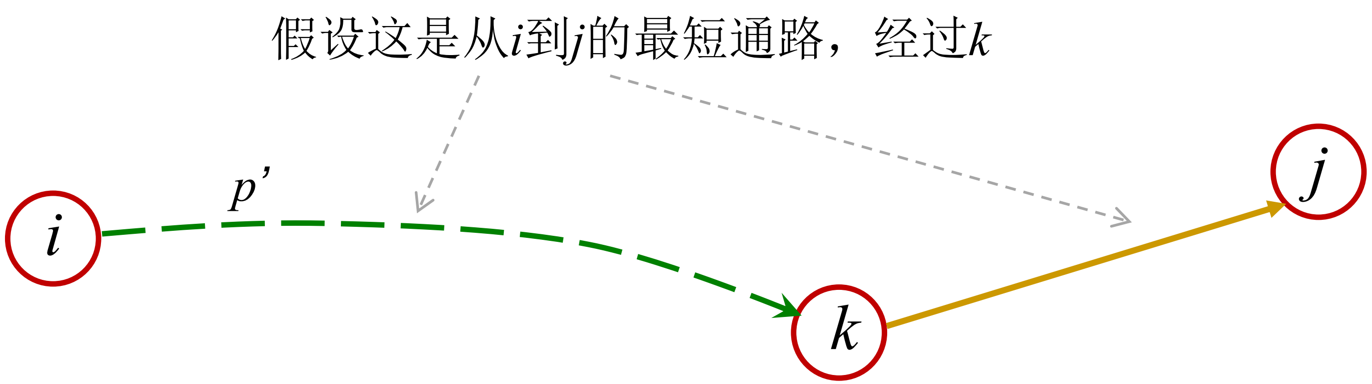
计算机问题求解 — 论题3-9
– All-Pair Shortest Paths

2016年11月2日

复习

- 动态规划的解题基本规律是什么？
  - 最优子结构分析
  - 递归定义最优解
  - 设计自底向上算法依次求解

最优子结构



If vertices i and j are distinct, then we decompose path p into $i \overset{p'}{\leadsto} k \to j$, where path p' now contains at most $m-1$ edges. By Lemma 24.1, p' is a shortest path from i to k, and so $\delta(i,j) = \delta(i,k) + w_{kj}$.

递归定义最优解

Now, let $l_{ij}^{(m)}$ be the minimum weight of any path from vertex i to vertex j that contains at most m edges. When $m = 0$, there is a shortest path from i to j with no edges if and only if $i = j$. Thus,

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

For $m \geq 1$, we compute $l_{ij}^{(m)}$ as the minimum of $l_{ij}^{(m-1)}$ (the weight of a shortest path from i to j consisting of at most $m-1$ edges) and the minimum weight of any path from i to j consisting of at most m edges, obtained by looking at all possible predecessors k of j. Thus, we recursively define

$$\begin{aligned} l_{ij}^{(m)} &= \min\left(l_{ij}^{(m-1)}, \min_{1 \leq k \leq n} \left\{l_{ik}^{(m-1)} + w_{kj}\right\}\right) \\ &= \min_{1 \leq k \leq n} \left\{l_{ik}^{(m-1)} + w_{kj}\right\} \, . \end{aligned} \tag{25.2}$$

The latter equality follows since $w_{jj} = 0$ for all j.

问题1：

在有10个点的图中，l_{ij}^{6}的直观含义是什么？

如果$l_{ij}^{6}=7$，能认定*ij*节点间的最短路径长度是7吗？

问题2:

为什么 $\delta(i,j)=l_{ij}^{(n-1)}$?

问题3：

如果定义矩阵$\boldsymbol{L}^m=(l_{ij}^m)$, $\boldsymbol{L}^1$, $\boldsymbol{L}^2$, …, $\boldsymbol{L}^{n-1}$分别表示什么含义？

如何去计算$\boldsymbol{L}^m$？

自底向上计算

The heart of the algorithm is the following procedure, which, given matrices $L^{(m-1)}$ and W, returns the matrix $L^{(m)}$. That is, it extends the shortest paths computed so far by one more edge.

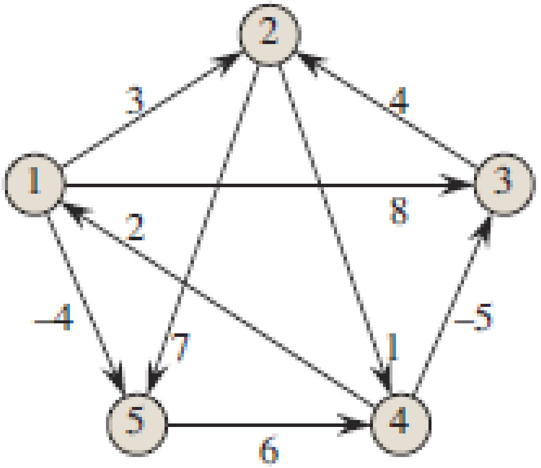
EXTEND-SHORTEST-PATHS(L, W)

```
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'
```

问题4：

“one more edge”体现在哪里？

只需要扩展n-2次



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)
```

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

问题5：

为什么上述算法被称为“慢”算法，为什么它可能被加快？

Extending和矩阵乘法

```
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'
```

？？？
真的一样吗？

并将W中的∞换为0

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

```
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
```

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)
```

一次"扩"的可不只是一条边了。

Because each of the $\lceil \lg(n-1) \rceil$ matrix products takes $\Theta(n^3)$ time, FASTER-ALL-PAIRS-SHORTEST-PATHS runs in $\Theta(n^3 \lg n)$ time. Observe that the code is tight, containing no elaborate data structures, and the constant hidden in the Θ-notation is therefore small.

这段话是什么意思?

问题6：

上面那个“快”算法有问题吗？具体说如果某两点之间最短路含3条边，是否会出错？

问题7：

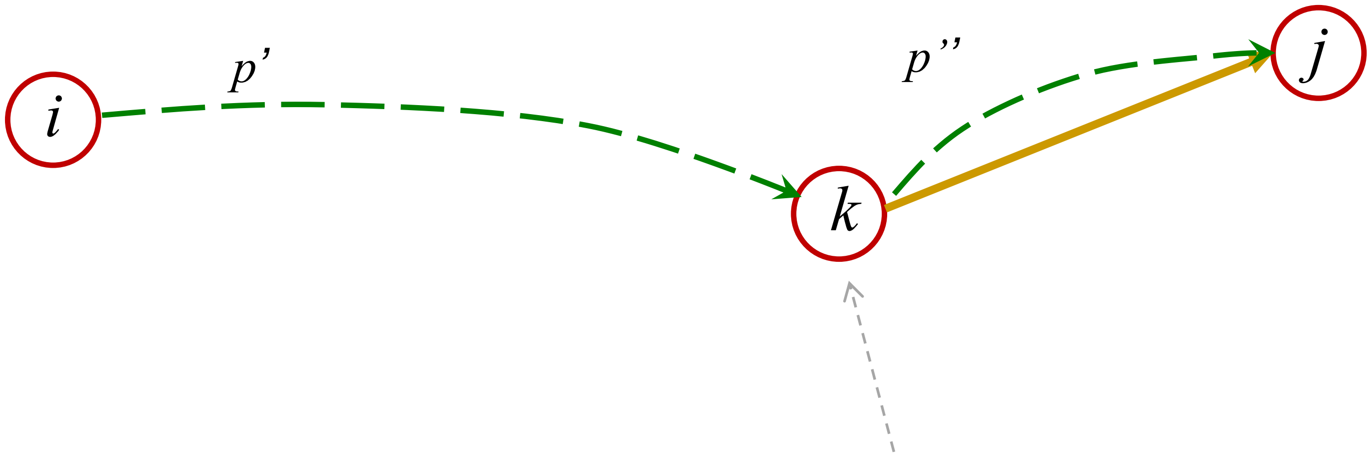
为什么说这是一种“动态规划”算法？

你个人觉得，哪一步是至关重要的？

一种新的“子结构”观察视角：

$$\&(i,j)=\&(i,k)+\&(k,j)$$

K不再是j的直接前驱节点



假设这是从*i*到*j*的最短通路，经过*k*

选一个“特定”的k点

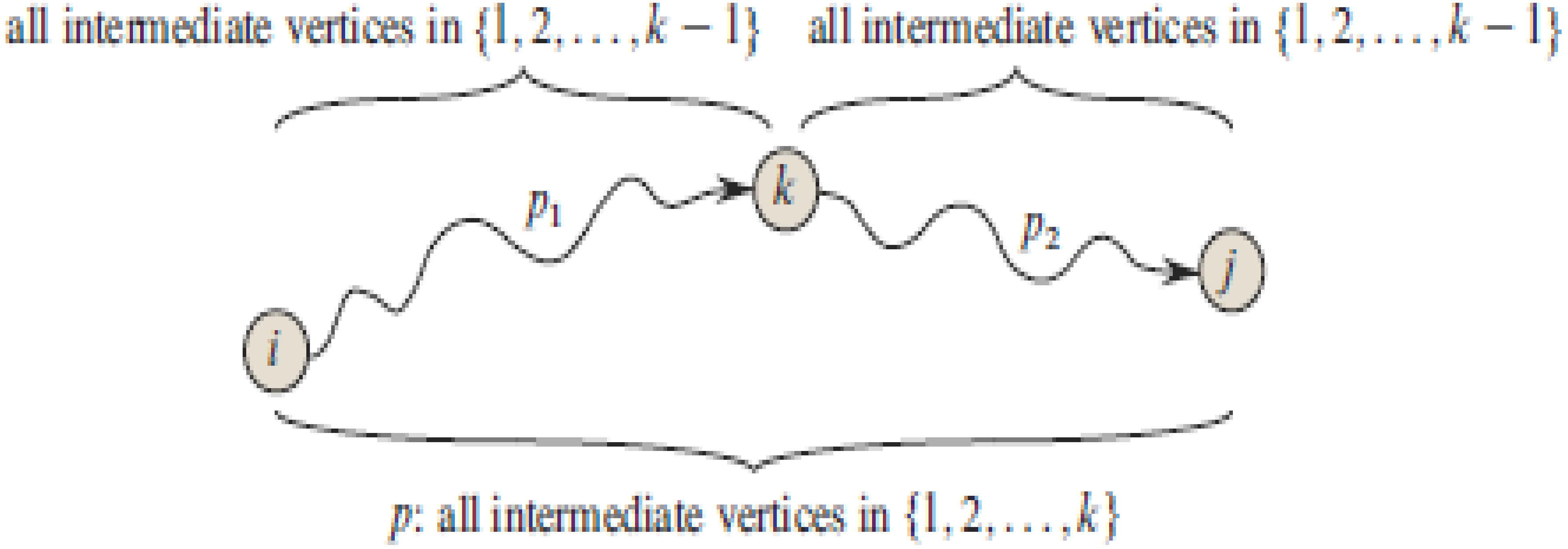


Figure 25.3 Path p is a shortest path from vertex i to vertex j, and k is the highest-numbered intermediate vertex of p. Path p_1, the portion of path p from vertex i to vertex k, has all intermediate vertices in the set $\{1, 2, \ldots, k-1\}$. The same holds for path p_2 from vertex k to vertex j.

问题8：

从动态规划的视角考虑，现在“子问题”有什么不同了？

The Floyd-Warshall algorithm exploits a relationship between path p and shortest paths from i to j with all intermediate vertices in the set $\{1, 2, \ldots, k-1\}$. The relationship depends on whether or not k is an intermediate vertex of path p.

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

对比这两个递归式，你有什么感觉？

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

对于all-pair shortest path问题，

输入、输出与*D*是什么关系？

FLOYD-WARSHALL(W)

$n = W.rows$
$D^{(0)} = W$
for $k = 1$ **to** n
    let $D^{(k)} = \left(d_{ij}^{(k)}\right)$ 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 $D^{(n)}$

问题11：

你觉得Floyd-Washall算法是如何将复杂度的阶降下来的？

问题12:

传递闭包问题与最短路径问题为什么能够联系在一起，并用基本相同的方法解决？

采用布尔矩阵，利用逻辑运算

$$t_{ij}^{(0)} = \begin{cases} 0 & \text{if } i \neq j \text{ and } (i,j) \notin E\ , \\ 1 & \text{if } i = j \text{ or } (i,j) \in E\ , \end{cases}$$

and for $k \geq 1$,

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}\right) \ .$$

```
TRANSITIVE-CLOSURE(G)
n = |G.V|
let T^(0) = (t_ij^(0)) be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        if i == j or (i, j) ∈ G.E
            t_ij^(0) = 1
        else t_ij^(0) = 0
for k = 1 to n
    let T^(k) = (t_ij^(k)) be a new n × n matrix
    for i = 1 to n
        for j = 1 to n
            t_ij^(k) = t_ij^(k-1) ∨ (t_ik^(k-1) ∧ t_kj^(k-1))
return T^(n)
```

问题13:

为什么在计算传递闭包时, 矩阵计算能够发挥更加直接的作用?

你试试证明: 如果M是图的邻接矩阵, 则M^2中某一项等于1 当且仅当 其对应的两点之间存在长度恰好是2的通路。这个结论很容易利用归纳法推广到传递闭包算法的证明。

问题14：

可以说Johnson算法体现了“尽可能”“有效”利用单源最短路算法的思想。你能否说说“尽可能”和“有效”体现在何处？

利用Bellman-Ford算法判定负回路

而且只能让Bellman-Ford算法执行1次！

问题15：

这件事是怎么做到的？

重复执行Dijstra算法

但是Dijstra算法不能用于边带有负值权的图！

问题16：

这个问题是如何解决的？

. If G has negative-weight edges but no negative-weight cycles, we simply compute a new set of nonnegative edge weights that allows us to use the same method. The new set of edge weights $\hat{w}$ must satisfy two important properties:

1. For all pairs of vertices $u, v \in V$, a path p is a shortest path from u to v using weight function w if and only if p is also a shortest path from u to v using weight function $\hat{w}$.

2. For all edges (u, v), the new weight $\hat{w}(u, v)$ is nonnegative.

$$\hat{w}(u, v) = w(u, v) + h(u) - h(v)$$

在什么情况下“有效（率）”

JOHNSON(G, w)

compute G', where $G'.V = G.V \cup \{s\}$,
  $G'.E = G.E \cup \{(s, v) : v \in G.V\}$, and
  $w(s, v) = 0$ for all $v \in G.V$
if BELLMAN-FORD(G', w, s) == FALSE
  print “the input graph contains a negative-weight cycle”
else for each vertex $v \in G'.V$
    set $h(v)$ to the value of $\delta(s, v)$
      computed by the Bellman-Ford algorithm
  for each edge $(u, v) \in G'.E$
    $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$
  let $D = (d_{uv})$ be a new $n \times n$ matrix
  for each vertex $u \in G.V$
    run DIJKSTRA$(G, \hat{w}, u)$ to compute $\hat{\delta}(u, v)$ for all $v \in G.V$
    for each vertex $v \in G.V$
      $d_{uv} = \hat{\delta}(u, v) + h(v) - h(u)$
  return D

$$\boxed{O(V^2 \lg V + VE)}$$

如果$|E| \in O(|V|)$，则此算法效率好于Floyd-Washall算法。

Open Topics

- 我们如何在Floyd-warshall算法基础上，构造最短路径？
  - 请按照动态规划解题基本思路来解决该问题

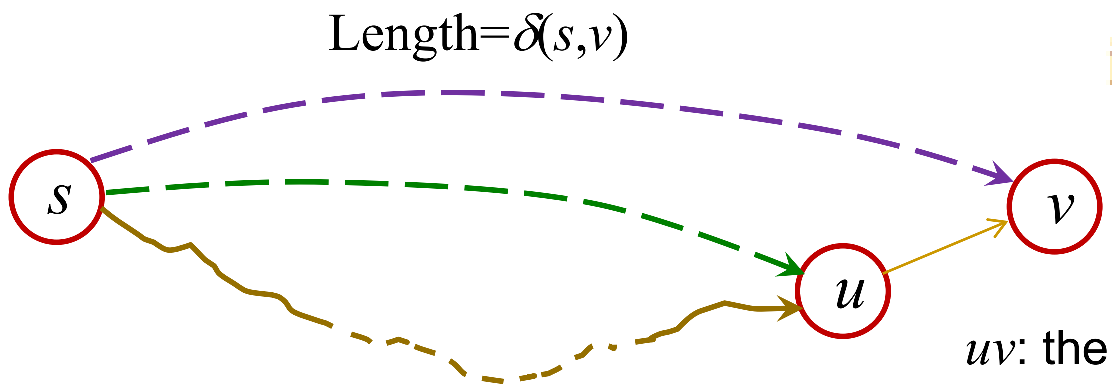
- 四川省决定在省内建设一个炼钢厂，集中冶炼省内开采出来的铁矿石。请问你对炼钢厂的选址有什么建议？

课外作业

- TC Ex.25.1: 4, 5, 6, 9, 10
- TC Ex.25.2: 2, 4, 6, 8
- TC Ex.25.3: 2, 3
- TC Prob 25: 2

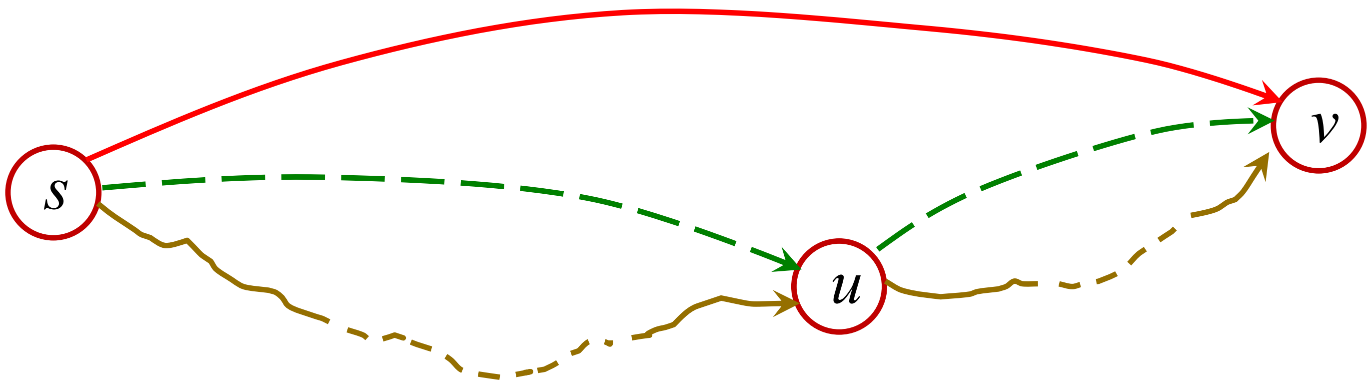
问题1:

你能否借助下图说说单源最短通路算法的核心思想?



$$\text{if } v.d > u.d + w(u,v)$$
$$v.d = u.d + w(u,v)$$

uv: the edge to be relaxed



问题2:

假如前面图中uv不是一条边, 而是一条路，类似的图对你考虑从s到v的最短通路问题有什么启发吗？

要的结果不仅是距离，还有“路”

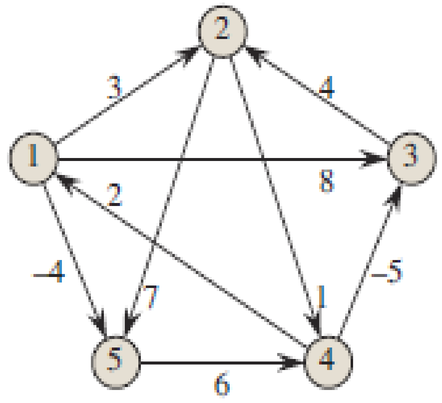
$$\pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{if } i = j \text{ or } w_{ij} = \infty \,, \\ i & \text{if } i \neq j \text{ and } w_{ij} < \infty \,. \end{cases} \quad (25$$

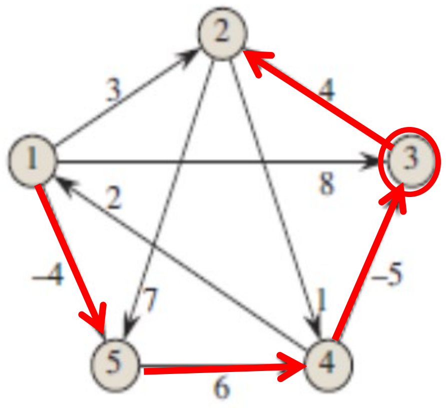
For $k \geq 1$, if we take the path $i \leadsto k \leadsto j$, where $k \neq j$, then the predecessor of j we choose is the same as the predecessor of j we chose on a shortest path from k with all intermediate vertices in the set $\{1, 2, \ldots, k-1\}$. Otherwise, we choose the same predecessor of j that we chose on a shortest path from i with all intermediate vertices in the set $\{1, 2, \ldots, k-1\}$. Formally, for $k \geq 1$,

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \,, \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \,. \end{cases} \tag{25.7}$$

要的结果不仅是距离，还有“路”

从predecessor matrix到predecessor subgraph



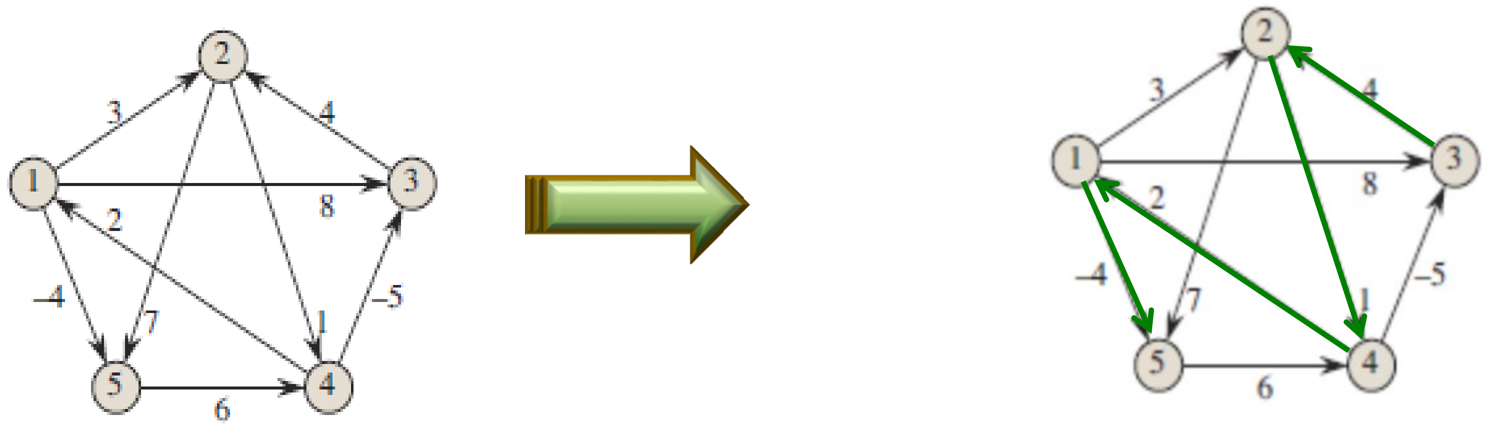


$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(5)} = \begin{pmatrix} \boxed{\text{NIL} \quad 3 \quad 4 \quad 5 \quad 1} \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4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

1=>2

要的结果不仅是距离，还有“路”

从predecessor matrix到predecessor subgraph



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

3=》5