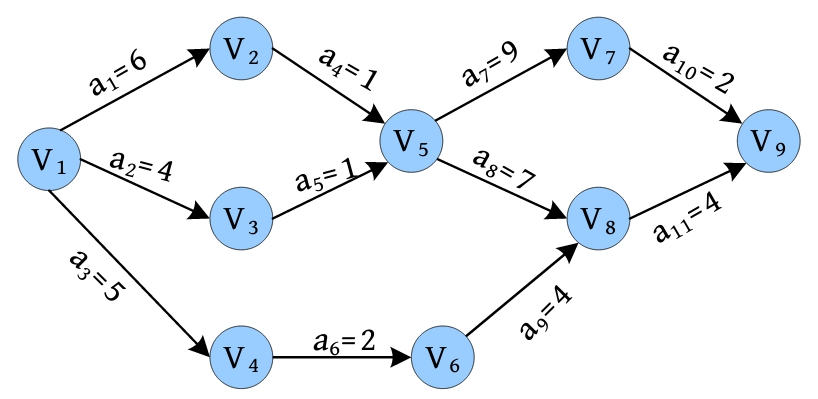
**SDUTOJ2498**

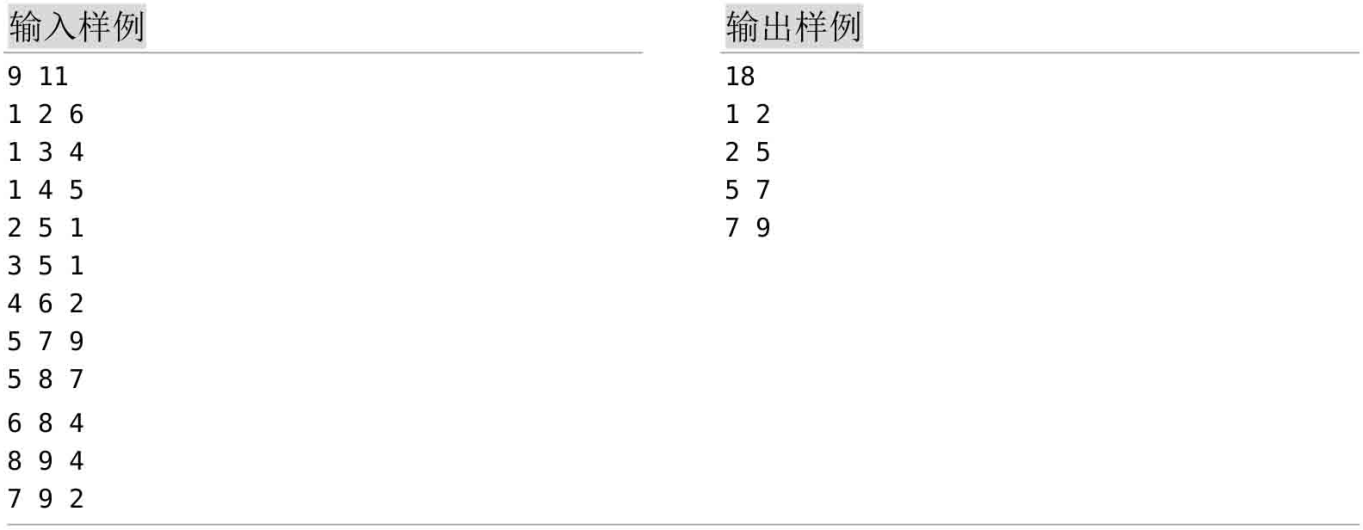
**题目描述（SDUTOJ2498）：**一个无环的有向图被称为有向无环图（Directed Acyclic Graph，之后简称DAG）。AOE（Activity On Edge）网是指以边表示活动的网，如下图所示。



在上图中共有11个活动、9个事件。整个工程只有一个开始点和一个完成点，即只有一个入度为零的点（源点）和一个出度为零的点（汇点）。关键路径指从开始点到完成点的最长路径。路径的长度是边上活动耗费的时间。如上图所示，1-2-5-7-9是关键路径（关键路径不止一条，输出字典序最小的），权值之和为18。

**输入：**输入包含多组数据，不超过10组。第1行包含节点数n（2≤n≤10000）和边数m（1≤m≤50000），接下来的m行，包含每条边的起点s和终点e，权值w（1≤s,e≤n，s!=e，1≤w≤20）。数据保证图连通，且只有一个源点和汇点。

**输出：**单行输出关键路径的权值和，并且从源点输出关键路径上的路径（如果有多条，则输出字典序最小的）。



**题解：**本题求解关键路径**实际上就是求解最长路径**。**求解最长路径时可以将权值加负号求解最短路径**，**也可以改变松弛条件，若距离较大则更新**。

• 对有向无环图，可以按拓扑序列松弛求解最长路径，也可以用Bellman或SPFA算法权值加负号求解最短路径，或者改变松弛条件求解最长路径。

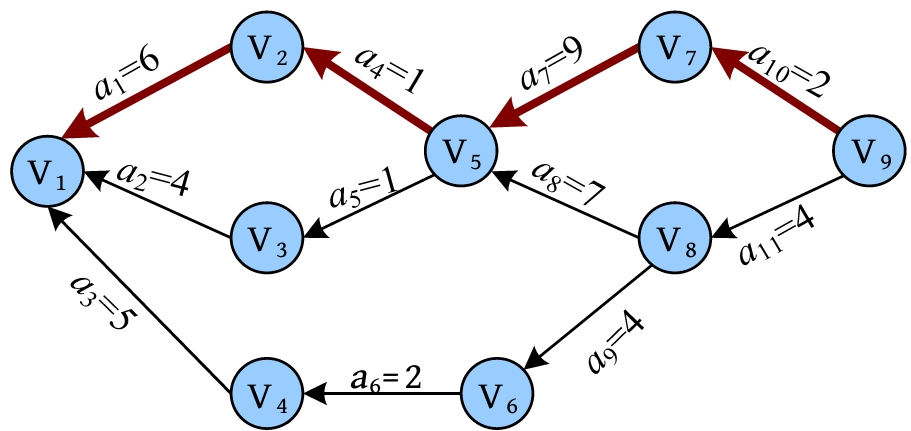
• 对有向有环图，可以用Bellman或SPFA算法判断环，若有正环，则不存在最长路径。

需要注意的是，Dijkstra算法不可以用于处理负权边，也无法通过改变松弛条件得到最长路径。Bellman算法的时间复杂度为O(n×m)，可能会超时，所以可以采用SPFA算法，该算法的时间复杂度为O(k×m)，k是一个较小的常数，最多为O(n×m)。其次，该题需要输出路径，而且该路径需要按字典序选取，所以**反向建图会更便于记录路径。**

路径的字典序最小就是走到一个点，继续向下一步走时，选择编号最小的，这就是字典序，但是在最短路径的更新过程中，如果dis[y]==dis[x]+w&&x<pre[y]，路径长度相等但是x比y的前驱节点编号更小，则更新y的前驱节点为x，即pre[y]=x。

在本题的AOE网中，V5-V7-V9和V5-V8-V9的路径长度是一样的，按字典序应该走前者。如果逆向走，从V9到V7，则dis[7]=2；从V9到V8，则dis[8]=4；从V8到V5，则dis[5]=11，pre[5]=8；从V7到V5，则dis[7]+9=11=dis[5]，但是7比8的字典序小，更新5的前驱为7，pre[5]=7。

在原图的逆向图上，从后向前走一条最长路径，然后根据前驱数组，1的前驱为2，输出1 2；2的前驱为5，输出2 5；5的前驱为7，输出5 7；7的前驱为9，输出7 9。

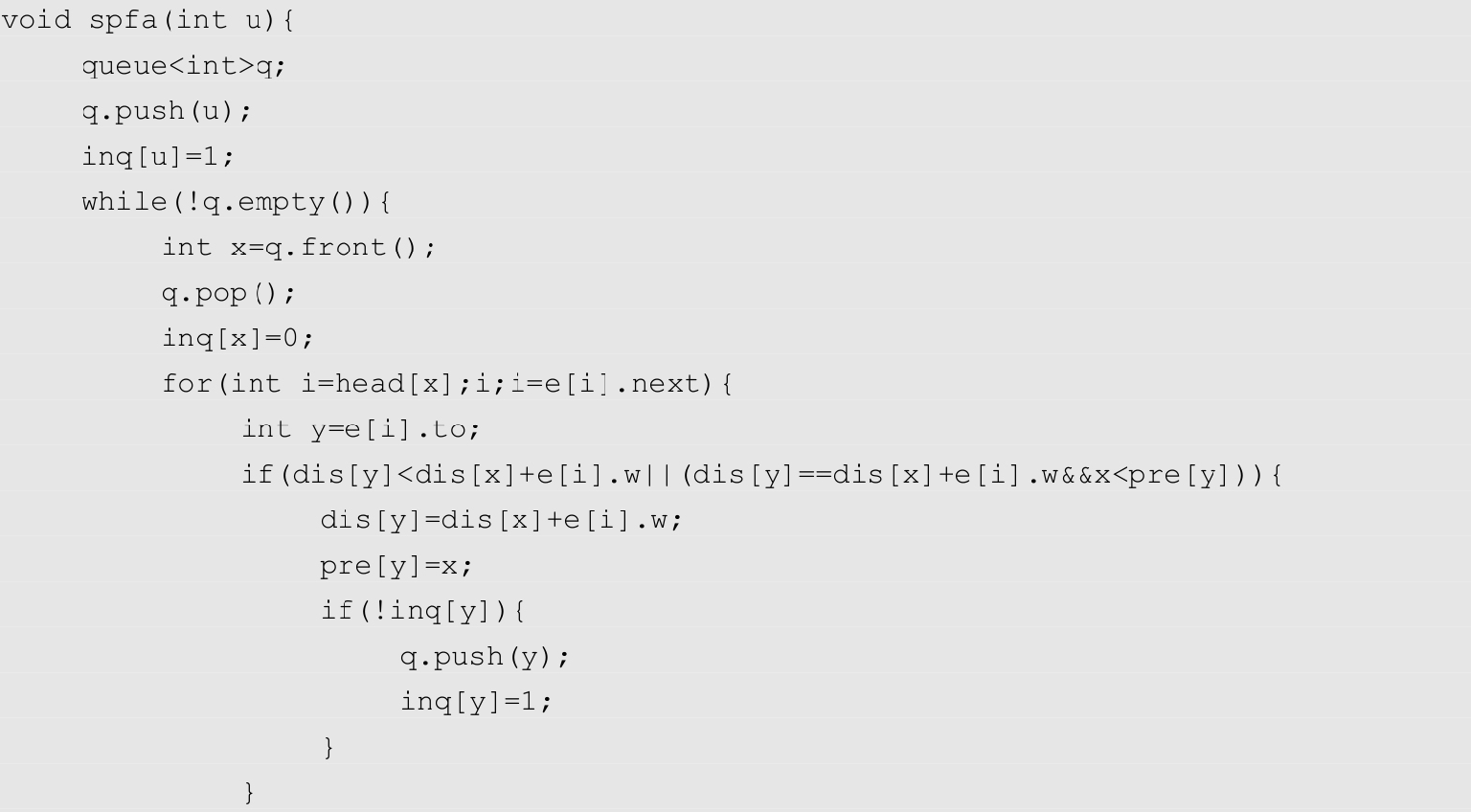


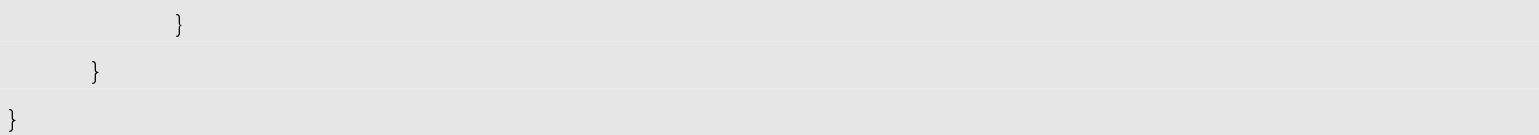
**1. 算法设计**

（1）建立原图的逆向图。检查入度为0的节点s和出度为0的节点t。

（2）使用SPFA算法求最长路径。如果dis[y]<dis[x]+e[i].w||(dis[y]==dis[x]+e[i].w&&x<pre[y]))，则更新dis[y]=dis[x]+e[i].w; pre[y]=x。

**2. 算法实现**

****

****

**HDU4109**

**题目描述（HDU4109）：**阿里本学期开设了计算机组成原理课程。他了解到指令之间可能存在依赖关系，例如WAR（写入后读取）、WAW、RAW。

如果两个指令之间的距离小于安全距离，则会导致危险，这可能导致错误的结果。所以需要设计特殊的电路以消除危险。然而，解决此问题的最简单方法是添加气泡（无用操作），这意味着浪费时间以确保两条指令之间的距离不小于安全距离。对两条指令之间距离的定义是它们的开始时间之间的差。

现在有很多指令，已知指令之间的依赖关系和安全距离，可以根据需要同时运行多个指令，并且CPU速度非常快，只需花费1ns即可完成任何指令。你的工作是重新排列指令，以便CPU用最短的时间完成所有指令。

**输入：**输入包含几个测试用例。每个测试用例的第1行都包含两个整数N和M（N≤1000,M ≤10000），表示N个指令和M个依赖关系。以下M行，每行都包含3个整数X、Y、Z，表示X和Y之间的安全距离为Z，Y在X之后运行。指令编号为0～N-1。

**输出：**单行输出一个整数，即CPU运行所需的最短时间。

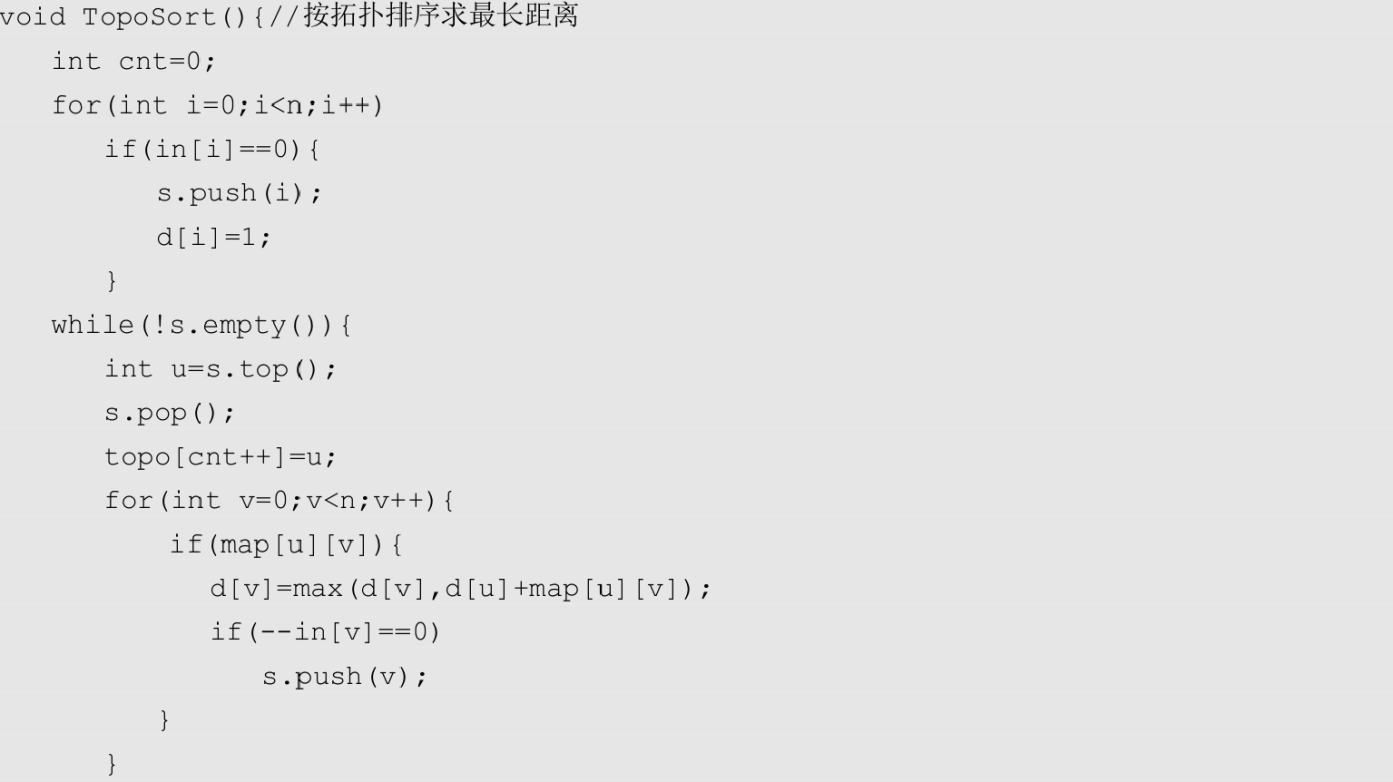


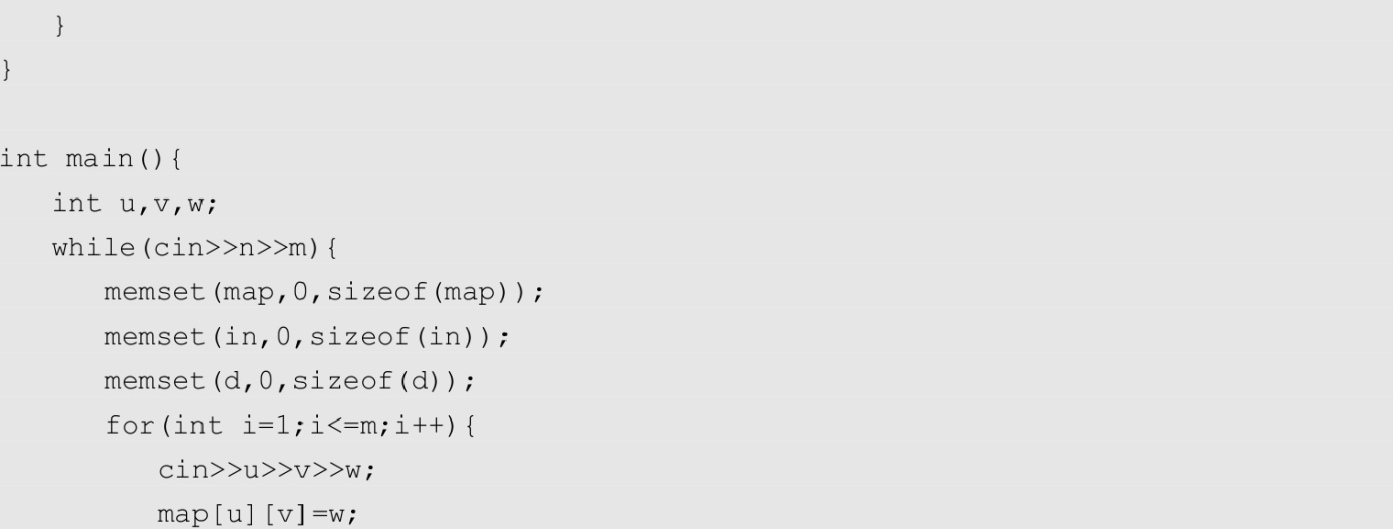
**题解：**根据测试用例，构建的图形结构如下图所示。在第1ns中，执行指令0、1和3；在第2ns中，执行指令2和4。答案是2。

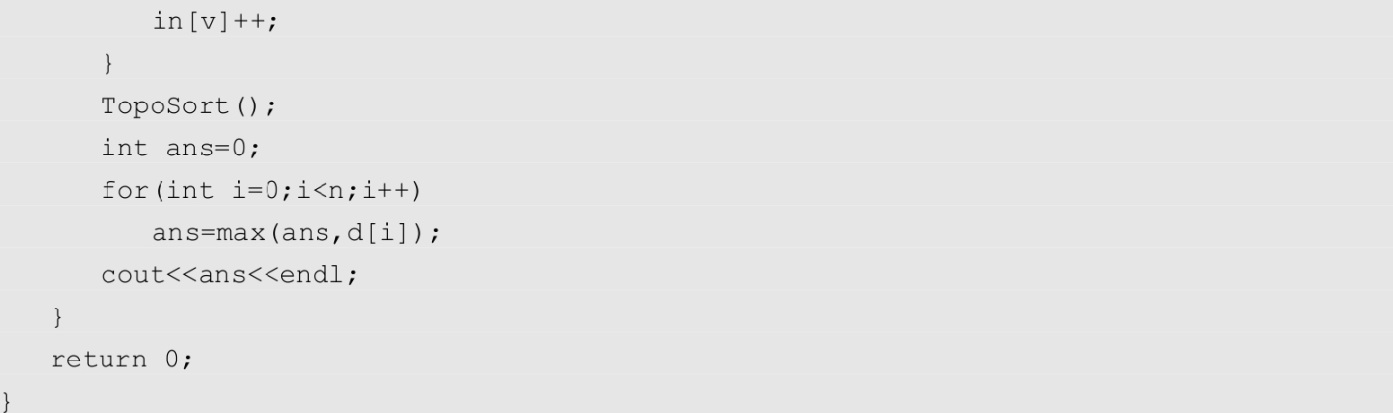


按照拓扑排序求每个节点的最长距离，然后求各个节点最长距离的最大值。

**算法代码：**

****

****

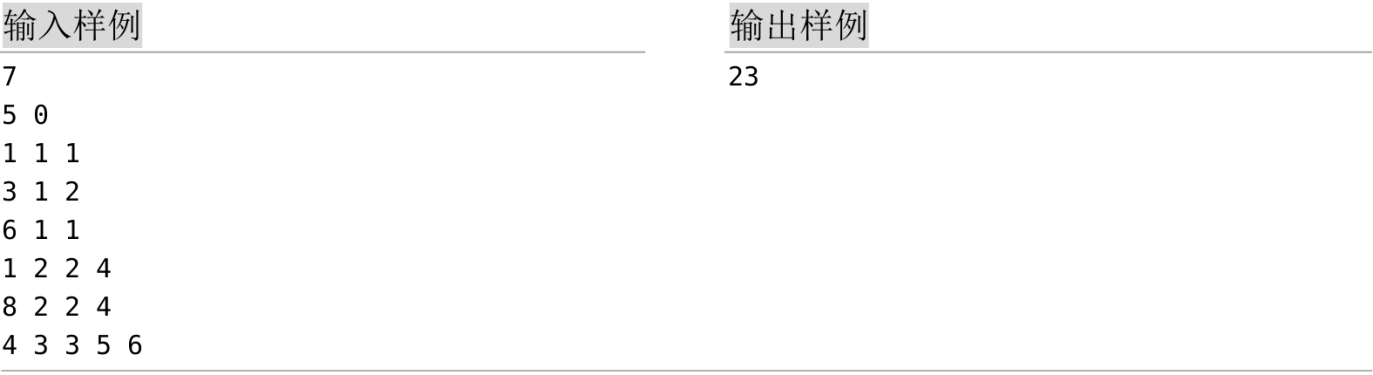
****

**POJ1949**

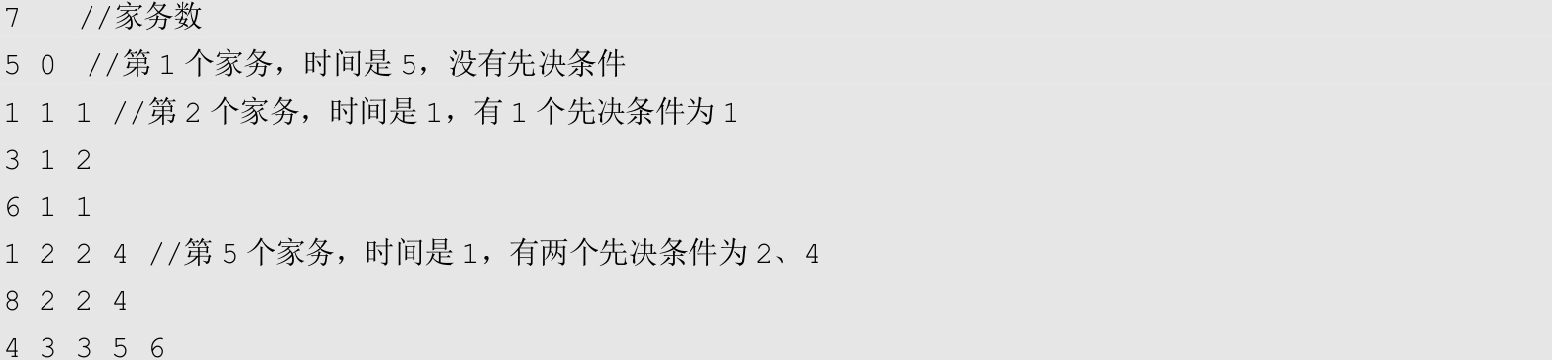
**题目描述（POJ1949）：**约翰有一份必须完成的N（3≤N≤10 000）个家务的清单。每个家务都需要一个整数时间T（1≤T≤100）才能完成，并且可能还有其他家务必须在这个家务开始之前完成。至少有一个家务没有先决条件：第1号。家务K（K>1）只能以家务1～K-1作为先决条件。计算完成所有N个家务所需的最少时间。当然，可以同时进行彼此不依赖的家务。

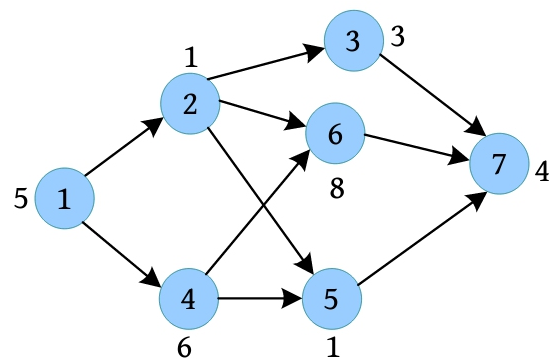
**输入：**第1行包含一个整数N。第2～N+1行描述每个家务，第2行包含家务1；第3行包含家务2，以此类推。每行都包含完成家务的时间、先决条件的数量Pi（0≤Pi≤100）和Pi个先决条件。

**输出：**单行输出完成所有家务所需的最少时间。



**题解：**根据输入样例1，构建的图形结构如下图所示。





**分析：**

• 家务1在时间0开始，在时间5结束；

• 家务2在时间5开始，在时间6结束；

• 家务3在时间6开始，在时间9结束；

• 家务4在时间5开始，在时间11结束；

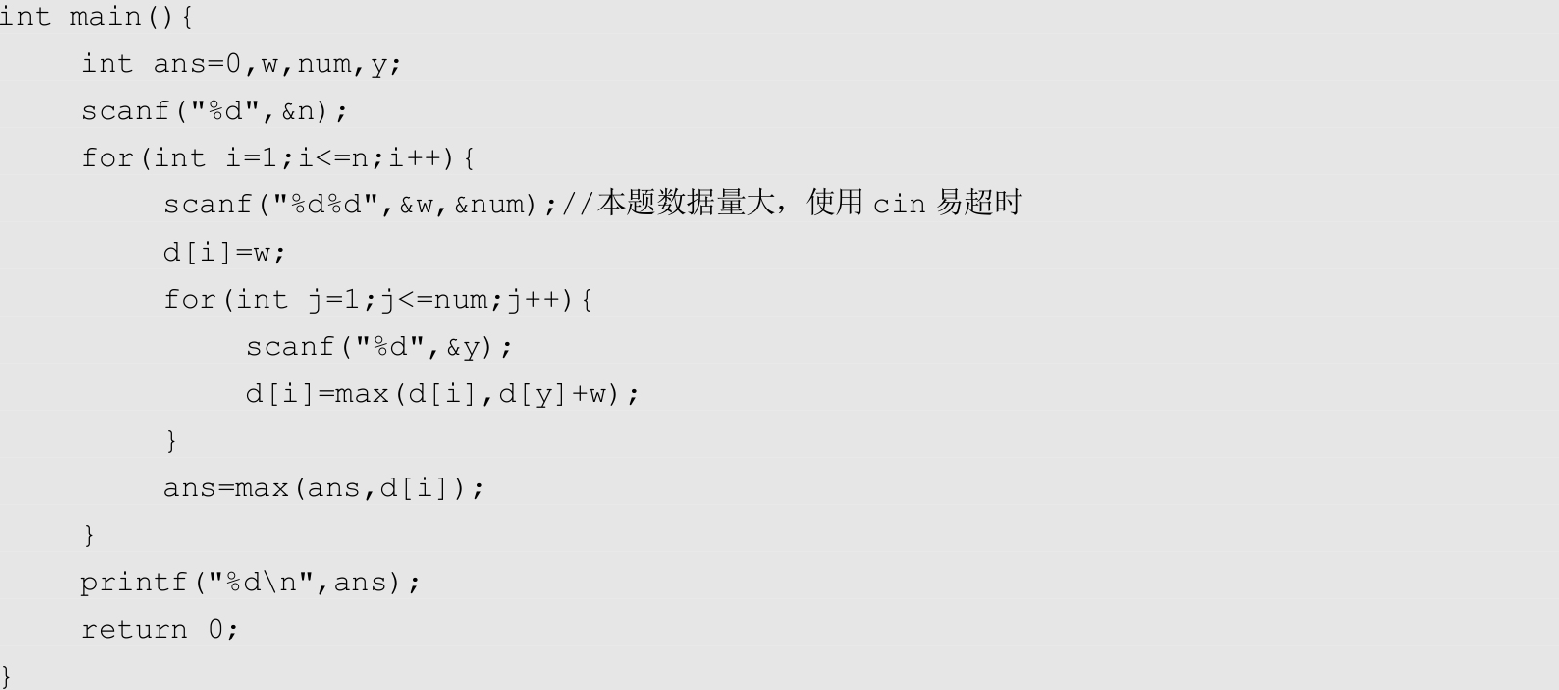
• 家务5在时间11开始，在时间12结束；

• 家务6在时间11开始，在时间19结束；

• 家务7在时间19开始，在时间23结束。

本题的关键在于，**家务K（K>1）只能以家务1～K-1作为先决条件**。也就是说，输入第K个家务时，它的先决条件均已确定什么时间结束。因此在输入过程中直接求最长距离即可。如果没有先决条件限制，则不可以这样计算。

**算法代码：**

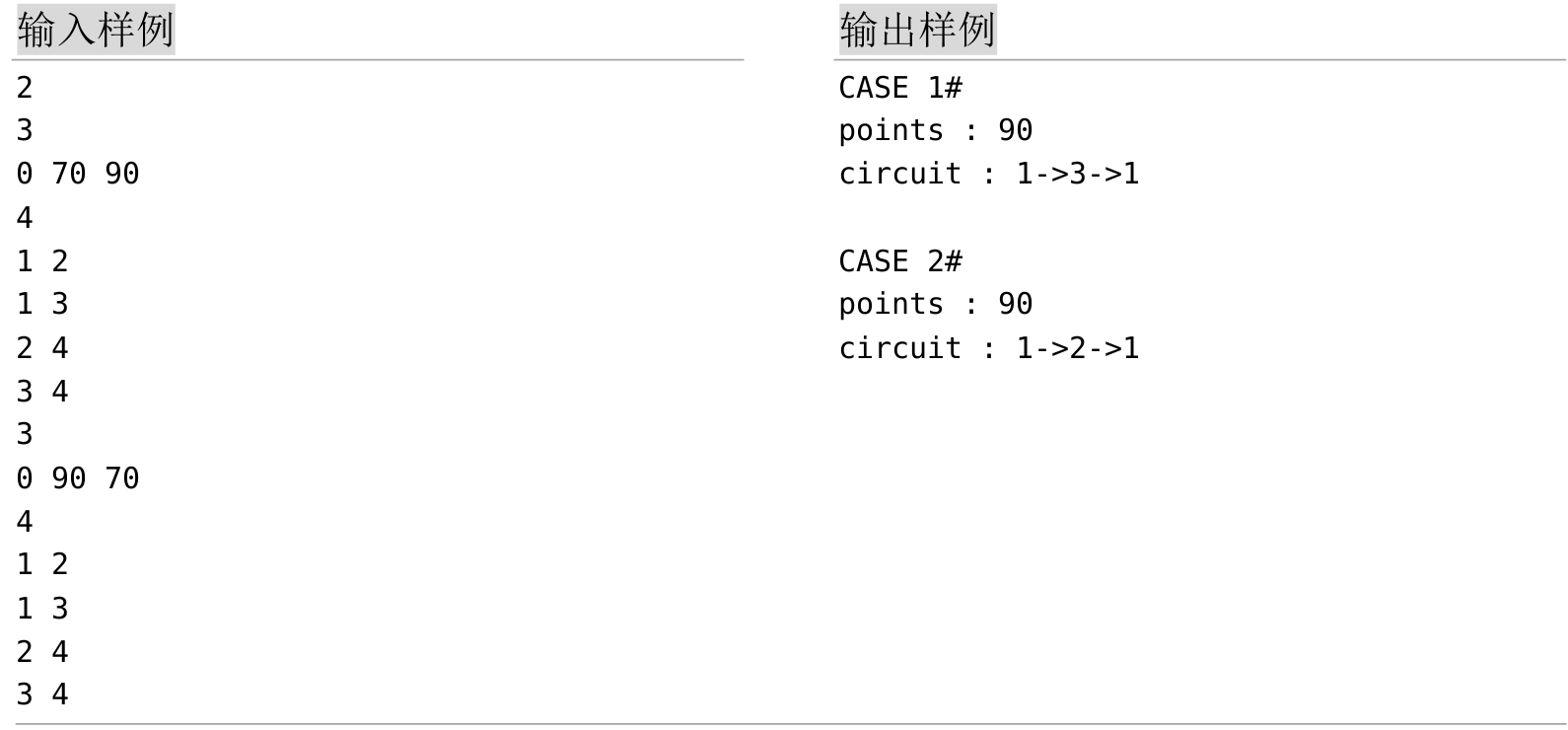


**HDU1224**

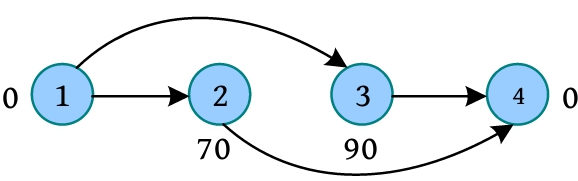
**题目描述（HDU1224）：**旅游公司展示了一种新型DIY线路。各线路都包含一些可由游客自己选择的城市。根据该公司的统计数据，每个城市都有自己的评分，评分越高越有趣。例如，巴黎的评分是90，纽约的评分是70，等等。世界上不是任何两个城市之间都可以直飞的，因此旅游公司提供了一张地图，告诉游客是否可以在地图上任意两个城市之间直飞。在地图上用一个数字标记每个城市，一个数字较大的城市不能直接飞往数字较小的城市。薇薇从杭州出发（杭州是第1个城市，也是最后1个城市，所以杭州被标记为1和N+1），它的评分为0。薇薇希望尽可能地让游览变得有趣。

**输入：**第1行是整数T，表示测试用例数。每个测试用例的第1行都是一个整数N（2≤N≤100），表示城市数。然后是N个整数，表示城市的评分。接着是整数M，后跟M对整数Ai、Bi（1≤i≤M），表示从城市Ai可以直飞到城市Bi。

**输出：**对于每个测试用例，都单行输出评分之和的最大值和最佳DIY线路。在测试用例之间都输出一个空行。



**题解：**本题其实是**求解1～N+1的最长路径**。根据输入样例1，构建的图如下图所示。



起点和终点的评分为0，终点4实际上也是起点1，因为起点编号为1和N+1。1→3→1这条路径的评分之和最大，因此答案为90。

**1. 算法设计**

可以使用邻接矩阵存储，使用两个for语句更新。也可以使用SPFA算法求最长路径。

（1）读入每个节点的评分，将第N+1个节点的评分设置为0。

（2）读入可以直飞的城市编号，采用邻接矩阵存储。

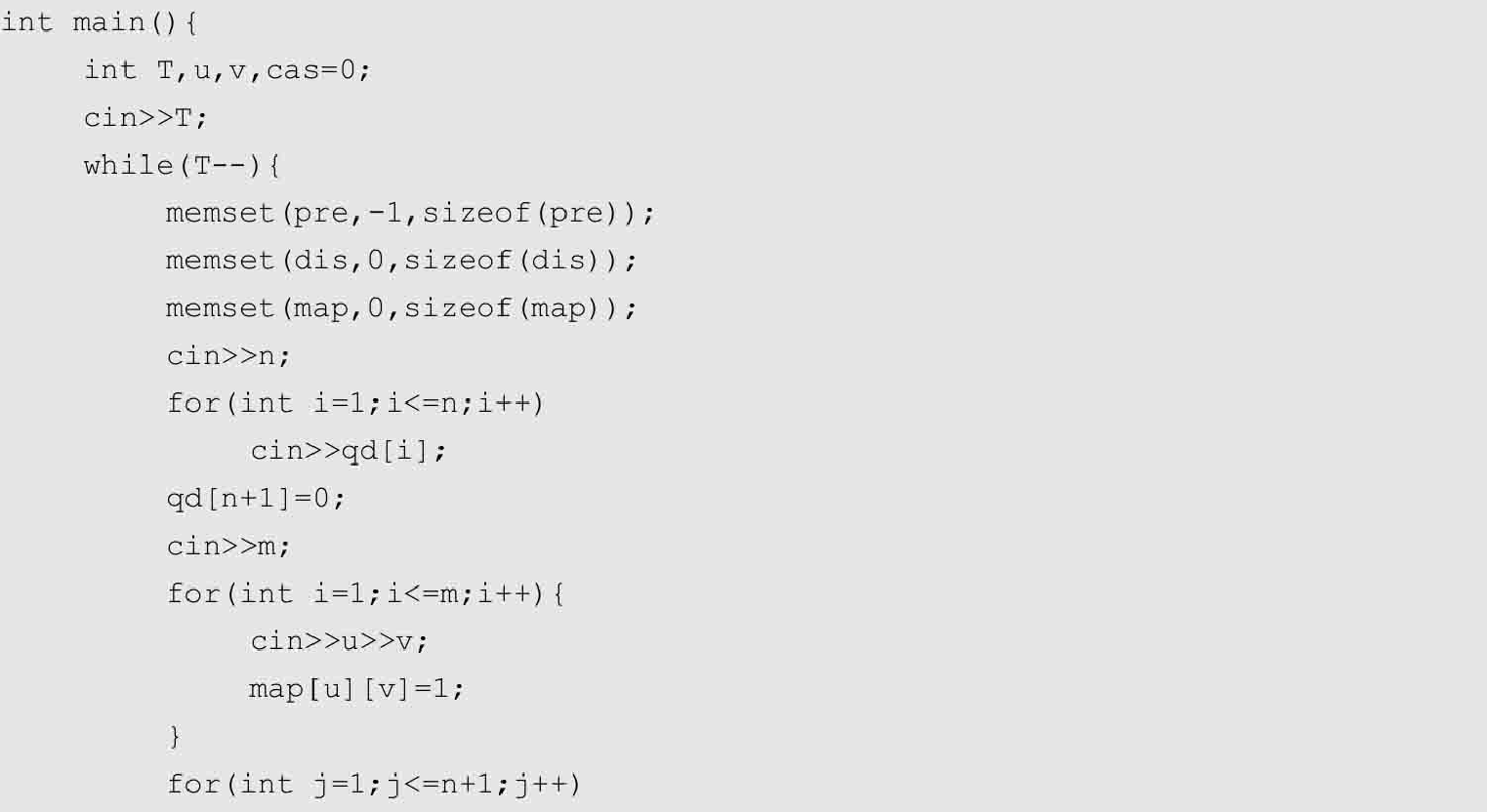
（3）枚举j=1...n+1，i=1...j-1，如果map[i][j]&&dis[j]<dis[i]+qd[j]，则

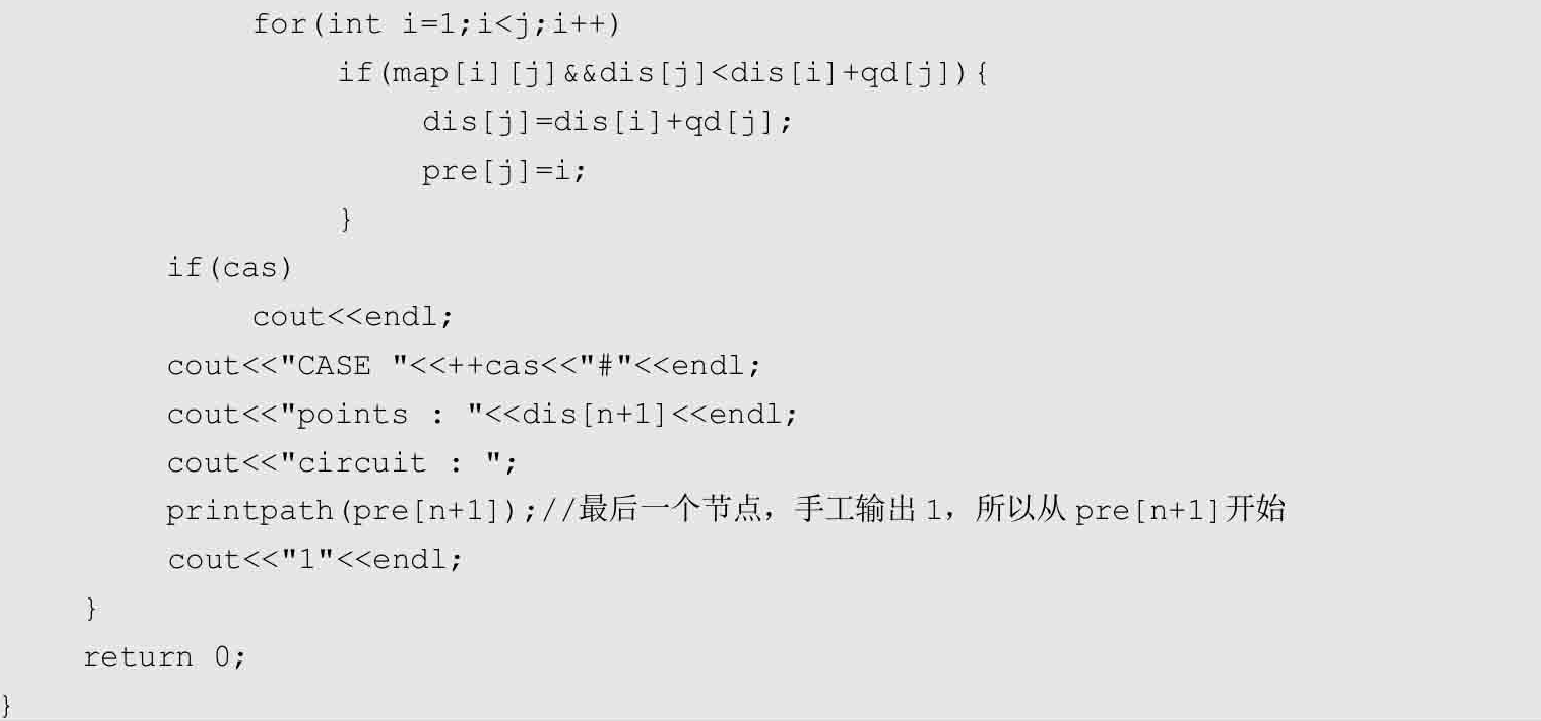


（4）递归输出最长的回路。

**2. 算法实现**

****

****

****

**HDU1317**

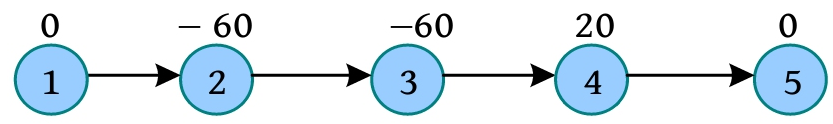
**题目描述（HDU1317）：**有n（n≤100）个房间，每个房间都有一个能量值（范围是-100～+100）。以单向门连接两个房间，可以通过任何连接所在房间的门到达另一个房间，从而进入另一个房间，到达该房间时会自动获得该房间的能量。可以多次进入同一个房间，每次都能获得能量。初始能量值为100，初始位置是1号房间，要走到n号房间。1号房间和n号房间的能量值均为0。到达n号房间可获胜，如果中途能量值小于或等于0，则会因能量耗尽而死亡。

**输入：**输入包含几个测试用例。每个测试用例的第1行都为n，表示房间数。接下来是n个房间的信息。每个房间的信息都包括：房间i的能量值、离开房间i的门数量、房间i可以通过门到达的房间列表。在最后一个测试用例之后是包含-1的行。

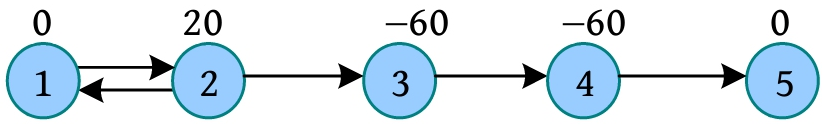
**输出：**如果玩家有可能获胜，则输出winnable，否则输出hopeless。



**题解：**根据输入样例1，构建的图如下图所示，到不了5号房间能量就耗尽了，输出“hopeless”。



根据输入样例4，构建的图如下图所示，有正环且可以到达终点，输出“winnable”。



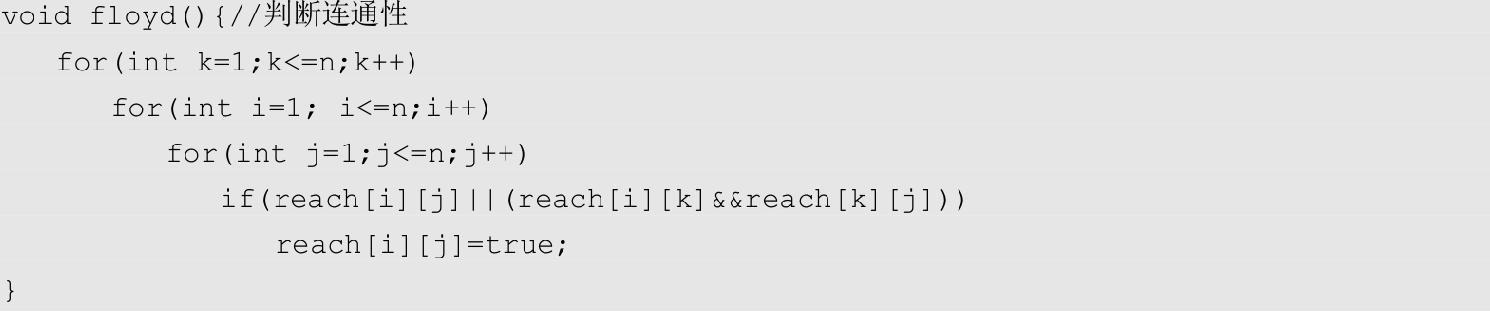
如果从1号房间到n号房间不连通，则必然不能获胜。如果有正环，则环上的点到n号房间连通即可获胜。如果没有环，则到达终点的最长路径的能量值大于0即可获胜。

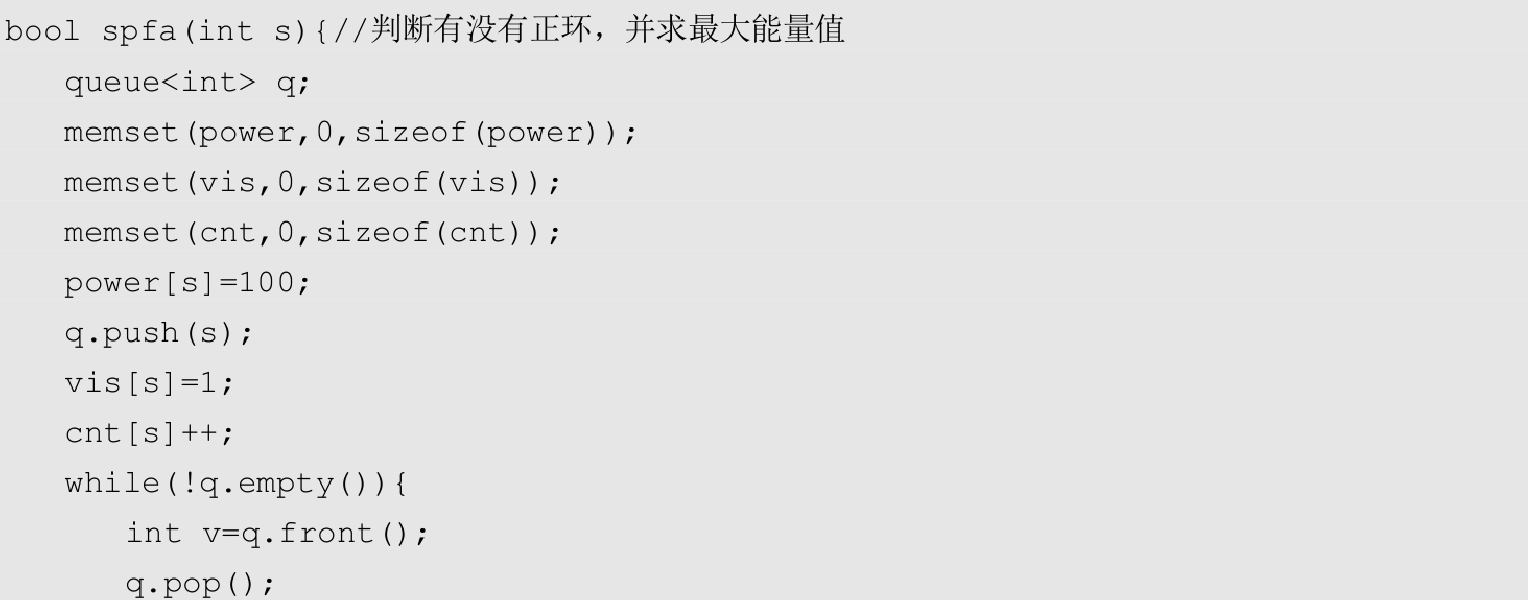
**1. 算法设计**

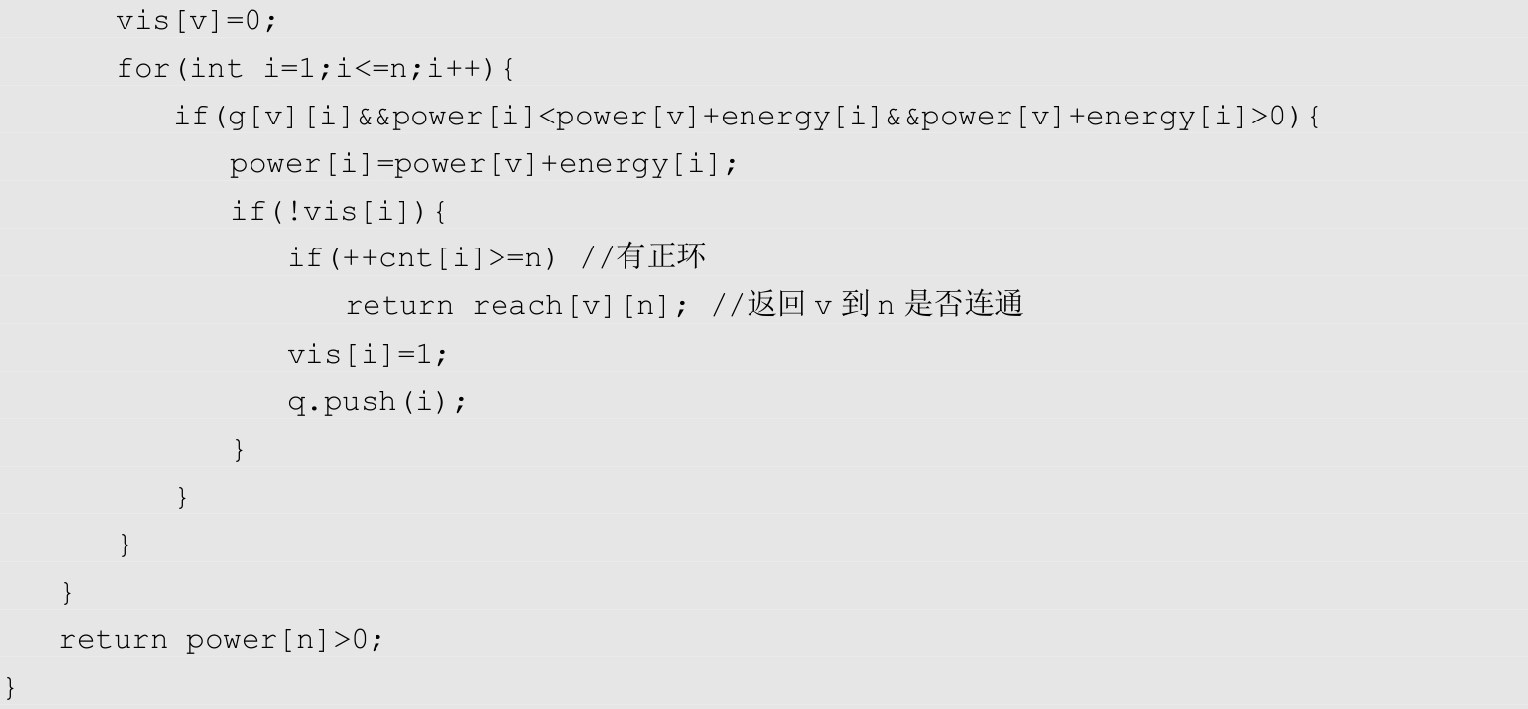
（1）用Floyd算法判断连通性，判断能否从1号房间走到n号房间，如果不连通则结束。

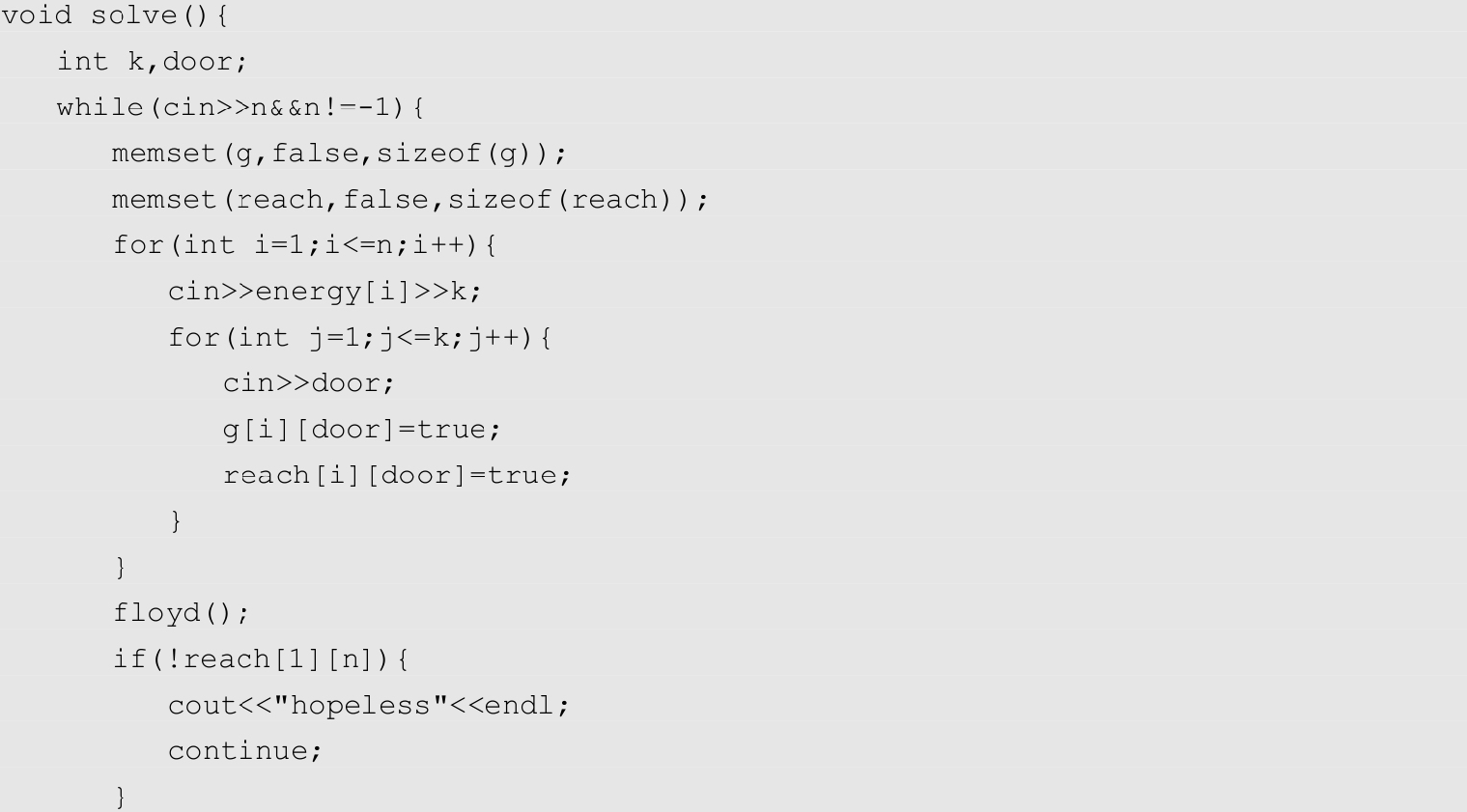
（2）用SPFA算法判断有没有正环，在cnt[v]≥n时有正环，判断环上一点到终点是否连通。如果没有正环，则判断到达终点的最长路径的能量值是否大于0即可。注意：由于该题给出的数据是每个节点的能量值，而不是边的能量值，需要用Floyd算法判断连通性，因此用邻接矩阵来存储图。

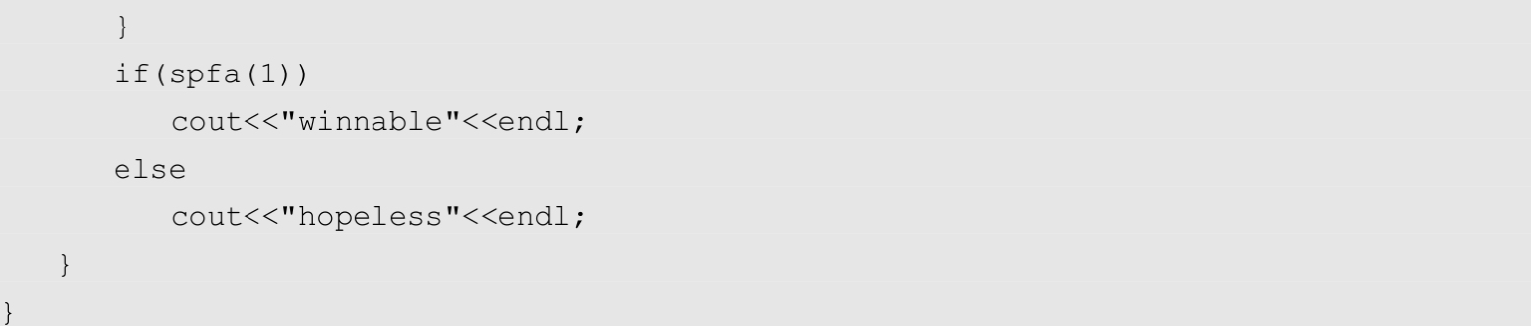
**2. 算法实现**

****

****

****

****

****