Problema do Caminho Mínimo

Prof. Gilberto Farias

Sumário

- Definição do Problema
- Algoritmo de Dijkstra (Estratégia Gulosa)

Problema do Caminho Mínimo

Em um problema de caminhos mais curtos, temos um grafo orientado ponderado $G=(V,E)$, com função peso w : $E \rightarrow \mathbb{R}$ mapeando arestas para pesos de valores reais.

Exemplo do problema:

Determine o menor caminho entre João Pessoa e o Rio de Janeiro.

Problema do Caminho Mínimo

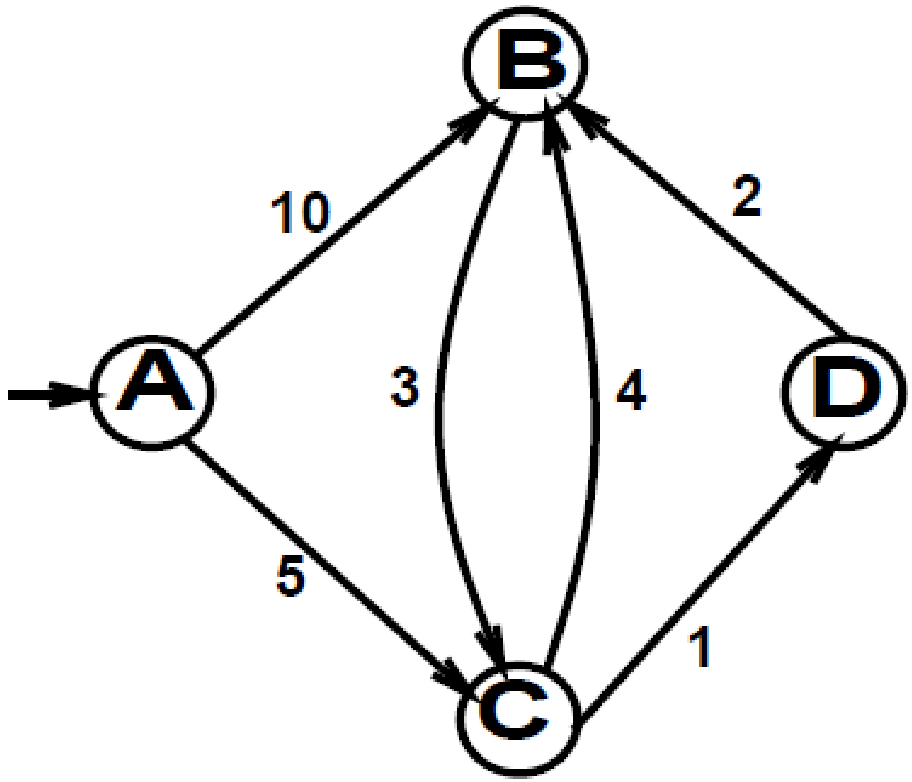
O peso do caminho p=<v_0, v_1, .., v_k> é o somatório dos pesos de suas arestas constituintes.

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

O peso do menor caminho ente *v* e *u* é definido como:

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

Exemplo de caminho mínimo



$\delta(A,B) = ??$

Algoritmos de Caminho Mínimo: Inicialiação

Os algoritmos de Caminho Mínimo de fonte única operam grafo direcionado ponderado *G(V,E)* e usam dois arrays, π e *d*, para calcular o menor caminho de *s* (fonte) até *t* (destino) pertencentes a *V*. Quando o algoritmo termina, para todo vértice $v \in V$, $\pi[v]$ é o vértice predecessor de v no caminho de *s* até v e $d[v]$ é o peso do caminho.

A seguinte rotina é usada pelo algoritmo de Caminho Mínimo para inicializar os arrays.

INITIALIZE-SINGLE-SOURCE(G, s)

for each vertex $v \in V[G]$
    do $d[v] \leftarrow \infty$
        $\pi[v] \leftarrow \mathrm{NIL}$
$d[s] \leftarrow 0$

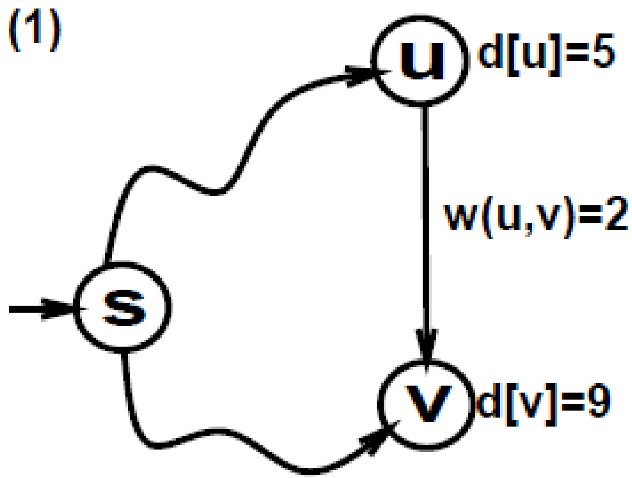
Algoritmos de Caminho Mínimo: Relaxação

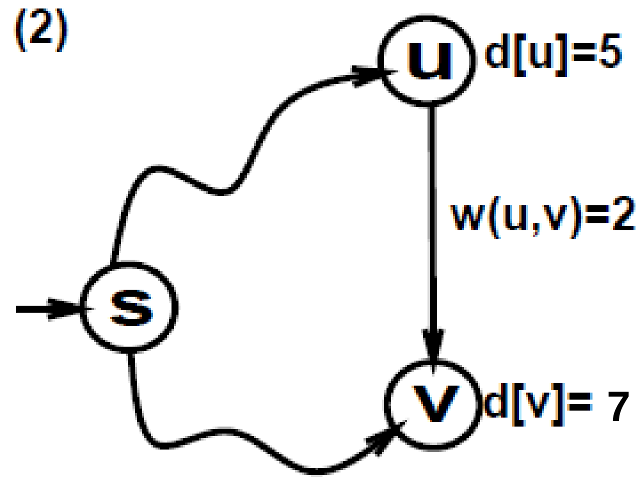
Os algoritmos de Caminho Mínimo de fonte única usam uma técnica chamada **relaxação**. A ideia básica é que $d[v]$ é um valor estimado do peso do caminho entre *s* e v, que é reduzido de ∞ até encontrar o valor do peso do Caminho Mínimo $\delta(s, v)$.

RELAX(u, v, w) atualiza *d*$[v]$ (e $\pi[v]$) pelo exame do impacto do peso da aresta (u, v).

RELAX(u, v, w)
1. **if** $d[v] > d[u] + w(u, v)$
2. **then** $d[v] \leftarrow d[u] + w(u, v)$
3. $\pi[v] \leftarrow u$

Exemplo de Relaxação





Algoritmo de Dijkstra

- O algoritmo de Dijkstra usa uma estratégia gulosa, com a suposição que não há arestas negativas no grafo, para determinar o caminho mínimo de uma fonte s em um grafo direcionado e ponderado $G = (V, E)$ para todos os vértices v pertencentes a V.
- O algoritmo seleciona o vértice v mínimo de uma fila de prioridade Q, classificada pelo valor de $d[v]$, enquanto Q não for vazio. Similar ao algoritmo de PRIM.

Algoritmo de Dijkstra

DIJKSTRA(G, w, s)

INITIALIZE-SINGLE-SOURCE(G, s)

$S \leftarrow \emptyset$

$Q \leftarrow V[G]$

while $Q \neq \emptyset$

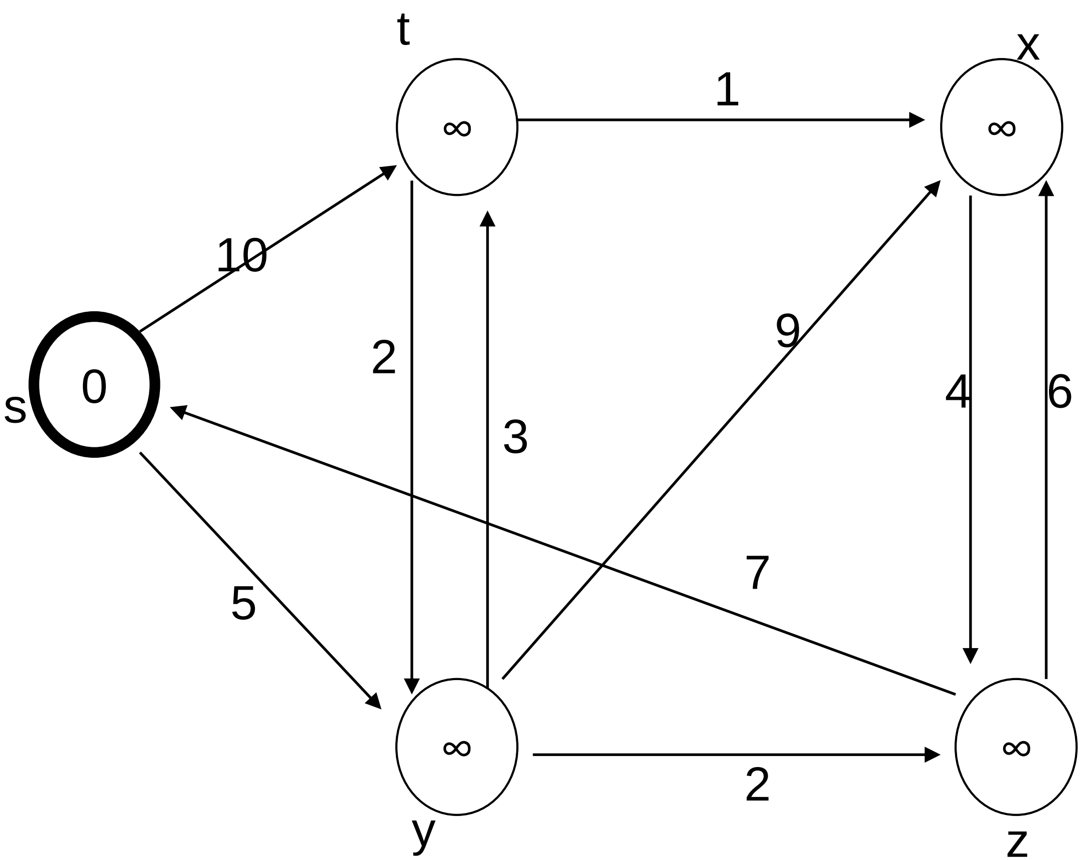
    do $u \leftarrow$ EXTRACT-MIN(Q)

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

        for each vertex $v \in Adj[u]$

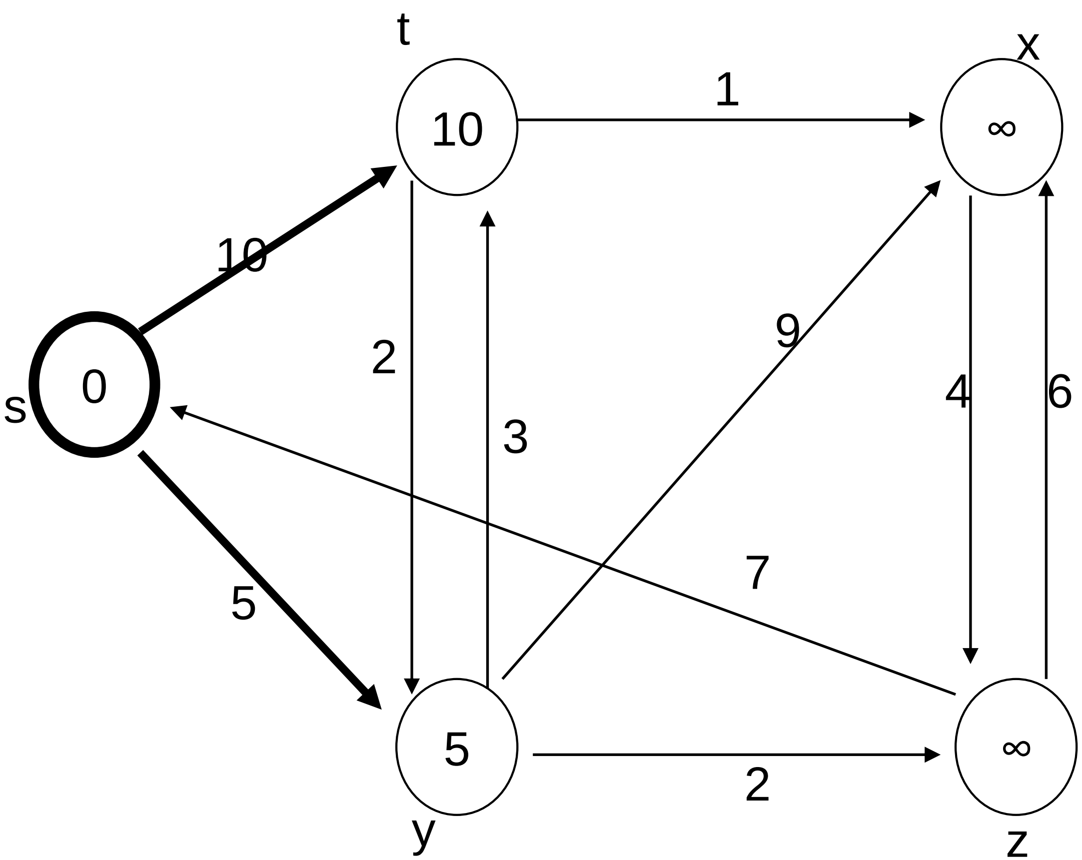
            RELAX(u, v, w)

Algoritmo de Dijkstra



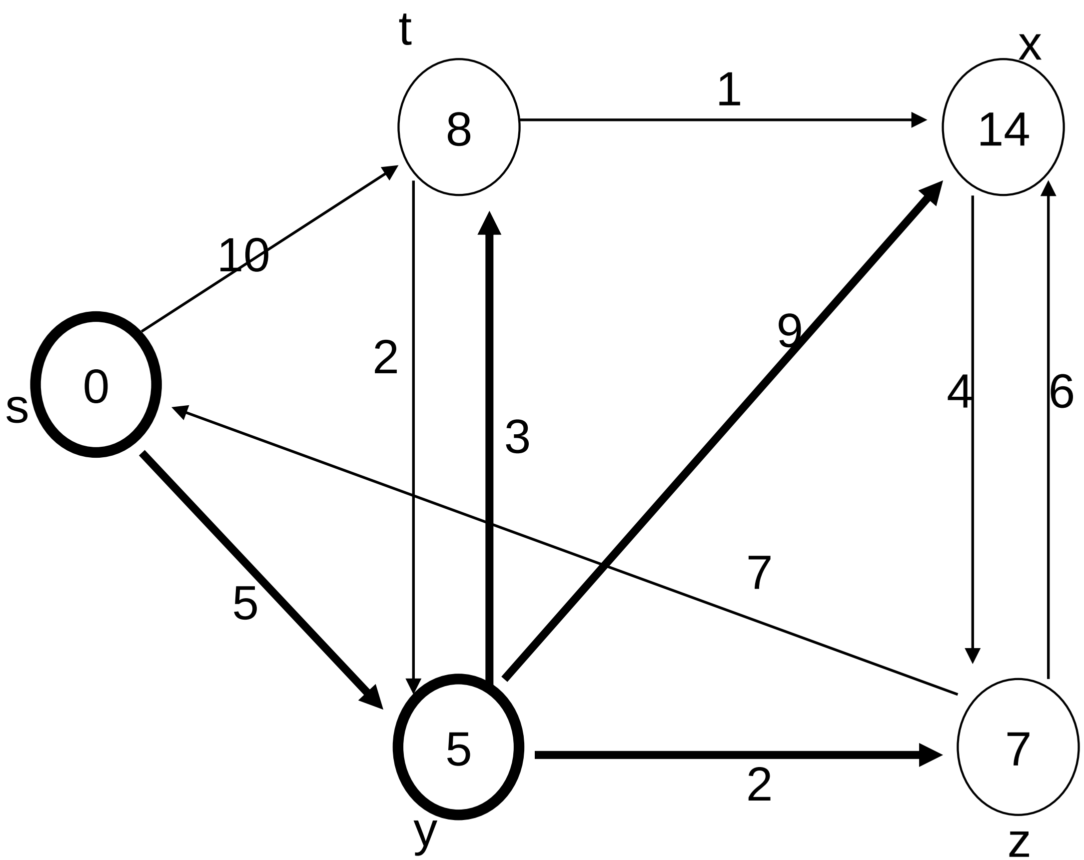

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S ← ∅
Q ← V[G]
while  Q ≠ ∅
        do u ← EXTRACT-MIN(Q)
            S ← S ∪ {u}
            for each vertex v ∈ Adj[u]
                RELAX(u, v, w)
```

Algoritmo de Dijkstra



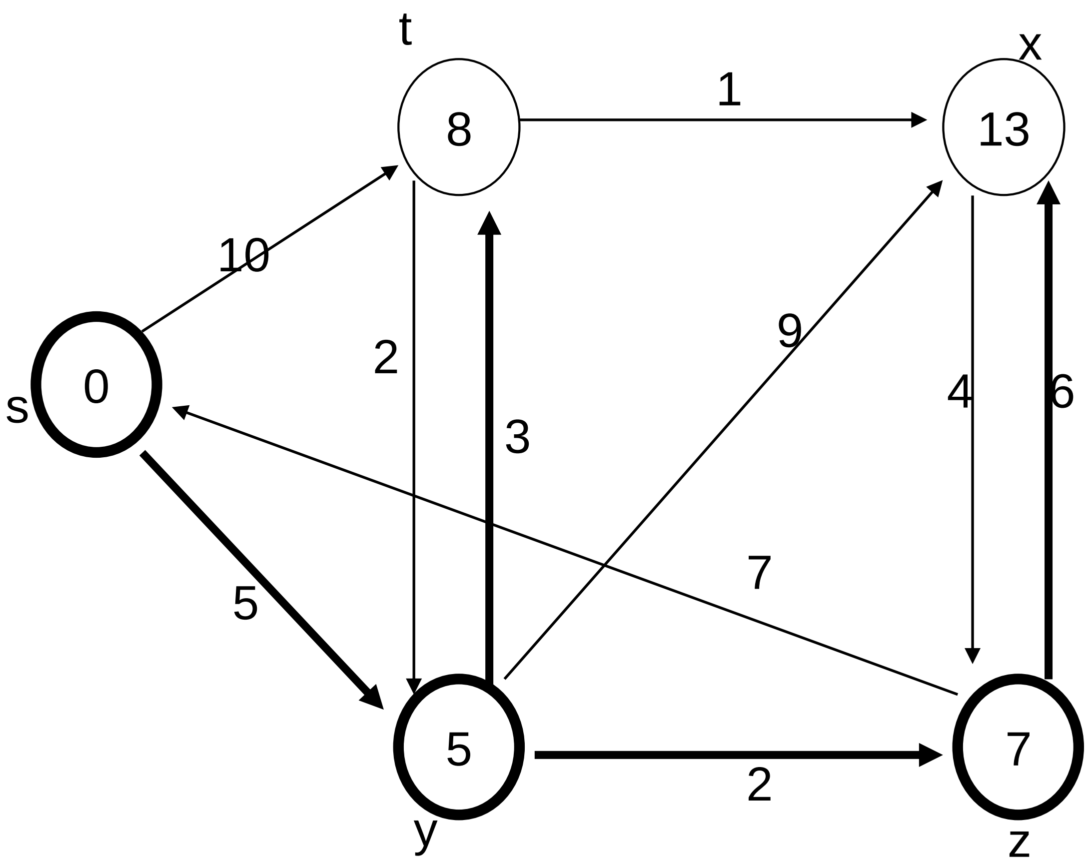

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S ← ∅
Q ← V[G]
while  Q ≠ ∅
        do u ← EXTRACT-MIN(Q)
            S ← S ∪ {u}
            for each vertex v ∈ Adj[u]
                RELAX(u, v, w)
```

Algoritmo de Dijkstra



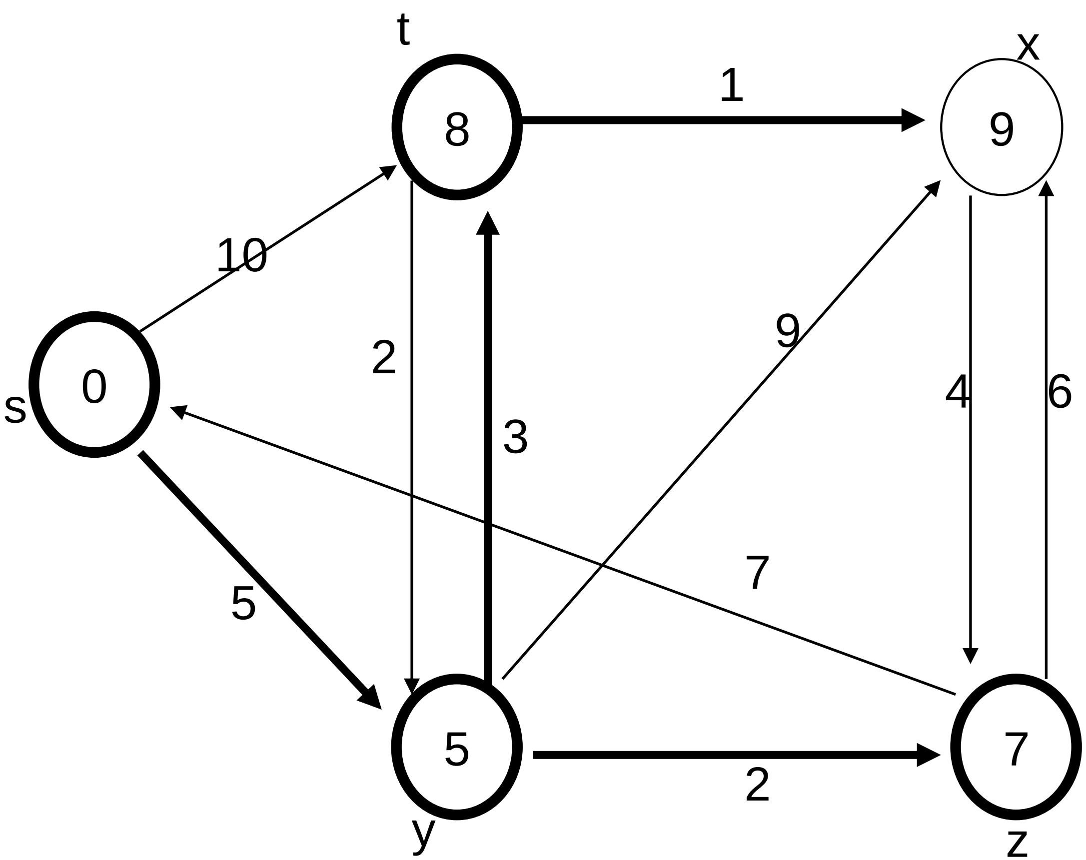

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S ← ∅
Q ← V[G]
while Q ≠ ∅
        do u ← EXTRACT-MIN(Q)
            S ← S ∪ {u}
            for each vertex v ∈ Adj[u]
                RELAX(u, v, w)
```

Algoritmo de Dijkstra




```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S ← ∅
Q ← V[G]
while  Q ≠ ∅
        do u ← EXTRACT-MIN(Q)
            S ← S ∪ {u}
            for each vertex v ∈ Adj[u]
                RELAX(u, v, w)
```

Algoritmo de Dijkstra



DIJKSTRA(G, w, s)

INITIALIZE-SINGLE-SOURCE(G, s)

$S \leftarrow \emptyset$

$Q \leftarrow V[G]$

while $Q \neq \emptyset$

    do $u \leftarrow$ EXTRACT-MIN(Q)

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

        for each vertex $v \in Adj[u]$

            RELAX(u, v, w)