

# Grafos: caminhos mínimos

Prof. Jefferson T. Oliva

Material da Profa. Luciene de Oliveira Marin

Algoritmos e Estrutura de Dados II (AE23CP)

Engenharia de Computação

Departamento Acadêmico de Informática (Dainf)

Universidade Tecnológica Federal do Paraná (UTFPR)

Campus Pato Branco

## Problema do(s) Caminho(s) Mínimo(s)

# Problema do(s) Caminho(s) Mínimo(s)



# Problema do(s) Caminho(s) Mínimo(s)

Seja  $G$  um grafo **orientado** e suponha que para cada aresta  $(u, v)$  associamos um **peso** (custo, distância)  $(u, v)$ . Usaremos a notação  $(G, w)$ .

- **Problema do Caminho Mínimo entre Dois Vértices:**

Dados dois vértices  $s$  e  $t$  em  $(G, w)$ , encontrar um caminho (de peso) mínimo de  $s$  a  $t$ .

- Aparentemente, este problema não é mais fácil do que o **Problema dos Caminhos Mínimos com Mesma Origem:**

Dados  $(G, w)$  e  $s \in V[G]$ , encontrar para cada vértice  $v$  de  $G$ , um caminho mínimo de  $s$  a  $v$ .

# Problema do(s) Caminho(s) Mínimo(s)

**Teorema.** Seja  $(G, w)$  um grafo orientado e seja

$$P = (v_1, v_2, \dots, v_k)$$

um **caminho mínimo** de  $v_1$  a  $v_k$  .

Então para quaisquer  $i, j$  com  $1 \leq i \leq j \leq k$

$$P_{ij} = (v_i, v_{i+1}, \dots, v_j)$$

é um **caminho mínimo** de  $v_i$  a  $v_j$  .

# Representação de caminhos mínimos

- Usamos uma idéia similar à usada em **Busca em Largura** no algoritmo de caminhos mínimos que veremos.
- Para cada vértice  $v \in V[G]$  associamos um predecessor  $\pi[v]$ .
- Ao final do algoritmo obtemos uma **Árvore de Caminhos Mínimos** com raiz  $s$ .
- Um caminho de  $s$  a  $v$  nesta árvore é um caminho mínimo de  $s$  a  $v$  em  $(G, w)$ .

# Estimativa de distâncias

- Para cada  $v \in V[G]$  queremos determinar  $dist(s, v)$ , o peso de um caminho mínimo de  $s$  a  $v$  em  $(G, w)$  (distância de  $s$  a  $v$ .)
- Os algoritmos de caminhos mínimos associam a cada  $v \in V[G]$  um valor  $d[v]$  que é uma estimativa da distância  $dist(s, v)$ .

# Algoritmo - inicialização

INITIALIZE-SINGLE-SOURCE( $G, s$ )

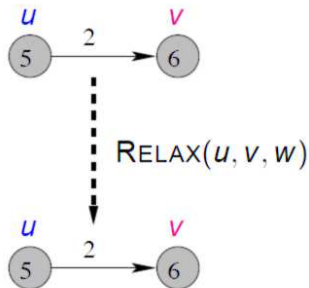
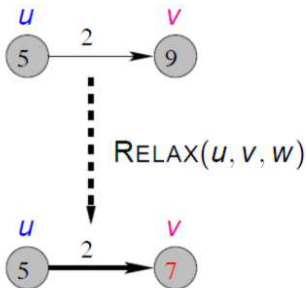
```
1  para cada vértice  $v \in V[G]$  faça
2       $d[v] \leftarrow \infty$ 
3       $\pi[v] \leftarrow \text{NIL}$ 
4   $d[s] \leftarrow 0$ 
```

- O valor  $d[v]$  é uma **estimativa superior** para o peso de um caminho mínimo de  $s$  a  $v$ .
- Ele indica que o algoritmo encontrou até aquele momento um caminho de  $s$  a  $v$  com peso  $d[v]$ .
- O caminho pode ser recuperado por meio dos predecessores  $\pi[]$ .



# Algoritmo - relaxação

Tenta melhorar a estimativa  $d[v]$  examinando  $(u, v)$ .

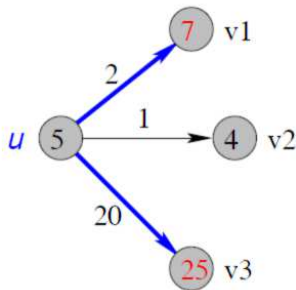
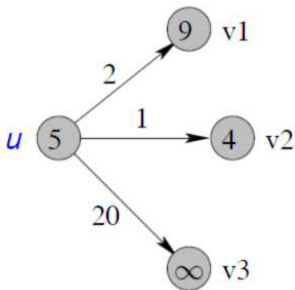


$\text{RELAX}(u, v, w)$

```
1  se  $d[v] > d[u] + w(u, v)$ 
2    então  $d[v] \leftarrow d[u] + w(u, v)$ 
3           $\pi[v] \leftarrow u$ 
```

## Algoritmo - relaxação dos vizinhos

Em cada iteração o algoritmo seleciona um vértice  $u$  e para cada vizinho  $v$  de  $u$  aplica  $\text{RELAX}(u, v, w)$ .

RELAX( $u, v, w$ )

1 se  $d[v] > d[u] + w(u, v)$  faça

$$2 \quad d[v] \leftarrow d[u] + w(u, v)$$
3  $\pi[v] \leftarrow u$

Existem três algoritmos baseados em relaxação para tipos de instâncias diferentes de **Problemas de Caminhos Mínimos**.

- $G$  é acíclico: aplicação de ordenação topológica
- $(G, w)$  não tem arestas de peso negativo: **algoritmo de Dijkstra**
- $(G, w)$  tem arestas de peso negativo, mas não contém ciclos negativos: algoritmo de Bellman-Ford.

## Algoritmo de Dijkstra

## Objetivo:

- encontrar o caminho de distância mínima de um vértice origem  $x$  a um vértice destino  $y$ .
- resolve o problema do caminho mínimo em um grafo direcionado ou não direcionado com arestas de peso não negativo
- É semelhante ao **algoritmo BFS (busca em largura)**, utilizando uma estratégia gulosa.

# Algoritmo de Dijkstra

## Entrada

O algoritmo de Dijkstra recebe um grafo orientado (ou não)  $(G, w)$  (sem arestas de peso negativo) e um vértice  $s$  de  $G$

e devolve

## Saída

- para cada  $v \in V[G]$ , o peso de um caminho mínimo de  $s$  a  $v$
- e uma **Árvore de Caminhos Mínimos** com raiz  $s$ .  
Um caminho de  $s$  a  $v$  nesta árvore é um caminho mínimo de  $s$  a  $v$  em  $(G, w)$ .

# Algoritmo de Dijkstra

DIJKSTRA( $G, w, s$ )

1   INITIALIZE-SINGLE-SOURCE( $G, s$ )

2    $S \leftarrow \emptyset$

3    $Q \leftarrow V[G]$

4   **enquanto**  $Q \neq \emptyset$  **faça**

5      $u \leftarrow \text{EXTRACT-MIN}(Q)$

6      $S \leftarrow S \cup \{u\}$

7     **para cada** vértice  $v \in \text{Adj}[u]$  **faça**

8       RELAX( $u, v, w$ )

O conjunto  $Q$  é implementado como uma fila de prioridade.

O conjunto  $S$  não é realmente necessário, mas simplifica a análise do algoritmo.

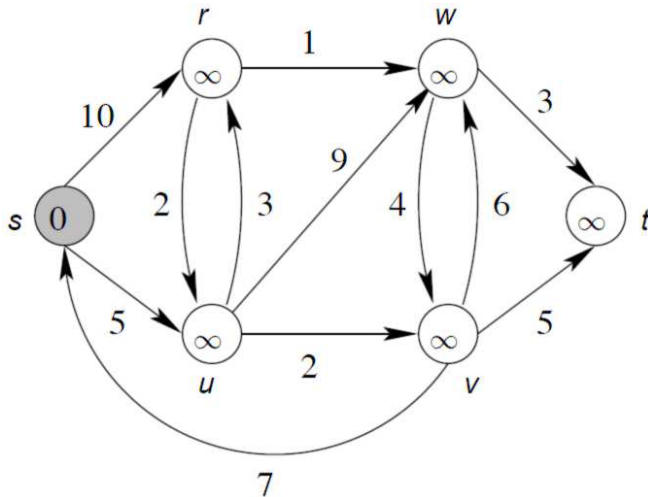
# Algoritmo de Dijkstra - intuição do algoritmo

Em cada iteração, o algoritmo de **DIJKSTRA**

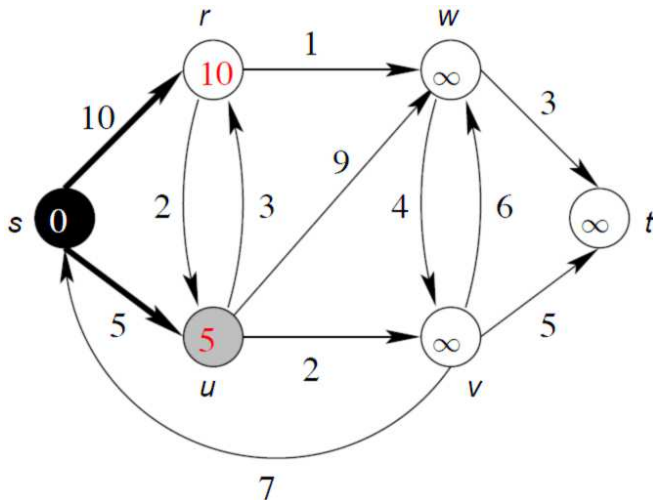
- escolhe um vértice  $u$  fora do conjunto  $S$  que esteja **mais próximo** a esse e acrescenta-o a  $S$ ,
- atualiza as distâncias estimadas dos vizinhos de  $u$  e
- atualiza a **Árvore dos Caminhos Mínimos**.



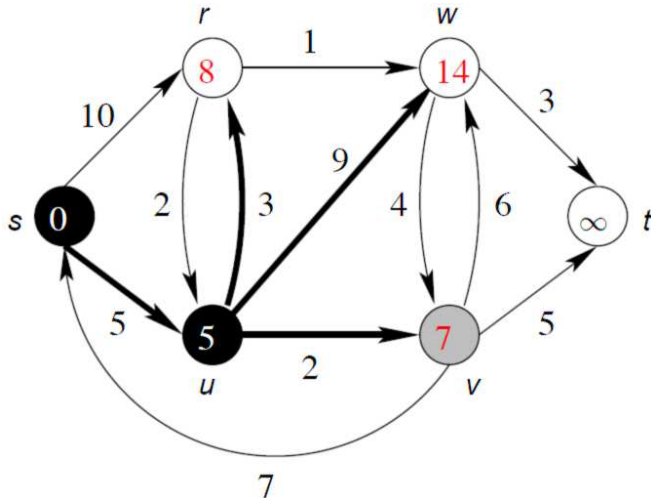
# Algoritmo de Dijkstra - Exemplo



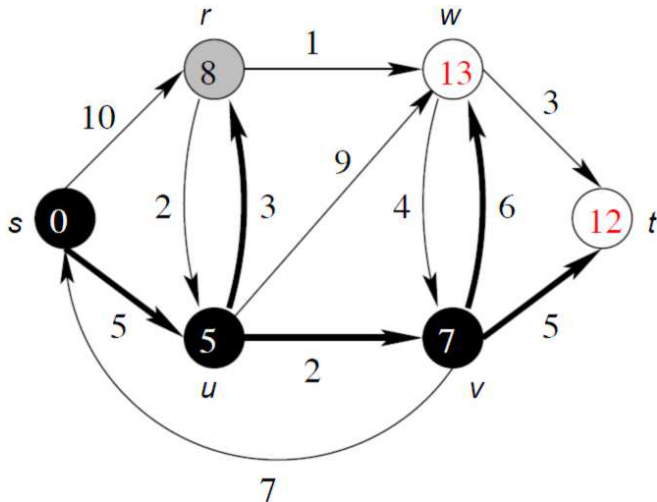
# Algoritmo de Dijkstra - Exemplo



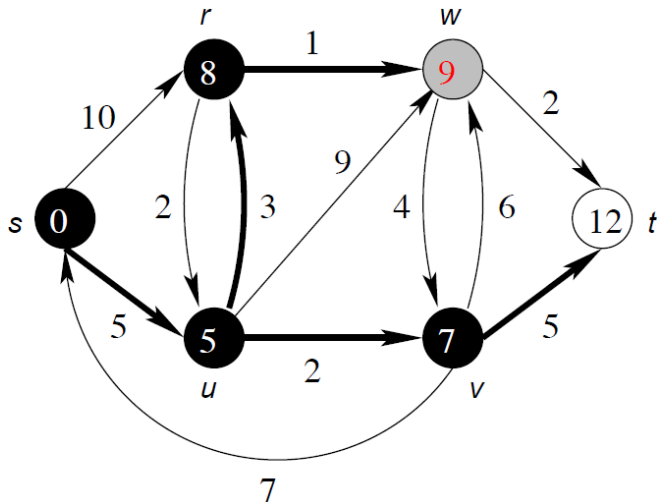
# Algoritmo de Dijkstra - Exemplo



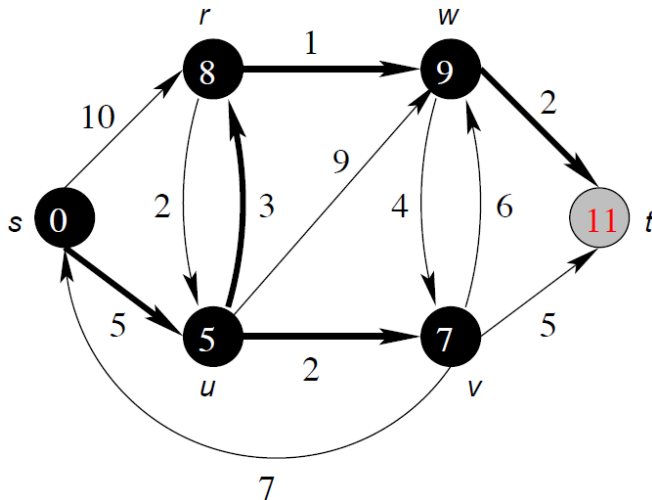
# Algoritmo de Dijkstra - Exemplo



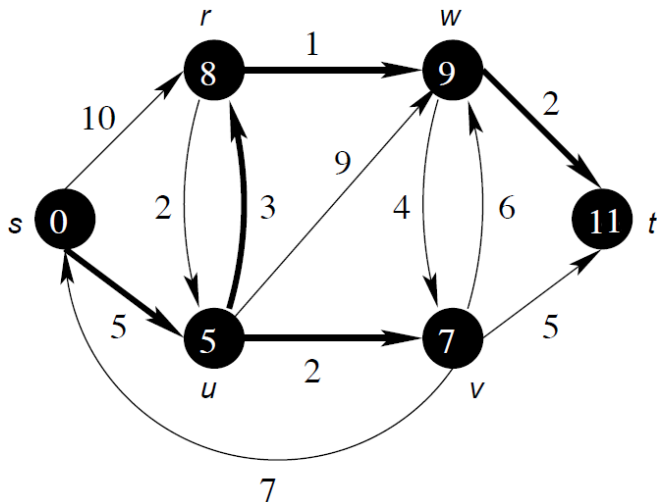
# Algoritmo de Dijkstra - Exemplo



# Algoritmo de Dijkstra - Exemplo

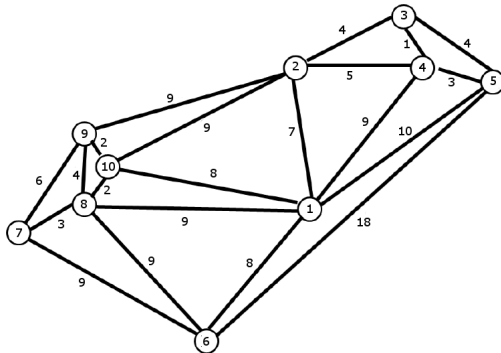


# Algoritmo de Dijkstra - Exemplo



# Exercício

Considere o grafo abaixo, em seguida faça:



- Execute o algoritmo de Dijkstra e encontre a árvore de caminhos mínimos do grafo acima, considerando o nó 1 como origem.
- Implemente o algoritmo de Dijkstra em linguagem C.



- Cormen et. al. *Algoritmos - Teoria e Prática*.
- Nivio Ziviani. *Projeto de Algoritmos com Implementações em Pascal e C*.