Part VI

图算法

图算法

- 基本图算法
  - 图的表示
  - **BFS，DFS**
  - 拓扑排序
- 单源最短路径
  - 寻找从一个给定的源顶点到所有其他顶点的最短路径。
- 全对最短路径
  - 计算每对顶点之间的最短路径。
- 最大流

25 全对最短路径

- 怎么找到途中所有成对顶点间最短路径。
- 在为地图集制作所有城市对之间的距离表时，可能会出现这个问题 。
- 我们可以通过运行单源最短路径算法|V|次来解决全对最短路径问题，每个顶点依次作为源点。

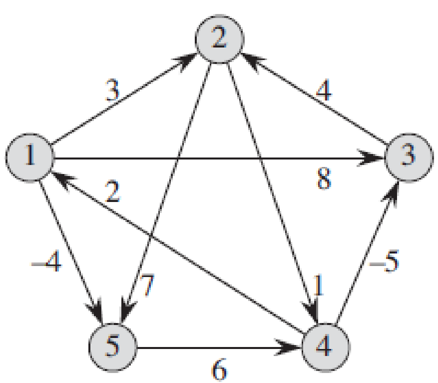
25 全对最短路径

- 本章中的大多数算法都使用邻接矩阵表示。
- 输入是 $n \times n$ 矩阵 $\boldsymbol{W}$ 表示n顶点有向图的边权值 $G = (V, E)$，其中

$$w_{ij} = \begin{cases} 0 & \text{if } i = j , \\ \text{the weight of directed edge } (i, j) & \text{if } i \neq j \text{ and } (i, j) \in E , \\ \infty & \text{if } i \neq j \text{ and } (i, j) \notin E . \end{cases} \qquad (25.1)$$

- **最短路径权值**: 输出是 $n \times n$ 矩阵 $\boldsymbol{D} = (d_{ij})$.

  终止时 $d_{ij} = \delta(i, j)$ 。



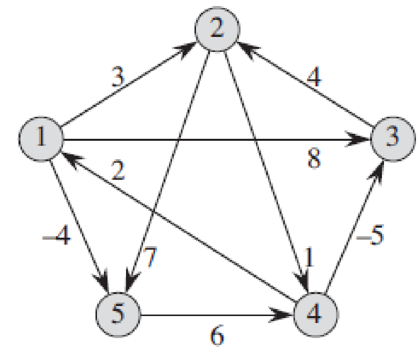
$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

25 全对最短路径

- 我们不仅要计算**最短路径权值 *D***

- 还要算前驱矩阵 : $\Pi = (\pi_{ij})$, 其中 π_{ij} 为 NIL 当 $i = j$ 或没有 i 到 j 的路径, 否则 π_{ij} 是 j 在从 i 出发的最短路径的前驱节点

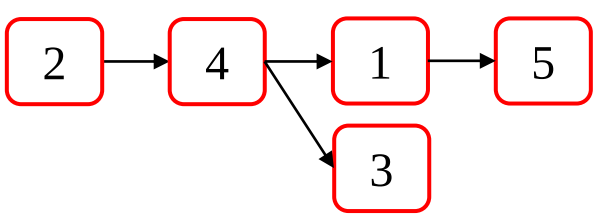
- 由Π矩阵的第 i 行引出的子图应该是根为 i 的最短路径树。例如对于第二行



$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \boldsymbol{W}$$

$$\begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \boldsymbol{D}$$

$$\begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix} \quad \boldsymbol{\Pi}$$



25.1 最短路径和矩阵乘法*

- 基于**矩阵乘法**的动态规划算法 , $\Theta(V^4)$.
- "重复平方," $\Theta(V^3 \lg V)$.

25.2 The Floyd-Warshall algorithm (DP)

最短路径的结构

- Floyd-Warshall 算法考虑最短路径的中间顶点

  简单路径 $p = <v_1, v_2, \ldots, v_l>$的端点是 v_1 和 v_l， 其他是 p 的“中间”节点)

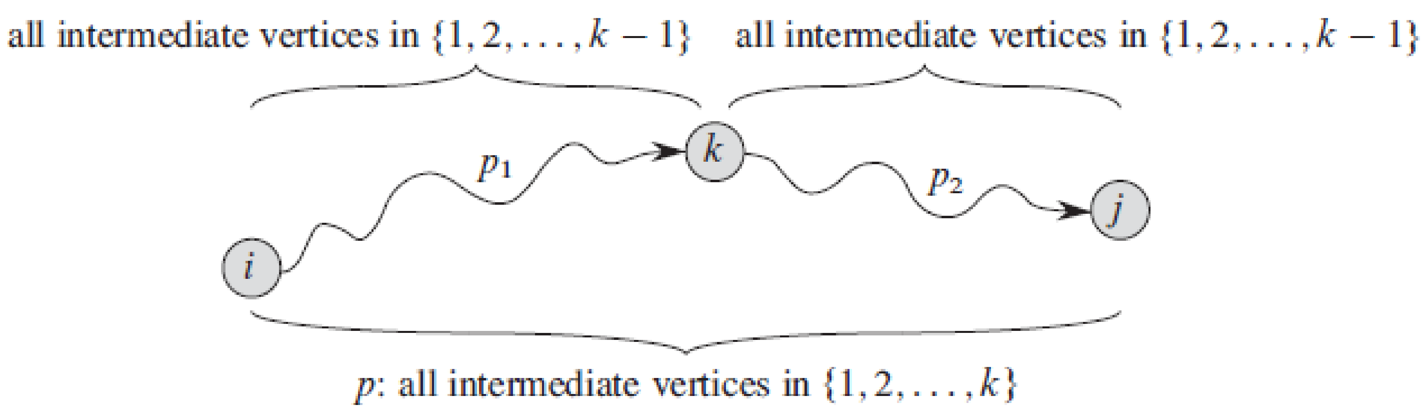
- 考虑某个 k 的顶点子集$\{1, 2, \ldots, k\}$。对于任意一对顶点 $i, j \in V$，考虑从 i 到 j 的所有中间顶点都来自$\{1, 2, \ldots, k\}$的路径，设 p 为其中的最小权值路径。(p 很简单)

25.2 The Floyd-Warshall algorithm

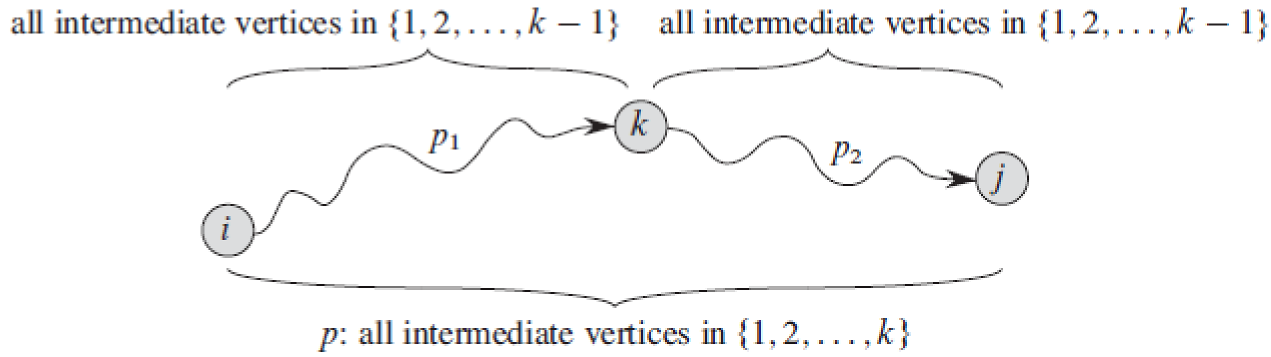
最短路径的结构

对于所有成对顶点 (i, j), 考虑从 i 到 j 的所有中间顶点都来自{1 , 2 , … , k}的路径，设 p 为其中的最小权值路径。

- **如果k不是路径p的中间顶点**, 则p的所有中间顶点都在集合{1, 2, … , k-1}中。因此，所有中间顶点在集合{1, 2, … , k-1} 中的*st-path-*$\delta(i, j)$也是所有中间顶点在集合{1, 2, … , k}中的*st-path-*$\delta(i, j)$。
- **若k是**, 把 p 分解为 $i \overset{p_1}{\leadsto} k \overset{p_2}{\leadsto} j$ 。p_1是所有中间顶点在集合{1, 2, … , k-1} 中的 *st-path-*$\delta(i, k)$ ， 同样对于 p_2 , …



25.2 The Floyd-Warshall algorithm



全对最短路径问题的递归解

令 $d_{ij}^{(k)}$ 是所以中间节点都在{1, 2, … , k}的***st-path-δ(i, j)*** 的权。 当 $k = 0$, ***path-p(i, j)***没有中间节点。这样的路径最多有一条边。 我们递归定义

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

对于任何路径, 所有中间节点在集合 {1, 2, … , n}中, 矩阵$D^{(n)} = (d_{ij}^{(n)})$ 给出了最终结果: $d_{ij}^{(n)} = \delta(i, j)$ 对于所有 $i, j \in V$.

25.2 The Floyd-Warshall algorithm

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

Direct recursion algorithm ? complexity ?

自底向上计算最短路径权值

我们可以使用下面的自底向上的过程来按k的递增顺序计算$d_{ij}^{(k)}$的值。

```
FLOYD-WARSHALL(W)
n = W.rows
D^(0) = W
for k = 1 to n
    let D^(k) = (d_ij^(k)) be a new n × n matrix
    for i = 1 to n
        for j = 1 to n
            d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
return D^(n)
```

running time?

$\Theta(n^3)$

25.2 The Floyd-Warshall algorithm

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

构造最短路径

计算一系列矩阵 $\Pi^{(0)}, \Pi^{(1)}, \ldots, \Pi^{(n)}$, 其中 $\Pi = \Pi^{(n)}$ 我们将$\pi_{ij}^{(k)}$ 定义为顶点 j 在从顶点i和集合$\{1, 2, \ldots, k\}$中所有中间顶点的最短路径的前驱。

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

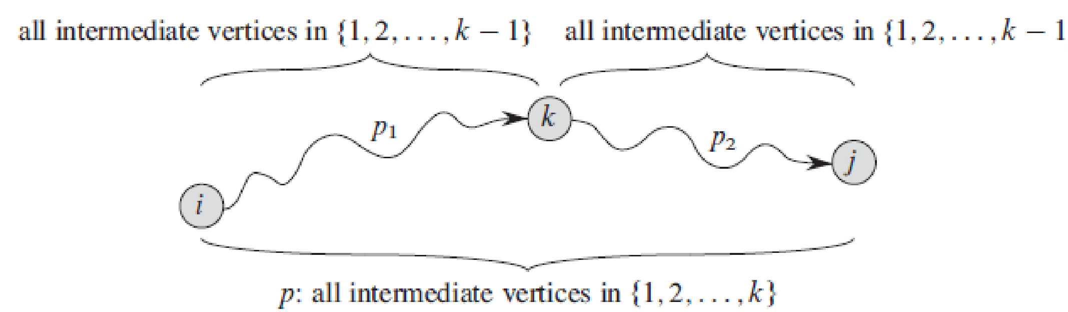
$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} , \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} . \end{cases}$$

对最短路径$i \to j$,

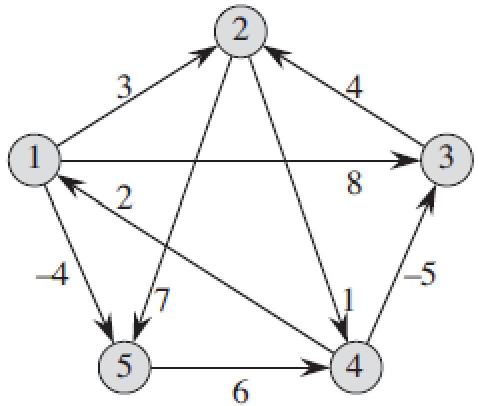
1) k 不在最短路径上，最短路径的顶点是$\{1, 2, \ldots, k\text{-}1\}$，因此$\pi^{(k)}=\pi^{(k\text{-}1)}$

2) k 在最短路径上，$i \to j$ 中 j 的前驱 $\pi_{ij}^{(k)}$ 显然就是的$k \to j$ 中 j 的前驱 $\pi_{kj}^{(k\text{-}1)}$

Exercises…



25.2 The Floyd-Warshall algorithm



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

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

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

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

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

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

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

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

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

25.2 The Floyd-Warshall algorithm

有向图的传递闭包*

25.3 Johnson’s algorithm for sparse graphs*

总结

- 队列 (Priority Queue)
- 枚举 (BFS, …)
- 递归 (DFS, …)
- 动态规划 (All-Pairs Shortest Paths)
- 贪心策略 (Single-Source Shortest Paths)
- 聚集分析
- …