

Grafos

Caminhos mais curtos de todos os pares

Notas

Conteúdo

Introdução

Algoritmos

- Baseado em multiplicação de matrizes
- Algoritmo de Floyd-Warshall
- Algoritmo de Johnson para grafos esparsos

Referências

Notas

Introdução

- ▶ Dado um grafo orientado $G = (V, E)$ e uma função peso $w : E \rightarrow \mathbb{R}$, queremos encontrar, para todo par de vértices $u, v \in V$, o caminho mais curto (de peso mínimo) de u até v

3 / 20

Notas

Introdução

- ▶ Podemos resolver este problema aplicando um algoritmo de caminhos mais curtos de única origem $|V|$ vezes, uma vez para cada vértice
 - ▶ Dijkstra (sem arestas com pesos negativos)
 - ▶ Arranjo linear: $O(V^3 + VE) = O(V^3)$
 - ▶ Heap binário: $O(VE \lg V)$, se o grafo é denso $O(V^3 \lg V)$
 - ▶ Heap de fibonacci: $O(V^2 \lg V + VE)$, se o grafo é denso $O(V^3)$
 - ▶ Bellman-Ford (grafos gerais)
 - ▶ $O(V^2 E)$, se o grafo é denso $O(V^4)$
- ▶ Veremos um algoritmo $O(V^3)$ que não usa nenhuma estrutura de dados especial

4 / 20

Notas

Introdução

- ▶ Considerações
 - ▶ Supomos que não existem ciclos de pesos negativos
 - ▶ Os vértices estão numerados como $1, 2, 3, \dots, n$, onde $n = |V|$
- ▶ Entrada
 - ▶ Uma matriz $Wn \times n$ 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 $Dn \times n = d_{ij}$, onde a entrada d_{ij} contém o peso do caminho mais curto 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 mais curto a partir de i

5 / 20

Notas

Baseado em multiplicação de matrizes

- ▶ Não estudaremos este algoritmo

6 / 20

Notas

O algoritmo de Floyd-Warshall

- ▶ Algoritmo de programação dinâmica com tempo $\Theta(V^3)$
- ▶ Para um caminho $p = (v_1, v_2, \dots, v_l)$, um **vértice intermediário** é qualquer vértice de p que não seja v_1 ou v_l

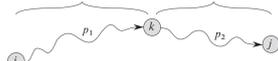
7 / 20

Notas

O algoritmo de Floyd-Warshall

- ▶ Considere um menor caminho $i \xrightarrow{p} j$ com todos os vértices intermediários em $\{1, 2, \dots, k\}$
 - ▶ Se k não é um vértice intermediário de p , então, todos os vértices intermediários de p estão em $\{1, 2, \dots, k-1\}$. Deste modo, um caminho mais curto $i \rightsquigarrow j$ com todos os vértices intermediários no conjunto $\{1, 2, \dots, k-1\}$, também é um caminho mais curto $i \rightsquigarrow j$ com todos os vértices intermediários no conjunto $\{1, 2, \dots, k\}$
 - ▶ Se k é um vértice intermediário do caminho p , então desmembramos o caminho p em $i \xrightarrow{p_1} k \xrightarrow{p_2} j$. p_1 é um caminho mais curto de i até k , com todos os vértices intermediários no conjunto $\{1, 2, \dots, k-1\}$. A mesma ideia se aplica a p_2

all intermediate vertices in $\{1, 2, \dots, k-1\}$ all intermediate vertices in $\{1, 2, \dots, k-1\}$



p : all intermediate vertices in $\{1, 2, \dots, k\}$

8 / 20

Notas

O algoritmo de Floyd-Warshall

- ▶ Algoritmo de programação dinâmica
 - ▶ O menor caminho pode ser calculado baseado nos menores caminhos para subproblemas já calculados e memorizados
 - ▶ Subproblemas ótimos
 - ▶ Superposição de subproblemas
- ▶ Formulação
 - ▶ Seja $d_{ij}^{(k)}$ o peso de um menor caminho $i \rightsquigarrow j$ com todos os vértices intermediários em $\{1, 2, \dots, k\}$

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

- ▶ Considerando-se que, para qualquer caminho, todos os vértices intermediários estão no conjunto $\{1, 2, \dots, n\}$, a matriz $D^{(n)} = (d_{ij}^{(n)})$ fornece a resposta desejada: $d_{ij}^{(n)} = \delta(i, j)$ para todo $i, j \in V$

9 / 20

Notas

O algoritmo de Floyd-Warshall

floyd-warshall(W)

```

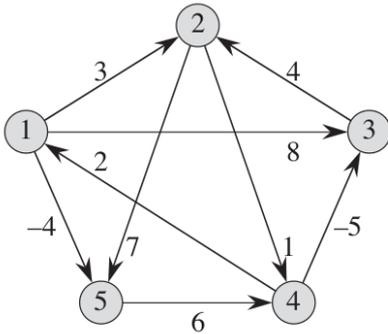
1 n = W.linhas
2 D(0) = W
3 for k = 1 to n
4   for i = 1 to n
5     for j = 1 to n
6       dij(k) = min(dij(k-1), dik(k-1) + dkj(k-1))
7 return D(n)
    
```

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

10 / 20

Notas

O algoritmo de Floyd-Warshall



11 / 20

Notas

O algoritmo de Floyd-Warshall

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

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

12 / 20

Notas

O algoritmo de Floyd-Warshall

- ▶ Como construir um caminho mais curto
 - ▶ Calcular a matriz predecessora Π , durante o cálculo da matriz de distância de caminhos mais curtos D
 - ▶ Quando $k = 0$, um caminho mais curto de i até j não tem nenhum vértice intermediário, então

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

- ▶ Quando $k \geq 1$

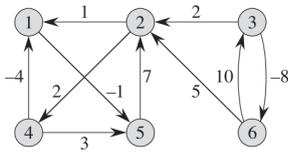
$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{se } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ \pi_{kj}^{(k-1)} & \text{se } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}$$

17 / 20

Notas

Exercícios

- 25.2-1 Execute o algoritmo de Floyd-Warshall sobre o grafo orientado ponderado da figura 25.2. Mostre a matriz $D^{(k)}$ que resulta de cada iteração do laço externo.



- 25.2-3 Modifique o procedimento floyd-warshall para incluir o cálculo das matrizes $\Pi^{(k)}$ de acordo com as equações 25.6 e 25.7.
- 25.2-6 De que modo a saída do algoritmo de Floyd-Warshall pode ser usada para detectar a presença de um ciclo de peso negativo?

18 / 20

Notas

Algoritmo de Johnson para grafos esparsos

- ▶ Não estudaremos este algoritmo

19 / 20

Notas

Referências

- ▶ Thomas H. Cormen et al. Introdução a Algoritmos. 2ª edição em português. Capítulo 25.

20 / 20

Notas
