Part VI

Graph Algorithms (II)

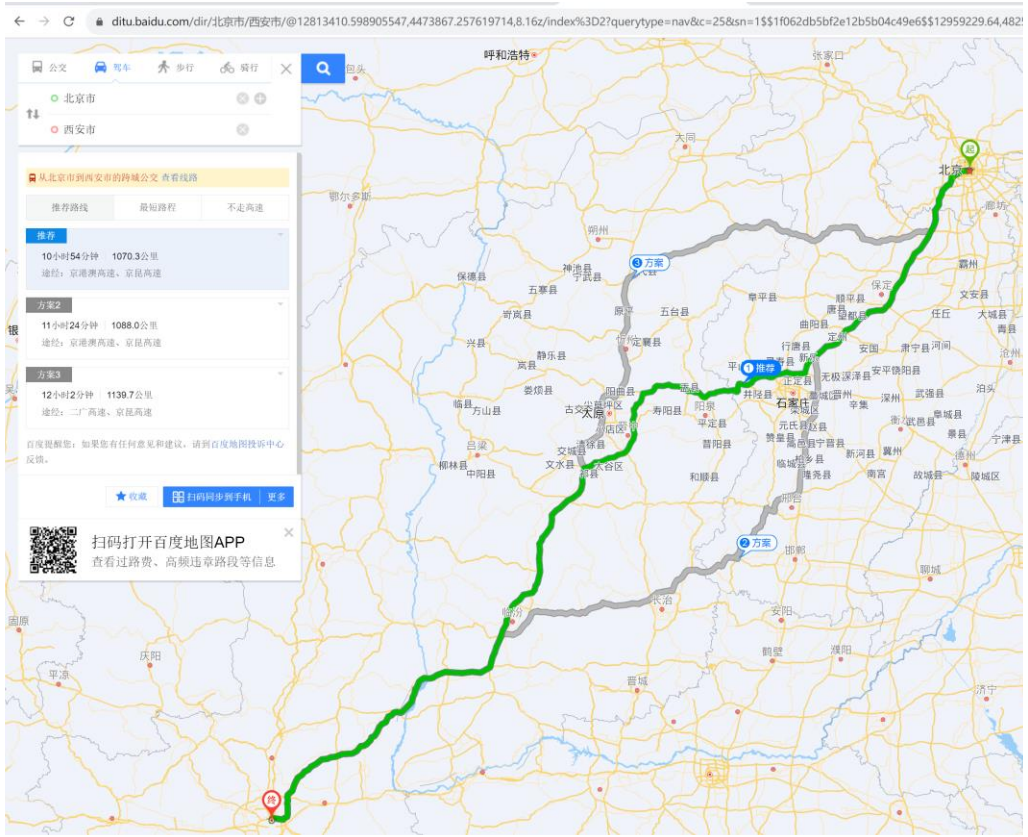
songyou@buaa.edu.cn

Graph Algorithms

- **Elementary Graph Algorithms**（图算法基础）
  - **Representations of Graphs**
  - **BFS, DFS**
  - **Sort Topologically**
- **Single-Source Shortest Paths**（最短路径问题）
  - **Finding shortest paths from a given source vertex to all other vertices.**
  - **Relaxation**（松弛）
- **All-Pairs Shortest Paths**（任意两点的最短路径问题）
  - **Computing shortest paths between every pair of vertices.**
- **Maximum Flow**（最大流）

25 All-Pairs Shortest Paths

- How to find shortest paths between all pairs of vertices in a graph.
- This problem might arise in making a table of distances between all pairs of cities for a road atlas. 制作道路地图集，求出所有城市间的距离
- We can solve an all-pairs shortest-paths problem by running a single-source shortest-paths algorithm $|V|$ times.



运行单源最短路径 n 次

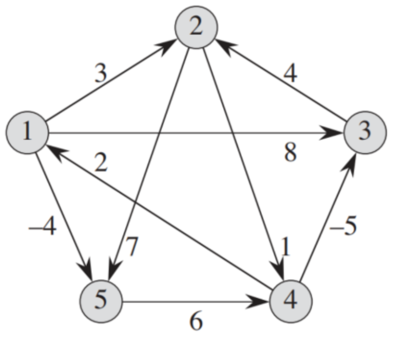
25 All-Pairs Shortest Paths

- Most of the algorithms in this chapter use an **adjacency-matrix** representation.
- The input is an $n \times n$ matrix $\boldsymbol{W}$ representing the edge weights of an n-vertex directed graph $G = (V, E)$, where

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

- **shortest-path weights:** The output is an $n \times n$ matrix $\boldsymbol{D} = (d_{ij})$.

$d_{ij} = \delta(i, j)$ at termination. （在后文，$\delta(i, j)$ 有时既指最短路径，也指最短路径距离，请根据上下文进行区分。）



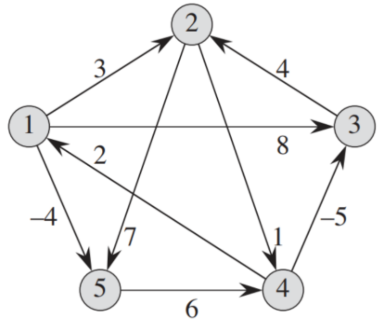
$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0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

input output

25 All-Pairs Shortest Paths

- We need to compute not only the shortest-path weights D, but also a *predecessor matrix* : $\Pi = (\pi_{ij})$, where π_{ij} is NIL if either $i = j$ or there is no path from i to j , and otherwise π_{ij} is the predecessor of j on some shortest path from i.

  不仅求最短路径矩阵 $\boldsymbol{D}$，通常也求前驱矩阵 $\boldsymbol{\Pi}$，其元素 π_{ij} 表示从 i 到 j 的最短路径中 j 的前驱节点



- The subgraph induced by the ith row of the Π matrix should be a shortest-paths tree with root i, For example, choose line 2.

  前驱矩阵 $\boldsymbol{\Pi}$ 的第 i 行，称为 $\boldsymbol{\Pi}$ 的一个诱导子图，表示以 i 为根节点的最短路径树

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

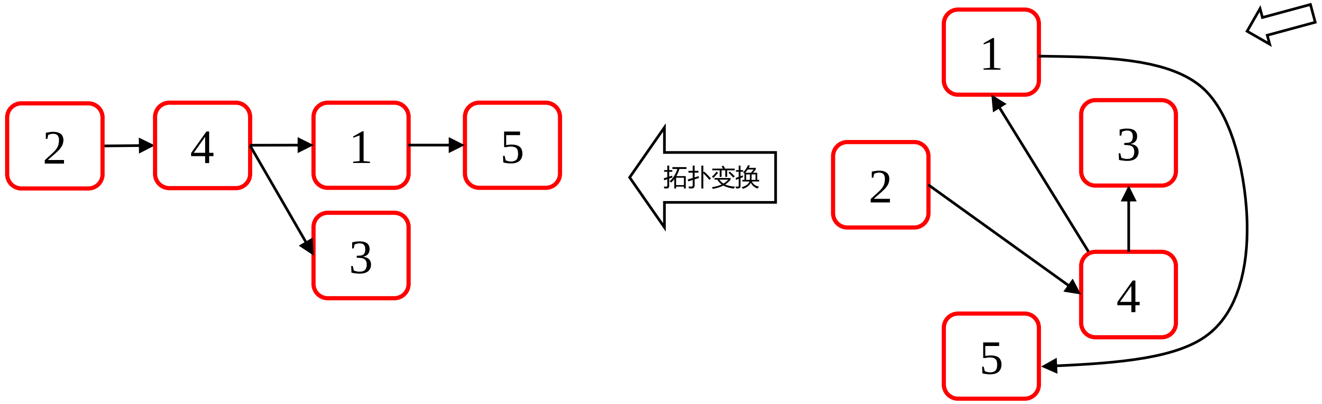
$$\begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix} \boldsymbol{\Pi}$$

2→1: 最短路径上1的前驱为4

2→3: 最短路径上3的前驱为4

2→4: 最短路径上4的前驱为2

……



*25.1 Shortest paths and matrix multiplication

A dynamic-programming algorithm based on matrix multiplication. 基于矩阵相乘求最短路径问题的DP算法

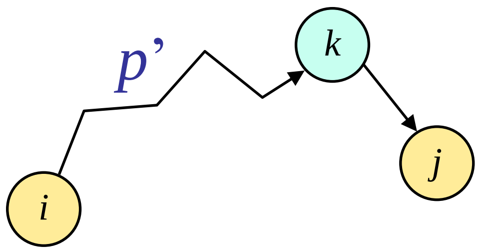
The structure of a shortest path

Vertices i and j are distinct, then we decompose shortest path p into

$$\delta(i, j) \text{ is } \underbrace{i \overset{p'}{\leadsto} k \rightarrow j}_{p}$$

then p' is a shortest path i to k, and so

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



最短路径 $\delta(i, j)$ 上，j 的前驱节点为 k，$p = \delta(i, j) = p' + w_{kj} = d(i, k) + w_{kj}$，
则 $p' = d(i, k)$ 一定是 $p' = d(i, k) = \delta(i, k)$.

p 是最短路径 ⟹ 子路径 p' 也是最短路径

*25.1 Shortest paths and matrix multiplication

A recursive solution to the all-pairs shortest-paths problem

$l_{ij}^{(m)}$: the minimum weight of any path from *i* and *j* that contains at most *m* edges.

$$l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \,, \\ \infty & \text{if } i \neq j \,. \end{cases}$$

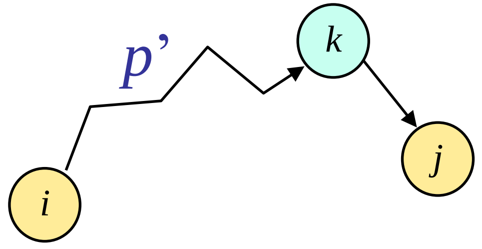
$$l_{ij}^{(m)} = \min_{1 \le k \le n} \{ l_{ik}^{(m-1)} + w_{kj} \} \,.$$

$l_{ij}^{(m)}$：从 *i* 到 *j*，包括最多 *m* 条边的最短路径 *p* 的长度（权值），其中 $p = p' + w_{kj}$.
则必有，p' 是从 *i* 到 *k*，包括最多 *m*-1 条边的最短路径，即 p' 的长度为 $l_{ik}^{(m-1)}$.

The graph($V = n$) contains no negative-weight cycles (无负环路), then

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

n 个顶点的图（无环路），任意两个顶点的最短路径的边不会超过 *n*-1 条边



*25.1 Shortest paths and matrix multiplication

Computing the shortest-path weights bottom up

Inputs : $W = (w_{ij})$

compute a series of matrices $L^{(i)}$, $i = 1, \ldots, n-1$. $L^{(m)} = (l_{kj}^{(m)})$. The final matrix $L^{(n-1)}$ contains the actual shortest-path weights.

$L^{(1)} = W$.

$l_{ij}^{(m)}$：从 i 到 j，包括最多 m 条边的最短路径 p 的长度（权值），其中 $p = p' + w_{kj}$.
则必有 p' 是从 i 到 k 包括最多 m-1 条边的最短路径，即 p' 的长度为 $l_{ik}^{(m-1)}$.

```
EXTEND-SHORTEST-PATHS(L, W)
n = L.rows
let L' = (l'_ij) be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        l'_ij = ∞
        for k = 1 to n
            l'_ij = min(l'_ij, l_ik + w_kj)
return L'
```

$$l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \,, \\ \infty & \text{if } i \neq j \,. \end{cases}$$

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

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

这个算法是一次迭代，即，从 i 到 j，已知包括最多 m-1 条边时的最短路径问题，求包括最多 m 条边时的最短路径问题。

*25.1 Shortest paths and matrix multiplication

Computing the shortest-path weights bottom up

```
EXTEND-SHORTEST-PATHS(L, W)
n = L.rows
let L' = (l'_ij) be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        l'_ij = ∞
        for k = 1 to n
            l'_ij = min(l'_ij, l_ik + w_kj)
return L'
```

$$l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \ , \\ \infty & \text{if } i \neq j \ . \end{cases}$$

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

```
SLOW-ALL-PAIRS-SHORTEST-PATHS(W)
n = W.rows
L^(1) = W
for m = 2 to n - 1
    let L^(m) be a new n × n matrix
    L^(m) = EXTEND-SHORTEST-PATHS(L^(m-1), W)
return L^(n-1)
```

循环n-2次，求得 $l_{ij}^{(n-1)}$

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

Running time?

$\Theta(n^4)$

最开始，输入 W 就是 i 到 j 只有 1 条边时的最短路径问题，进行 n-2 次迭代，求出 i 到 j 之间最多有 n-1 条边时的最短路径问题（即，目标问题）。

*25.1 Shortest paths and matrix multiplication

Improving the running time from matrix multiplication.

$$l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j , \\ \infty & \text{if } i \neq j . \end{cases}$$

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

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

$l_{ij}^{(m)}$：从 i 到 j，包括最多 m 条边的最短路径 p 的长度（权值），其中 $p = p' + w_{kj}$.
则必有，p' 是从 i 到 k，包括最多 m-1 条边的最短路径，即 p' 的长度为 $l_{ik}^{(m-1)}$.

EXTEND-SHORTEST-PATHS(L, W)

```
n = L.rows
let L' = (l'_ij) be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        l'_ij = ∞
        for k = 1 to n
            l'_ij = min(l'_ij, l_ik + w_kj)
return L'
```

product $C = A \cdot B$ of two $n \times n$ matrices A and B

$$\boxed{c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} .}$$

Observe that if we make the substitutions

$$\begin{aligned} l^{(m-1)} &\to a , \\ w &\to b , \\ l^{(m)} &\to c , \\ \min &\to + , \\ + &\to \cdot \end{aligned}$$

SQUARE-MATRIX-MULTIPLY(A, B)

```
n = A.rows
let C be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        c_ij = 0
        for k = 1 to n
            c_ij = c_ij + a_ik · b_kj
return C
```

*25.1 Shortest paths and matrix multiplication

Improving the running time from matrix multiplication.

$$
\begin{aligned}
L^{(1)} &= L^{(0)} \cdot W &&= W\,, \\
L^{(2)} &= L^{(1)} \cdot W &&= W^2\,, \\
L^{(3)} &= L^{(2)} \cdot W &&= W^3\,, \\
&\vdots \\
L^{(n-1)} &= L^{(n-2)} \cdot W &&= W^{n-1}\,.
\end{aligned}
$$

n to lg$n$ $\Longrightarrow$

$$
\begin{aligned}
L^{(1)} &= W\,, \\
L^{(2)} &= W^2 &&= W \cdot W\,, \\
L^{(4)} &= W^4 &&= W^2 \cdot W^2 \\
L^{(8)} &= W^8 &&= W^4 \cdot W^4\,, \\
&\vdots \\
L^{(2^{\lceil \lg(n-1) \rceil})} &= W^{2^{\lceil \lg(n-1) \rceil}} &&= W^{2^{\lceil \lg(n-1) \rceil - 1}} \cdot W^{2^{\lceil \lg(n-1) \rceil - 1}}\,.
\end{aligned}
$$

```
SLOW-ALL-PAIRS-SHORTEST-PATHS(W)
n = W.rows
L^(1) = W
for m = 2 to n − 1
    let L^(m) be a new n × n matrix
    L^(m) = EXTEND-SHORTEST-PATHS(L^(m−1), W)
return L^(n−1)
```

$\Longrightarrow$

```
FASTER-ALL-PAIRS-SHORTEST-PATHS(W)
n = W.rows
L^(1) = W
m = 1
while m < n − 1
    let L^(2m) be a new n × n matrix
    L^(2m) = EXTEND-SHORTEST-PATHS(L^(m), L^(m))
    m = 2m
return L^(m)
```

$\Theta(n^4)$ $\Longrightarrow$ $\Theta(n^3 \lg n)$

*25.1 Shortest paths and matrix multiplication

- A dynamic-programming algorithm based on **matrix multiplication** , $\Theta(V^4)$.

- “Repeated squaring,” $\Theta(V^3 \lg V)$.

25.2 The Floyd-Warshall algorithm (DP)

The structure of a shortest path

- **The Floyd-Warshall** algorithm considers the intermediate vertices of a shortest path, where an *intermediate* vertex of a simple path $p = \langle v_1, v_2, \ldots, v_l \rangle$ is any vertex of p other than v_1 or v_l, that is, any vertex in the set $\{v_2, v_3, \ldots, v_{l-1}\}$.

简单路径 p 的端点是 v_1 和 v_l，其他点是 p 的“之间”顶点

Floyd-warshall 来源于floyd，其原理基于warshall提出的基于布尔矩阵的传递闭包。

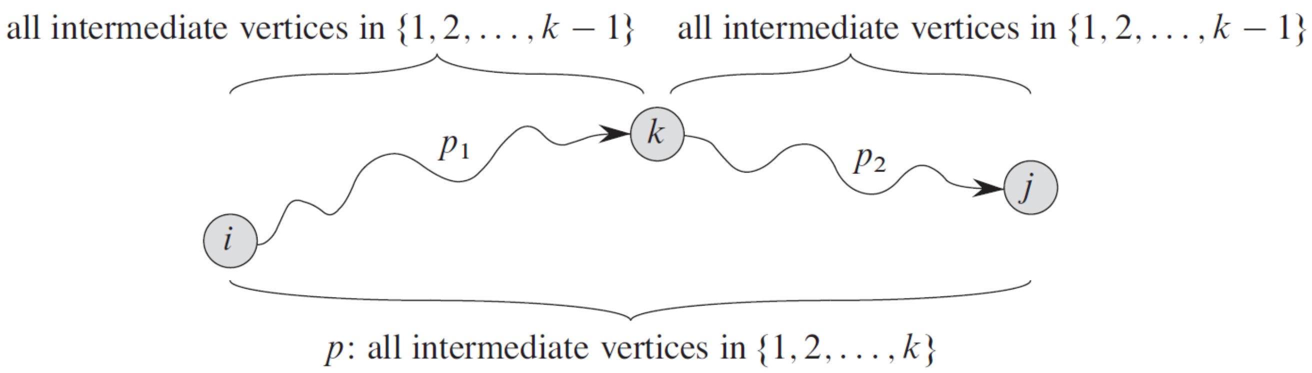
[PDF] Algorithm 97: shortest path
RW Floyd - Communications of the ACM, 1962 - dl.acm.org
¢ onlment This procedure will perform different order arithmetic operations with b and c, putting the result in a. The order of the operation is given by op. For op= 1 addition is performed. …
☆ 保存 ワワ 引用 被引用次数: 6015 相关文章 所有 3 个版本

A theorem on boolean matrices
S Warshall - Journal of the ACM (JACM), 1962 - dl.acm.org
… Given two boolean matrices A arid B, we define the boolean product AAB as that matrix whose (i, j)th entry is vk(a~/, A bki). We define tile boolean sum AVB as that matrix whose (i, j)th …
☆ 保存 ワワ 引用 被引用次数: 2563 相关文章 所有 7 个版本

25.2 The Floyd-Warshall algorithm (DP)

The structure of a shortest path

- The Floyd-Warshall algorithm considers the intermediate vertices of a shortest path, where an *intermediate* vertex of a simple path $p = \langle v_1, v_2, \ldots, v_l \rangle$ is any vertex of p other than v_1 or v_l, that is, any vertex in the set $\{v_2, v_3, \ldots, v_{l-1}\}$.
  简单路径 p 的端点是 v_1 和 v_l， p 的“之间”顶点是$\{v_2, v_3, \ldots, v_{l-1}\}$之间的任意顶点
- Consider a subset $\{1, 2, \ldots, k\}$ of vertices for some k. For any pair of vertices $i, j \in V$, consider all paths from i to j whose intermediate vertices are all drawn from $\{1, 2, \ldots, k\}$, and let p be a minimum-weight path from among them. (Path p is simple.)
  考虑 i to j 的所有路径，其“之间”顶点from $\{1, 2, \ldots, k\}$， p 是这所有路径中最短的一条



25.2 The Floyd-Warshall algorithm (DP)

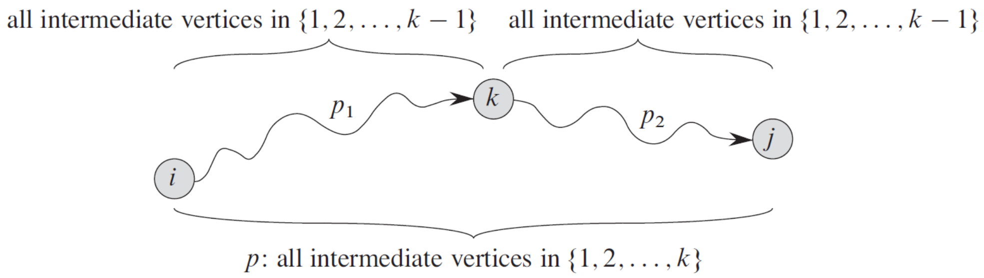
The structure of a shortest path

For any pair of vertices (i, j), all paths from i to j whose intermediate vertices are all drawn from $\{1, 2, \dots, k\}$, and let p be a minimum path.

- If k is not an intermediate vertex of path p, then all intermediate vertices of p are in the set $\{1, 2, \dots, k\text{-}1\}$. Thus, a *st-path*-$\delta(I, j)$ with all intermediate vertices in the set $\{1, 2, \dots, k\text{-}1\}$ is also a *st-path*-$\delta(i, j)$ with all intermediate vertices in the set $\{1, 2, \dots, k\}$.
- If k is … , then we decompose p into $i \overset{p_1}{\leadsto} k \overset{p_2}{\leadsto} j$. p_1 is a *st-path*-$\delta(i, k)$ with all intermediate vertices in the set $\{1, 2, \dots, k\text{-}1\}$. Similarly, for p_2, …

p 是从 i 到 j 的最短路径 *st-path*-$\delta(i, j)$，“之间”顶点 from $\{1, 2, \dots, k\}$:

- 如果 k 不是路径 p 上的顶点，则 *st-path*-$\delta(i, j)$ 的之间顶点来自于 $\{1, 2, \dots, k\text{-}1\}$，即，之间顶点 from $\{1, 2, \dots, k\}$ 的 *st-path*-$\delta(i, j)$ 就是之间顶点 from $\{1, 2, \dots, k\text{-}1\}$ 的 *st-path*-$\delta(i, j)$；
- 如果 k 是路径 p 上的顶点，则 *st-path*-$\delta(i, j)$ 由两部分构成 $i \overset{p_1}{\leadsto} k \overset{p_2}{\leadsto} j$ ，其中 p_1 是之间顶点来自于 $\{1, 2, \dots, k\text{-}1\}$ 的 *st-path*-$\delta(i, k)$， p_2 同理。

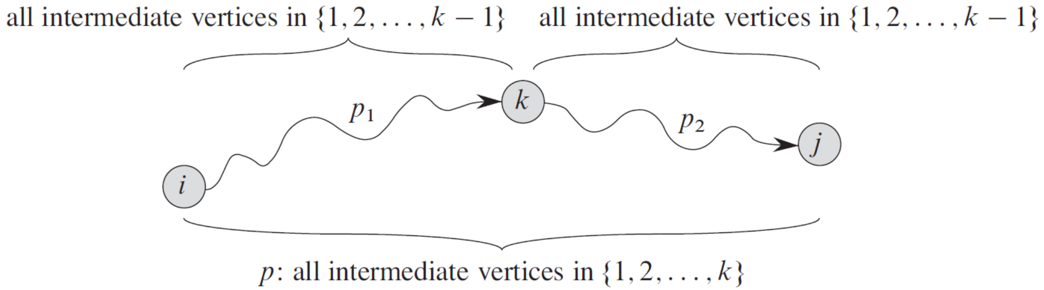


25.2 The Floyd-Warshall algorithm (DP)

A recursive solution to the all-pairs shortest-paths problem

Let $d_{ij}^{(k)}$ be the weight of a *st-path-δ*(*i*, *j*) for which all intermediate vertices are in the set {1, 2, … , *k*}. When *k* = 0, a *path-p*(*i*, *j*) has no intermediate vertices at all. Such a path has at most one edge. We define recursively

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



Because for any path, all intermediate vertices are in the set {1, 2, … , *n*}, the matrix $D^{(n)} = (d_{ij}^{(n)})$ gives the final answer: $d_{ij}^{(n)} = \delta(i, j)$ for all *i*, *j* ∈ *V*.

p 是从 *i* 到 *j* 的最短路径 *st-path-δ*(*i*, *j*)，*p* 的“之间”顶点 from {1 , 2 , … , *k*}，其长度（权值）为 $d_{ij}^{(k)}$:

- 如果 *k* 不是路径 *p* 上的顶点，则 *st-path-δ*(*i*, *j*) 的之间顶点来自于 {1 , 2 , … , *k*-1}，即，之间顶点 from {1 , 2 , … , *k*} 的 *st-path-δ*(*i*, *j*) 就是之间顶点 from {1 , 2 , … , *k*-1} 的 *st-path-δ*(*i*, *j*);
- 如果 *k* 是路径 *p* 上的顶点，则 *st-path-δ*(*i*, *j*) 由两部分构成 $i \overset{p_1}{\leadsto} k \overset{p_2}{\leadsto} j$，其中 p_1 是之间顶点来自于 {1 , 2 , … , *k*-1} 的 *st-path-δ*(*i*, *k*)，p_2 同理。

25.2 The Floyd-Warshall algorithm (DP)

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

Direct recursion algorithm? complexity?

Computing the shortest-path weights bottom up

We can use the following bottom-up procedure to compute the values $d_{ij}^{(k)}$ in order of increasing values of k.

按 k 增加（自底向上）的方式进行计算

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 (DP)

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

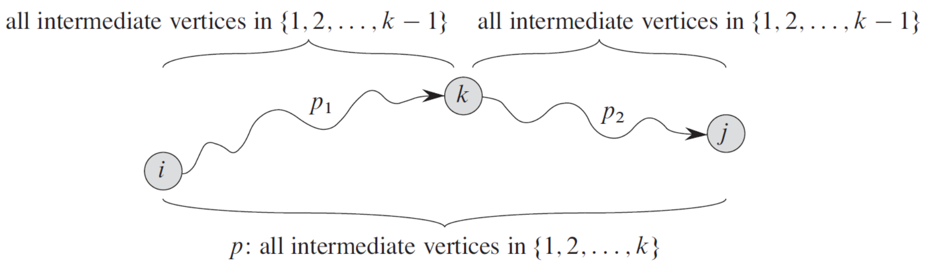
Constructing a shortest path （构造最短路径）

We compute a sequence of matrices $\Pi^{(0)}, \Pi^{(1)}, \ldots, \Pi^{(n)}$, where $\Pi = \Pi^{(n)}$ and we define $\pi_{ij}^{(k)}$ as the predecessor of vertex j on a shortest path from vertex i with all intermediate vertices in the set $\{1, 2, \ldots, k\}$.

$\pi_{ij}^{(k)}$ 表示从 i 到 j 的最短路径（之间顶点 from $\{1, 2, \ldots, k\}$）中 j 的前驱节点

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

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

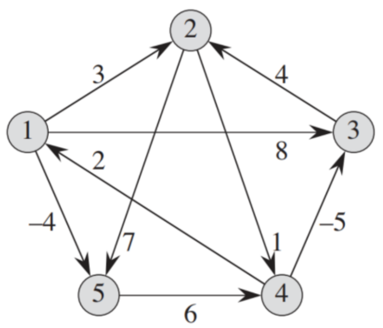


对最短路径 $i \rightarrow j$,

(1) k 不在最短路径上，最短路径的之间顶点是$\{1, 2, \ldots, k\text{-}1\}$，因此 $\pi_{ij}^{(k)} = \pi_{ij}^{(k-1)}$

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

25.2 The Floyd-Warshall algorithm (DP)



$$d_{42}^{(1)} = \min(d_{42}^{(0)}, d_{41}^{(0)} + d_{12}^{(0)}) = \min(\infty, 2 + 3) = 5$$

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

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

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

$$d_{12}^{(5)} = \min(d_{12}^{(4)}, d_{15}^{(4)} + d_{52}^{(4)}) = \min(3, -4 + 5) = 1$$

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

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

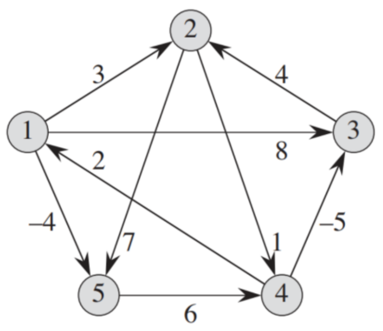
$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL}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

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

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

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

25.2 The Floyd-Warshall algorithm (DP)



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

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

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

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

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

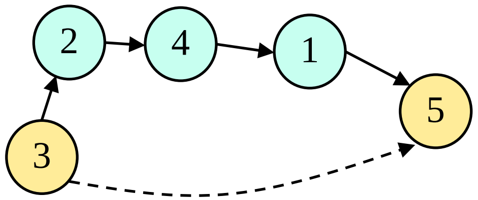
$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL}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

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

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

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

用前驱矩阵计算
最短路径



$\delta(3,5) = 3$

The Floyd-Warshall vs matrix multiplication

Floyd-Warshall $\Theta(n^3)$

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

上标和下标都是 k，一个 n 遍历（k from 1 to n）实现两个维度的计算

FLOYD-WARSHALL(W)

$n = W.rows$
$D^{(0)} = W$
for $k = 1$ **to** n
    let $D^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            $d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$
return $D^{(n)}$

Floyd-Warshall is fast. Why?

matrix multiplication $\Theta(n^4)$

$$l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j\,, \\ \infty & \text{if } i \neq j\,. \end{cases}$$

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

上标 m 和下标 k，两个 n 遍历

EXTEND-SHORTEST-PATHS(L, W)

$n = L.rows$
let $L' = (l'_{ij})$ be a new $n \times n$ matrix
for $i = 1$ **to** n
    for $j = 1$ **to** n
        $l'_{ij} = \infty$
        for $k = 1$ **to** n
            $l'_{ij} = \min(l'_{ij}, l_{ik} + w_{kj})$
return L'

SLOW-ALL-PAIRS-SHORTEST-PATHS(W)

$n = W.rows$
$L^{(1)} = W$
for $m = 2$ **to** $n - 1$
    let $L^{(m)}$ be a new $n \times n$ matrix
    $L^{(m)} = $ EXTEND-SHORTEST-PATHS($L^{(m-1)}, W$)
return $L^{(n-1)}$

25.2 The Floyd-Warshall algorithm

Transitive closure of a directed graph*

25.3 Johnson's algorithm for sparse graphs*

Summary of Graph Algorithms

- Queue, Priority Queue
- Enumeration (BFS, …)
- Recursion (DFS, …)
- Dynamic Programming (All-Pairs Shortest Paths)
- Greedy Strategy (Single-Source Shortest Paths)
- Relaxation
- Aggregate analysis
- …