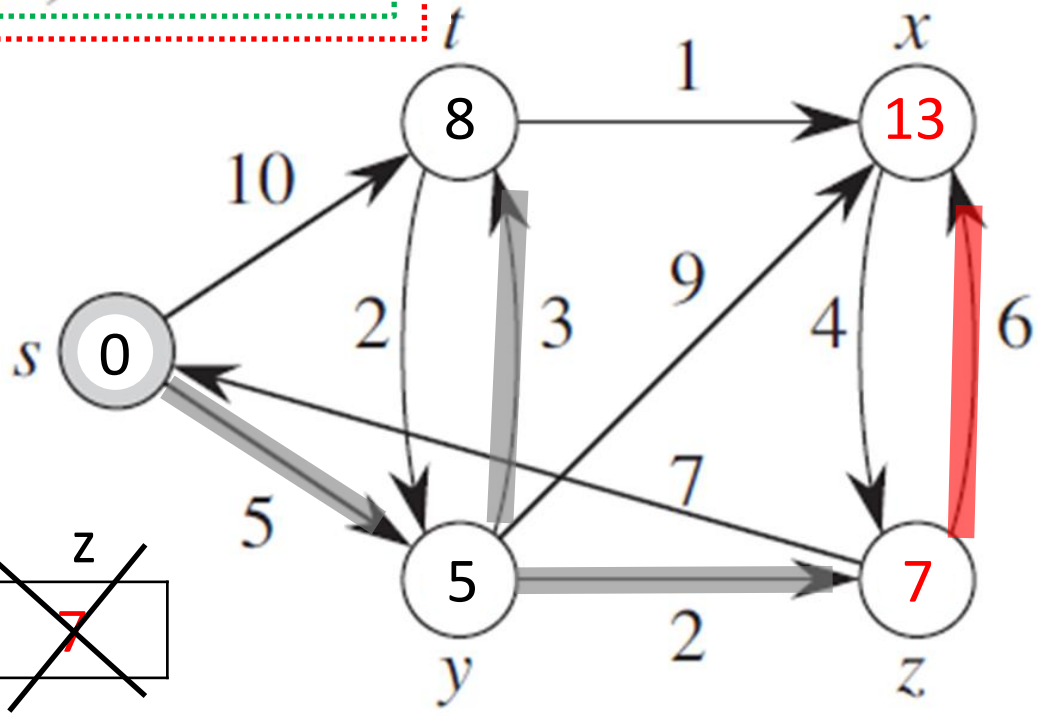
```
5  for each vertex v ∈ G.Adj[u]
6      RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	8	13	~~5~~	~~7~~

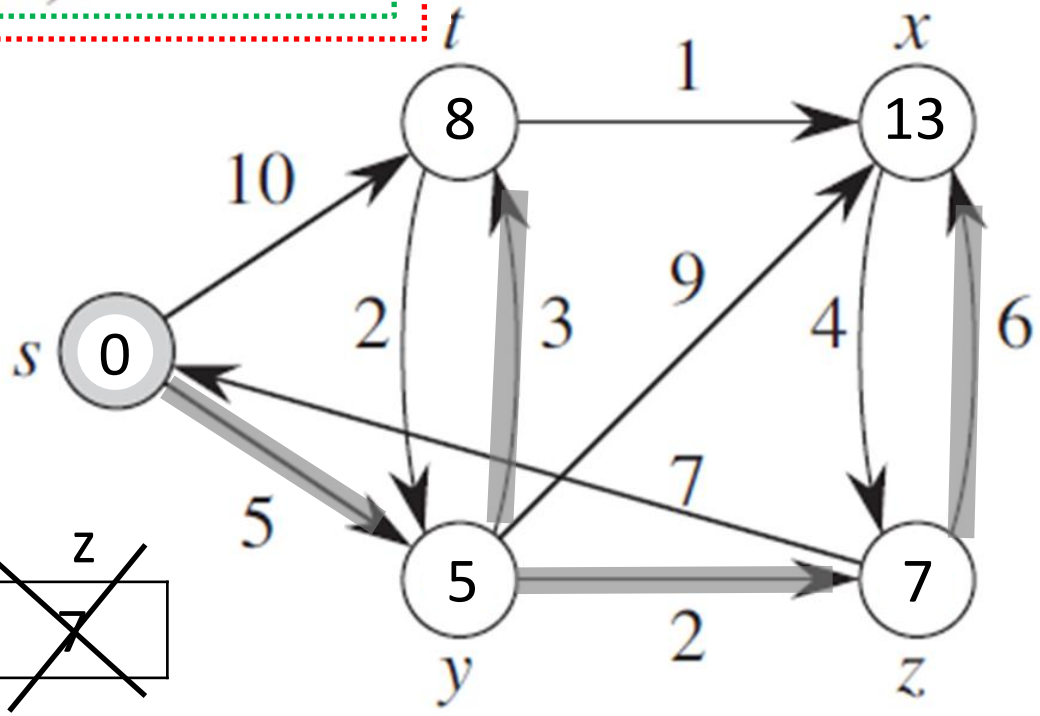
```
RELAX(u, v, w)
1   if u.d + w(u, v) < v.d
2       v.d = u.d + w(u, v)
3       v.π = u
```

```
5   for each vertex v ∈ G.Adj[u]
6       RELAX(u, v, w)
```



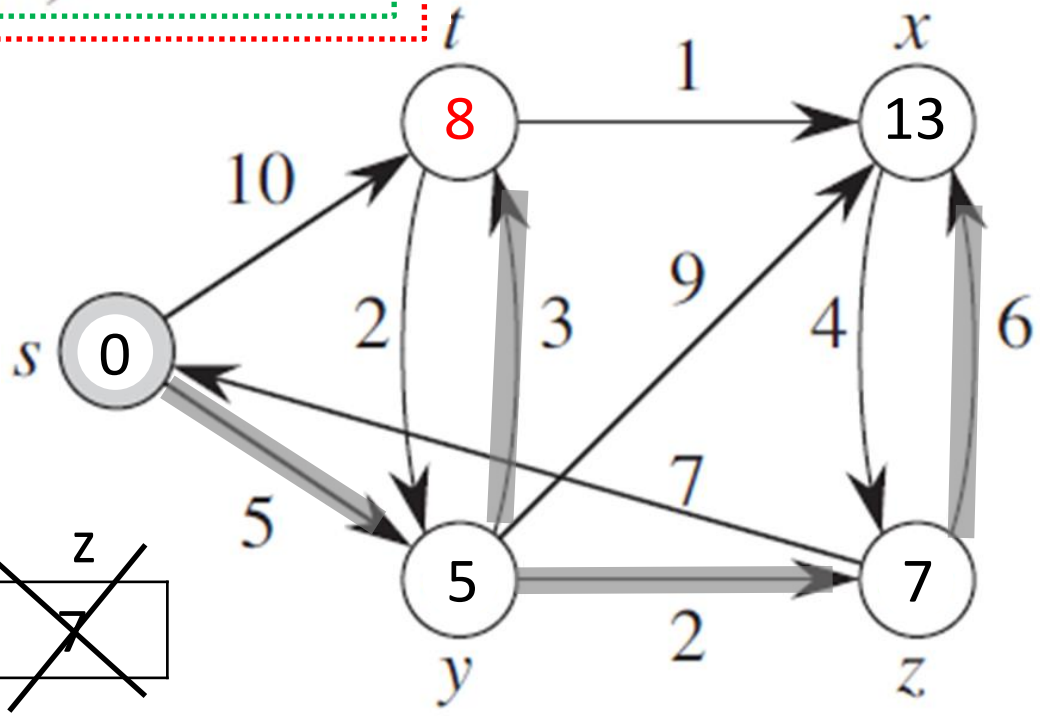
s	t	x	y	z
~~0~~	8	13	~~5~~	~~7~~

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4       u = EXTRACT-MIN(Q)
5       for each vertex v ∈ G.Adj[u]
6            RELAX(u, v, w)
```



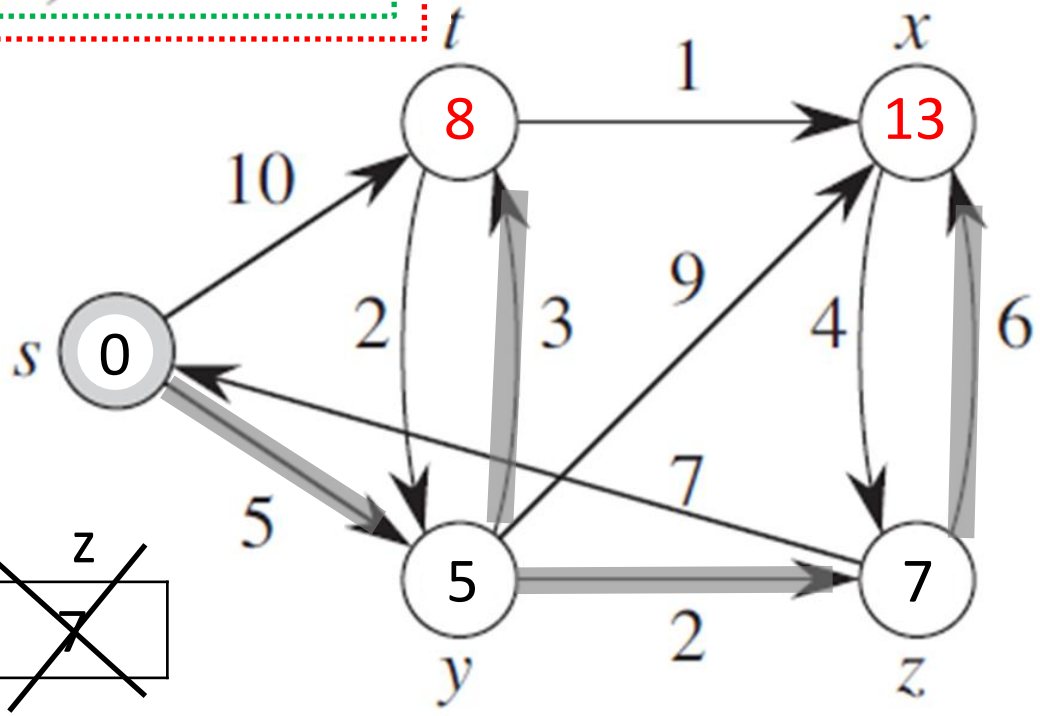
s	t	x	y	z
~~0~~	8	13	~~5~~	~~7~~

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	~~8~~	13	~~5~~	~~7~~

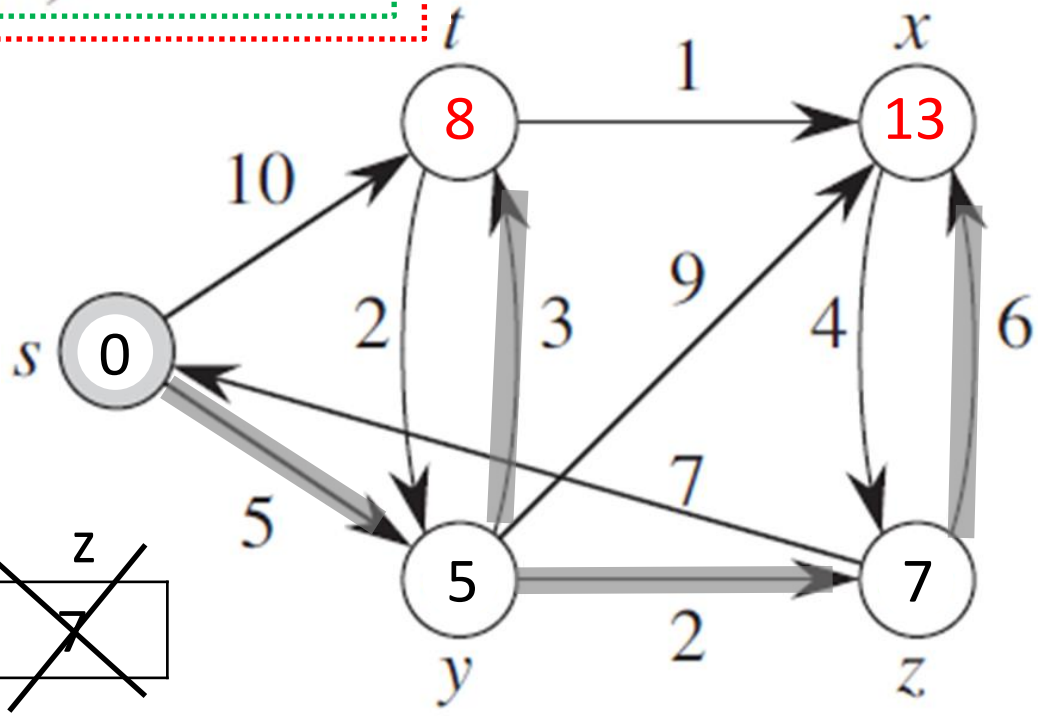
```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4       u = EXTRACT-MIN(Q)
5       for each vertex v ∈ G.Adj[u]
6            RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	~~8~~	13	~~5~~	~~7~~

```
RELAX(u, v, w)
1   if u.d + w(u, v) < v.d
2       v.d = u.d + w(u, v)
3       v.π = u
```

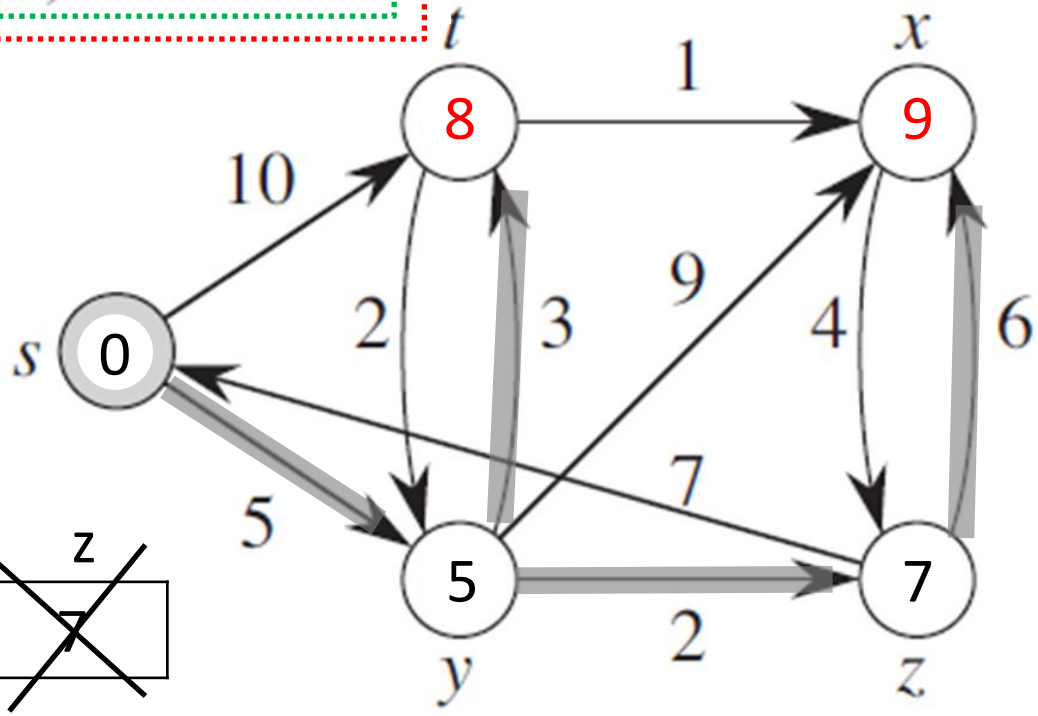
```
5   for each vertex v ∈ G.Adj[u]
6       RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	~~8~~	13	~~5~~	~~7~~

```
RELAX(u, v, w)
1   if u.d + w(u, v) < v.d
2       v.d = u.d + w(u, v)
3       v.π = u
```

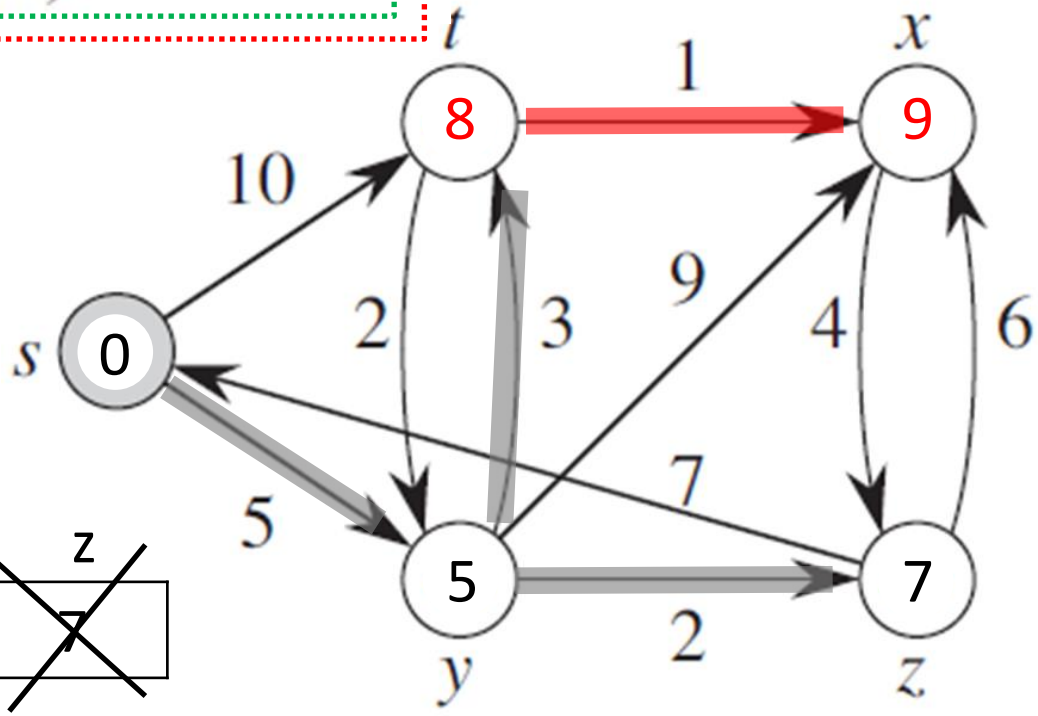
```
5   for each vertex v ∈ G.Adj[u]
6       RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	~~8~~	9	~~5~~	~~7~~

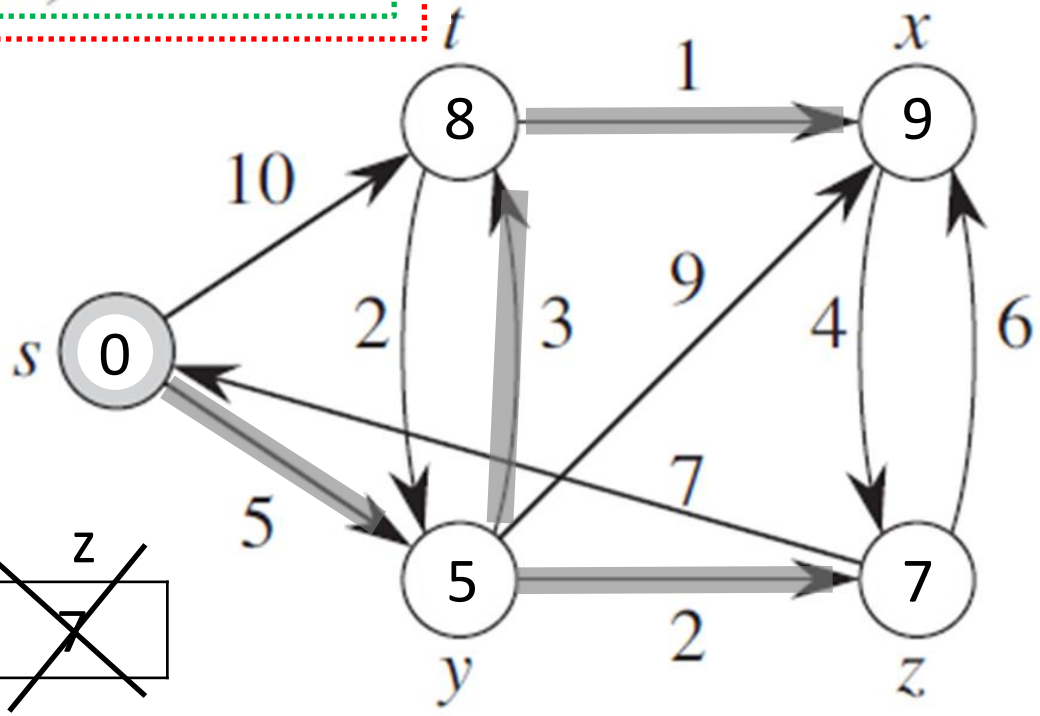
```
RELAX(u, v, w)
1   if u.d + w(u, v) < v.d
2       v.d = u.d + w(u, v)
3       v.π = u
```

```
5   for each vertex v ∈ G.Adj[u]
6       RELAX(u, v, w)
```



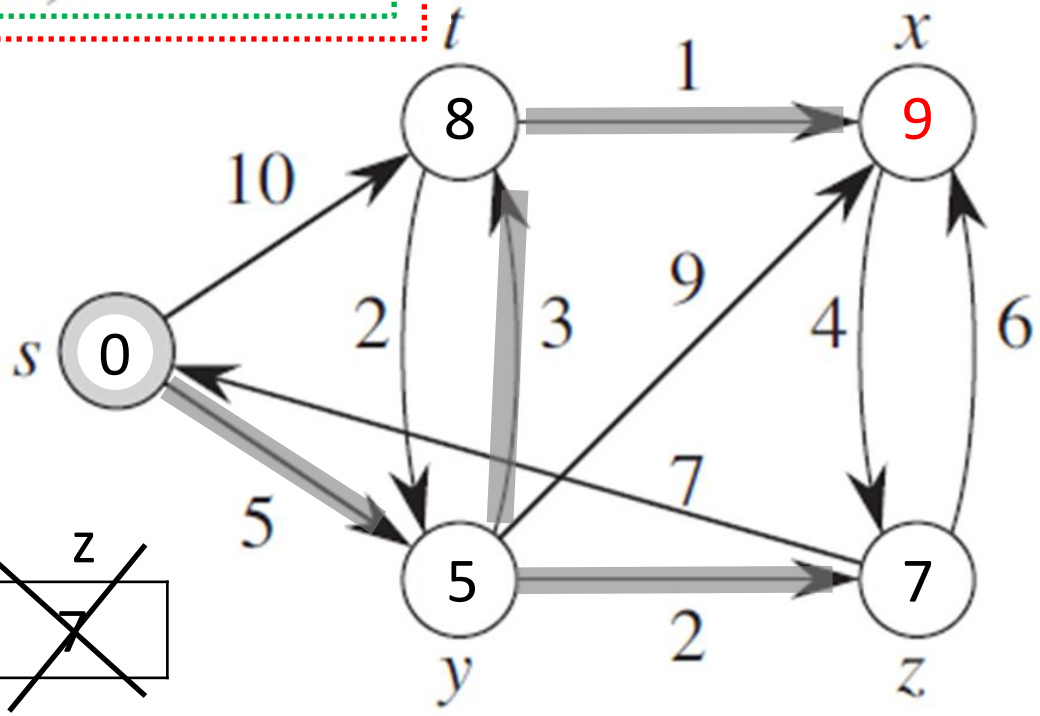
s	t	x	y	z
~~0~~	~~8~~	9	~~5~~	~~7~~

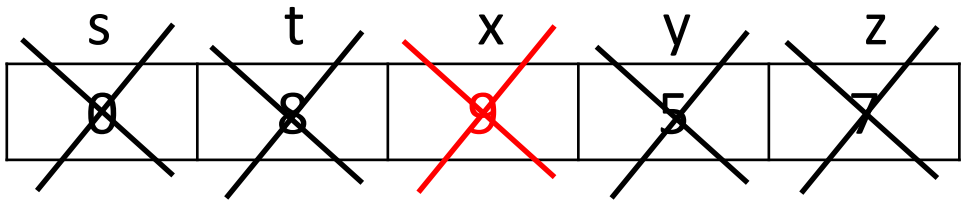
```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4        u = EXTRACT-MIN(Q)
5        for each vertex v ∈ G.Adj[u]
6              RELAX(u, v, w)
```



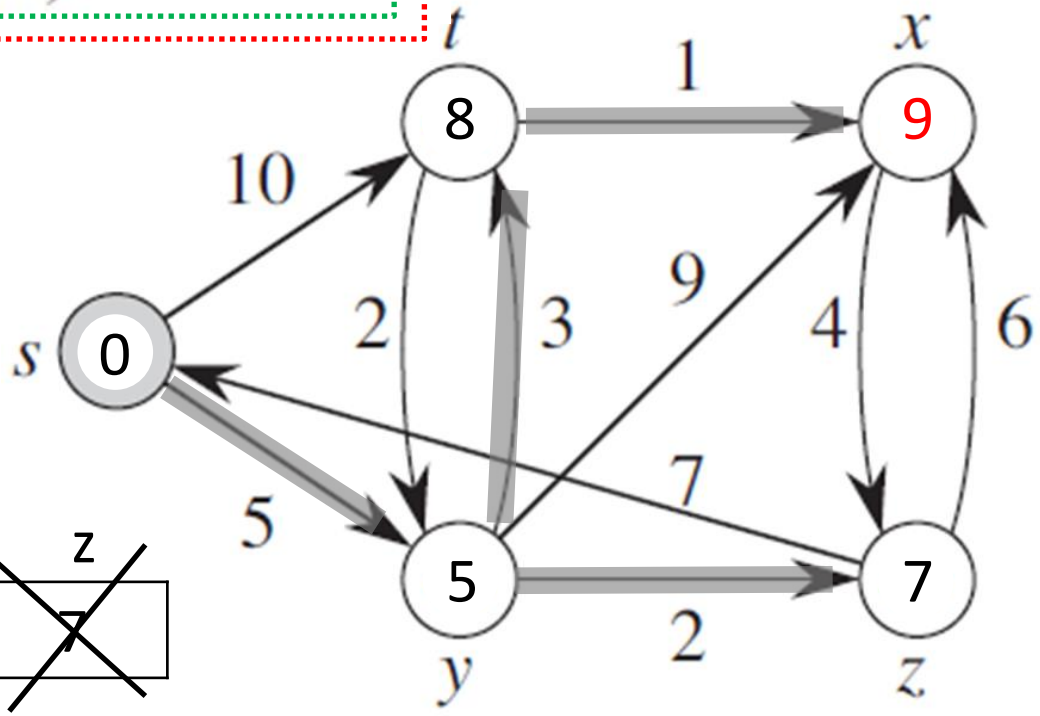
s	t	x	y	z
~~0~~	~~8~~	9	~~5~~	~~7~~

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```



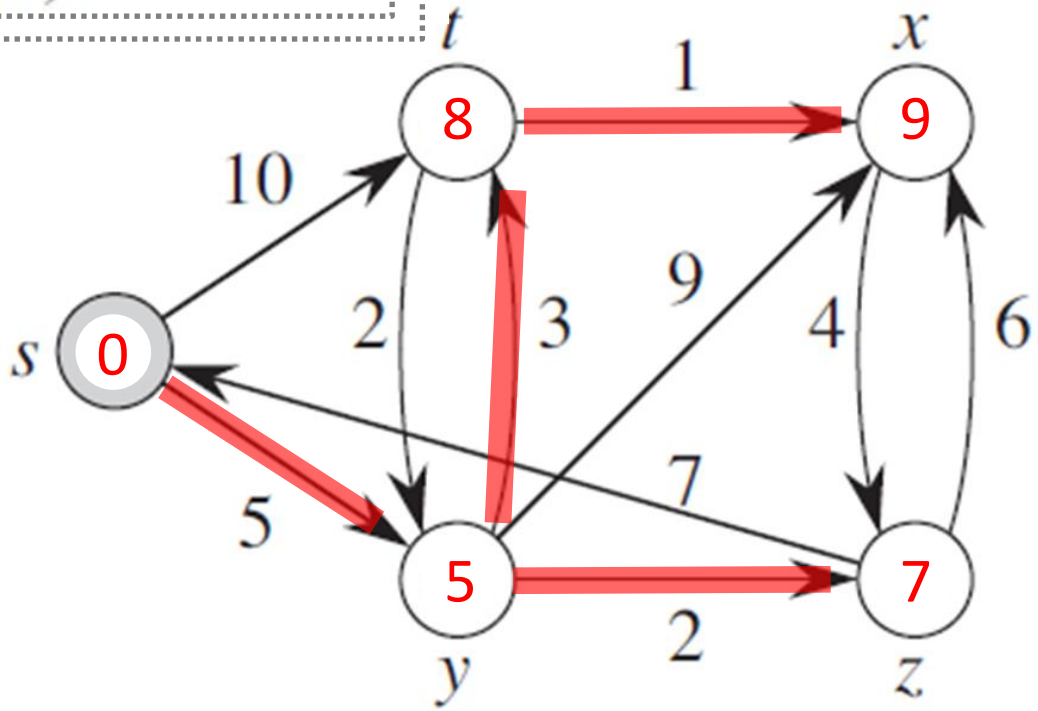


```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4       u = EXTRACT-MIN(Q)
5       for each vertex v ∈ G.Adj[u]
6            RELAX(u, v, w)
```



s	t	x	y	z
~~0~~	~~8~~	~~9~~	~~5~~	~~7~~

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```



Dijkstra

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```

```
INITIALIZE-SINGLE-SOURCE(G, s)
1  for each vertex v ∈ G.V
2      v.d = ∞
3      v.π = NIL
4  s.d = 0
```

Dijkstra

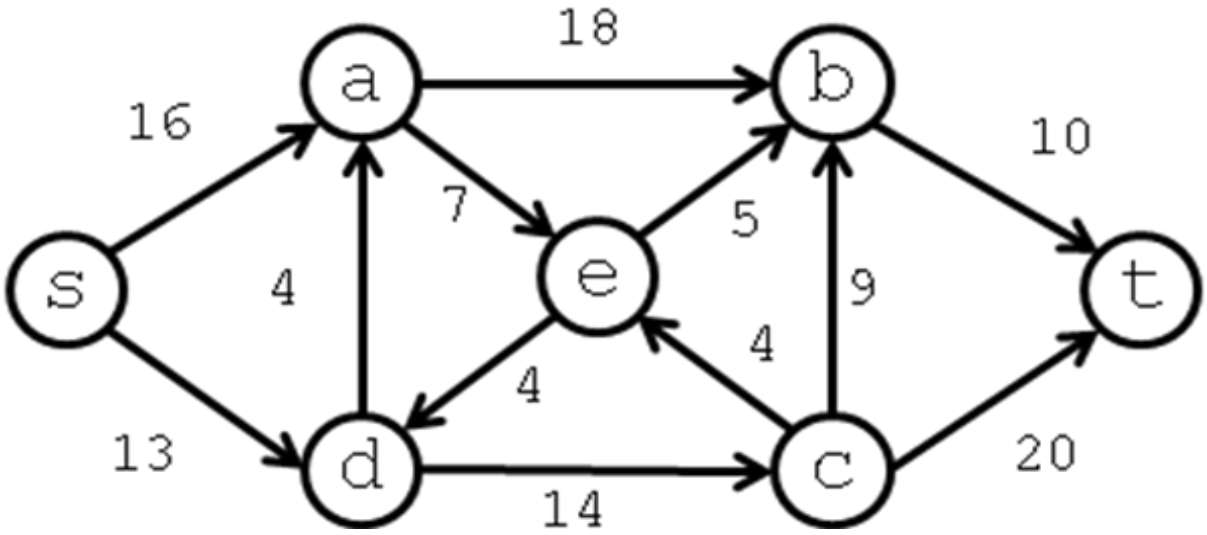
```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4      u = EXTRACT-MIN(Q)
5      for each vertex v ∈ G.Adj[u]
6          RELAX(u, v, w)
```

```
RELAX(u, v, w)
1  if u.d + w(u, v) < v.d
2      v.d = u.d + w(u, v)
3      v.π = u
```

Exercícios

- Simule o algoritmo de Dijkstra:

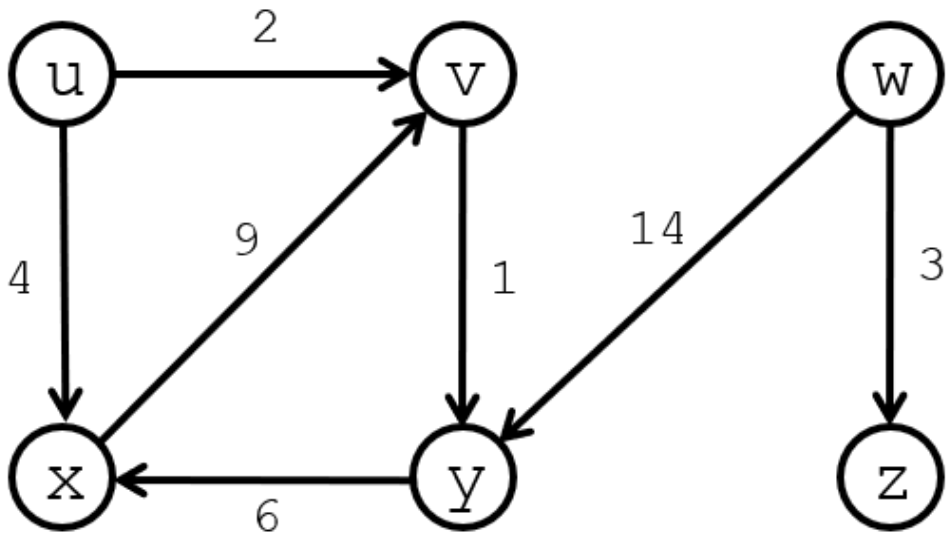
(a)



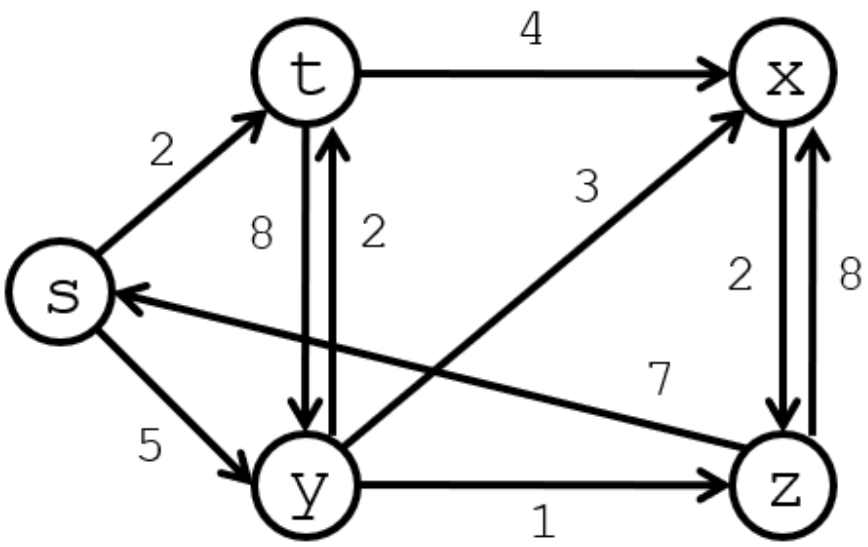
Exercícios

- Simule o algoritmo de Dijkstra:

(b)



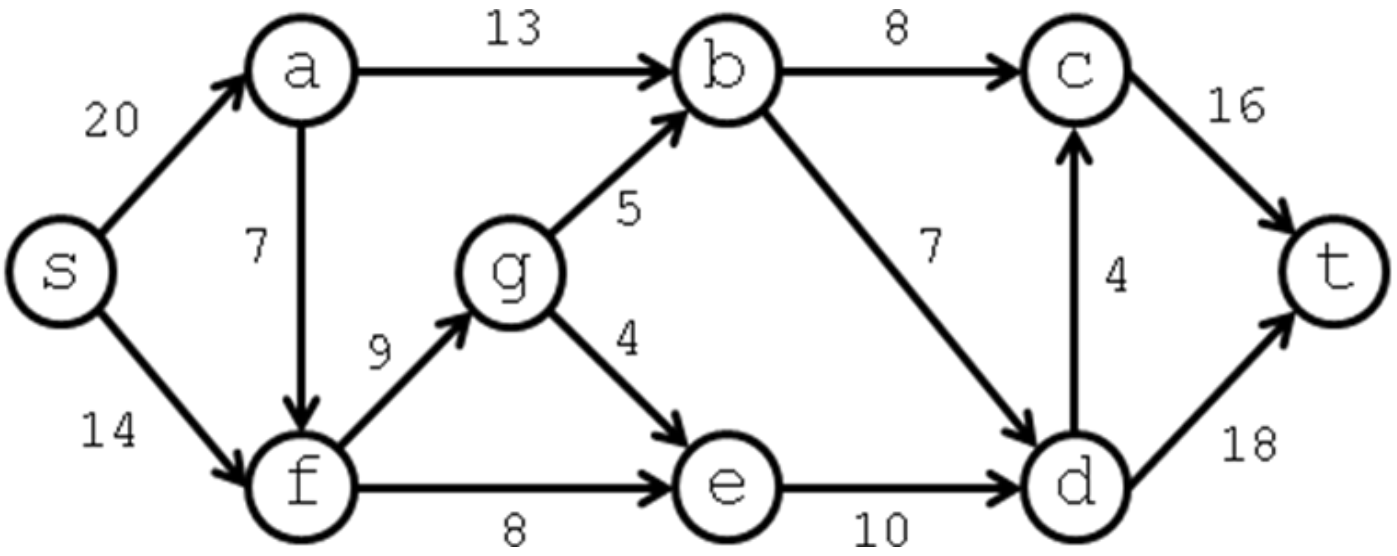
(c)



Exercícios

- Simule o algoritmo de Dijkstra:

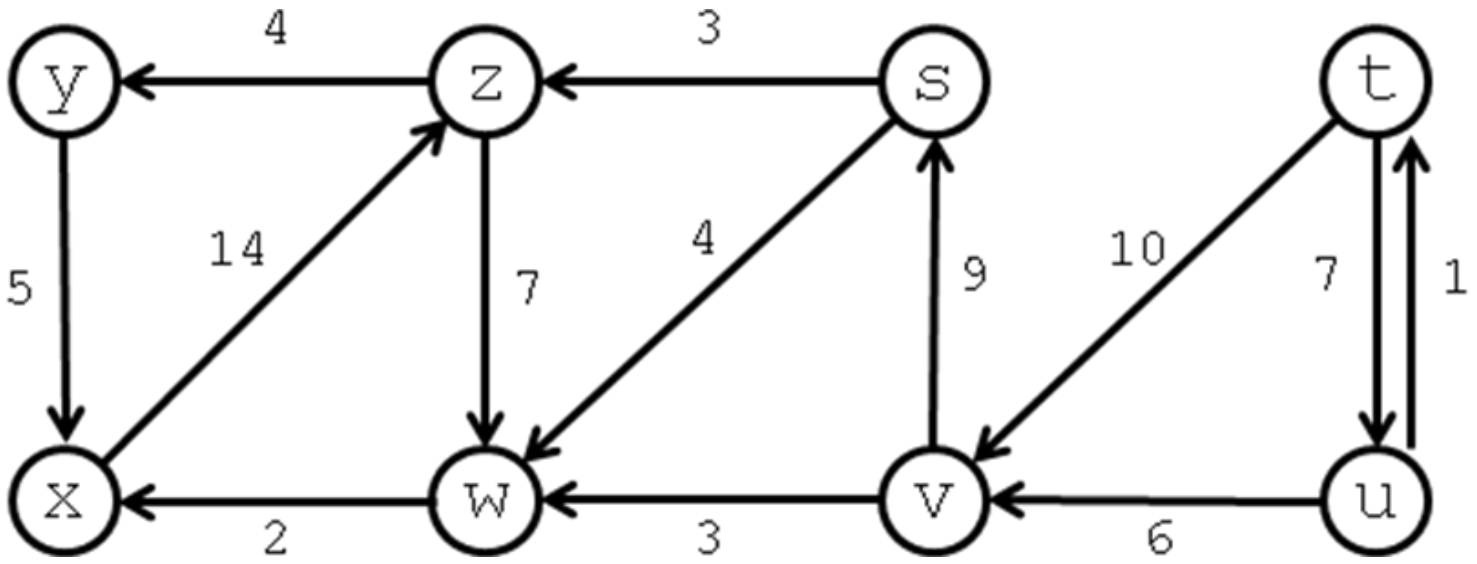
(d)



Exercícios

- Simule o algoritmo de Dijkstra:

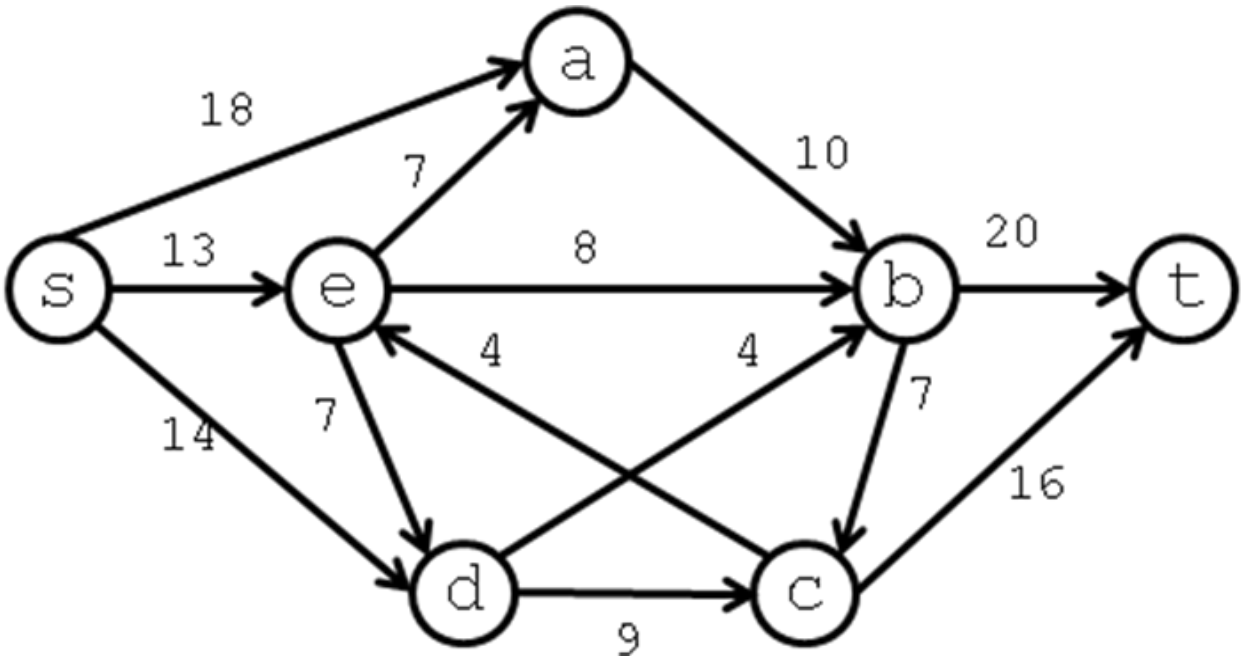
(e)



Exercícios

- Simule o algoritmo de Dijkstra:

(f)



Consumo de tempo?

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  Q = G.V
3  while Q ≠ ∅
4       u = EXTRACT-MIN(Q)
5       for each vertex v ∈ G.Adj[u]
6            RELAX(u, v, w)
```

```
INITIALIZE-SINGLE-SOURCE(G, s)
1  for each vertex v ∈ G.V
2       v.d = ∞
3       v.π = NIL
4  s.d = 0
```

```
RELAX(u, v, w)
1  if u.d + w(u, v) < v.d
2       v.d = u.d + w(u, v)
3       v.π = u
```

Exercício Programa

- 10-dijkstra.py