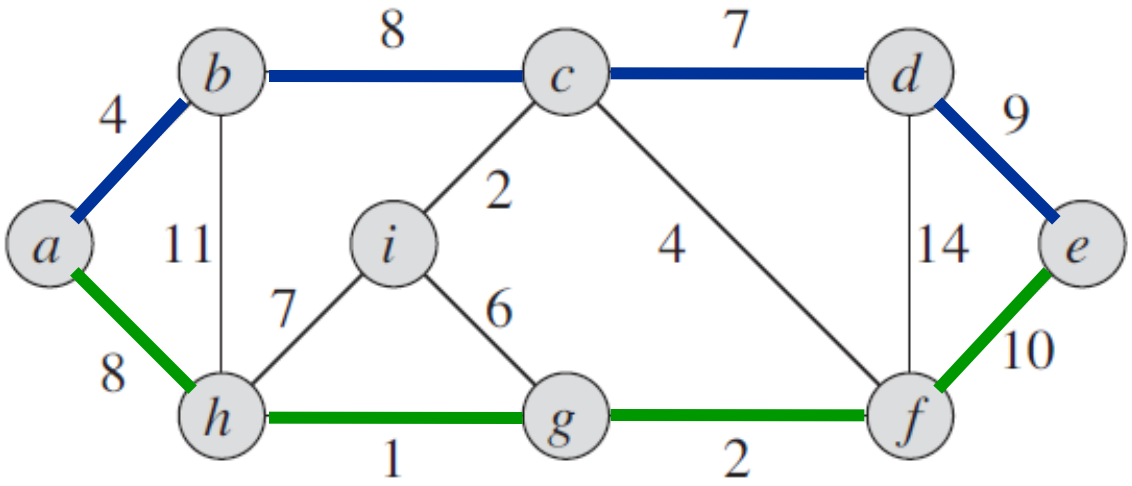
Teoria dos Grafos

Universidade Federal do ABC

- Dado um grafo ponderado,
  - o comprimento de um caminho é a soma dos pesos das arestas no caminho.

$$w(P) = 4 + 8 + 7 + 9 = 28$$

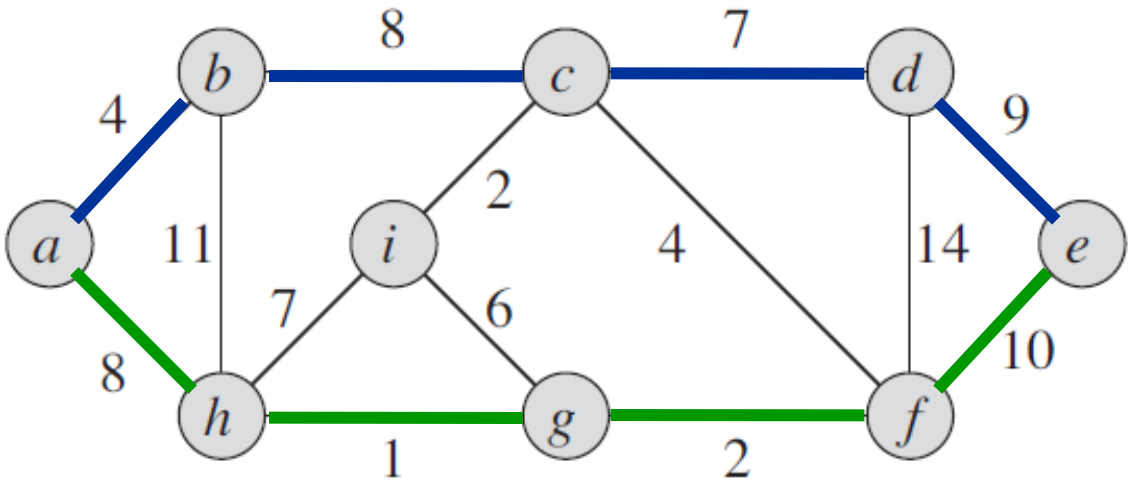
$$w(P') = 8 + 1 + 2 + 10 = 21$$



- Um caminho entre **s** e **t** é mínimo se não existe outro caminho (entre as mesmas extremidades) de comprimento menor.

$$w(P) = 4 + 8 + 7 + 9 = 28$$

$$w(P') = 8 + 1 + 2 + 10 = 21$$



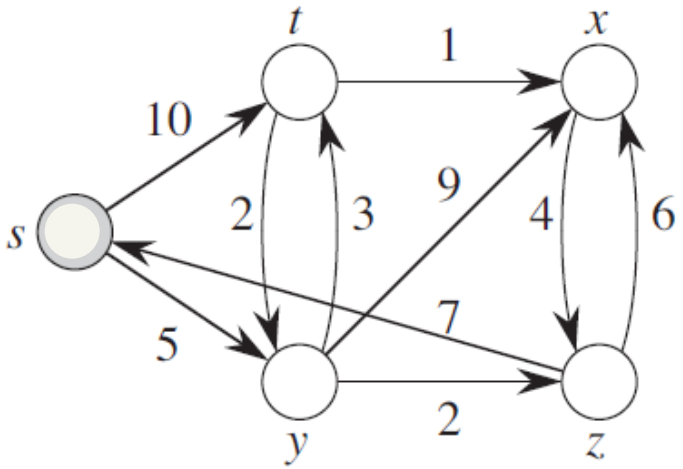
- A distância entre **s** e **t** é dada pelo comprimento de um caminho mínimo entre as mesmas extremidades e é denotada por:

$$\delta(s,t) = \begin{cases} \text{compr. cam. mín. entre } s \text{ e } t & \text{se existir caminho} \\ \infty & \text{caso contrário} \end{cases}$$

- 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")

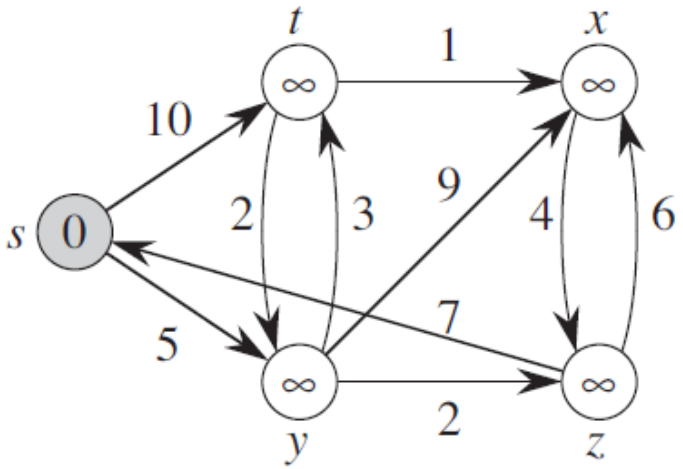
Fila de prioridade: operações

- Insert(Q, x)
  - insere elemento x no conjunto Q
- Minimum(Q)
  - devolve o elemento de Q com a menor chave
- ExtractMin (Q)
  - remove e devolve o elemento de Q com a menor chave
- DecreaseKey(Q, x, k)
  - diminui o valor da chave de x para o novo valor k.



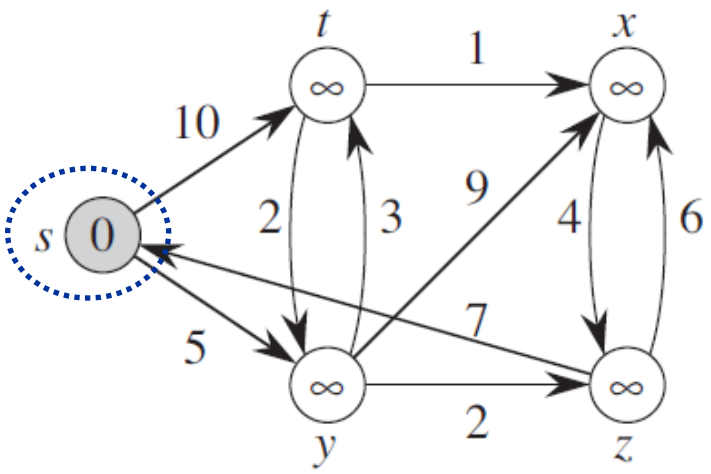

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

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



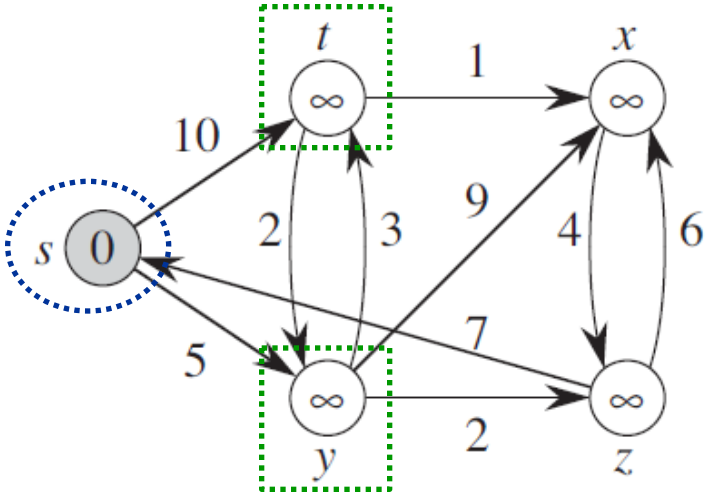

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```



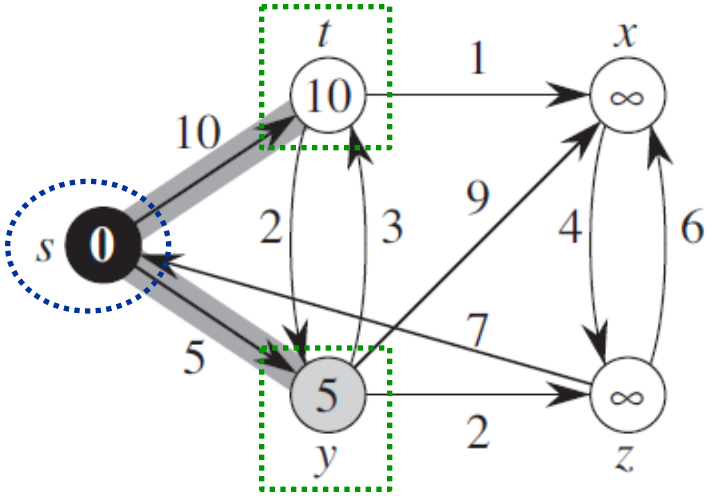

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```



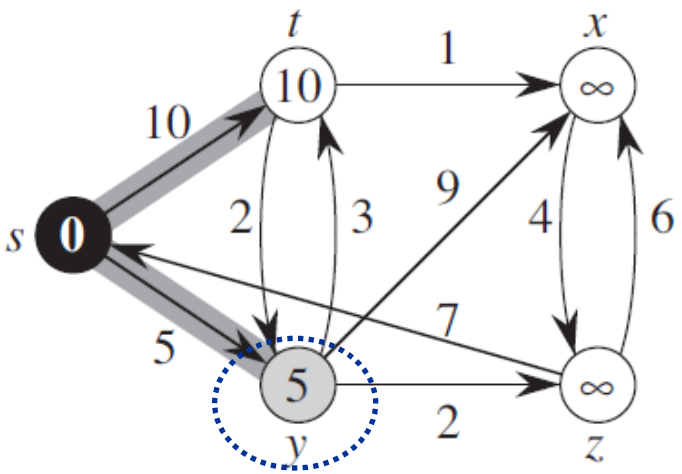

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```



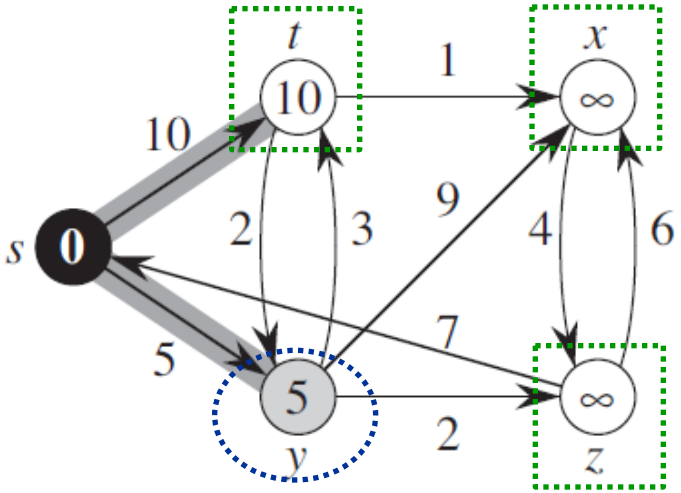

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```



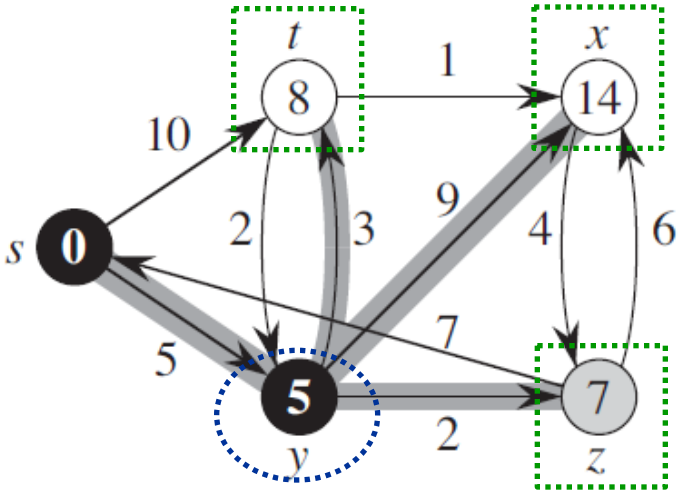

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```



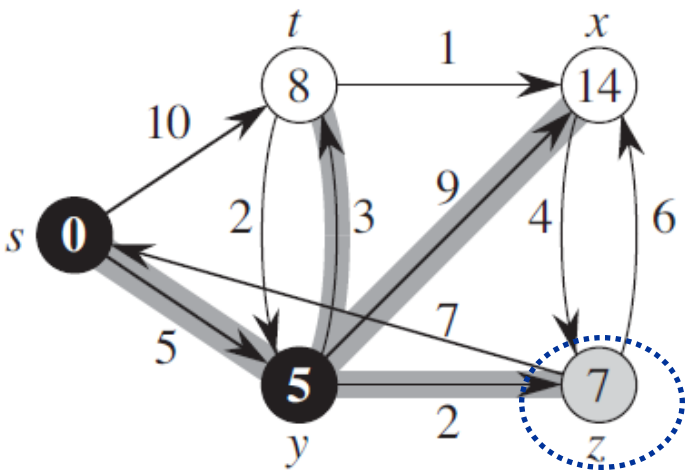

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```



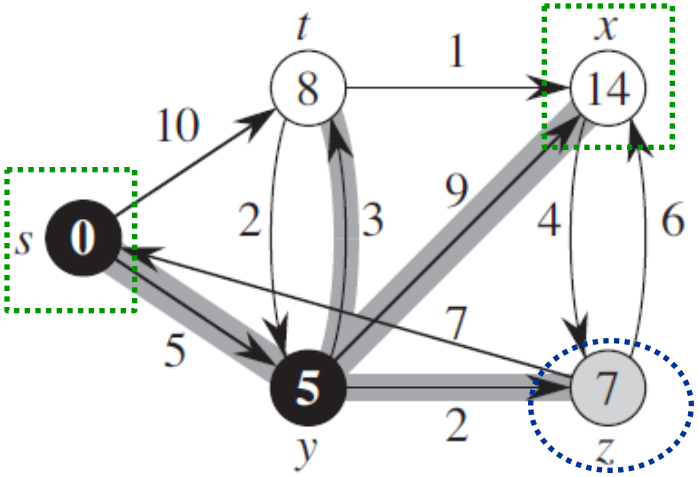

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```



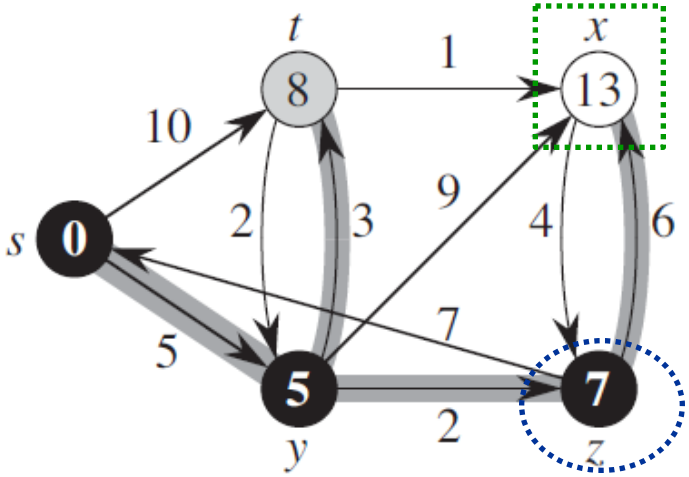

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```



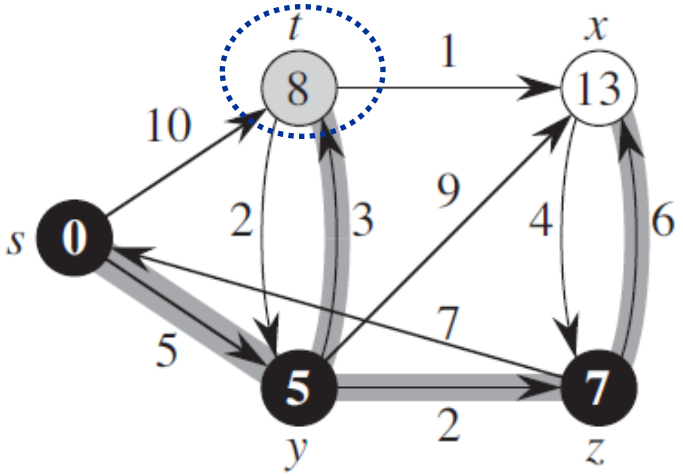

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```



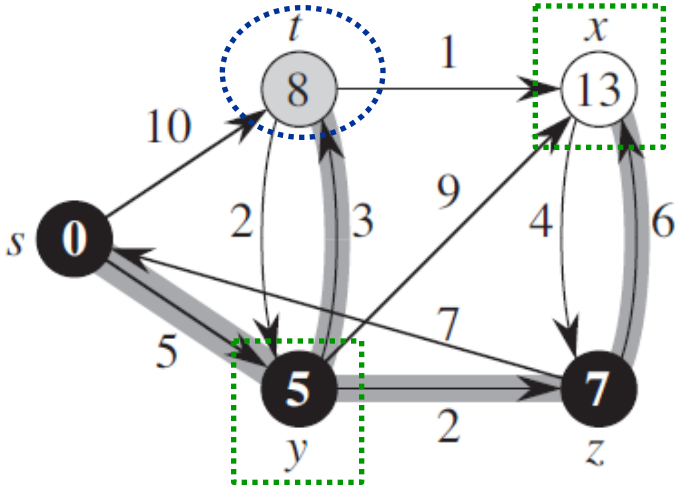

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```



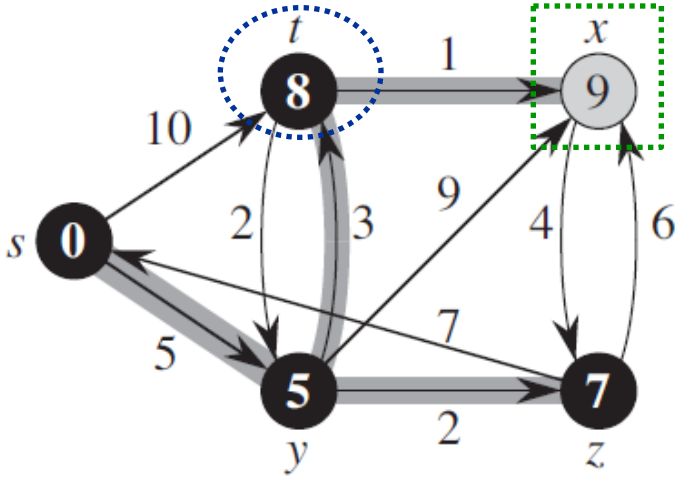

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```



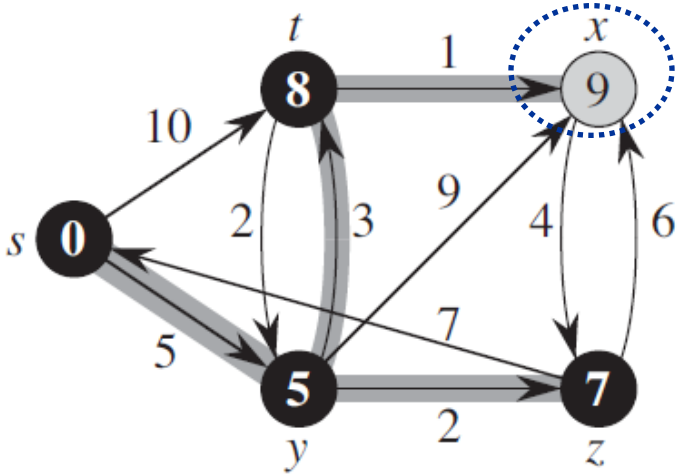

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```



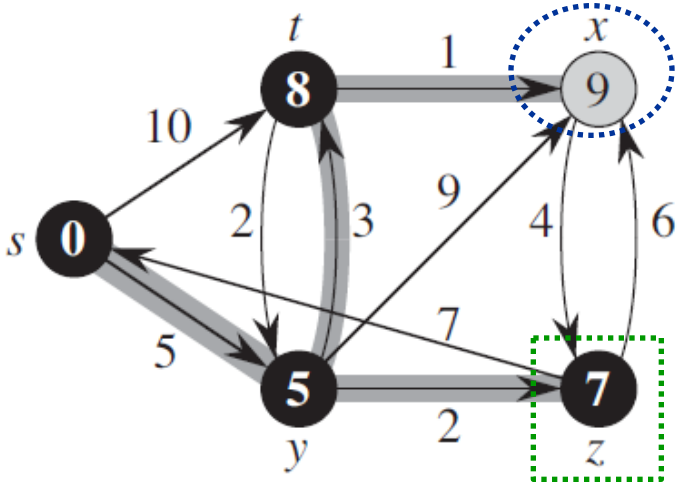

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```



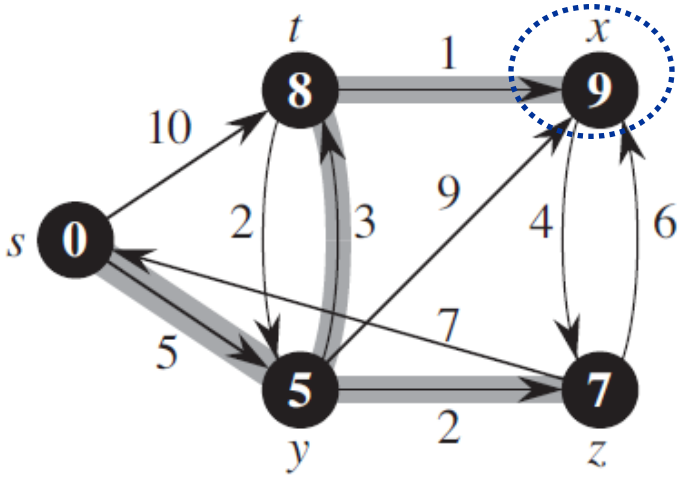

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

```
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
```




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




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

Exercícios

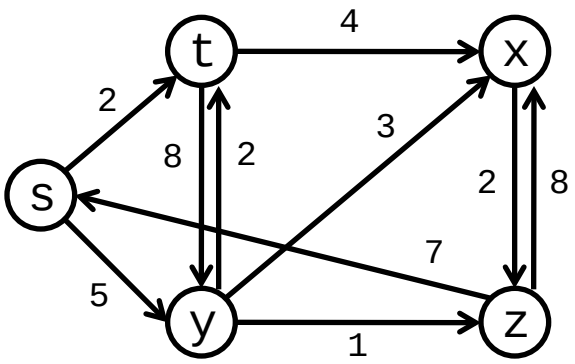
Caminhos mínimos

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

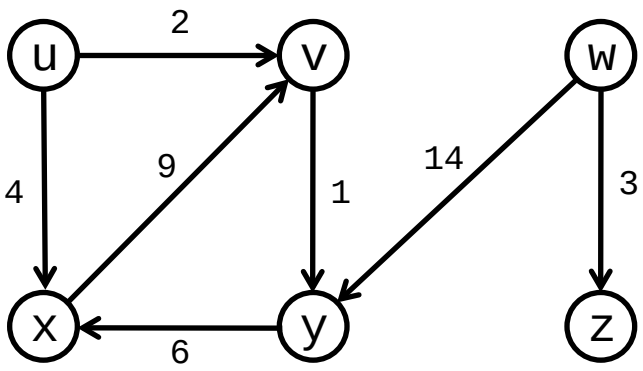
```
RELAX(u, v, w)
1  if v.d > u.d + w(u, v)
2      v.d = u.d + w(u, v)
3      v.π = u
```

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

(a)



(b)



Tarefa

- Exercícios:
  - Lista 4

Tarefa

- EP 4

- Página da disciplina:
  - https://sites.google.com/site/alexnoma/home/grafos