SCC0216 - Modelagem Computacional em Grafos

Caminhos Mínimos

Prof.: Rosane Minghim (rminghim@icmc.usp.br) 1° sem. 2014

PAE: Rafael Martins (rmartins@icmc.usp.br)
Bilzã Marques (bmarques@icmc.usp.br)

Baseado no material de aula original: Profª. Josiane M. Bueno

Caminho mínimo

- **Problema: encontrar o caminho de menor custo (ou o menor caminho) entre dois vértices em um grafo valorado**
  - ***Algoritmo de Djikstra***
    Uma única origem
  - ***Algoritmo de Floyd-Warshall***
    Caminhos mais curtos de todos os pares possíveis

Caminho mínimo

Grafo dirigido G(V, E) com função peso w:
E→ℜ que mapeia as arestas em pesos.

Peso (custo) do caminho p = <v0, v1, ..., vk>

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

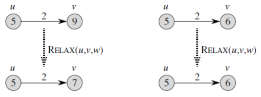
Custo do caminho de menor peso entre u e v:

$$\delta(u,v) = \begin{cases} \min\{w(p) : u \overset{p}{\Rightarrow} v\} se \exists\ rota\ de\ u\ \ p/v \\ \infty\ cc \end{cases}$$

Dijkstra (Cormen, 2001)

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

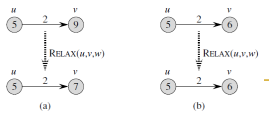
```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2     then d[v] ← d[u] + w(u, v)
3          π[v] ← u
```



Algoritmo Dijkstra (Cormen, 2001)

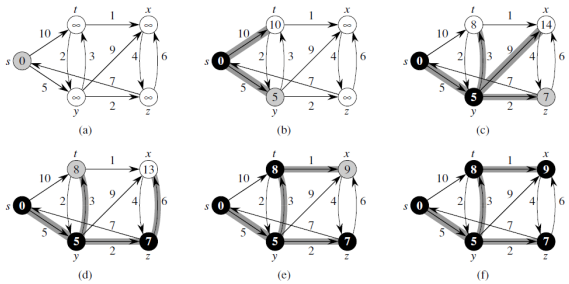
```
INITIALIZE-SINGLE-SOURCE(G, s)
1  for each vertex v ∈ V[G]
2       do d[v] ← ∞
3          π[v] ← NIL
4  d[s] ← 0
```

```
RELAX(u, v, w)
1  if d[v] > d[u] + w(u, v)
2     then d[v] ← d[u] + w(u, v)
3          π[v] ← u
```




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


Exemplo Dijkstra (Cormen, 2001)



Complexidade Dijkstra (Cormen, 2001)

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

$O(E)$ DECREASE-KEY's implícitos

Tempo = $V.T_{\text{EXTRACT-MIN}} + E.T_{\text{DECREASE-KEY}}$

Complexidade Dijkstra (Cormen, 2001)

Tempo = $V.T_{\text{EXTRACT-MIN}} + E.T_{\text{DECREASE-KEY}}$

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total
array	$O(V)$	$O(1)$	$O(V^2)$
binary heap	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$

Caminhos entre todos os pares

Entrada: Grafo direcionado $G = (V, E)$, com uma função de peso $w : E \rightarrow \mathrm{R}$.

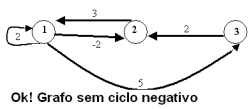
Saída: matriz $n \times n$ com os pesos dos menores caminhos $\delta(i, j)$ entre todos os vértices $i, j \in V$.

Floyd-Warshall

O algoritmo de Floyd-Warshall determina as distâncias dos menores caminhos entre todos os pares de vértices de um grafo.

Trabalha com arestas com pesos negativos.

Mas não funciona quando existem ciclos negativos no grafo.



Ok! Grafo sem ciclo negativo

Nada feito. Grafo com ciclo negativo (arestas vermelhas)

Floyd-Warshall

Ideia

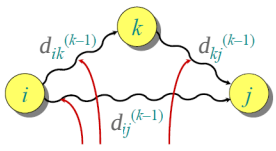
$d_{ij}^{(k)}$: peso do menor caminho do vértice i ao vértice j cujos vértices intermediários pertencem ao conjunto $\{1,2,\ldots,k\}$

Começar com k=0 e ir atualizando a matriz incrementando o valor de k, ou seja, inserindo mais um vértice no conjunto de vértices intermediários permitidos.

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 , \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1 . \end{cases}$$

Floyd-Warshall

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 , \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1 . \end{cases}$$



Floyd-Warshall

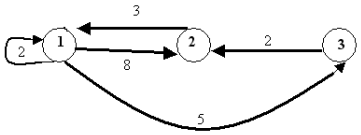
W: matriz representando os pesos das arestas:

$$w_{ij} = \begin{cases} 0 & \text{if } i = j, \\ \text{the weight of directed edge } (i, j) & \text{if } i \neq j \text{ and } (i, j) \in E, \\ \infty & \text{if } i \neq j \text{ and } (i, j) \notin E. \end{cases}$$

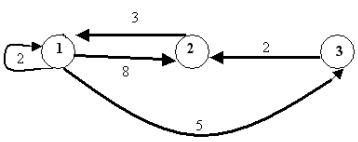
Algoritmo:

```
FLOYD-WARSHALL(W)
1  n ← rows[W]
2  D(0) ← W
3  for k ← 1 to n
4       do for i ← 1 to n
5            do for j ← 1 to n
6                 do d_ij(k) ← min(d_ij(k−1), d_ik(k−1) + d_kj(k−1))
7  return D(n)
```


Exemplo de Floyd-Warshall



Exemplo de Floyd-Warshall



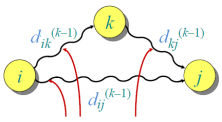
$$D^{(0)} = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & \infty \\ \infty & 2 & 0 \end{bmatrix} \quad D^{(1)} = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ \infty & 2 & 0 \end{bmatrix} \quad D^{(2)} = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{bmatrix} \quad D^{(3)} = \begin{bmatrix} 0 & 7 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{bmatrix}$$

Floyd-Warshall

Caminho?

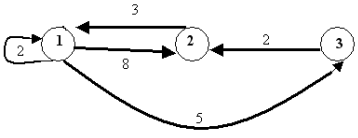
π_{ij}: predecessor do vértice j no menor caminho de i para j com todos os intermediários pertencendo ao conjunto $\{1,2,\ldots,k\}$.

$$\pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{if } i = j \text{ or } w_{ij} = \infty, \\ i & \text{if } i \neq j \text{ and } w_{ij} < \infty. \end{cases}$$



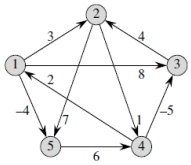
$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}, \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)}. \end{cases}$$

Exemplo de Floyd-Warshall
Armazenando caminho



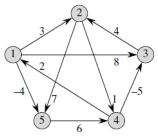
Quadro…

Exemplo de Floyd-Warshall
Armazenando caminho (Cormen, 2001)



Exemplo de Floyd-Warshall

Armazenando caminho (Cormen, 2001)



$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix} \quad D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix} \quad D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix} \quad D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

Floyd-Warshall

SE $i \neq j$ E $(i,j) \in A$ então $B_0[i,j]=C[i,j]$

SE $(i,j) \notin A$ então $B_0[i,j]=\infty$

SE $i=j$ então $B_0[i,j]=0$

Para $k=1$ até N faça

$B_k[i,j]=\min(B_{k-1}[i,j], B_{k-i}[i,k]+B_{k-1}[k,j])$

B_n contém a distância dos caminhos mínimos de todos os pares de vértices

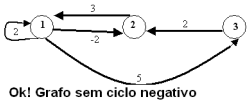
C[i,j]: custo para ir de i a j

A: conjunto de arestas do grafo

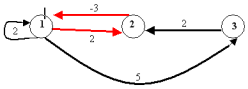
N: número de vértices do grafo

Floyd-Warshall

- O algoritmo de Floyd-Warshall determina as distâncias dos menores caminhos entre todos os pares de vértices de um grafo
- Trabalha com arestas com pesos negativos
- Mas não funciona quando existem ciclos negativos no grafo



Ok! Grafo sem ciclo negativo



Nada feito. Grafo com ciclo negativo (arestas vermelhas)