



算法设计与分析

Computer Algorithm Design & Analysis
2019.11

王多强

dqwang@mail.hust.edu.cn

群名称: 2019-算法
群 号: 835135560



群名称: 2019-算法
群 号: 835135560

A decorative graphic on the left side of the slide, consisting of a black crosshair with blue, red, and yellow squares at the intersections.

Chapter 25

All-Pairs Shortest Paths

所有结点对的最短路径问题

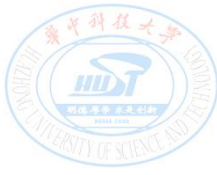
一个新的最短路径问题：

给定一个带权重的**有向图** $G=(V,E)$ ，其权重函数为 $\omega:E\rightarrow\mathbb{R}$ 。
在图中，对**所有的结点对** $u,v\in V$ ，找出从结点 u 到结点 v 的最短路径。

该问题的解以**表格**（二维数组）的形式给出：第 u 行第 v 列给出从结点 u 到结点 v 的最短路径权重。

- **一条路径的权重是组成该路径的所有边的权重之和。**

如何求解？



用单源最短路径算法求解：

执行 $|V|$ 次单源最短路径算法，每次使用一个不同的结点作为源点，从而可以求出每个结点到其他所有结点的最短路径。

- 如果所有的边的权重为非负值，用**Dijkstra**算法：

- 用线性数组实现最小优先队列： $O(V^3 + VE) = O(V^3)$;
- 用二叉堆实现最小优先队列： $O(VE \lg V)$ ；（对稀疏图较好）
- 用斐波那契堆实现最小优先队列： $O(V^2 \lg V + VE)$;

- 如果有权重为负值边，用**Bellman-Ford**算法：

- 一般的运行时间： $O(V^2E)$;
- 对稠密图，运行时间为 $O(V^4)$ 。

怎么更好地求解？

本章约定:

- 1) **结点编号**: 不失一般性, 结点编号为 $1, 2, \dots, |V|$ 。
- 2) **成本邻接矩阵**: 图 G 用一个 $n \times n$ 的邻接矩阵 $W = (w_{ij})$ 表示, 其中,

$$w_{ij} = \begin{cases} 0 & \text{若 } i = j \\ \text{有向边 } (i, j) \text{ 的权重} & \text{若 } i \neq j \text{ 且 } (i, j) \in E \\ \infty & \text{若 } i \neq j \text{ 且 } (i, j) \notin E \end{cases}$$

- 3) 允许存在权重为负值的边, 但不能包含权重为负值的环路。

➤ 否则无解。

- 4) **最短路径矩阵**: 算法的输出为一个 $n \times n$ 的最短路径矩阵

$D = (d_{ij})$, 其中 d_{ij} 表示从结点 i 到结点 j 的一条最短路径的权重。算法结束时有 $d_{ij} = \delta(i, j)$ 。

5) 前驱结点矩阵:

前驱结点矩阵记为: $\Pi = (\pi_{ij})$, 其中

$$\pi_{ij} = \begin{cases} \text{NIL}, & \text{若 } i = j \text{ 或者从 } i \text{ 到 } j \text{ 不存在路径} \\ \text{从结点 } i \text{ 到结点 } j \text{ 的某条最短路径上结点 } j \text{ 的前驱结点, 否则} \end{cases}$$

- 利用前驱结点矩阵 Π 可以计算出每对结点间的最短路径。

前驱结点矩阵 Π 的**第 i 行所诱导(induce)的子图**是一棵根结点为 i 的最短路径树。

对于每个结点 $i \in V$, 定义图 G 对于结点 i 的**前驱子图**为 $G_{\pi,i} = (V_{\pi,i}, E_{\pi,i})$, 其中

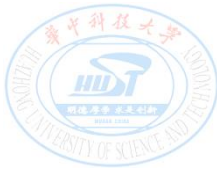
$$V_{\pi,i} = \{j \in V : \pi_{ij} \neq \text{NIL}\} \cup \{i\}$$

$$E_{\pi,i} = \{(\pi_{ij}, j) : j \in V_{\pi,i} - \{i\}\} .$$

6) PRINT-ALL-PAIRS-SHORTEST-PATH过程:

如果 $G_{\pi,i}$ 是一棵最短路径树, 则**PRINT-ALL-PAIRS-SHORTEST-PATH**过程输出从结点*i*到结点*j*的一条最短路径。

```
PRINT-ALL-PAIRS-SHORTEST-PATH( $\Pi, i, j$ )  
1  if  $i == j$   
2      print  $i$   
3  elseif  $\pi_{ij} == \text{NIL}$   
4      print “no path from”  $i$  “to”  $j$  “exists”  
5  else PRINT-ALL-PAIRS-SHORTEST-PATH( $\Pi, i, \pi_{ij}$ )  
6      print  $j$ 
```



25.1 最短路径和矩阵乘法

本节给出有向图所有结点对最短路径问题的一种**动态规划算法**。时间复杂度 $\Theta(V^4)$ ，然后改进到 $\Theta(V^3 \lg V)$ 。

■ 最短路径的最优子结构性质：每条路径都是最短路径

考虑从结点*i*到结点*j*的一条最短路径*p*。假定*p*至多包含*m*条边（假定没有权重为负值的环路），且*m*为有限值。

- 如果*i=j*，则*p*中不包含任何边，所以*p*的权重等于0；
- 如果*i≠j*，则将路径*p*分解为 $i \xrightarrow{p'} k \rightarrow j$ ，其中
 - *p'*至多包含*m-1*条边，根据引理24.1，*p'*是从*i*到*k*的一条最短路径，
 - 且 $\delta(i, j) = \delta(i, k) + w_{kj}$ 。



■ 递归解

设 $l_{ij}^{(m)}$ 是从结点i到结点j的**至多包含m条边**的任意路径中的最小权重。

- ▶ 当 $m=0$ 时，表示从结点i到结点j**中间没有边**的最短路径，
所以有

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

- ▶ 当 $m \geq 1$ 时，可以在 $l_{ij}^{(m-1)}$ 的基础上计算从i到j的**最多由m条边**组成的任意路径的最小权重 $l_{ij}^{(m)}$ 。
- ▶ $l_{ij}^{(m-1)}$ 是从i到j最多由m-1条边组成的最短路径的权重。

- $l_{ij}^{(m)}$ 的计算：通过对j的所有可能的前驱k进行检查获得。

$$\begin{aligned} l_{ij}^{(m)} \text{ 的递归定义: } l_{ij}^{(m)} &= \min \left(l_{ij}^{(m-1)}, \min_{1 \leq k \leq n} \{ l_{ik}^{(m-1)} + w_{kj} \} \right) \\ &= \min_{1 \leq k \leq n} \{ l_{ik}^{(m-1)} + w_{kj} \} . \end{aligned}$$

含有 $l_{ij}^{(m-1)}$

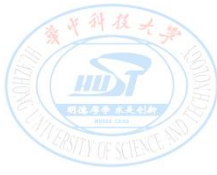
- 结点间的最短路径权重 $\delta(i,j)$

如果图G不包含权重为负值的环路，则对于每一对结点i和j，如果 $\delta(i,j) < \infty$ ，则从i到j之间存在一条最短路径。并且，由于最短路径是简单路径，其中至多包含n-1条边，因此有：

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

且：

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



■ 自底向上计算最短路径权重

(1) $I_{ij}^{(1)}$ 表示从结点i到结点j中间至多包含1条边的路径的最短权重，因此 $I_{ij}^{(1)} = w_{ij}$ 。

(2) 自底向上计算方法：

根据输入矩阵 $W = (w_{ij})$ ，计算 $L(1) = (I_{ij}^{(1)})$ ，然后根据递归关系式计算 $L(2)$ 、 \dots 、 $L(n-1)$ 。其中，

$$L^1 = (w_{ij}) = W$$

$$L^m = I_{ij}^{(m)}, \quad m=1, 2, \dots, n-1.$$

(3) $L(n-1)$ 即是最后的最短路径权重结果矩阵，有


$$I_{ij}^{(n-1)} = \delta(i, j)$$

自底向上计算算法的伪代码表示

EXTEND-SHORTEST-PATHS算法在给定 W 和 $L^{(m-1)}$ 的情况下计算 $L^{(m)}$ ：

EXTEND-SHORTEST-PATHS(L, W)

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



- 算法将最近计算出的最短路径扩展一条边：
 - 找一条从 i 到某结点 k ，再经过边 (k, j) 到达 j 的更短路径。
- 算法结束时返回结果矩阵 $L' = L^{(m)}$ 。
- 时间复杂度 $O(n^3)$ 。

最短路径和矩阵乘法

上述一次计算最短路径结果矩阵 $L^{(m)}$ 的过程和矩阵乘法在整体框架上是一致的：(详见P401)

EXTEND-SHORTEST-PATHS(L, W)

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

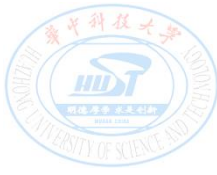
SQUARE-MATRIX-MULTIPLY(A, B)

```
1   $n = A.rows$ 
2  let  $C$  be a new  $n \times n$  matrix
3  for  $i = 1$  to  $n$ 
4      for  $j = 1$  to  $n$ 
5           $c_{ij} = 0$ 
6          for  $k = 1$  to  $n$ 
7               $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$ 
8  return  $C$ 
```


$$L \cdot W \equiv A \cdot B$$

时间复杂度都是 $O(n^3)$

一次“ L 与 W 的乘 $L \cdot W$ ”完成一次从 $L^{(m-1)}$ 向 $L^{(m)}$ 的计算。



所以从上述L的计算过程以及与矩阵乘法的对比可以看出：

设 $A \bullet B$ 表示由算法EXTEND-SHORTEST-PATHS(A, B)返回的矩阵“乘积”，可以得到以下从 $L^{(1)}$ 到 $L^{(n-1)}$ 的计算序列：

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

矩阵 $L^{(n-1)}=W^{(n-1)}$ 将包含最短路径权重

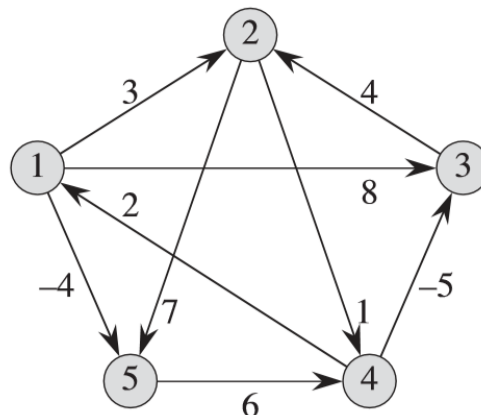
计算矩阵序列的过程：

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

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

时间复杂度： $O(n^4)$

例：已知有向图：



EXTEND-SHORTEST-PATHS(L, W)

```

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

SLOW-ALL-PAIRS-SHORTEST-PATHS的计算过程：

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

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

矩阵序列： $L^{(1)} \rightarrow L^{(2)} \rightarrow L^{(3)} \rightarrow L^{(4)}$ 。而任何 $m \geq 4$ ， $L^{(m)} = L^{(4)}$ 。



改进的算法:

我们的目标是算出矩阵 $L^{(n-1)}$, 而对所有 $m \geq n-1$, $L^{(m)} = L^{(n-1)}$, 因此可用以下 $\lceil \lg(n-1) \rceil$ 个矩阵乘过程来计算矩阵 $L^{(n-1)}$, 而不是依次计算所有的 $L^{(1)} \sim L^{(n-1)}$ 矩阵。

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

注: 由于 $2^{\lceil \lg(n-1) \rceil} \geq n-1$, 所以最后的乘积 $L^{(2^{\lceil \lg(n-1) \rceil})}$ 等于 $L^{(n-1)}$ 。

改进的计算过程如下:

下面的过程使用 “**重复平方技术**” (**repeated squaring**) 来计算上述矩阵序列:

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  $n \times n$  matrix
6       $L^{(2m)} = \text{EXTEND-SHORTEST-PATHS}(L^{(m)}, L^{(m)})$ 
7       $m = 2m$ 
8  return  $L^{(m)}$ 
```

```
EXTEND-SHORTEST-PATHS( $L, W$ )
1   $n = L.rows$ 
2  let  $L' = (l'_{ij})$  be a new  $n \times n$  matrix
3  for  $i = 1$  to  $n$ 
4      for  $j = 1$  to  $n$ 
5           $l'_{ij} = \infty$ 
6          for  $k = 1$  to  $n$ 
7               $l'_{ij} = \min(l'_{ij}, l_{ik} + w_{kj})$ 
8  return  $L'$ 
```

- 计算从 $m=1$ 开始, 每次迭代后对 m 加倍;
- 最后计算的 $L^{(2m)}$ 即是 $L^{(n-1)}$: $L^{(2m)} = L^{(n-1)}$, 其中 $n-1 \leq 2m \leq 2n-2$ 。
- 算法的运行时间是: $O(n^3 \lg n)$ 。



25.2 Floyd-Warshall算法

本节讨论另一种**动态规划**策略来求解有向图的所有结点对最短路径问题——**Floyd-Warshall算法**。

- 算法的时间复杂度 $\Theta(V^3)$ 。
- 算法允许图中存在负权重的边，但不能存在权重为负值的环路。

■ 最短路径结构的重新描述：

中间结点：一条简单路径 $p = \langle v_1, v_2, \dots, v_l \rangle$ 上的**中间结点**是指路径 p 上除 v_1 和 v_l 之外的其它任意结点。

假定图 G 的结点集为 $V=\{1,2,\dots,n\}$ 。考虑其中的一个子集 $\{1,2,\dots,k\}$ ，这里 k 是小于 n 的某个整数，并是其中的最大编号。

对于任意一对结点 $i, j \in V$ ，定义 p 是从 i 到 j 、且所有中间结点均取自于集合 $\{1,2,\dots,k\}$ 的最短路径。含义：

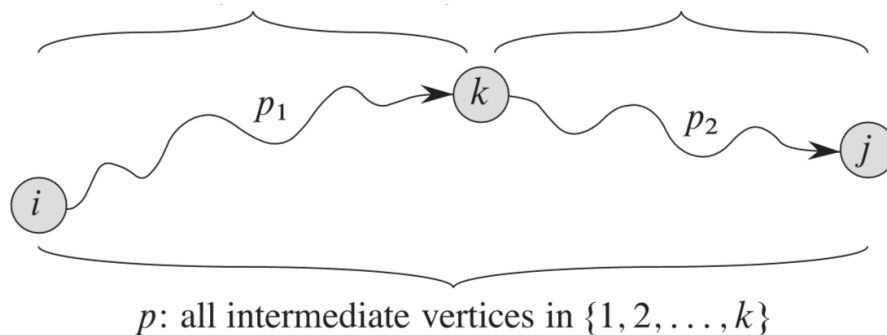
- p 是简单路径，且 p 的中间结点都不大于 k 。
- p 从 i 到 j ，仅经过集合 $\{1, 2, \dots, k\}$ 中的结点，但，
 - 不一定经过其中的每一个结点，且与顺序无关；
 - 也可能不存在这样的路径，此时 p 的权重等于 ∞ 。

在从 i 到 j 之间中间结点均取自集合 $\{1,2,\dots,k-1\}$ 的基础上，试图回答这样一个问题：结点 k 是否是路径 p 上的一个中间结点？

结点k是否是路径p上的一个中间结点?

- 情况1: 如果结点k不是路径p上的中间结点, 则p上的所有中间结点都属于集合 $\{1, 2, \dots, k-1\}$ 。
 - 此时, 从结点i到结点j的中间结点取自集合 $\{1, 2, \dots, k-1\}$ 的一条最短路径也是从结点i到结点j的中间结点取自集合 $\{1, 2, \dots, k\}$ 的一条最短路径。

- 情况2：如果结点k是路径p上的中间结点，则k将路径p分解为两段： $i \xrightarrow{p_1} k \xrightarrow{p_2} j$



- 根据引理24.1（最优子结构性）， p_1 是从结点i到结点k的一条最短路径，且中间结点全部取自集合 $\{1, 2, \dots, k-1\}$ 。

这是因为结点k不是路径 p_1 上的中间结点，所以路径 p_1 上的所有结点都属于集合 $\{1, 2, \dots, k-1\}$ 。

- 同理， p_2 是从结点k到结点j的一条最短路径，且中间结点全部取自集合 $\{1, 2, \dots, k-1\}$ 。

状态转移方程：

设 $d_{ij}^{(k)}$ 为从结点i到结点j的所有中间结点全部取自集合 $\{1, 2, \dots, k\}$ 的一条最短路径的权重，则有：

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

其中，

- $k=0$ 时，代表从结点i到结点j的一条不包含编号大于0的中间结点的路径，这样的路径没有任何中间结点，最多只有一条边。所以 $d_{ij}^{(0)} = w_{ij}$ 。

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

- 而因为任何路径的中间结点都属于集合 $\{1, 2, \dots, n\}$ ，所以 $k=n$ 时， $d_{ij}^{(n)}$ 给出所有可能的从结点 i 到结点 j 的中间结点均取自集合 $\{1, 2, \dots, n\}$ 的一条最短路径的权重，也就是从结点 i 到结点 j 的最短路径的权重。所以对所有的 $i, j \in V$ 有：

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

- 矩阵 $D^{(n)} = (\hat{d}_{ij}^{(n)})$ 为结果矩阵。

算法描述:

FLOYD-WARSHALL(W)

```
1   $n = W.rows$ 
2   $D^{(0)} = W$ 
3  for  $k = 1$  to  $n$ 
4      let  $D^{(k)} = (d_{ij}^{(k)})$  be a new  $n \times n$  matrix
5      for  $i = 1$  to  $n$ 
6          for  $j = 1$  to  $n$ 
7               $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$ 
8  return  $D^{(n)}$ 
```

- $W_{n \times n}$: 权重邻接矩阵;
- $D_{n \times n}$: 最短路径权重矩阵
- FLOYD-WARSHALL自底向上地完成D矩阵的计算。
- D矩阵的计算可以在原址上完成。

构建最短路径

在计算矩阵 $D^{(k)}$ 的同时，计算**前驱矩阵** Π 序列： $\Pi^{(0)}$ ， $\Pi^{(1)}$ ，
 \dots ， $\Pi^{(n)} = \Pi$ 。

其中，

- $\pi_{ij}^{(k)}$ 为从结点 i 到结点 j 的一条所有中间结点都取自集合 $\{1, 2, \dots, k\}$ 的最短路径上 j 的前驱结点。

- $k=0$ 时：

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

- $k=0$ 时，是一条从 i 到 j 的没有中间结点的最短路径，所以当路径存在时($w_{ij} < \infty$)， j 的前驱就是 i 。

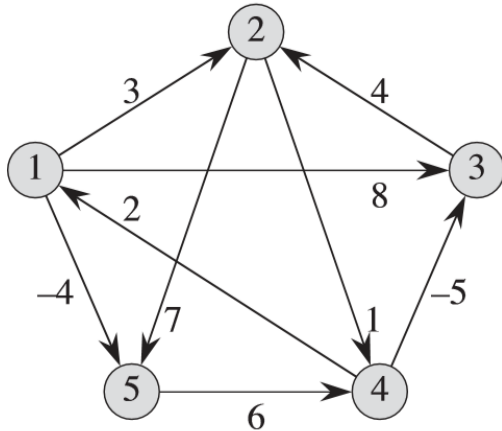
- $k \geq 1$ 时, 从结点i到结点j的一条所有中间结点都取自集合 $\{1, 2, \dots, k\}$ 的最短路径或者经过k, 或者不经过k。

- 若不经过k, 则有 $d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$
 - 此时求从结点i到结点j的所有中间结点都取自集合 $\{1, 2, \dots, k\}$ 的最短路径上的j的前驱等价于求从结点i到结点j的所有中间结点都取自集合 $\{1, 2, \dots, k-1\}$ 的最短路径上的j的前驱。
- 若经过k, 则有 $d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$
 - 此时求从结点i到结点j的所有中间结点都取自集合 $\{1, 2, \dots, k\}$ 的最短路径上的j的前驱等价于求从结点k到结点j的所有中间结点都取自集合 $\{1, 2, \dots, k-1\}$ 的最短路径上的j的前驱。

所以有:

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

例：对下例， FLOYD-WARSHALL算法的计算过程如下：



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

$$\Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

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

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

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

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

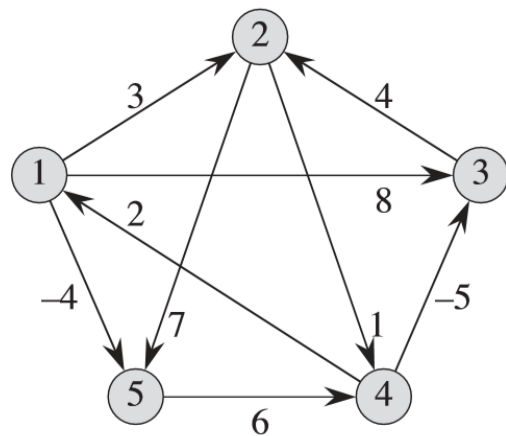
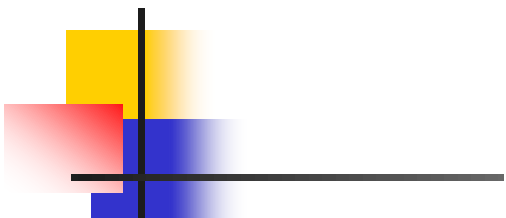
$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & \underline{-1} & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

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

FLOYD-WARSHALL(W)

```

1   $n = W.rows$ 
2   $D^{(0)} = W$ 
3  for  $k = 1$  to  $n$ 
4      let  $D^{(k)} = (d_{ij}^{(k)})$  be a new  $n \times n$  matrix
5      for  $i = 1$  to  $n$ 
6          for  $j = 1$  to  $n$ 
7               $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$ 
8  return  $D^{(n)}$ 
    
```



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



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



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

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



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



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

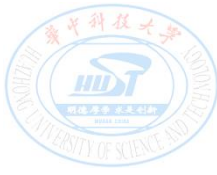
$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}$$

算法时间分析

- 3层嵌套的for循环，所以时间是： $\Theta(n^3)$ 。

```
FLOYD-WARSHALL( $W$ )
1   $n = W.rows$ 
2   $D^{(0)} = W$ 
3  for  $k = 1$  to  $n$ 
4      let  $D^{(k)} = (d_{ij}^{(k)})$  be a new  $n \times n$  matrix
5      for  $i = 1$  to  $n$ 
6          for  $j = 1$  to  $n$ 
7               $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$ 
8  return  $D^{(n)}$ 
```

思考：怎么在上述算法中加入计算 Π 的代码？



有向图的传递闭包

对有向图 $G=(V, E)$ ，定义图 G 的传递闭包 $G^*=(V, E^*)$ ，其中

$E^*=\{(i, j): \text{如果图} G \text{中包含一条从结点} i \text{到结点} j \text{的路径}\}$ 。

求有向图的传递闭包：

方法一： 给 E 中每条边赋权重1，然后运行FLOYD-WARSHALL算法，可以在 $\Theta(n^3)$ 求出权重路径矩阵 D 。在 D 中若 $d_{ij} < n$ ，则表示存在一条从结点 i 到结点 j 的路径；否则 $d_{ij} = \infty$ 。

方法二： 定义矩阵 $T = \{t_{ij}\}$ ，若存在一条从结点 i 到结点 j 的路径， $t_{ij}=1$ ，否则 $t_{ij}=0$ 。

计算T

对FLOYD-WARSHALL算法进行改造：用**逻辑或**操作（**V**）和**逻辑与**操作（**^**）替换算术操作min和+，得以下计算公式：

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

$$k \geq 1\text{时}, \quad t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}) .$$

有向图的传递闭包计算过程如下:

TRANSITIVE-CLOSURE(G)

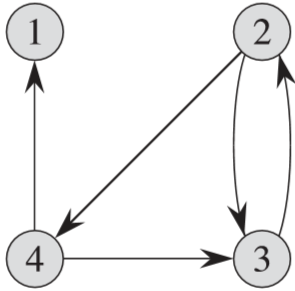
```
1   $n = |G.V|$ 
2  let  $T^{(0)} = (t_{ij}^{(0)})$  be a new  $n \times n$  matrix
3  for  $i = 1$  to  $n$ 
4      for  $j = 1$  to  $n$ 
5          if  $i == j$  or  $(i, j) \in G.E$ 
6               $t_{ij}^{(0)} = 1$ 
7          else  $t_{ij}^{(0)} = 0$ 
8  for  $k = 1$  to  $n$ 
9      let  $T^{(k)} = (t_{ij}^{(k)})$  be a new  $n \times n$  matrix
10     for  $i = 1$  to  $n$ 
11         for  $j = 1$  to  $n$ 
12              $t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$ 
13 return  $T^{(n)}$ 
```

按照 k 的递增次序依次
计算 $T^{(k)} = (t_{ij}^{(k)})$

时间复杂度: $\Theta(n^3)$

注: 逻辑运算比算术运算快; 空间需求也较小

例，求如图所示的图G的传递闭包矩阵T。



$$T^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$T^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & \underline{1} \\ 0 & \underline{1} & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$T^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & \underline{1} \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$T^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & \underline{1} & 1 & 1 \end{pmatrix}$$

$$T^{(4)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \underline{1} & 1 & 1 & 1 \\ \underline{1} & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$



25.4 用于稀疏图的Johnson算法

对于有 $|V|$ 个结点、 $|E|$ 条边的有向图，求每对结点之间的最短路径：

- FLOYD-WARSHALL算法的时间复杂度： $\Theta(n^3)$
- 用单源最短路径算法，执行 $|V|$ 次单源最短路径算法，每次使用一个不同的结点作为源点，求出每个结点到其他所有结点的最短路径。
 - 如果所有的边的权重为非负值，用Dijkstra算法：
 - 用线性数组实现最小优先队列： $O(V(V^2+E))=O(V^3)$ ；
 - 用二叉堆实现最小优先队列： $O(VE \lg V)$ ；（对稀疏图较好）
 - 用斐波那契堆实现最小优先队列： $O(V^2 \lg V + VE)$ ；

A decorative graphic in the top-left corner consisting of overlapping yellow, red, and blue squares with a black crosshair.

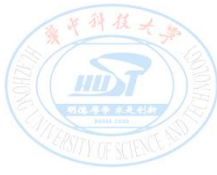
■ 用单源最短路径算法

- 如果有权重为负值边，用Bellman-Ford算法：
 - 一般的运行时间： $O(V^2E)$ ；
 - 对稠密图，运行时间为 $O(V^4)$ 。



Johnson算法：在**稀疏图**中求每对结点之间的最短路径权重。

- 对稀疏图，Johnson算法优于Floyd-Warshall算法，时间复杂度可达 $O(V^2 \lg V + VE)$ 。
- Johnson算法使用**Dijkstra算法**和**Bellman-Ford算法**作为自己的子程序，可处理带有**负权重的图**。
- 如果图中包含所有结点对的最短路径，Johnson算法输出一个包含所有结点对的最短路径权重矩阵；否则报告图中包含权重为负值的环路。



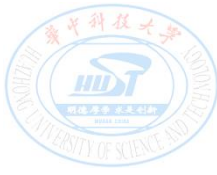
重赋权重：Johnson算法使用**重新赋予权重**的技术求解。

工作原理：

- 如果图 $G=(V, E)$ 中所有的边权重 w 皆为非负值，则通过对每个结点运行一次**Dijkstra**算法来找到所有结点对之间的最短路径；
- 如果图 G 包含权重为负值的边，但没有权重为负值的环路，则通过**重赋权重**，构造出一组**新的非负权重值**，然后使用上面同样的方法求解(**Dijkstra**算法)。

新赋予的权重函数记为： \hat{w}

- \hat{w} 必须满足以下两个重要性质：



新赋予的权重 \hat{w} 必须满足以下两个重要性质：

1. **路径等价性**：对于所有结点对 $u, v \in V$ ，一条路径 p 是在使用权重函数 w 时的从结点 u 到结点 v 的一条最短路径，**当且仅当** p 是在使用权重 \hat{w} 时的从 u 到 v 的一条最短路径。
 - 即不管是使用原来的权重函数还是新的权重函数，所能求出来的最短路径应是一致的。
2. **非负性**：对于所有的边 (u, v) ，新权重 $\hat{w}(u, v)$ 为非负值。
 - 即，需要经过技术处理，把负权重的边的权重改造成非负值。

如何找这样的新权重函数呢？



下面的引理，使得我们可以对边的权重重新赋值，
以满足上面的两个条件：

这里，

- 用 δ 表示从权重函数 ω 所导出的最短路径权重；
- 用 $\hat{\delta}$ 表示从权重函数 $\hat{\omega}$ 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 $\omega: E \rightarrow \mathbb{R}$, 设 $h: V \rightarrow \mathbb{R}$ 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

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

设 $p = \langle v_0, v_1, \dots, v_k \rangle$ 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 ω 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: $\omega(p) = \delta(v_0, v_k)$ 当且仅当 $\hat{w}(p) = \hat{\delta}(v_0, v_k)$ 。

而且, 图 G 在使用权重函数 ω 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} 也不包含权重为负值的环路。

证明:

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

根据定义, 对于边有: $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$.

所以有: $\hat{w}(p) = \sum_{i=1}^k \hat{w}(v_{i-1}, v_i)$ $p = \langle v_0, v_1, \dots, v_k \rangle$

$$\begin{aligned} &= \sum_{i=1}^k (w(v_{i-1}, v_i) + h(v_{i-1}) - h(v_i)) \\ &= \sum_{i=1}^k w(v_{i-1}, v_i) + h(v_0) - h(v_k) \\ &= w(p) + h(v_0) - h(v_k). \end{aligned}$$

因为 $h(v_0)$ 和 $h(v_k)$ 不依赖于任何具体路径, 因此, 如果从结点 v_0 到结点 v_k 的一条路径在使用权重函数 w 时比另一条路径短, 则其在使用权重函数 \hat{w} 时也比另一条短。因此 $w(p) = \delta(v_0, v_k)$ 当且仅当 $\hat{w}(p) = \hat{\delta}(v_0, v_k)$

然后证明：

若G在使用权重函数 w 时包含一个权重为负值的环路当且仅当p在使用权重函数 \hat{w} 也包含一个权重为负值的环路。

考虑任意环路 $c = \langle v_0, v_1, \dots, v_k \rangle$ ，其中 $v_0 = v_k$ 。

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

所以有
$$\begin{aligned}\hat{w}(c) &= w(c) + \underline{h(v_0) - h(v_k)} \\ &= w(c),\end{aligned}$$

因此，环路c在使用权重函数 w 时为负当且仅当在使用权重函数 \hat{w} 时也为负值。证毕。

剩下的问题就是到哪里去找能使 \hat{w} 为非负的h函数映射

通过重新赋值来生成非负权重

对于图 G 构造一幅新图 $G'=(V', E')$ ，其中

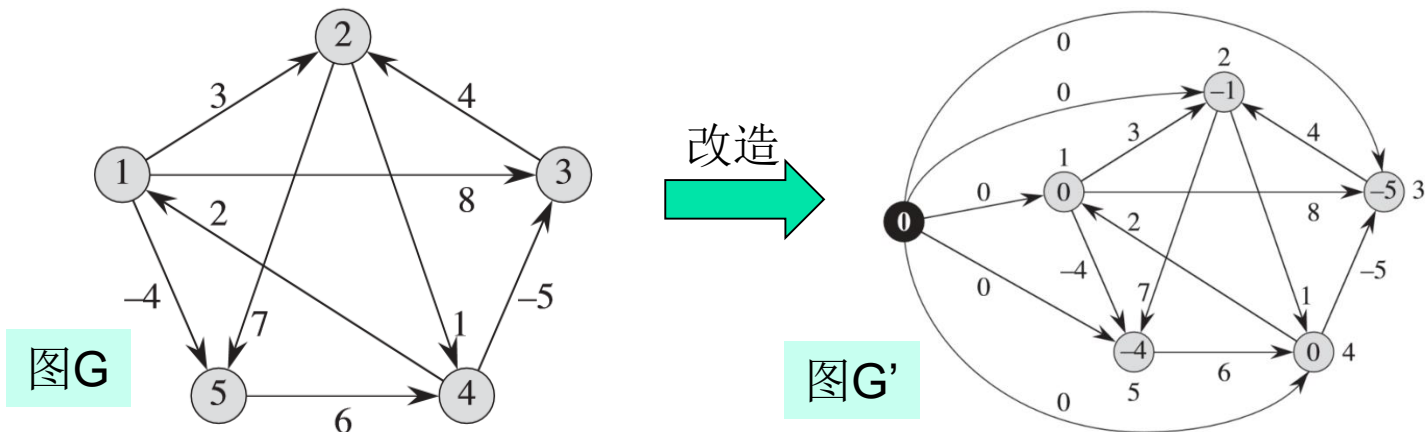
$$V' = V \cup \{s\}, \quad s \text{ 是一个新结点}, \quad s \notin V,$$

$$E' = E \cup \{(s, v) : v \in V\} \quad .$$

并令，对于所有结点 $v \in V$ ，有 $\omega(s, v)=0$ 。

注：由于结点 s 没有入边，所以除了以 s 为源点的最短路径外，图 G' 中没有其它包含 s 的最短路径。而且 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。

如：



假定图 G 和图 G' 都不包含权重为负值的环路。

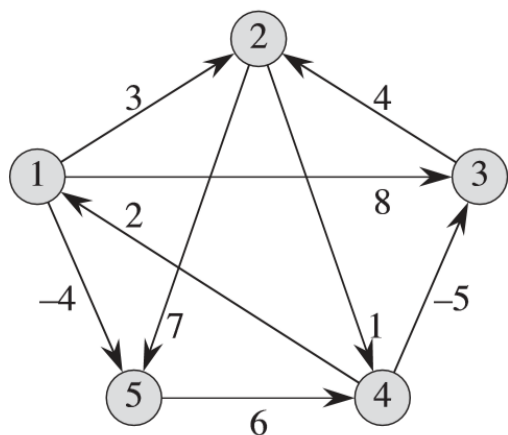
对于所有的结点 $v \in V'$ ，定义： $h(v) = \delta(s, v)$ 。

- 根据三角不等式，对于所有的边 $(u, v) \in E'$ ，有

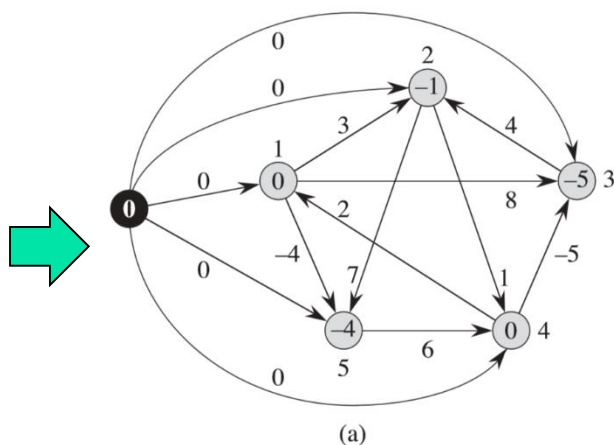
$$h(v) \leq h(u) + \omega(u, v)。$$

定义新权重 \hat{w} ，有 $\hat{w}(u, v) = w(u, v) + \underline{h(u) - h(v)} \geq 0$

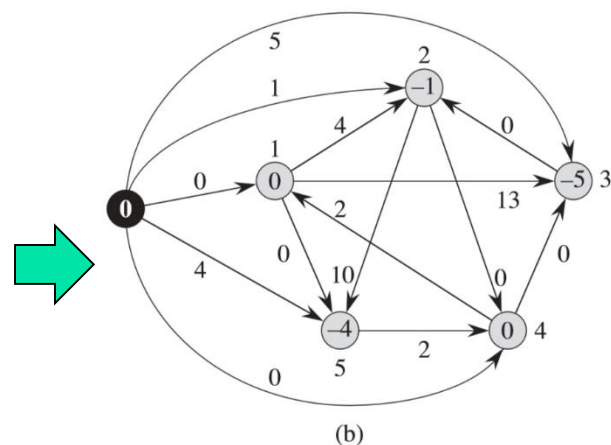
即满足性质二的要求（性质1已经证明成立）。



图G



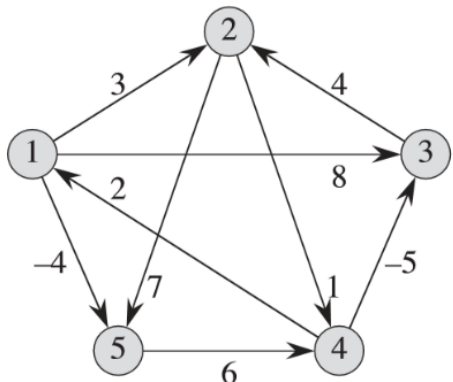
(a)



(b)

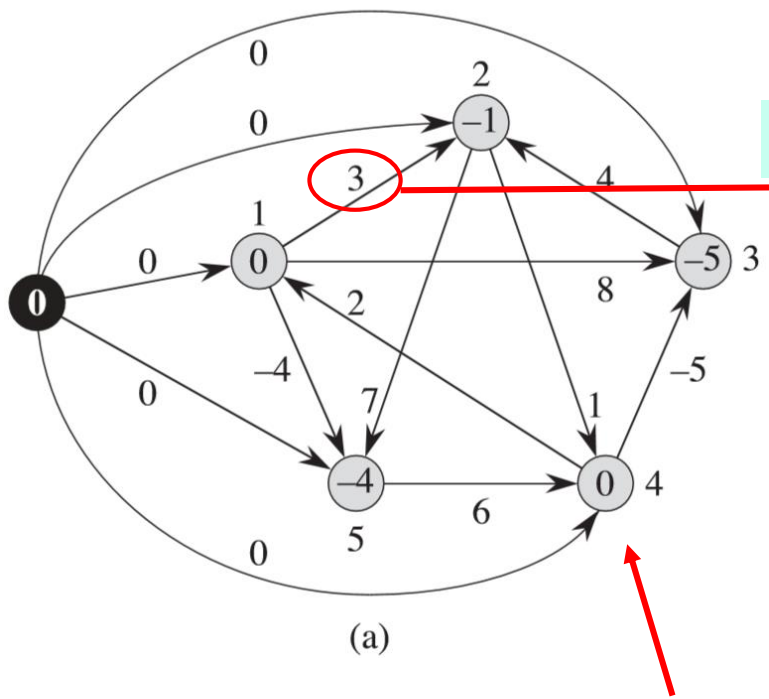
重新赋予权重的图 G'
每条边的新权重函数为：
 $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$

结点里标记的是 $h(v) = \delta(s, v)$ 的值



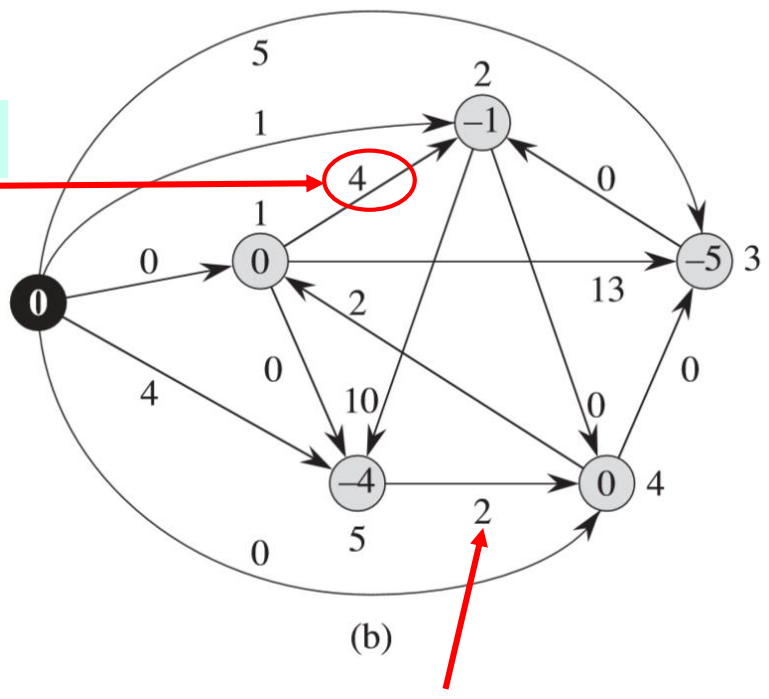
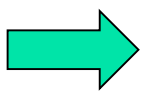
图G

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



(a)

结点里标记的是 $h(v) = \delta(s, v)$ 的值

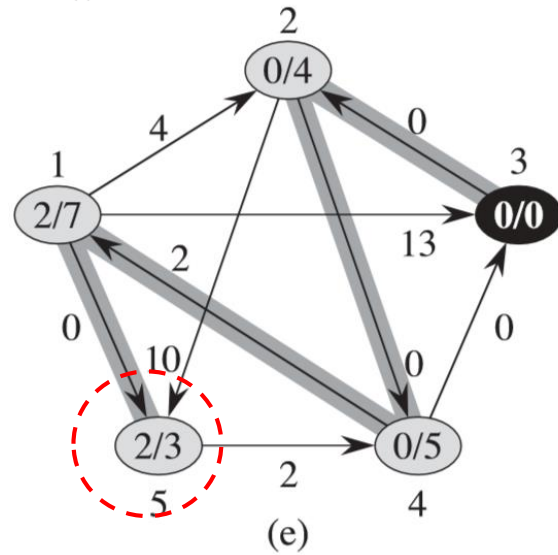
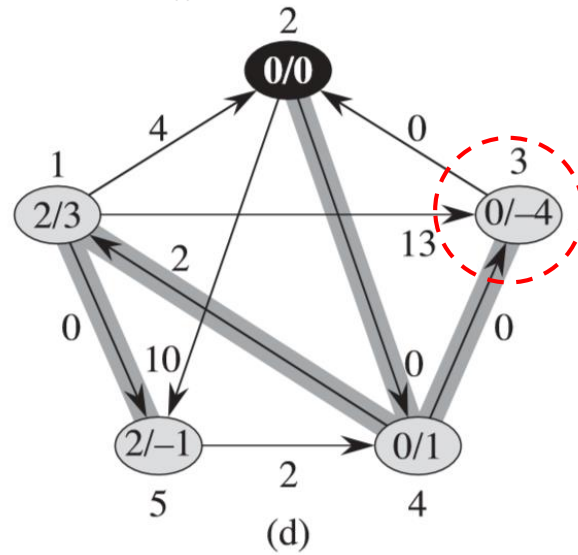
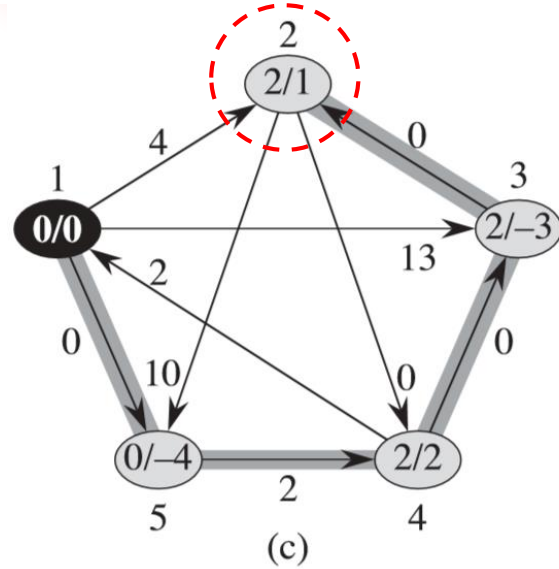
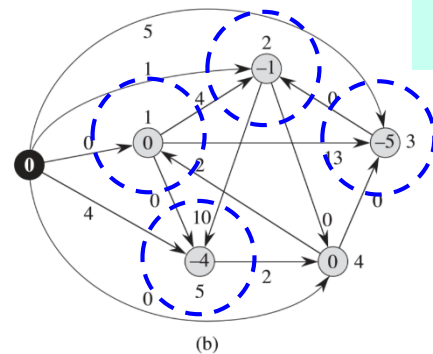
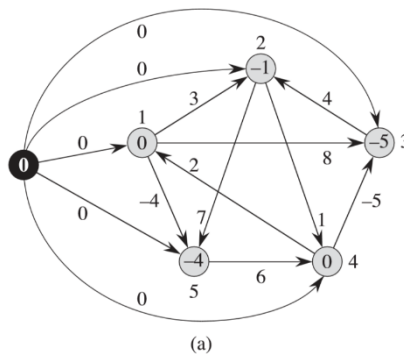
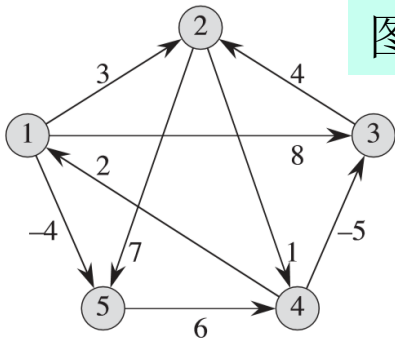


(b)

重新赋予权重的图G'
 每条边的新权重函数为:
 $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$

图G

图G'



- (c)~(g)使用权重函数 \hat{w} 在G的每个结点上运行Dijkstra算法。
- 黑色结点是当前源结点u。
- 加了阴影的边是由算法计算出来的属于最短路径树里面的边。
- 每个结点里面标记的是 $\hat{\delta}(u, v) / \delta(u, v)$ 的值。还原：

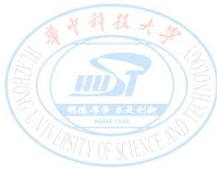
$$d_{uv} = \delta(u, v) = \hat{\delta}(u, v) + h(v) - h(u)$$

如：

$$\begin{aligned} d_{12} &= 2 + h(2) - h(1) \\ &= 2 + (-1) - 0 = 1 \end{aligned}$$

$$\begin{aligned} d_{23} &= 0 + h(3) - h(2) \\ &= 0 + (-5) - (-1) = -4 \end{aligned}$$

$$\begin{aligned} d_{35} &= 2 + h(5) - h(3) \\ &= 2 + (-4) - (-5) = 3 \end{aligned}$$



Johnson算法计算所有结点对之间的最短路径

Johnson算法使用Bellman-Ford算法和Dijkstra算法作为子程序来计算所有结点之间的最短路径。

JOHNSON(G, w)

```
1  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  $v \in G.V$   
2  if BELLMAN-FORD( $G', w, s$ ) == FALSE  
3    print "the input graph contains a negative-weight cycle"  
4  else for each vertex  $v \in G'.V$   
5    set  $h(v)$  to the value of  $\delta(s, v)$   
   computed by the Bellman-Ford algorithm  
6  for each edge  $(u, v) \in G'.E$   
7     $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$   
8  let  $D = (d_{uv})$  be a new  $n \times n$  matrix  
9  for each vertex  $u \in G.V$   
10   run DIJKSTRA( $G, \hat{w}, u$ ) to compute  $\hat{\delta}(u, v)$  for all  $v \in G.V$   
11   for each vertex  $v \in G.V$   
12      $d_{uv} = \hat{\delta}(u, v) + h(v) - h(u)$   
13  return  $D$ 
```

先用Bellman-Ford算法计算s到各个结点的最短路径，并判断是否存在负权重的环路。

利用Bellman-Ford算法计算得到的 $\delta(s, v)$ 定义 $h(v)$ 的值，并重新赋边的权重。

对每个结点执行Dijkstra算法，得到路径权重。

还原路径权重，得到原图的路径权重矩阵D。

JOHNSON(G, w)

```
1  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  $v \in G.V$   
2  if BELLMAN-FORD( $G', w, s$ ) == FALSE  
3      print “the input graph contains a negative-weight cycle”  
4  else for each vertex  $v \in G'.V$   
5      set  $h(v)$  to the value of  $\delta(s, v)$   
        computed by the Bellman-Ford algorithm  
6  for each edge  $(u, v) \in G'.E$   
7       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$   
8  let  $D = (d_{uv})$  be a new  $n \times n$  matrix  
9  for each vertex  $u \in G.V$   
10     run DIJKSTRA( $G, \hat{w}, u$ ) to compute  $\hat{\delta}(u, v)$  for all  $v \in G.V$   
11     for each vertex  $v \in G.V$   
12          $d_{uv} = \hat{\delta}(u, v) + h(v) - h(u)$   
13  return  $D$ 
```

- Johnson算法假定所有的边都保存在邻接表里。返回一个 $|V| \times |V|$ 的矩阵 $D = d_{ij}$ ，其中 $d_{ij} = \delta(i, j)$ 。
- 或者报告图 G 中包含权重为负值的环路。



时间分析:

算法的运行时间依赖于Dijkstra算法中最小优先队列的实现方式:

- 如果使用斐波那契堆实现, 则Johnson算法的运行时间为 $O(V^2 \lg V + VE)$ 。
 $O(V \lg V + E)$
- 如果使用二叉最小堆实现, 则Johnson算法的运行时间为 $O(VE \lg V)$ 。
 $O((V+E) \lg V)$
- 在稀疏图的情况下, 该算法的时间比Floyd-Warshall算法的表现($O(V^3)$)要好。

22-25章作业

- 计算题:
 - 24.1-1
 - 24.4-1
 - 25.2-1
- 设计、证明题:
 - 24.1-3
 - 24-3
 - 25.2-7
- 思考题: 25.2-6