Note 14 Graph Algorithm II

All Pairs Shortest Path, AOE Network

I. All Pairs Shortest Path

(1) Overview

Input: Digraph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$, with edge-weight function $w: E \to \mathbb{R}$.
Output: $n \times n$ matrix of shortest-path lengths $\delta(i, j)$ for all $i, j \in V$.

Naive Ideas: Run Single Source Shortest Path from each vertex once

- Dijkstra Algorithm $O(E\lg V)$ —|V| times→ $O(EV\lg V)$
- Bellman-Ford Algorithm $O(EV)$ —|V| times→ $O(EV^2)$

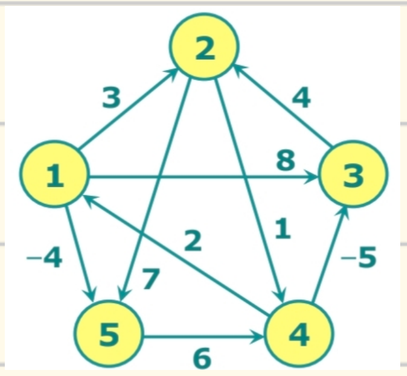
(II) "Matrix Multiplication" Algorithm

1. Dynamic Programming Analysis

在这里，我们使用邻接矩阵存储图

我们对于最短至路中边的长度 m 进行动态规划，显然 $m \leq n-1$

- Consider the n×n adjacency matrix $A = (a_{ij})$ of the digraph



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

（这里顶点的标号是随意的）

- We define matrix $D^{(m)} = \{d_{ij}^{(m)}\}$:

$d_{ij}^{(m)}$ = weight of a shortest path from i to j that uses at most m edges.

Claim: $d_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{if } i \neq j \end{cases}$

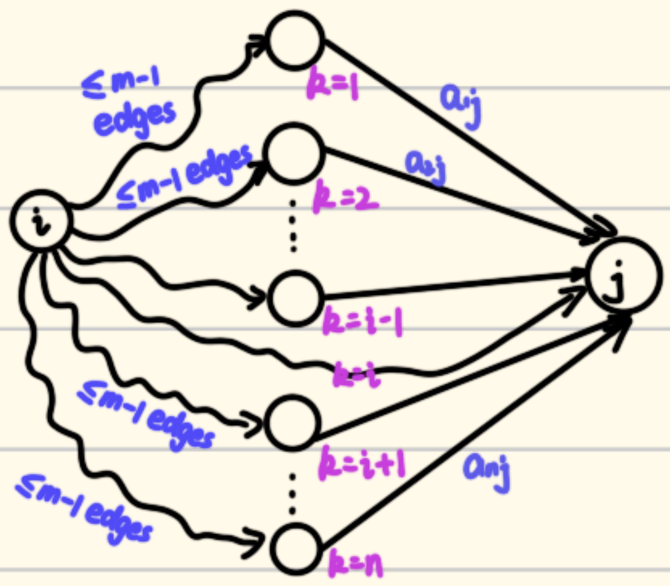
- For $m = 1, 2, \cdots, n-1$:

$$d_{ij}^{(m)} = \min_k \{d_{ik}^{(m-1)} + a_{kj}\}$$

$k = 1, 2, 3, \cdots, n$

当 $k=i$ 时，表示-继续 $d_{ij}^{(m)} = d_{ij}^{(m-1)}$，无需加入新边，等效于 $a_{ii} = 0$

Proof:

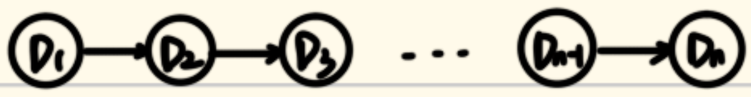


Relaxation Operation!

```
for k ← 1 to n
    do if d_ij > d_ik + a_kj
        then d_ij ← d_ik + a_kj
```

2. Algorithm

```
SLOW-ALL-PAIRS-SHORTEST-PATHS(W)
1  n = W.rows
2  D^(1) = W
3  for m = 2 to n − 1                          O(n^4)
4      let D^(m) be a new n × n matrix
5      D^(m) = EXTEND-SHORTEST-PATHS(D^(m−1), W)
6  return D^(n−1)
```




```
EXTEND-SHORTEST-PATHS(D, W)
1  n = D.rows
2  let D' = (d'_ij) be a new n × n matrix
3  for i = 1 to n                              O(n^3)
4      for j = 1 to n
5          d'_ij = ∞
6          for k = 1 to n
7              d'_ij = min(d'_ij, d_ik + a_kj)
8  return L'
```

if $d'_{ij} > d_{ik} + a_{kj}$: $d_{ij}' = k$

关于Π矩阵(用于重建路径，Πij的值表示从i到j的最短路径下到j的上一个结点的序号

Time Complexity:

$O(n^4)$

3. "Matrix-Multiplication" Representation

将上述 EXTEND-SHORTEST-PATHS(L, W) 改写为类似矩阵乘法的形式

$$d_{ij}^{(m)} = \min_k \{ d_{ik}^{(m-1)} + a_{kj} \} \qquad k = 1, 2, \cdots, n$$

定义矩阵新乘法 $A_{n\times n} \odot B_{n\times n}$

$$(A \odot B)_{ij} = \min_k \{ a_{ik} + b_{kj} \} \qquad k = 1, 2, \cdots, n$$

c.f. $(AB)_{ij} = \sum_k (a_{ik} + b_{kj}) \qquad k = 1, 2, \cdots, n$

则可改写成：$D^{(m)} = D^{(m-1)} \odot A$

每次⊙操作复杂度为 $O(n^3)$，假设不用类似 strassen 的话

$D^{(n)} = D^{(\frac{n}{2})} \odot D^{(\frac{n}{2})} = (D^{(\frac{n}{4})} \odot D^{(\frac{n}{4})}) \odot (D^{(\frac{n}{4})} \odot D^{(\frac{n}{4})}) = \cdots$

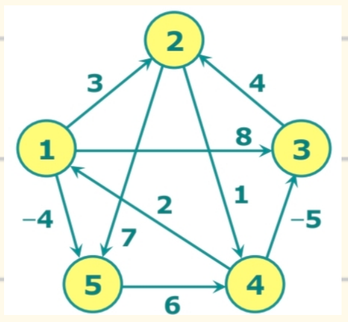
```
FASTER-ALL-PAIRS-SHORTEST-PATHS(W)
1  n = W.rows
2  L^(1) = W
3  m = 1
4  while m < n − 1
5      let L^(2m) be a new n × n matrix
6      L^(2m) = EXTEND-SHORTEST-PATHS(L^(m), L^(m))
7      m = 2m
8  return L^(m)
```

$D^{(n)} \leftarrow D^{(\frac{n}{2})} \leftarrow D^{(\frac{n}{4})} \cdots \leftarrow D^{(1)}$

共 lgn 次乘法，每次复杂度为 $O(n^3)$ ⇒ Time Complexity: $\Theta(n^3 \lg n)$

4. Example :

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



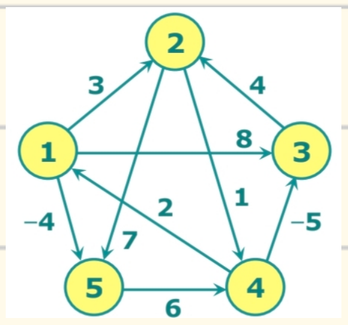
$$D^{(0)}=\begin{pmatrix}0&\infty&\infty&\infty&\infty\\ \infty&0&\infty&\infty&\infty\\ \infty&\infty&0&\infty&\infty\\ \infty&\infty&\infty&0&\infty\\ \infty&\infty&\infty&\infty&0\end{pmatrix}$$

$D^{(1)}=A$

max len = 0

↓ iteration 1

各边

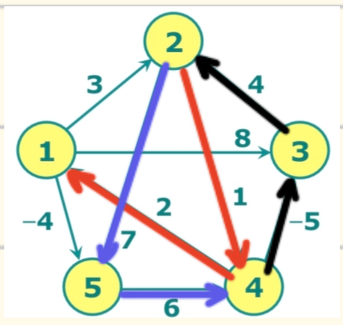


max len = 1

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

$$\Pi^{(1)}=\begin{pmatrix}\emptyset&1&1&\emptyset&1\\ \emptyset&\emptyset&\emptyset&2&2\\ \emptyset&3&\emptyset&\emptyset&\emptyset\\ 4&\emptyset&4&\emptyset&\emptyset\\ \emptyset&\emptyset&\emptyset&5&\emptyset\end{pmatrix}$$

↓ iteration 2

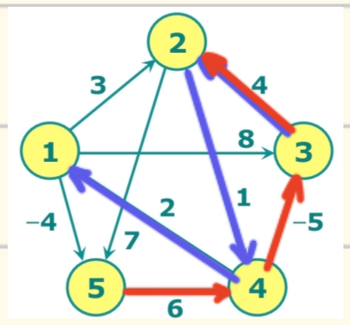


max len = 2

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

$$\Pi^{(2)}=\begin{pmatrix}\emptyset&1&1&5&1\\ 4&\emptyset&4&2&2\\ \emptyset&3&\emptyset&2&2\\ 4&3&4&\emptyset&1\\ 4&\emptyset&4&5&\emptyset\end{pmatrix}$$

↓ iteration 3

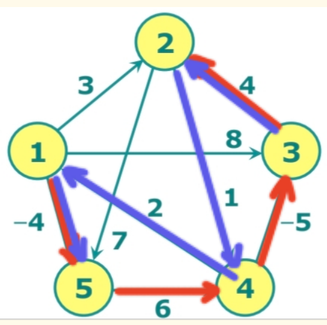


max len = 3

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

$$\Pi^{(3)}=\begin{pmatrix}\emptyset&1&4&5&1\\ 4&\emptyset&4&2&1\\ 4&3&\emptyset&2&2\\ 4&3&4&\emptyset&1\\ 4&3&4&5&\emptyset\end{pmatrix}$$

↓ iteration 4



max len = 4

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

$$\Pi^{(4)}=\begin{pmatrix}\emptyset&3&4&5&1\\ 4&\emptyset&4&2&1\\ 4&3&\emptyset&2&1\\ 4&3&4&\emptyset&1\\ 4&3&4&5&\emptyset\end{pmatrix}$$

Find Path From 3 to 5

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

$Path_{35}: 3\to2\to4\to1\to5$

$d_{35}=4+1+2-4=3$

Find Path From 1 to 2

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

$Path_{12}=1\to5\to4\to3\to2$

$d_{12}=-4+6-5+4$

(III). Floyd-Warshall Algorithm

1. Dynamic Programming Analysis

We define:

$d_{ij}^{(k)}$ = weight of a shortest path from i to j

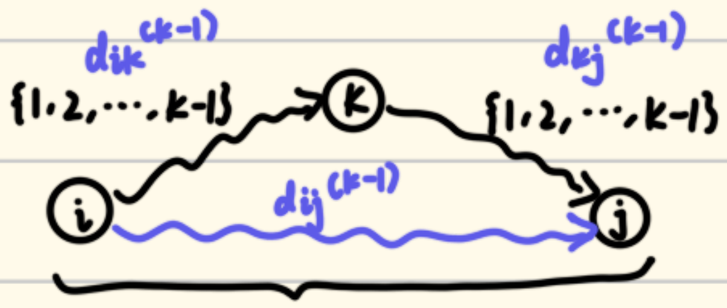
with intermediate vertices belonging to the set $\{1,2,\cdots,k\}$

$d_{ij}^{(0)} = w_{ij}$ 这里顶点序号$\{1,2,\cdots,n\}$是随意的。

我们对可利用的节点资源数进行动态规划，即我们可利用$\{1,2,\cdots,m\}$，$m \le n$

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

Proof:



p: All intermediate vertices in $\{1,2,\cdots,k\}$

2. Algorithm

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** $D^{(n)}$

Time Complexity: $O(n^3)$

- 前驱节点

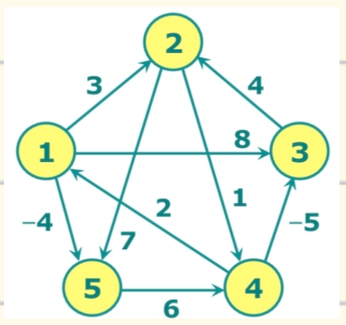
设 $\pi_{ij}^{(k)}$ 为从i至j上最短路上j的前驱节点

则有:

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

初始值 $\pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{若 } i=j \text{ 或 } w_{ij}=\infty \\ i & \text{若 } i \neq j \text{ 且 } w_{ij}<\infty \end{cases}$

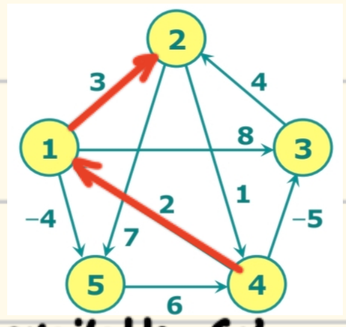
3. Example



available: $\varnothing$

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

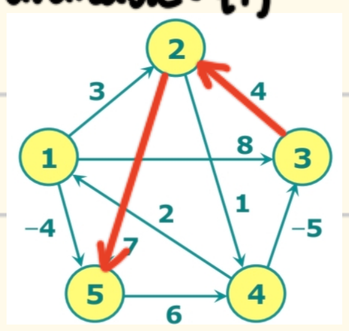
↓ iteration 1



available : {1}

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

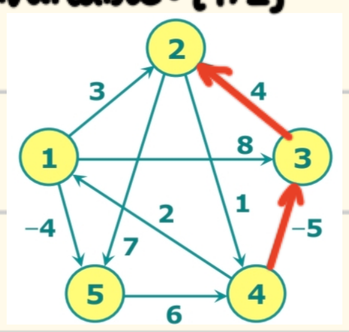
↓ iteration 2



available : {1, 2}

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

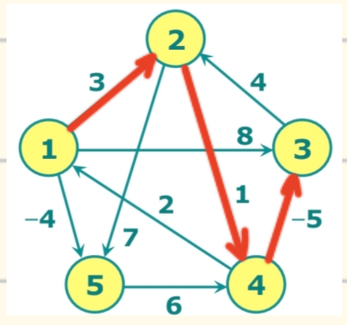
↓ iteration 3



available : {1, 2, 3}

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

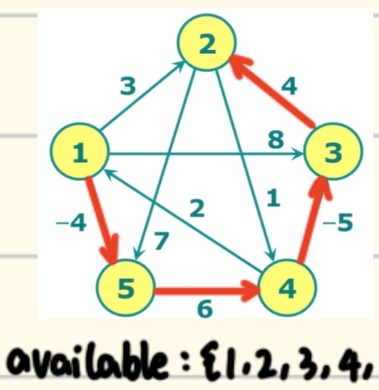
↓ iteration 4



available : {1, 2, 3, 4}

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

↓ iteration 5

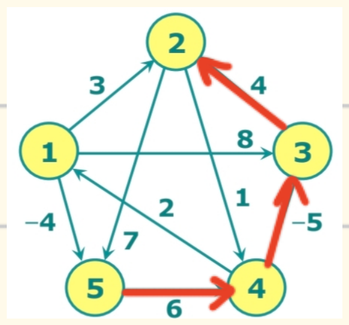


available : {1, 2, 3, 4, 5}

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

Find path from 5 to 2

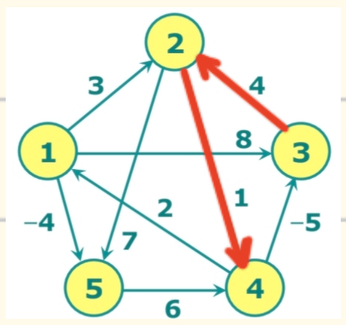
$$D^{(5)} = \begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix} \quad \Pi^{(5)} = \begin{bmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1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



5 → 4 → 3 → 2

Find path from 3 to 4

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



3 → 2 → 4

(IV). Johnson's Algorithm

1. Basic Idea:

Is it any possible to change all ***negative-weight*** edges to ***nonnegative***? (Graph reweighting)

Then, we can use ***Dijkstra's algorithm*** to compute the shortest path for all edges.

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

2. Graph-reweighting method

下面的引理描述的是我们可以很容易地对边的权重进行重新赋值来满足上面的两个条件。我们使用 δ 表示从权重函数 w 所导出的最短路径权重，而用 $\hat{\delta}$ 表示从权重函数 $\hat{w}$ 所导出的最短路径权重。

引理 25.1（重新赋予权重并不改变最短路径） 给定带权重的有向图 $G=(V, E)$，其权重函数为 $w: E \to \mathbf{R}$，设 $h: V \to \mathbf{R}$ 为任意函数，该函数将结点映射到实数上。对于每条边 $(u, v) \in E$，定义

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v) \tag{25.9}$$

设 $p=\langle v_0, v_1, \cdots, v_k \rangle$ 为从结点 v_0 到结点 v_k 的任意一条路径，那么 p 是在使用权重函数 w 时从结点 v_0 到结点 v_k 的一条最短路径，当且仅当 p 是在使用权重函数 $\hat{w}$ 时从结点 v_0 到结点 v_k 的一条最短路径，即 $w(p)=\delta(v_0, v_k)$ 当且仅当 $\hat{w}(p)=\hat{\delta}(v_0, v_k)$。而且，图 G 在使用权重函数 w 时不包含权重为负值的环路，当且仅当 p 在使用权重函数 $\hat{w}$ 也不包括权重为负值的环路。

证明 $\hat{w}(p) = w(p) + h(v_0) - h(v_k)$:

$$\begin{aligned}
\hat{w}(p) &= \sum_{i=1}^{k} \hat{w}(v_{i-1}, v_i) = \sum_{i=1}^{k} [w(v_{i-1}, v_i) + h(v_{i-1}) - h(v_i)] \\
&= \sum_{i=1}^{k} w(v_{i-1}, v_i) + \sum_{i=1}^{k} [h(v_{i-1}) - h(v_i)] = \sum_{i=1}^{k} w(v_{i-1}, v_i) + h(v_0) - h(v_k) \\
&= w(p) + h(v_0) - h(v_k) \quad \rightarrow \text{unrelated to path, only determined by vertices}
\end{aligned}$$

也就是说，通过对每个节点作映射 h，对每条边 (u,v) 加上 h(u) - h(v) 这样的变换不会影响属性！

如何找到这个 h 呢？

我们的下一个目标是确保第二个属性保持成立，即对于所有的边 $(u, v) \in E$，$\hat{w}(u, v)$ 为非负值。给定带权重的有向图 $G=(V, E)$，其权重函数为 $w: E \to \mathbf{R}$。我们制作一幅新图 $G'=(V', E')$，这里 $V'=V \cup \{s\}$，s 是一个新结点，$s \notin V$，$E'=E \cup \{(s, v): v \in V\}$。我们对权重函数 w 进行扩展，使得对于所有的结点 $v \in V$，$w(s, v)=0$。注意，因为结点 s 没有进入的边，除了以 s 为源结点的最短路径外，图 G' 中没有其他包含结点 s 的最短路径。而且，图 G' 不包含权重为负值的环路当且仅当图 G 不包含权重为负值的环路。图 25-6(a)描述的是对应图 25-1 图 G 的图 G'。

现在假定图 G 和图 G' 都不包含权重为负值的环路。让我们定义，对于所有的结点 $v \in V'$，$h(v)=\delta(s, v)$。根据三角不等式(引理 24.10)，对于所有的边 $(u, v) \in E'$，有 $h(v) \leqslant h(u) + w(u, v)$。因此，如果我们根据式(25.9)来定义新的权重 $\hat{w}$，则有 $\hat{w}(u, v)=w(u, v)+h(u)-h(v) \geqslant 0$，至此我们满足了第二条性质。图 25-6(b)描述的是对图 25-6(a)的图进行权重重新赋值后的图 G'。

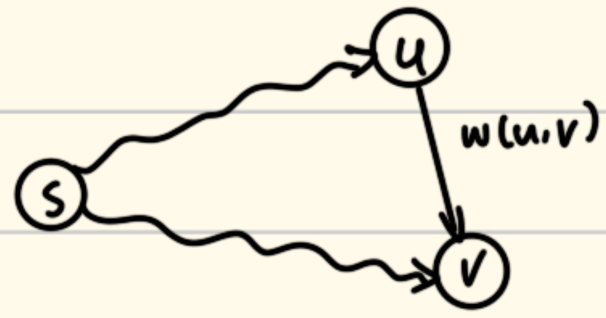
定义 $h(v) = \delta(s,v)$ (可用 Bellman-Ford 求)

证明性质 2: $\hat{w}(u,v) \geq 0$

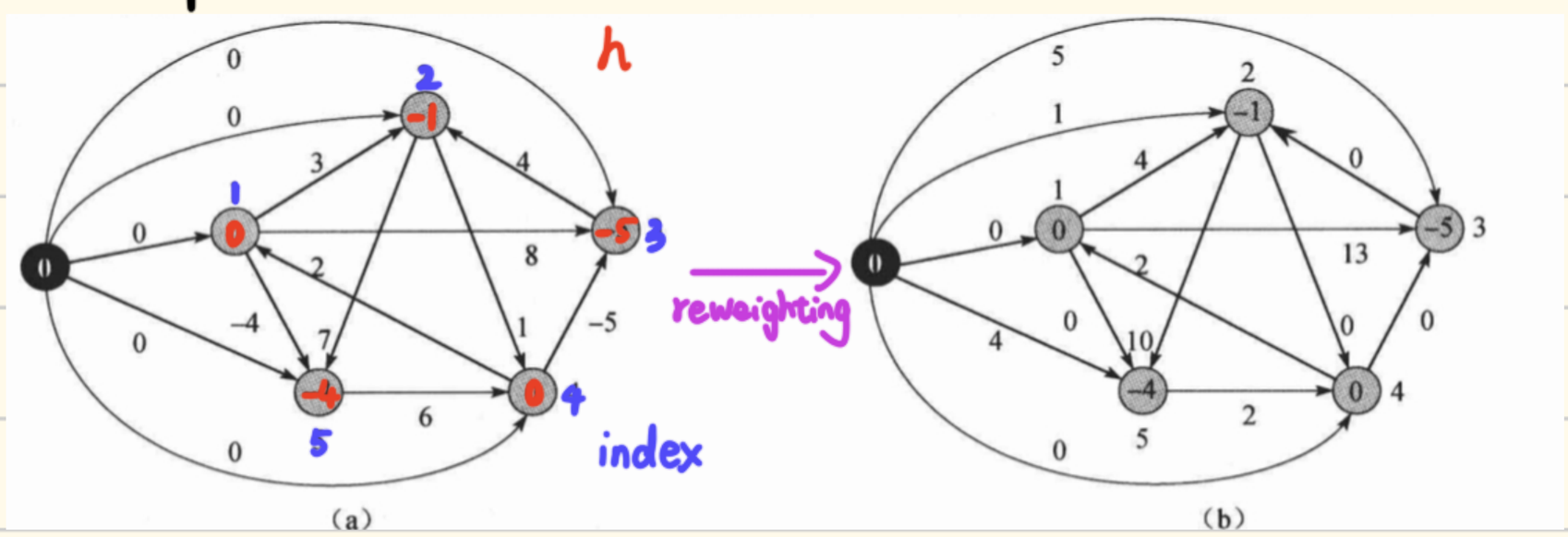
由三角不等式: $\delta(s,v) \leq \delta(s,u) + w(u,v)$

$\Rightarrow w(u,v) + \delta(s,u) - \delta(s,v) \geq 0$

$\therefore \hat{w}(u,v) \geq 0$



example:



3. Algorithms

```
JOHNSON(G, w)
1  compute G', where G'.V = G.V ∪ {s},
       G'.E = G.E ∪ {(s,v) : v ∈ G.V}, and
       w(s,v) = 0 for all v ∈ G.V
2  if BELLMAN-FORD(G', w, s) == FALSE
3      print "the input graph contains a negative-weight cycle"
4  else for each vertex v ∈ G'.V
5      set h(v) to the value of δ(s,v)
           computed by the Bellman-Ford algorithm
6      for each edge (u,v) ∈ G'.E
7          ŵ(u,v) = w(u,v) + h(u) − h(v)
8      let D = (d_uv) be a new n × n matrix
9      for each vertex u ∈ G.V
10         run DIJKSTRA(G, ŵ, u) to compute δ̂(u,v) for all v ∈ G.V
11         for each vertex v ∈ G.V
12             d_uv = δ̂(u,v) + h(v) − h(u)
13     return D
```

BF detects negative-weight cycle

BF and reweighting

run dijkstra

cut down reweight addition

Algorithm	Time Complexity	Idea
"Matrix Multiplication" (improved)	$O(V^4)/O(V^3 \lg V)$	Dynamic Programming
Floyd-Warshall Algorithm	$O(V^3)$	Dynamic Programming
Johnson Algorithm	$O(VE \lg V)$	Reweighting

II. AOE Network　　Activity On Edges

1. Critical Path (关键路径)

完成整个工程所需的时间取决于从源点到汇点的最长路径长度

即在这条路径上所有活动的时间之和

这条最长路径称为关键路径 (Critical Path)

可能有多条关键路径。不是任一关键活动加速就一定能使工程提前

想要使工程提前，要所有关键路径都能缩短

2. Critical Activity (关键活动)

① 关键活动：不按期完成就会影响整个工程完成的活动

② 关键路径上所有的活动都是关键活动

只要找到关键活动，就可找到关键路径

3. Notations：　　e: early　　l: last

(1) ev：顶点v的最早开始时间

ev(i)是从源点 V_1 至顶点 V_i 的最长路径长度

递推公式：$ev(i)=max\{ev(j)+dur(<V_j,V_i>)\}$

(2) lv：顶点 V 的最晚开始时间

保证汇点 V_n 在 ev(n) 时刻完成的前提下，事件 V_i 允许的最迟开始时间

lv(i)是ee(n)减去从 V_i 到 V_n 的最长路径长度

递推公式：$lv(i)=min\{lv(j)-dur(<V_i,V_j>)\}$

(3) ea：活动 a 的最早开始时间

设 a_x 在边 $<V_j,V_k>$ 上，则 $ea[x]=ee[j]$

(4) la：活动 a 的最迟开始时间

保证汇点 V_n 在 ev(n) 时刻完成的前提下，活动 a_x 允许的最迟开始时间

设 a_x 在边 $<V_j,V_k>$ 上，则 $la[x]=le[k]-dur(<j,k>)$

4. 时间余量/松弛时间

$la[x]-ea[x]$

在不增加工程所需总时间的情况下，活动 a_x 可拖延的时间

当 $la[x]==ea[x]$，即松弛时间为0时，则 a_x 是关键活动

5. 求关键路径的步骤

① 进行拓扑排序，对顶点进行标号

② 顺拓扑序求出 eV

③ 逆拓扑序求出 lV

④ 求 ea, la:

对于边 $<V_i, V_j>$ $ea = eV(i)$, $la = lV(j) - dur(i,j)$

⑤ 若 $ea == la$，则在关键路径上

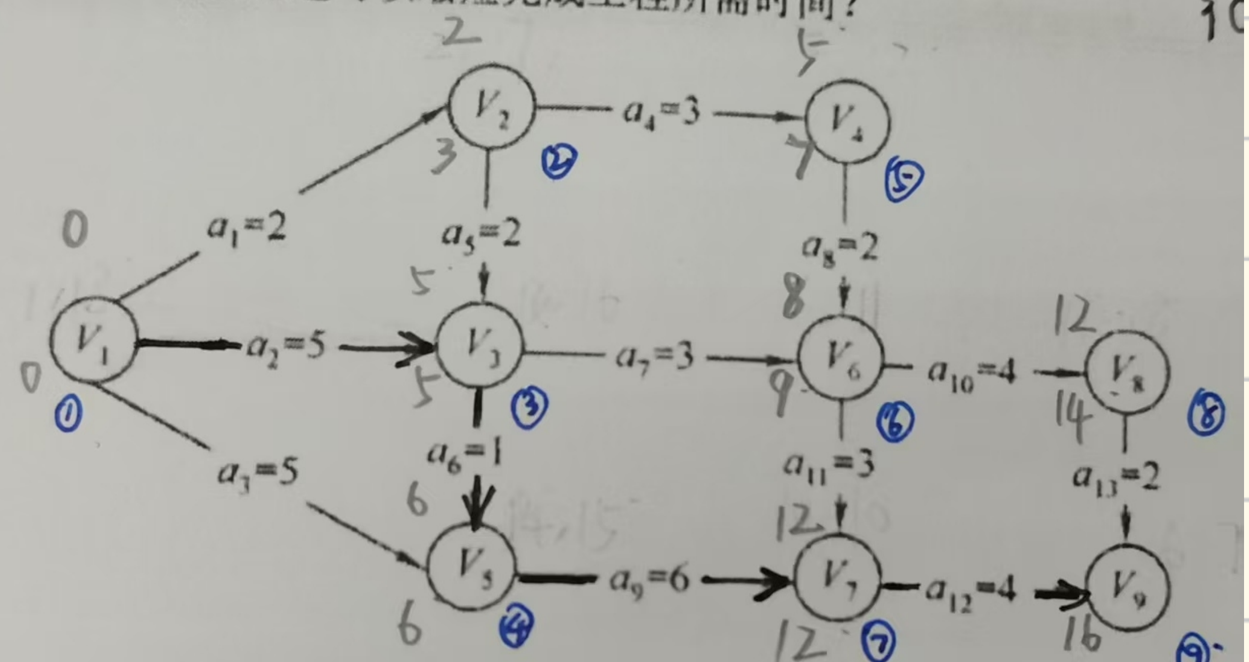
visited: 1a:0, c:2, e:5, f

Activity On Edges.

5、 下图所示为一个用 AOE 网表示的工程，试回答：

(1) 完成此工程至少需要多少时间？

(2) 那些活动加速可以缩短完成工程所需时间？



解：

(1) $5+1+6+4 = 6+6+4 = 16$.

(2) a_2, a_6, a_9, a_{12}.