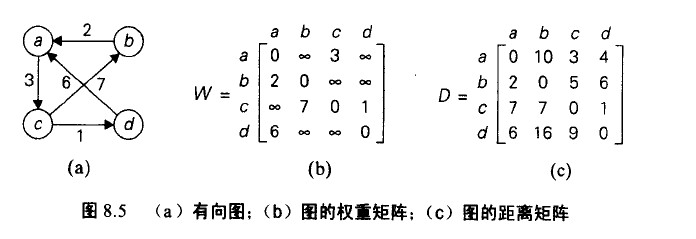
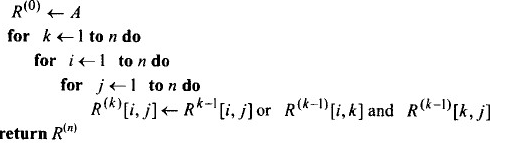
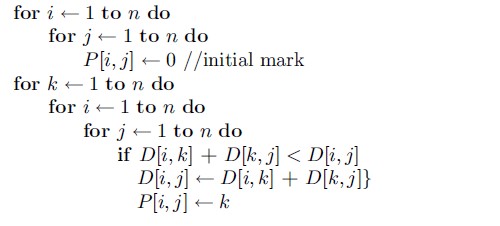
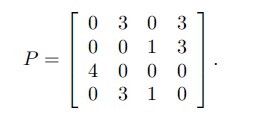
**Floyd求关系传递闭包**  
Floyd算法可以用于构造无向或有向加权图（不包含长度为负的回路）的完全最短路径也可以来求完全闭包（即有a[i][j]和a[j][k]，则要有a[i][k]）  
  
  
floyd求闭包算法是通过每次加入一个顶点，看把这个顶点作为中间顶点是否能改进传递闭包的矩阵（通过这个新加入的顶点作为中间桥梁，使得原来不可达的2个顶点可达，以此逐步向传递闭包逼近）。

我们可以通过一次加入一个点的方式（一共n次，加入n个点，n步决策）来构造最终的传递闭包：  
----用R0表示邻接矩阵，以后每次加入一个顶点来构造R1，R2.......Rn。  
---如果 r（i , j） 在Rk-1中为1，那么加入顶点k作为中间节点后，r(i , j) 在Rk中的值仍为1（如果可达了，加入点之后肯定还是可达）  
--如果r(i , j)在Rk-1中不为1，仅当 r(i , k) = 1 且 r(k , j) = 1，r(i , j)在Rk中才为1（如果现在不可达，仅当加入的一个中间节点可以作为一个桥梁使之可达，才可达）  


**Floyd求最短路径**   
**解答：**  
可以再弄一个矩阵p（大小为n\*n），p[i , j] = k，表明从 i 到 j 的最短路径要经过顶点 k （**注意不是只经过 k**）。这样在原来的Floyd算法里，每次加入顶点 k 来改  
变矩阵时，若 k 起到了作用（加入k后对某个最短路径起到了修正作用），那么就把进行把 k 记在 p 矩阵里：  
  
  
  
相比原算法，就是多用了一个p矩阵，加了最后一行，每次加入k起到了作用的时候，记下。注意这个伪代码是空间优化了的Floyd算法，直接在原矩阵上填。  
这样最终结果就得到了2个矩阵：D矩阵记录了所有 i 到 j 顶点的最短路径。**p矩阵间接表达了每条最短路径的具体路径**。例如：最终可能得到下列p矩阵：  
  
根据p[i , j] 的定义， p[1][2] = 3，表明从顶点1（即顶点a）到顶点2（即顶点b）的最短路路径至少要经过顶点3  
即顶点3是顶点1到2最短路径上的一个桥梁，那么从1到3以及从3到2之间还有没有别的桥梁呢？再看p[1][3] = 0，p[3][2] = 0，所以没有了，那么从1到2的最短  
路径就是1,3,2。

**Floyd求最小环**

Poj 1734 ：给出一张无向图，求一个最小环并输出环上路径。

这道题目是floyd的经典，floyd的核心思想就是动态规划从k=0->n来松弛i->j的路径，

这样我们可以这样思考这个环，里面肯定有最大的下标，设为k，与之相邻的是i和j,然后i->j的最短路肯定在用k松弛前已经更新完，换句话说就是

我们floyd时候先找环再松弛，因为floyd的外层到k时，i->j的最短路上肯定没有k所以这个环不会存在相同的点



**Spfa 求多个点之间的最短路**

## Hdu6166

给你一个10w个点，10w个点，再从中选k个点（已给定），求k个点之间最近的两个点的最短距离。

方法1，建立超级起点s,和超级终点t,将这k个点二进制枚举,一些与s连边，另一些与t连边，这样保证了每个点对I,j一定会有一次机会分别在s,t中，所以跑logk遍spfa即可

方法2 建立超级起点s,和超级终点t,把这k个点每个点拆成两个，分别与s和t相连接,然后跑一遍spfa即可，但是有个问题就是可能会出现从s->左i->j->右i->t (其中j不属于k)这种自环的局面，所以我们spfa更新时还要记录一个resorce表示这条路是从谁始发的，保留一个不同source次短路就行了

# 同余最短路

同余最短路是指给定若干数a[i]要你判断是否能够刚好凑成一个x或者凑出一个最小的K使得它大于X

## Bzoj 2612

给定a数组，问num能否被表示为a[1]\*x[1]+a[2]\*x[2]+……+a[n]\*x[n]，x为非负整数

设a1为最小数字，若方程a[1]\*x[1]+a[2]\*x[2]+……+a[n]\*x[n]=k存在非负整数解，那么k+a1也必然有解。建立a1个点的图，点编号为0到a1-1,dist[i](0<=i<a1-1)代表%a1==i最少需要几才能凑出来。

点i向(i+aj)%a1连边，边权为aj，求从0号点到各个点的最短路dist[i]，如果dis[k%a1]<=k，那么k有解。跑一遍对堆优化的dijkstra即可

HDU 6071

一个圆形跑道上有4个点，给定相邻两个点的距离，你从2号点出发最后也要回到2号点，看怎样在跑步距离>=K的情况下你跑步的距离最短.

设l为min(a[1],a[2])，若方程a[1]\*x[1]+a[2]\*x[2]+……+a[n]\*x[n]=k存在非负整数解，那么k+2\*l也必然有解。

建立4\*2\*l个点的图，d[i][j]（i属于 0到2\*-1，j属于1到4）表示最少需要几才能凑出来到达j点距离mod2\*l==i。

用点d[i][j]去更新d[i+a[i+1]][j+1]和d[i+a[i]][j-1]就可以了