Problema dos Caminhos Mínimos de Única Fonte

Prof. Fábio Dias

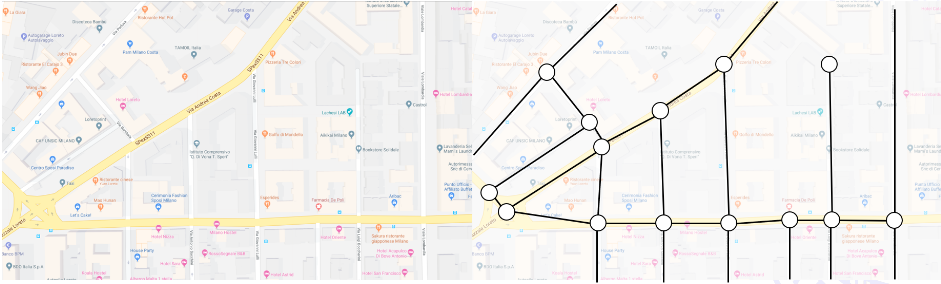
14 de janeiro de 2025

Introdução

Introdução

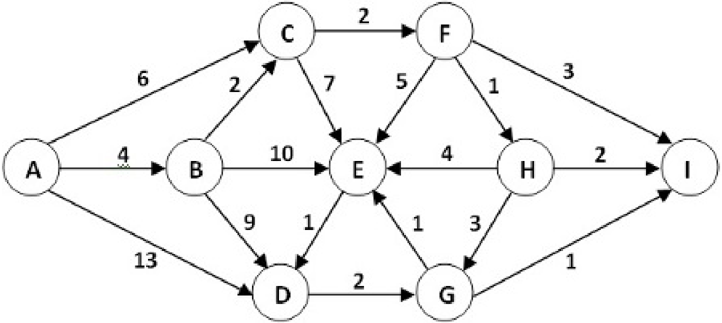
- Um motorista deseja encontrar a rota mais curta possível do Rio de Janeiro a São Paulo.
- Dado um mapa das rodovias do Brasil no qual a distância entre cada par de interseções adjacentes esteja marcada, como podemos determinar a rota mais curta?
- Uma solução possível seria enumerar todas as rotas existentes, somar as distâncias en cada rota e selecionar a mais curta.
- Esse algoritmo é eficiente??
- Algoritmo de Força Bruta.

Como um mapa se torna um grafo?



- As ruas são arcos, enquanto as interseções são vértices.

Como um mapa se torna um grafo?



Introdução

- Dado um grafo direcionado $G = (V, E)$, com $|V| = n$, $|E| = m$ e ponderado nas arestas: uma função $w : E \to R$.
- O peso de um caminho $p = (v_0, v_1, ..., v_k)$ é a soma dos pesos dos arcos: $w(p) = \sum_{i=0}^{k-1} w(v_i, v_{i+1})$
- Definimos o peso do caminho mais curto de um vértice u para v como:

$$\delta(u, v) = \begin{cases} \min\{w(p) : p \text{ é caminho de } u \rightsquigarrow v\} \\ +\infty \text{ se não existir caminho} \end{cases}$$

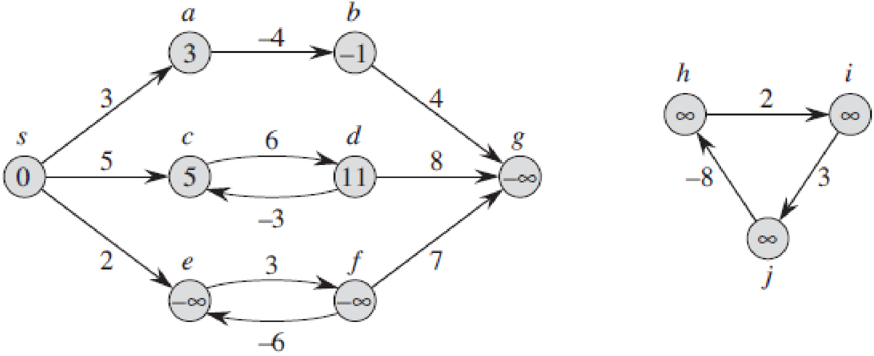
Versões do Problema

- **Problema de caminhos mínimos de fonte única**;
- 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 Pesos Negativos

- Em algumas instâncias do problema de caminhos mínimos com fonte única, podem existir arestas com pesos negativos.
- Se o grafo G não tiver ciclo com peso negativo alcançável a partir da fonte s, então $\delta(s, v)$ permanece bem definido para todo $v \in V$, mesmo se o grafo contiver algum ciclo negativo.
- Caso exista algum ciclo de peso negativo, então $\delta(s, v) = -\infty$.

Arestas de Pesos Negativos



Representação

- Assim como na busca em Largura, iremos manter para cada vértice $v \in V$ o seu predecessor π que é outro vértice ou nil.
- O grafo predecessor $G_\pi = (V_{pi}, E_{pi})$ induzido por π, após o término do algoritmo, produz a árvore de caminhos mínimos:

$$V_\pi = \{v \in V : \pi[v] \neq nil\} \cup \{s\}$$
$$E_\pi = \{(\pi[v], v) : v \in V_\pi - \{s\}\}$$

Inicialização

- Assim como na busca em Largura, teremos o atributo d para todo vértice.
- Armazena o valor do caminho encontrado pelo algoritmo, uma estimativa do caminho mínimo.
- Será um limite superior para o caminho mínimo da fonte s até v: $d[v] \geq \delta(s, v)$.

Initialize-Single-Source(G, s)

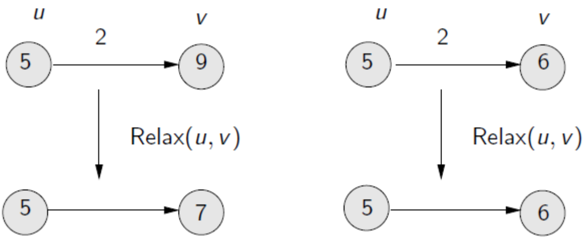
for each $v \in V[G]$
    do $d[v] \leftarrow \infty$
        $\pi[v] \leftarrow nil$
$d[s] = 0$

Após a inicialização, $\pi[v] = nil$ para todo $v \in V$, $d[s] = 0$, $d[v] = \infty$ para todo $v \in V - \{s\}$.

Relaxamento

- A técnica de relaxamento diminui esse limite superior para a distância mínima para cada vértice.

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



Relaxamento

- O propósito de relaxar uma aresta (u, v) consiste de testar se podemos melhorar a estimativa do caminho mais curto para v encontrado até então, por meio do caminho até u.
- A corretude de algoritmos de caminhos mínimos é baseada na propriedade de caminhos mínimos e das relaxações.

Algoritmo de Dijkstra

Algoritmo de Dijkstra

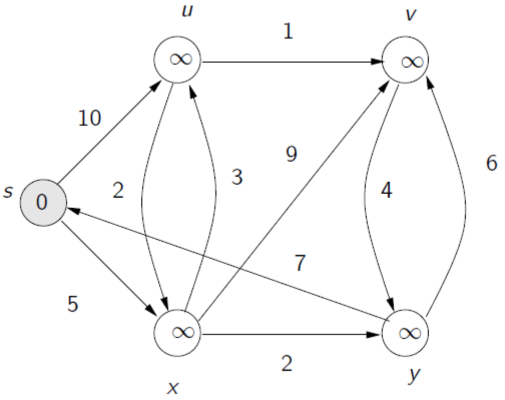
- Apenas em grafos sem arestas com pesos negativos.
- O algoritmo mantém um conjunto S de vértices para os quais a distância do caminho mais curto a partir de s já foi encontrada.
- Ou seja, para todo $v \in S$, temos $d[v] = \delta(s, v)$.
- O algoritmo iterativamente seleciona um vértice $u \in V - S$ que possua a menor estimativa de distância (atributo d), $d[u] = \min d[v] : v \in V - S$, relaxa as arestas incidentes a u e insere u em S.
- Faz uso de uma Lista de Prioridade.

Algoritmo de Dijkstra

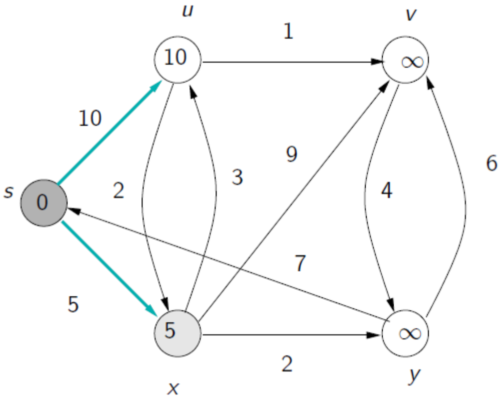
Dijkstra(G, w, s)

Initialize-Single-Source(G, s)
$S \leftarrow \emptyset$
$Q = V[G]$
while $Q \neq \emptyset$
    do $u \leftarrow$ Extract_min(Q)
        $S \leftarrow S \cup \{u\}$
        for each $(u, v) \in Adj[u]$
            do Relax(u, v, w)

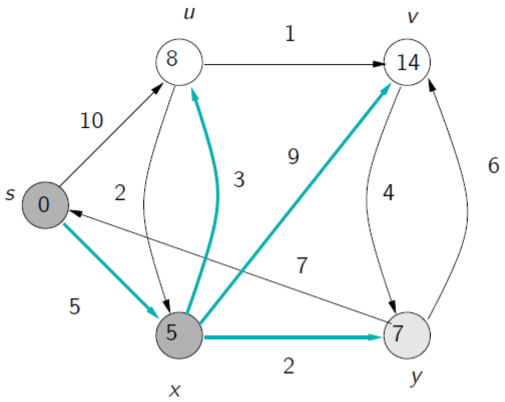
Algoritmo de Dijkstra - Inicialização



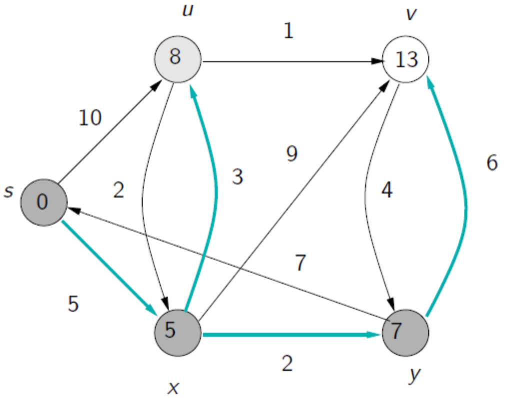
Algoritmo de Dijkstra - Iteração 1



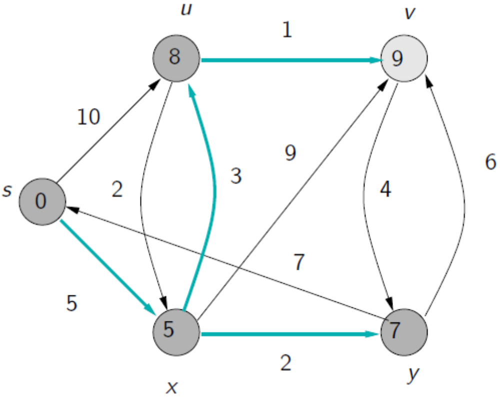
Algoritmo de Dijkstra - Iteração 2



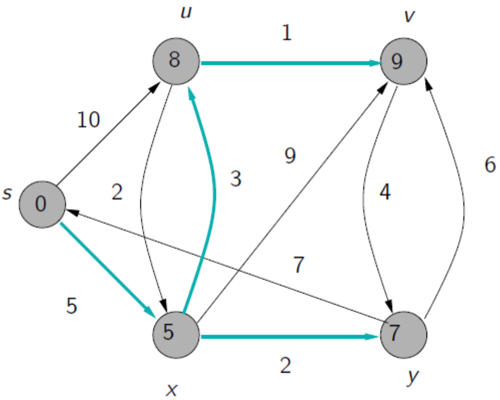
Algoritmo de Dijkstra - Iteração 3



Algoritmo de Dijkstra - Iteração 4



Algoritmo de Dijkstra - Iteração 5



Complexidade

Algoritmo de Dijkstra - Complexidade

Dijkstra(G, w, s)
Initialize-Single-Source(G, s)
$S \leftarrow \emptyset$
$Q = V[G]$
while $Q \neq \emptyset$
  do $u \leftarrow$ Extract_min(Q)
    $S \leftarrow S \cup \{u\}$
    for each $(u, v) \in Adj[u]$
      do Relax(u, v, w)

- Usando a implementação de Fila de Prioridade com vetor, temos o custo do extract-min de $O(n)$.
- Uma vez que cada aresta é examinada no máximo uma vez: $O(n^2 + m) = O(n^2)$.
- Usando a implementação com Heap Binário, extract-min tem custo $O(\log n)$ e alteração do Heap tem custo $O(\log n)$.
- Logo: $O(n \log n + m \log n) = O(m \log n)$, assumindo que $m >>> n$.

Perguntas?!