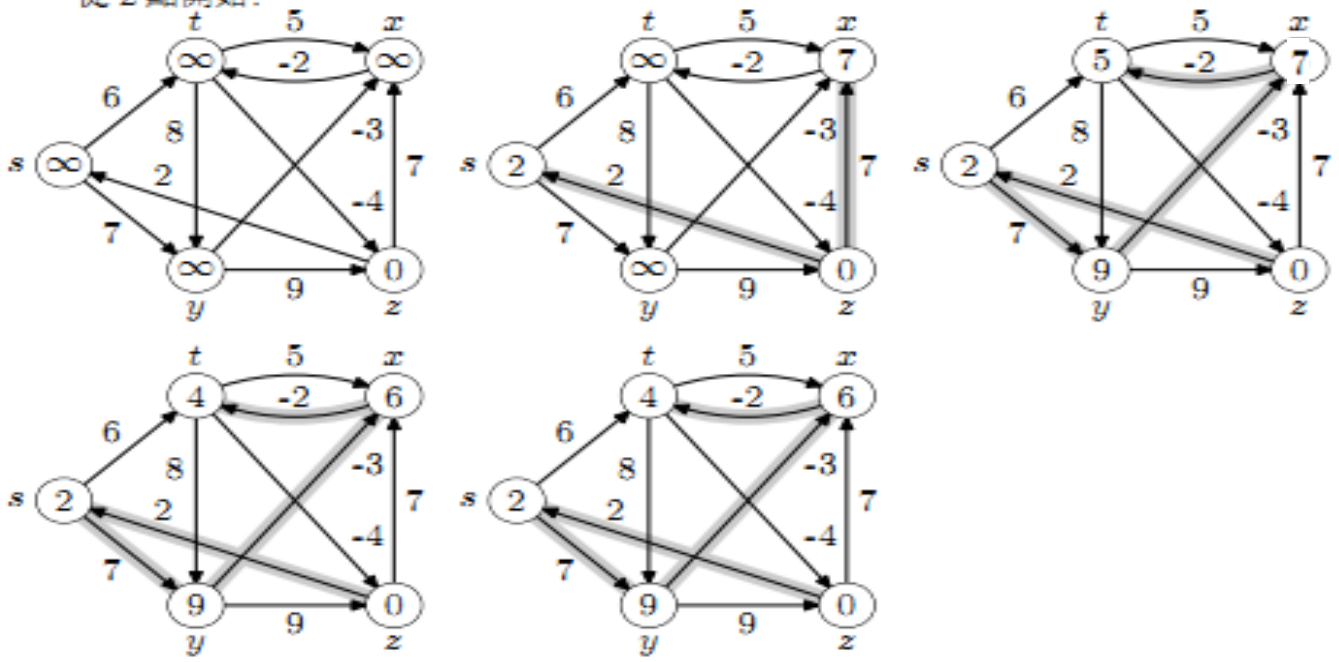
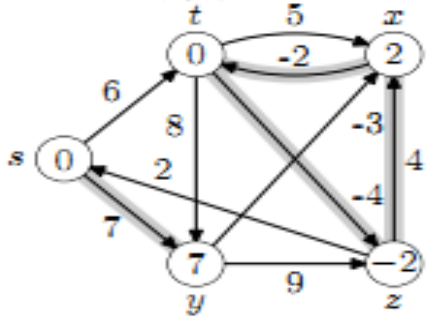
22-25章作业

24.1-1 在图 24-4 上运行 Bellman-Ford 算法，使用结点 z 作为源结点。在每一遍松弛过程中，以图中相同的次序对每条边进行松弛，给出每遍松弛操作后的 d 值和 π 值。然后，把边 (z, x) 的权重改为 4，再次运行该算法，这次使用 s 作为源结点。

從 z 點開始：



改了 (z, x) 的權重後，會出現總權重爲負值的環路。

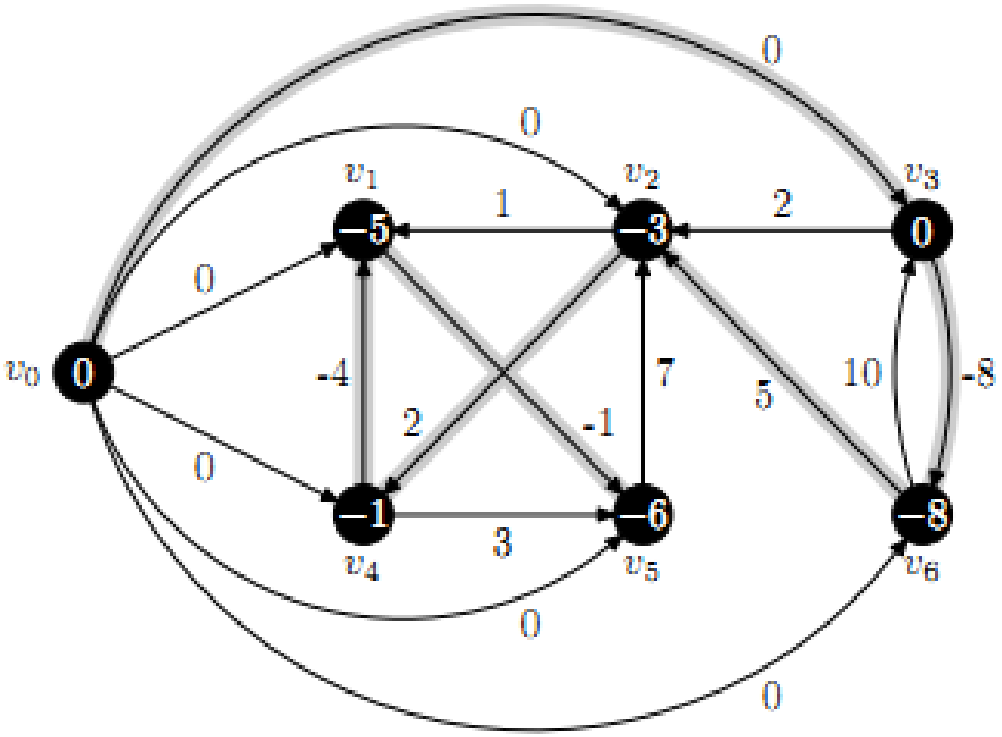


$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$.

24.4-1 请给出下面差分约束系统的可行解或证明该系统没有可行解。

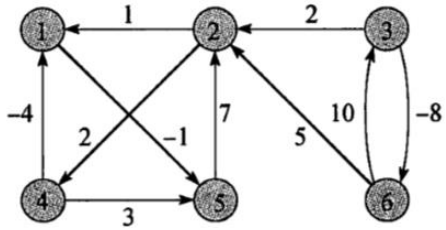
$$x_1 - x_2 \leqslant 1$$
$$x_1 - x_4 \leqslant -4$$
$$x_2 - x_3 \leqslant 2$$
$$x_2 - x_5 \leqslant 7$$
$$x_2 - x_6 \leqslant 5$$
$$x_3 - x_6 \leqslant 10$$
$$x_4 - x_2 \leqslant 2$$
$$x_5 - x_1 \leqslant -1$$
$$x_5 - x_4 \leqslant 3$$
$$x_6 - x_3 \leqslant -8$$

约束图



易犯的错误：边的方向画反了

25.2-1 在图 25-2 所示的带权重的有向图上运行 Floyd-Warshall 算法，给出外层循环的每一次迭代所生成的矩阵 $D^{(k)}$。



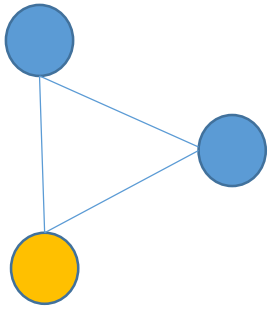
$$D^{(1)} = \begin{pmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ \infty & 2 & 0 & \infty & \infty & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ \infty & 7 & \infty & \infty & 0 & \infty \\ \infty & 5 & 10 & \infty & \infty & 0 \end{pmatrix} \quad D^{(2)} = \begin{pmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & 2 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 10 & 7 & 5 & 0 \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & 2 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 10 & 7 & 5 & 0 \end{pmatrix} \quad D^{(4)} = \begin{pmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ 0 & 2 & 0 & 4 & -1 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{pmatrix}$$

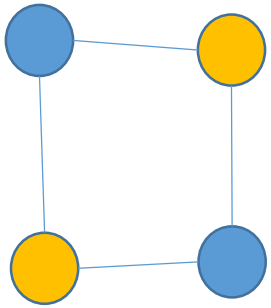
$$D^{(5)} = \begin{pmatrix} 0 & 6 & \infty & 8 & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ 0 & 2 & 0 & 4 & -1 & -8 \\ -4 & 2 & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{pmatrix} \quad D^{(6)} = \begin{pmatrix} 0 & 6 & \infty & 8 & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ -5 & -3 & 0 & -1 & -6 & -8 \\ -4 & 2 & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{pmatrix}$$

22.2-7 职业摔跤手可以分为两种类型："娃娃脸"("好人")型和"高跟鞋"("坏人")型。在任意一对职业摔跤手之间都有可能存在竞争关系。假定有 n 个职业摔跤手，并且有一个给出竞争关系的 r 对摔跤手的链表。请给出一个时间为 $O(n+r)$ 的算法来判断是否可以将某些摔跤手划分为"娃娃脸"型，而剩下的划分为"高跟鞋"型，使得所有的竞争关系均只存在于娃娃脸型和高跟鞋型选手之间。如果可以进行这种划分，则算法还应当生成一种这样的划分。

- 创建一个图G，其中每个顶点代表一个摔跤手，每个边代表一个对抗。 该图将包含n个顶点和r个边。
- 根据需要执行尽可能多的BFS来访问所有顶点。 指定所有距离为偶数的摔跤手为娃娃脸，而所有距离为奇数的摔跤手为高跟鞋。
- 然后检查每个边，以确认它是在娃娃脸和脚跟之间。
- 对于BFS,这个解决方案将花费O(n+r):O(n)时间把每个摔角手指定为娃娃脸或脚跟，O(r)时间检查边，整体O(n+r)。



a



b

24.1-3 给定 $G=(V, E)$是一带权重且没有权重为负值的环路的有向图，对于所有结点 $v \in V$，从源结点 s 到结点 v 之间的最短路径中，包含边的条数的最大值为 m。（这里，判断最短路径的根据是权重，不是边的条数。）请对算法 BELLMAN-FORD 进行简单修改，可以让其在 $m+1$ 遍松弛操作之后终止，即使 m 不是事先知道的一个数值。

（1）如果权重最小的路径上的边数最多为m，则根据路径松弛性质，在Bellman-ford算法执行了m次迭代后，每个结点v的v.d就会到达了它的最小路径权重。

再根据上界性质，经过m次迭代，就没有哪个结点的d再能被修改了。因此，m+1次循环后，d值不再改变了，算法终止。

（2）如果我们事先不知道m的大小，就不能使算法恰好执行m次然后终止。但是，如果我们可以检查是不是有改变发生，如果没有结点的d值再发生改变，则说明可以使算法终止了。发生最后一次改变的那次循环就是第m次循环。

```
BELLMAN-FORD-(M+1)(G, w, s)
  INITIALIZE-SINGLE-SOURCE(G, s)
  changes = TRUE
  while changes == TRUE
        changes = FALSE
        for each edge (u, v) ∈ G.E
              RELAX-M(u, v, w)

RELAX-M(u, v, w)
  if v.d > u.d + w(u, v)
        v.d = u.d + w(u, v)
        v.π = u
        changes = TRUE
```

注：这里没有检查负权重的环路。如果存在负权重的环路，while循环将无限循环下去。

24-3 （套利交易） **套利交易**指的是使用货币汇率之间的差异来将一个单位的货币转换为多于一个单位的同种货币的行为。例如，假定1美元可以购买49印度卢比，1印度卢比可以购买2日元，1日元可以购买0.0107美元。那么通过在货币之间进行转换，一个交易商可以从1美元开始，购买$49\times2\times0.0107=1.0486$美元，从而获得4.86%的利润。

假设给定n种货币c_1，c_2，…，c_n和一个$n\times n$的汇率表R，一个单位的c_i货币可以购买$R[i, j]$单位的c_j货币。

a. 给出一个有效的算法来判断是否存在一个货币序列$\langle c_{i_1}, c_{i_2}, \cdots, c_{i_k}\rangle$，使得

$$R[i_1, i_2]\cdot R[i_2, i_3]\cdot\cdots\cdot R[i_{k-1}, i_k]\cdot R[i_k, i_1]>1$$

请分析算法的运行时间。

b. 给出一个有效算法来打印出这样的一个序列（如果存在这样一种序列）。分析算法的运行时间。

a. 用bellman-ford算法。

首先建立一个有向图G=<V,E>，每种货币对应一个结点，根据汇率表，若$R[i_1, i_2]$，则从i_1到i_2引一条边。图中增加一个新结点v_0，从v_0引出到其它所有结点的边<v_0,v_i>

要使得$R[i_1, i_2] * \cdots R[i_k, i_1] > 1$，即有$\lg \frac{1}{R[i_1,i_2]} + \cdots + \lg \frac{1}{R[i_k,i_1]} < 0$。

因此，置边(v_i, v_j)权重$w(v_i, v_j) = lg \frac{1}{R[i,j]} = -lgR[i,j]$。

对v_0，它到其它所有节点v_l的边(v_0, v_l)权重为0的。

从v_0出发，运行bellman-ford，若存在一条负权重和的环路，则判断存在此种序列，反之，不存在此种序列。

运行时间：创建具有$O(n^2)$条边的有向图G需要$O(n^2)$，需要$O(n^3)$执行bellman-ford算法，故总时间为$O(n^3)$。

- b.假设bellman-ford算法执行结果存在false返回值，即a中序列存在。此时，根据结点的前驱π可以找到负权重环路上的一个结点，然后沿环路继续搜索，直到到达一个我们之前曾经访问过的结点x。该回路就是我们要找的回路。
- 可以使用22.2节中打印路径的算法，在返回至结点x时，打印路径终止即可。
- 时间分析：花费$O(n^3)$执行bellman-ford，花$O(n)$时间打印环路中节点，总共$O(n^3)$时间。

25-1 （动态图的传递闭包） 假定我们希望在将边插入到集合 E 中时维持有向图 $G=(V, E)$ 的传递闭包，即在插入每条边后，我们希望对到目前为止已插入边的传递闭包进行更新。假定图 G 开始时不包含任何边，并且传递闭包用布尔矩阵来表示。

a. 说明在加入一条新边到图 G 时，如何在 $O(V^2)$ 时间内更新图 $G=(V, E)$ 的传递闭包 $G^*=(V, E^*)$。

b. 给出一个图 G 和一条边 e，使得在将边 e 插入到图 G 后，更新传递闭包的时间复杂性为 $\Omega(V^2)$，而不管使用的是何种算法。

c. 描述一个有效的算法，使得在将边加入到图 G 中时更新传递闭包。对于任意 n 次插入的序列，算法运行的总时间应该是 $\sum_{i=1}^{n} t_i = O(V^3)$，其中 t_i 是插入第 i 条边时更新传递闭包所用的时间。请证明你的算法确实达到了这个时间效率。

a.设T=(t_{ij})为|V|*|V|的矩阵，t_{ij}为1代表从i到j存在一条路径，为0则不存在。当加入一条边(u,v)时，做以下修正：

```
TRANSITIVE-CLOSURE-UPDATE(T, u, v)
  let T be |V| × |V|
  for i = 1 to |V|
      for j = 1 to |V|
          if t_iu == 1 and t_vj == 1
              t_ij = 1
```

b.考虑以下情况：图G($v_1 \to v_2 \to \cdots \to v_{|V|}$)，插入边($v_{|V|}, v_1$)。

在插入边之前，在T中有|V|(|V| + 1)/2个元素为1,（主对角线之上的元素也为1）。在边插入后，图为一个环路，所有结点之间都可达，故T中$|V|^2$个元素为1，所以将有$|V|^2$-|V|(|V| + 1)/2=Θ(V^2)个元素需要修改，所以任何算法均可在$\Omega(V^2)$时间内更新闭包。

c.

```
TRANSITIVE-CLOSURE-UPDATE(T, u, v)
  let T be |V| × |V|
  for i = 1 to |V|
      if t_iu == 1 and t_iv == 0
          for j = 1 to |V|
              if t_vj == 1
                  t_ij = 1
```

- 证明：边数n ≤ $|V|^2$，假设超过n次插入，外循环中if语句会花费O(nV)=O(V^3)。最后三行中，花费O(|V|)，因为每次插入只在t_{iv}=0时执行，且最终在t_{ij}=1结束，故每次插入只会执行一次，故n此插入实际只需O(V^2)。综合起来即为O(V^3)。

```
TRANSITIVE-CLOSURE-UPDATE(T, u, v)
  let T be |V| × |V|
  for i = 1 to |V|
      for j = 1 to |V|
          if t_iu == 1 and t_vj == 1
              t_ij = 1
```

25.2-6　我们怎样才能使用 Floyd-Warshall 算法的输出来检测权重为负值的环路？

方法1：

检测结果矩阵主对角线是否存在负值。

方法2：

多执行一遍循环，如果还有矩阵元素更新，这表明有负权重的环路。

25.2-7 在 Floyd-Warshall 算法中构建最短路径的另一种办法是使用 $\phi_{ij}^{(k)}$，其中 i，j，$k=1$，2，…，n，$\phi_{ij}^{(k)}$ 是从结点 i 到结点 j 的一条中间所有结点都取自集合{1，2，…，k}的最短路径上编号最大的中间结点。请给出 $\phi_{ij}^{(k)}$ 的一个递归公式，并修改 Floyd-Warshall 过程来计算 $\phi_{ij}^{(k)}$ 的值，并重写 PRINT-ALL-PAIRS-SHORTEST-PATH 过程，使其以矩阵 $\Phi=(\phi_{ij}^{(n)})$ 作为输入。矩阵 Φ 与 15.2 节所讨论的链式矩阵乘法中的表格存在何种相似点？

1）递推公式

$$\emptyset_{ij}^{(0)} = 0$$

$$\emptyset_{ij}^{(k)} = \begin{cases} \emptyset_{ij}^{(k-1)}, d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ k, d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}$$

2）程序

Floyd-warshell(W)

n=W.row

$D^{(0)}=W, \Phi^{(0)}=0$

for k=1 to n

let $D^{(k)}$, $\Phi^{(k)}$ be new nXn matrices

for i=1 to n

for j=1 to n

if $d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$

$d_{ij}^{(k)} = d_{ij}^{(k-1)}$

$\emptyset_{ij}^{(k)} = \emptyset_{ij}^{(k-1)}$

else $d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$

$\emptyset_{ij}^{(k)} = k$

return D(n), Φ(0)

```
Print-all-paris-shortest-paths(Φ)
    for i=1 to n
      for j=1 to n
          Print-all-paris-shortest-path(Φ,i,j)

Print-all-paris-shortest-path(Φ,i,j)
     if i == j
           print I
     else  Print-all-paris-shortest-path(Φ,i, Φ ij)
            Print-all-paris-shortest-path(Φ, Φ ij,j)
```