Caminhos mínimos de todos os pares
5189-32

Rodrigo Calvo
rcalvo@uem.br

Departamento de Informática – DIN
Universidade Estadual de Maringá – UEM

1° semestre de 2016

Introdução

- O problema de **caminho mínimo de todos os pares** consiste em encontrar o caminho mínimo entre todos os pares de vértices de um grafo
- Os algoritmos que resolvem este problema recebem como entrada
  - Um grafo direcionado $G = (V, A)$
  - Uma função peso $w : A \rightarrow \mathbf{R}$

Introdução

- Para resolver o problema de **caminho mínimo de todos os pares**, pode-se utilizar algoritmos de caminho mínimo de única origem. Neste caso, o algoritmo é executado $|V|$ vezes, uma vez para cada vértice

Algoritmo	Fila de Prioridade	*Única origem*	Todos os pares
Dijkstra (sem arestas de pesos negativos	Arranjo Simples	$O(V^2 + A)$	$O(V^3 + VA) = O(V^3)$
	Heap	$O(A \lg V)$	$O(VA \lg V)$ $O(V^3 \lg V)$ - grafo denso
	Heap de Fibonacci	$O(V \lg V + A)$	$O(V^2 \lg V + VA)$ $O(V^3)$ – grafo denso
Bellman-Ford (grafos gerais)		$O(VA)$	$O(V^2A)$ $O(V^4)$ – grafo denso

Introdução

- Entrada
  - Uma matriz $W_{n \times n}$ que representa os pesos das arestas. Isto é, $W = w_{ij}$, onde

$$w_{ij} = \begin{cases} 0 & , \text{se } i = j \\ w(i, j) & , \text{se } i \neq j \text{ e } (i, j) \in A \\ \infty & , \text{se } i \neq j \text{ e } (i, j) \notin A \end{cases}$$

- Saída
  - Matriz $D_{n \times n} = d_{ij}$, onde a entrada d_{ij} contém o peso do caminho mínimo do vértice i até o vértice j, ou seja, $d_{ij} = \delta(i, j)$
  - Matriz predecessora $\Pi_{n \times n} = \pi_{ij}$, onde π_{ij} é o vértice predecessor de j em um caminho mínimo a partir de i

Introdução

- Algoritmos de caminhos mínimos de todos os pares
  - Floyd-Warshall
  - Baseado em multiplicação de matrizes
  - Johnson (grafos esparsos)

Introdução

- Considerações
  - Supomos que não existem ciclos de pesos negativos
  - Os vértices estão numerados como $1, 2, 3, \ldots, n$, onde $n = |V|$

Algoritmo de Floyd-Warshall

- Algoritmo de programação dinâmica com tempo $\Theta(V^3)$

- Ideia
  - O caminho mínimo pode ser calculado baseado nos caminhos mínimos para subproblemas já calculados e memorizados

- Etapas para resolver um problema com programação dinâmica
  - Caracterizar a estrutura de uma solução ótima
  - Definir recursivamente o valor da solução ótima
  - Computar o valor da solução ótima de baixo para cima (*bottom-up*)
  - Construir a solução ótima a partir das informações computadas

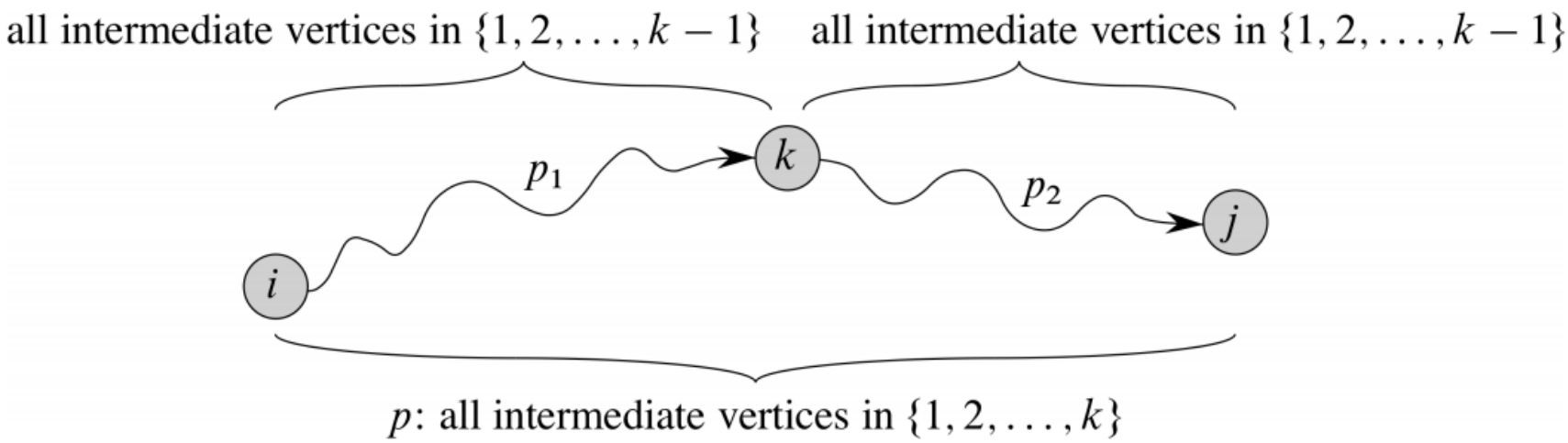
Algoritmo de Floyd-Warshall

- Para um caminho $p = \langle v_0, v_1, \ldots, v_l \rangle$, um vértice intermediário é qualquer vértice de p que não seja v_0 ou v_l
- Nota-se que $n = |V|$

Caracterização da estrutura de solução 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 $\{1, 2, \ldots, k - 1\}$. Deste modo, um caminho mínimo $i \rightsquigarrow j$ com todos os vértices intermediários no conjunto $\{1, 2, \ldots, k - 1\}$, também é um caminho mínimo $i \rightsquigarrow j$ com todos os vértices intermediários no conjunto $\{1, 2, \ldots, k\}$
  - Se k é um vértice intermediário do caminho p, então desmembra-se o caminho p em $i \overset{p_1}{\rightsquigarrow} k \overset{p_2}{\rightsquigarrow} j$. $p1$ é um caminho mínimo de i até k, com todos os vértices intermediários no conjunto $\{1, 2, \ldots, k - 1\}$. A mesma ideia se aplica a $p2$

Caracterização da estrutura de solução ótima



Definição recursiva do custo da solução ótima

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

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

- Observe que a matriz $D^{(n)} = d_{ij}^{(n)}$ fornece a reposta desejada:

  $d_{ij}^{(n)} = \delta(i, j)$ para todo *i, j* ∈ *V*, isto porque para qualquer caminho todos os vértices intermediários estão no conjunto {*1, 2, . . . , n*}

Algoritmo de Floyd-Warshall

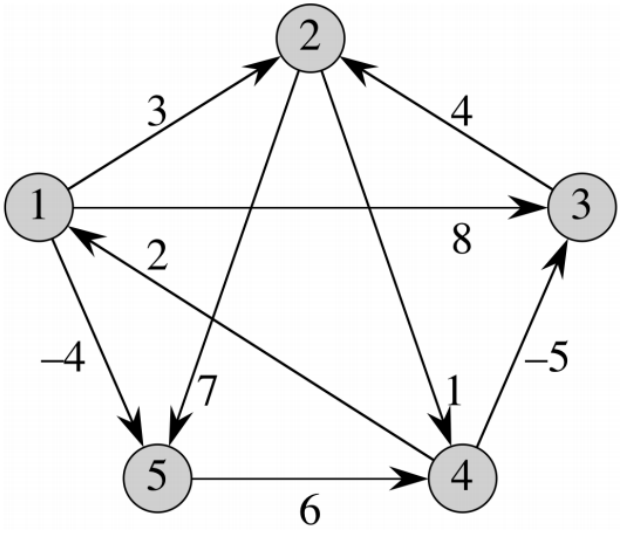
```
floyd-warshall(W)
1 n = W .linhas
2 D^(0) = W
3 for k = 1 to n
4   seja D^(k) = (d^(k)_ij) uma matriz n × n
5   for i = 1 to n
6     for j = 1 to n
7       d^(k)_ij = min(d^(k-1)_ij , d^(k-1)_ik + d^(k-1)_kj )
8 return D^(n)
```

Algoritmo de Floyd-Warshall

```
floyd-warshall(W)
1 n = W .linhas
2 D(0) = W
3 for k = 1 to n
4   seja D(k) = (d(k)ij) uma matriz n × n
5   for i = 1 to n
6     for j = 1 to n
7       d(k)ij = min(d(k-1)ij , d(k-1)ik + d(k-1)kj )
8 return D(n)
```

- Análise do tempo de execução
  - Cada execução da linha 7 demora O(*1*)
  - A linha 7 é executada n^3 vezes
  - Portanto, o tempo de execução do algoritmo é $\Theta(n^3) = \Theta(V^3)$

Algoritmo de Floyd-Warshall



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

Algoritmo de Floyd-Warshall

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

Algoritmo de Floyd-Warshall

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

Algoritmo de Floyd-Warshall

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

Algoritmo de Floyd-Warshall

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

Algoritmo de Floyd-Warshall

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

Algoritmo de Floyd-Warshall

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

Construção da solução ótima

- Como construir um caminho mínimo
  - Calcular a matriz predecessora *Π*, durante o cálculo da matriz de distância de caminhos mínimos *D*
  - Quando *k = 0*, um caminho mínimo de *i* até *j* não tem nenhum vértice intermediário, então:

$$\pi_{ij}^{(0)} = \begin{cases} \text{nil} & \text{, se } i = j \text{ ou } w_{ij} = \infty \\ i & \text{, se } i \neq j \text{ ou } w_{ij} < \infty \end{cases}$$

  - Quando *k ≥ 1*, então:

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

Construção da solução ótima

```
floyd-warshall( W )
1  n = W .linhas
2  D = W
3  Π = ( π_ij ) uma matriz n × n
4  for i = 1 to n
5    for j = 1 to n
6      if i = j ou w_ij = ∞
7        π_ij = nil
8      if i ≠ j e w_ij < ∞
9        π_ij = i
10 for k = 1 to n
11   for i = 1 to n
12     for j = 1 to n
13       if d_ij > d_ik + d_kj
14         d_ij = d_ik + d_kj
15         π_ij = π_kj
16 return D , Π
```

Bibliografia

- Thomas H. Cormen et al. Introduction to Algorithms. 3rd edition. Capítulo 25.