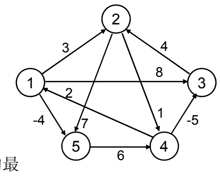
算法设计与分析
Algorithms Design & Analysis

第十三讲：全成对最短路径

华中科技大学软件学院
邱德红　主讲

全成对最短路径(All-Pairs Shortest Paths)

- **Given:(输入)**
  - Directed graph $G = (V, E)$ (有向图)
  - Weight function $w : E \to \mathbf{R}$ (权)
- **Compute:** (计算)
  - The shortest paths between all pairs of vertices in a graph (图中任何点对之间的最短距离)
  - Representation of the result: an $n \times n$ matrix of shortest-path distances $\delta(u, v)$ (结果表示: $n \times n$矩阵, 元素是对应的点对之间的最短距离$\delta(u, v)$)



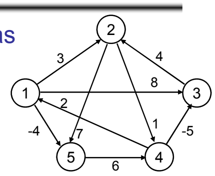
全成对最短路径解决方案(All-Pairs Shortest Paths – Solutions)

- Run BELLMAN-FORD once from each vertex:(对于每一个顶点, 运行BELLMAN-FORD(单源最短))
  - $O(V^2E)$, which is $O(V^4)$ if the graph is dense　　($E = \Theta(V^2)$)

  (计算开销: $O(V^2E)$, 甚至 $O(V^4)$)
- If no negative-weight edges, could run Dijkstra's algorithm once from each vertex: (如果不存在负权值边, 可利用Dijkstra's算法计算单源最短)
  - $O(VE\lg V)$ with binary heap, $O(V^3\lg V)$ if the graph is dense (计算开销: $O(VE\lg V)$, 甚至 $O(V^3\lg V)$: 对于边稠密的图)
- We can solve the problem in $O(V^3)$, with no elaborate data structures. (可以设计$O(V^3)$开销的全成对最短路径算法，且不需要特殊的数据结构)

全成对最短路径(All-Pairs Shortest Paths)

- Assume the graph (G) is given as adjacency matrix of weights
  - $W = (w_{ij})$, $n \times n$ matrix, $|V| = n$
  - Vertices numbered 1 to n



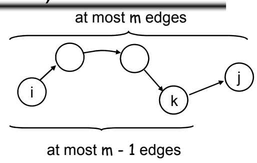
$$w_{ij} = \begin{cases} 0 & \text{if } i = j \\ \text{weight of } (i, j) & \text{if } i \neq j, (i, j) \in E \\ \infty & \text{if } i \neq j, (i, j) \notin E \end{cases}$$

(邻接矩阵表示, $n \times n$矩阵, 元素值如上定义)

- Output the result in an $n \times n$ matrix(输出$n \times n$矩阵$D = (d_{ij})$) $D = (d_{ij})$, where $d_{ij} = \delta(i, j)$
- Solve the problem using dynamic programming (动态规划?)

最短路径的优化结构(Optimal Substructure of a Shortest Path)

- All subpaths of a shortest path are shortest paths (最短路径的部分路径必然是最短的)

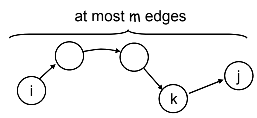


- Let p: a shortest path p from vertex i to j that contains at most m edges (p是从i到j至多含有m条边的最短路径)
- If $i = j$ (如果$i = j$, $w(p) = 0$)
  - $w(p) = 0$ and p has no edges
- If $i \neq j$: $p = i \overset{p'}{\leadsto} k \to j$
  - p' has at most m-1 edges(p'至多有m-1条边)
  - p' is a shortest path(p'是最短的)

$$\delta(i, j) = \delta(i, k) + w_{kj}$$

递归解(Recursive Solution)

- $l_{ij}^{(m)}$ = weight of shortest path $i \leadsto j$ that contains at most m edges($l_{ij}^{(m)}$从i到j至多含有m条边的最短路径的权)



- $m = 0$: $l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{if } i \neq j \end{cases}$
- $m \geq 1$: $l_{ij}^{(m)} = \min\limits_{1 \leq k \leq n} \{l_{ik}^{(m-1)} + w_{kj}\}$
  - Shortest path from i to j with at most m – 1 edges(从i到j的最短路径含有m – 1条边)
  - Shortest path from i to j containing at most m edges, considering all possible predecessors (k) of j (从i到j的最段路径含有m 条边, 考虑所有j的先前顶点k的情况)

计算最短路径(Computing the Shortest Paths)

- $m = 1$: $l_{ij}^{(1)} = w_{ij}$ $L^{(1)} = W$
  - The path between i and j is restricted to 1 edge (i到j只有1条边)
- Given $W = (w_{ij})$, compute: $L^{(1)}, L^{(2)}, \ldots, L^{(n-1)}$, where

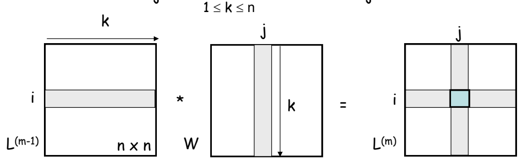
  $L^{(m)} = (l_{ij}^{(m)})$ (给出$W = (w_{ij})$, 计算$L^{(1)}, L^{(2)}, \ldots, L^{(n-1)}$)
- $L^{(n-1)}$ contains the actual shortest-path weights ($L^{(n-1)}$包含最短路径)

  Given $L^{(m-1)}$ and $W \Rightarrow$ compute $L^{(m)}$ (计算过程: 给出$L^{(m-1)}$和W, 求$L^{(m)}$)
  - Extend the shortest paths computed so far by one more edge (每次给当前的最短路径增加一条边)
- If the graph has no negative cycles: all simple shortest paths contain at most n - 1 edges (如果没有负权回路, 所有的最短路径至多含有n - 1条边, 因此有)

  $\delta(i, j) = l_{ij}^{(n-1)}$ and $l_{ij}^{(n)}, l_{ij}^{(n+1)} \ldots = l_{ij}^{(n-1)}$

最短路径的延伸(Extending the Shortest Path)

$$l_{ij}^{(m)} = \min_{1 \le k \le n} \{l_{ik}^{(m-1)} + w_{kj}\}$$



Replace: min → + , + → •

Computing $L^{(m)}$ looks like matrix multiplication (计算与矩阵乘法相似)

延伸算法:EXTEND(L, W, n)

1. create L', an n × n matrix
2. **for** i ← 1 **to** n
3. **do for** j ← 1 **to** n
4. **do** $l_{ij}' \leftarrow \infty$
5. **for** k ← 1 **to** n
6. **do** $l_{ij}' \leftarrow \min(l_{ij}', l_{ik} + w_{kj})$
7. **return** L'

$$l_{ij}^{(m)} = \min_{1 \le k \le n} \{l_{ik}^{(m-1)} + w_{kj}\}$$

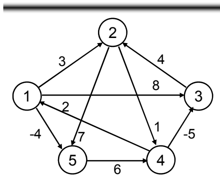
Running time: $\Theta(n^3)$

成对最短路径算法:SLOW-ALL-PAIRS-SHORTEST-PATHS(W, n)

1. $L^{(1)} \leftarrow W$
2. **for** m ← 2 **to** n - 1
3. **do** $L^{(m)} \leftarrow$ EXTEND ($L^{(m-1)}$, W, n)
4. **return** $L^{(n-1)}$

Running time: $\Theta(n^4)$

$$l_{ij}^{(m)} = \min_{1 \le k \le n} \{l_{ik}^{(m-1)} + w_{kj}\}$$



$L^{(m-1)} = L^{(1)}$

0	3	8	∞	-4
∞	0	∞	1	7
∞	4	0	∞	∞
2	∞	-5	0	∞
∞	∞	∞	6	0

W

0	3	8	∞	-4
∞	0	∞	1	7
∞	4	0	∞	∞
2	∞	-5	0	∞
∞	∞	∞	6	0

$L^{(m)} = L^{(2)}$

0	3	8	2	-4
3	0	-4	1	7
∞	4	0	5	11
2	-1	-5	0	-2
8	∞	1	6	0

… and so on until $L^{(4)}$

改进运行时间(Improving Running Time)

- No need to compute all $L^{(m)}$ matrices(不必计算所有$L^{(m)}$)
- If no negative-weight cycles exist: (如果没有负回路, 则有:)

  $L^{(m)} = L^{(n-1)}$ for all $m \ge n - 1$
- We can compute $L^{(n-1)}$ by computing the sequence: (因此可以通过计算下列序列来计算$L^{(n-1)}$)

  $L^{(1)} = W$ $L^{(2)} = W^2 = W \bullet W$

  $L^{(4)} = W^4 = W^2 \bullet W^2$ $L^{(8)} = W^8 = W^4 \bullet W^4 \ldots$

$$\Rightarrow 2^x = n-1$$

$$L^{(n-1)} = W^{2^{\lceil \lg(n-1) \rceil}}$$

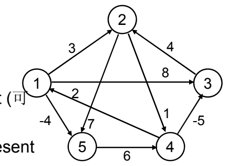
快速算法:FASTER-APSP(W, n)

1. $L^{(1)} \leftarrow W$
2. $m \leftarrow 1$
3. **while** $m < n - 1$
4. **do** $L^{(2m)} \leftarrow$ EXTEND($L^{(m)}$, $L^{(m)}$, n)
5. $m \leftarrow 2m$
6. **return** $L^{(m)}$

- OK to overshoot: products don't change after $L^{(n-1)}$ ($L^{(n-1)}$后积不变化)
- **Running Time:** $\Theta(n^3 \lg n)$ (运行开销)

Floyd-Warshall 算法

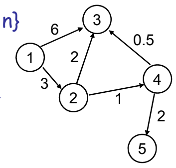
- **Given:**给定
  - Directed, weighted graph G = (V, E) (有向加权图G = (V, E))
  - Negative-weight edges may be present (可以存在负权边)
  - No negative-weight cycles could be present in the graph (但不能出现负权回路)
- **Compute:** 计算
  - The shortest paths between all pairs of vertices in a graph (点对之间的最短距离)



最短路径结构(The Structure of a Shortest Path)

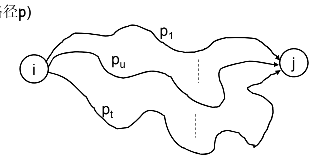
- Vertices in G are given by $V = \{1, 2, \ldots, n\}$ (图G中的顶点)
- Consider a path $p = \langle v_1, v_2, \ldots, v_l \rangle$ (对于路径$p = \langle v_1, v_2, \ldots, v_l \rangle$)
  - An **intermediate** vertex of p is any vertex in the set $\{v_2, v_3, \ldots, v_{l-1}\}$ (p的中间顶点是集合$\{v_2, v_3, \ldots, v_{l-1}\}$中的点)
  - *E.g.:* $p = \langle 1, 2, 4, 5 \rangle$: {2, 4}

    $p = \langle 2, 4, 5 \rangle$: {4}



最短路径结构(The Structure of a Shortest Path)

- For any pair of vertices i, j ∈ V, consider all paths from i to j whose intermediate vertices are all drawn from a subset {1, 2, …, k} (对于任何点对i和j, 考虑所有从i到j的路径, 这些路径的中间节点来自集合{1, 2, …, k})
  - Find p, a minimum-weight path from these paths (从这些路径中确定最短路径p)

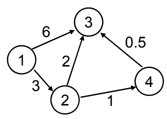


No vertex on these paths has index > k
这些路径不存在标号> k的顶点

Example

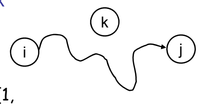
$d_{ij}^{(k)}$ = the weight of a shortest path from vertex i to vertex j with all intermediary vertices drawn from {1, 2, …, k} (dij(k)：从i到j的路径的权值, 中间节点来自集合{1, 2, …, k})

- $d_{13}^{(0)} =$ 6
- $d_{13}^{(1)} =$ 6
- $d_{13}^{(2)} =$ 5
- $d_{13}^{(3)} =$ 5
- $d_{13}^{(4)} =$ 4.5

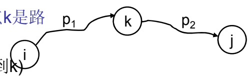


最短路径结构(The Structure of a Shortest Path)

- k is not an intermediate vertex of path p (顶点k不是路径p的中间节点)
  - Shortest path from i to j with intermediate vertices from {1, 2, …, k} is a shortest path from i to j with intermediate vertices from {1, 2, …, k - 1} (从i到j的中间节点来自{1, 2, …, k}的最短路径即是从i到j的中间节点来自{1, 2, …, k - 1}的最短路径)



- k is an intermediate vertex of path p(顶点k是路径p的中间节点)
  - p_1 is a shortest path from i to k(p_1从i到k)
  - p_2 is a shortest path from k to j (p_2从k到j)
  - k is not intermediary vertex of p_1, p_2(k不是p_1, p_2的中间节点)
  - p_1 and p_2 are shortest paths (是最短路径)



递归解:A Recursive Solution (cont.)

$d_{ij}^{(k)}$ = the weight of a shortest path from vertex i to vertex j with all intermediary vertices drawn from {1, 2, …, k} (从i到j的路径的权值, 中间节点来自集合{1, 2, …, k})

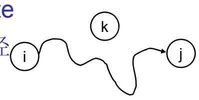
- $k = 0$
- $d_{ij}^{(k)} = w_{ij}$

没有中间节点

递归解: A Recursive Solution (cont.)

$d_{ij}^{(k)}$ = the weight of a shortest path from vertex i to vertex j with all intermediary vertices drawn from {1, 2, …, k} (从i到j的路径的权值, 中间节点来自集合{1, 2, …, k})

- $k \geq 1$ (有中间节点)
- **Case 1:** k is not an intermediate vertex of path p (顶点k不是路径p的中间节点)

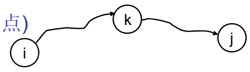


$d_{ij}^{(k)} = d_{ij}^{(k-1)}$

递归解: A Recursive Solution (cont.)

$d_{ij}^{(k)}$ = the weight of a shortest path from vertex i to vertex j with all intermediary vertices drawn from {1, 2, …, k} (从i到j的路径的权值, 中间节点来自集合{1, 2, …, k})

- $k \geq 1$
- **Case 2:** k is an intermediate vertex of path p(顶点k是路径p的中间节点)

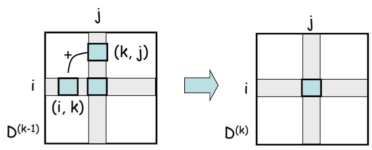


$d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$

计算方法(Computing the Shortest Path Weights)

- $d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 \\ \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\} & \text{if } k \geq 1 \end{cases}$
- The final solution: $D^{(n)} = (d_{ij}^{(n)})$: (最终解)

  $d_{ij}^{(n)} = \delta(i, j)\ \forall\ i, j \in V$

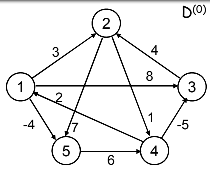


算法:FLOYD-WARSHALL(W)

1. $n \leftarrow$ rows[W]
2. $D^{(0)} \leftarrow W$
3. **for** $k \leftarrow 1$ **to** n
4.   **do for** $i \leftarrow 1$ **to** n
5.   **do for** $j \leftarrow 1$ **to** n
6.   **do** $d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
7. **return** $D^{(n)}$

- Running time: $\Theta(n^3)$

$d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$



$D^{(0)} = W$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

$D^{(1)}$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$D^{(2)}$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$D^{(3)}$

0	3	8	4	-4
∞	0	∞	1	7
∞	4	0	5	11
2	-1	-5	0	-2
∞	∞	∞	6	0

$D^{(4)}$

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

……