CAMINHOS MÍNIMOS EM GRAFOS: FLOYD-WARSHALL (PARTE III)

Prof. Daniel Kikuti

Universidade Estadual de Maringá

27 de abril de 2015

Sumário

Introdução

O Problema

Dado um grafo orientado $G = (V, E)$ e uma função peso $w : E \to R$, queremos encontrar o caminho de custo mínimo de u até v, para todo par de vértices $u, v \in V$.

Introdução

O Problema

Dado um grafo orientado $G = (V, E)$ e uma função peso $w : E \to R$, queremos encontrar o caminho de custo mínimo de u até v, para todo par de vértices $u, v \in V$.

Solução trivial

Aplicar um algoritmo de caminho mínimo de única origem $|V|$ vezes, uma para cada vértice.

	Complexidade	
Algoritmo	Grafo esparso	Grafo denso
Dijkstra	$O(VE \lg V)$	$O(V^3 \lg V)$
Bellman-Ford	$O(V^2 E)$	$O(V^4)$

Considerações iniciais

- Supomos que não existem ciclos negativos.
- Os vértices estão numerados de 1 a n, onde $n = |V|$.
- **Entrada:** matriz $W_{n \times n}$ que representa os pesos das arestas. Isto é, $W = w_{ij}$, onde

$$w_{ij} = \begin{cases} 0 & \text{se } i = j \\ \text{o peso da aresta } (i,j) & \text{se } i \neq j \text{ e } (i,j) \in E \\ \infty & \text{se } i \neq j \text{ e } (i,j) \notin E \end{cases}$$

- **Saída:** Matriz $D_{n \times n} = d_{ij}$, onde cada $d_{ij} = \delta(i,j)$, e a Matriz $\Pi_{n \times n} = \pi_{ij}$, onde cada π_{ij} é o vértice predecessor de j em um caminho a partir de i.

Ideia do algoritmo

Programação dinâmica

- Caracterizar estrutura da solução ótima.
- Definir solução recursiva.
- Computar os custos mínimos.
- Construir a solução ótima.

Caracterizar estrutura da solução ótima

Definição de vértice intermediário

Um **vértice intermediário** em um caminho $p = \langle v_1, v_2, \ldots, v_l \rangle$ é qualquer vértice de p, exceto v_1 ou v_l.

Subestrutura ótima

Considere um caminho mínimo $i \overset{p}{\rightsquigarrow} j$ com todos os vértices intermediários em $\{1, 2, \ldots, k\}$.

- Se k **não é um vértice intermediário** de p, então todos os vértices intermediários de p estão em $\{1, 2, \ldots, k-1\}$.
- Se k **é um vértice intermediário** de p, então podemos desmembrar p em $i \overset{p_1}{\rightsquigarrow} k \overset{p_2}{\rightsquigarrow} j$. p_1 e p_2 são caminhos mínimos com todos os vértices intermediários em $\{1, 2, \ldots, k-1\}$.

Solução recursiva

Seja $d_{ij}^{(k)}$ o custo de um caminho mínimo $i \rightsquigarrow j$ com todos os vértices intermediários em $\{1, 2, \ldots, k\}$.

Base da indução

Solução recursiva

Seja $d_{ij}^{(k)}$ o custo de um caminho mínimo $i \rightsquigarrow j$ com todos os vértices intermediários em $\{1, 2, \ldots, k\}$.

Base da indução

$k = 0$, o caminho de i a j sem um vértice com numeração maior que 0 não possui nenhum vértice intermediário. Este caminho possui no máximo uma aresta.

Solução recursiva

Seja $d_{ij}^{(k)}$ o custo de um caminho mínimo $i \rightsquigarrow j$ com todos os vértices intermediários em $\{1, 2, \ldots, k\}$.

Base da indução

$k = 0$, o caminho de i a j sem um vértice com numeração maior que 0 não possui nenhum vértice intermediário. Este caminho possui no máximo uma aresta.

Recursão

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

Solução recursiva

Seja $d_{ij}^{(k)}$ o custo de um caminho mínimo $i \rightsquigarrow j$ com todos os vértices intermediários em $\{1, 2, \ldots, k\}$.

Base da indução

$k = 0$, o caminho de i a j sem um vértice com numeração maior que 0 não possui nenhum vértice intermediário. Este caminho possui no máximo uma aresta.

Recursão

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

Observação importante

Para qualquer caminho, todos os vértices intermediários estão no conjunto $\{1, 2, \ldots, n\}$. Portanto, a matriz $D^{(n)} = d_{ij}^{(n)}$ fornece a reposta desejada: $d_{ij}^{(n)} = \delta(i, j) \forall i, j \in V$.

Computando os custos mínimos

O algoritmo de Floyd-Warshall

```
Floyd-Warshall(W)
1 n ← Número de linhas de W;
2 D^(0) = W
3 para k ← 1 até n faça
4     para i ← 1 até n faça
5         para j ← 1 até n faça
6             d_ij^(k) = min(d_ij^(k−1), d_ik^(k−1) + d_kj^(k−1))
7 devolva D^(n)
```

Computando os custos mínimos

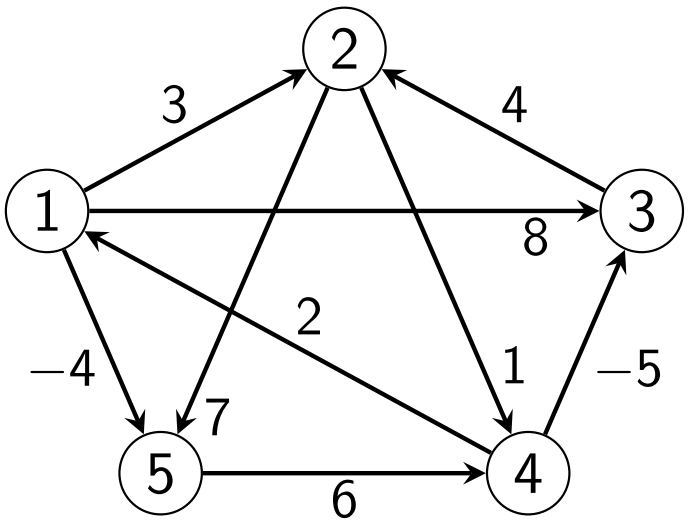
O algoritmo de Floyd-Warshall

```
Floyd-Warshall(W)
1 n ← Número de linhas de W;
2 D^(0) = W
3 para k ← 1 até n faça
4    para i ← 1 até n faça
5       para j ← 1 até n faça
6          d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
7 devolva D^(n)
```

Análise de complexidade

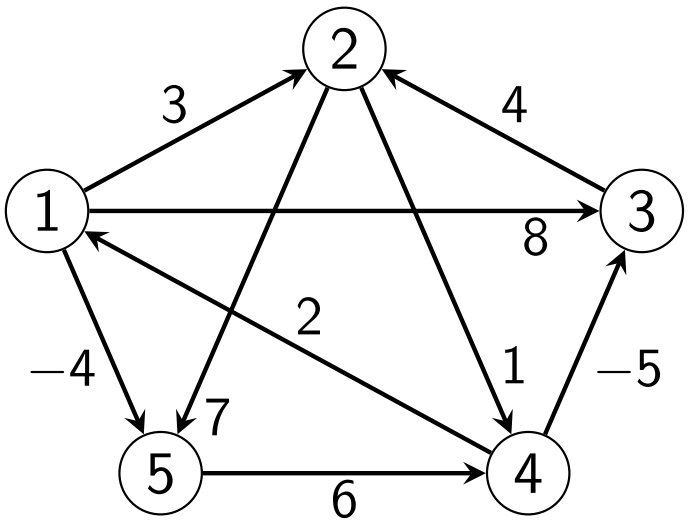
Cada execução da linha 6 demora $O(1)$. A linha 6 é executada n^3 vezes. Portanto, o tempo de execuão do algoritmo é $\Theta(n^3) = O(V^3)$.

Um exemplo



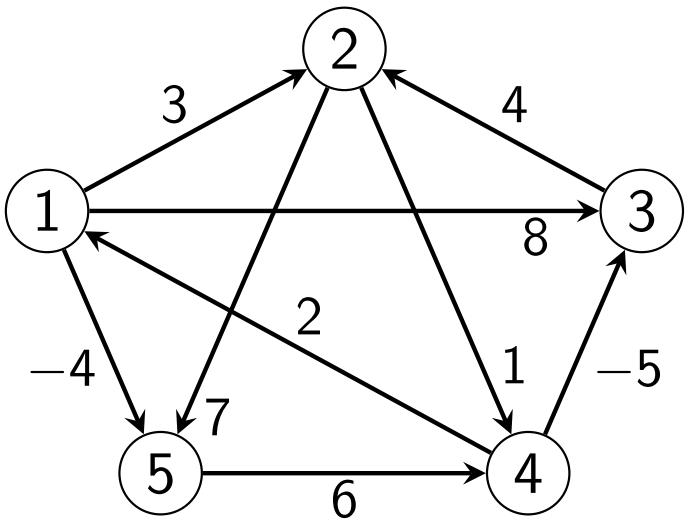
$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Um exemplo



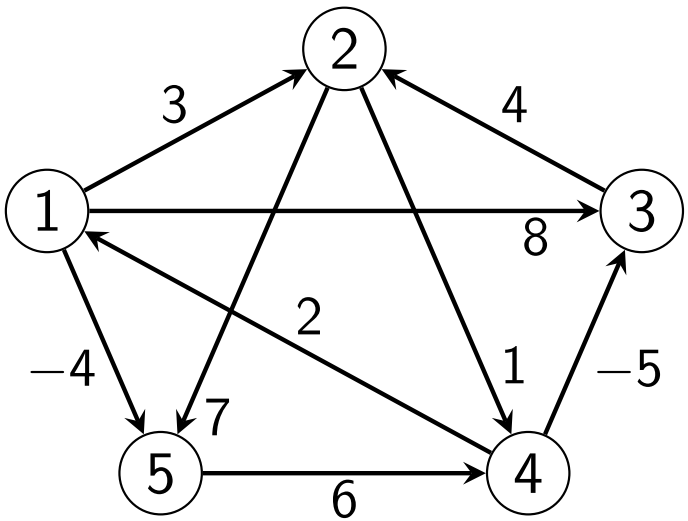
$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Um exemplo



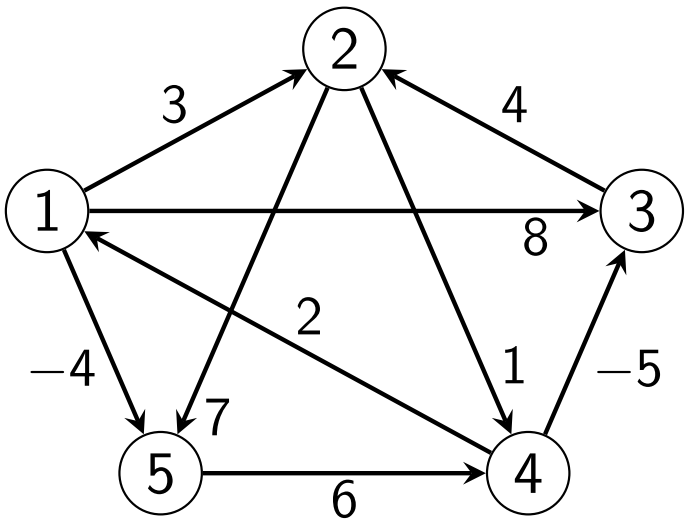
$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Um exemplo



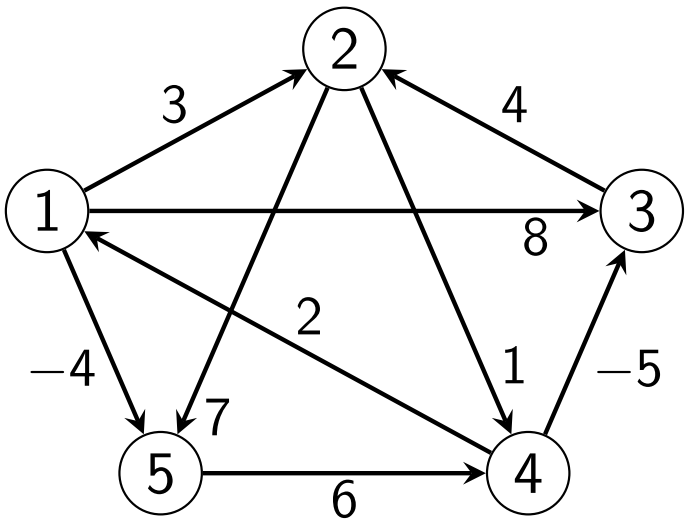
$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Um exemplo



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

Um exemplo



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

Exercícios

Exercício 1

Informe como podemos construir a solução ótima (matriz de predecessores $\Pi_{n \times n}$) e obter o caminho mínimo entre dois pares de vértices.

Exercício 2

Dado um grafo orientado $G = (V, E)$ com um conjunto de vértices $V = \{1, \ldots, n\}$, queremos saber se G tem um caminho de i a j para quaisquer pares de vértices $i, j \in V$. Definimos o **fecho transitivo** de G como sendo o grafo $G^* = (V, E^*)$, onde $E^* = \{(i, j) : \text{ existe um caminho de } i \text{ a } j \text{ em } G\}$. Mostre como podemos modificar o algoritmo de Floyd-Warshall para computar o fecho transitivo de maneira eficiente.