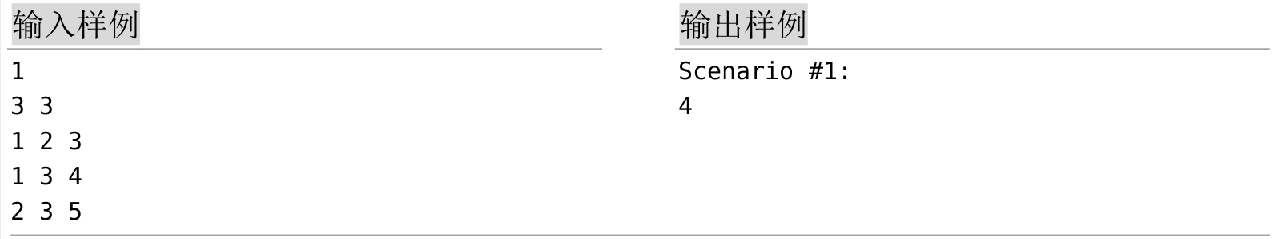
**POJ1797**

**题目描述（POJ1797）：**Hugo需要将巨型起重机从工厂运输到他的客户所在的地方，经过的所有街道都必须能承受起重机的重量。他已经有了所有街道及其承重的城市规划。不幸的是，他不知道如何找到街道的最大承重能力，以将起重机可以有多重告诉他的客户。

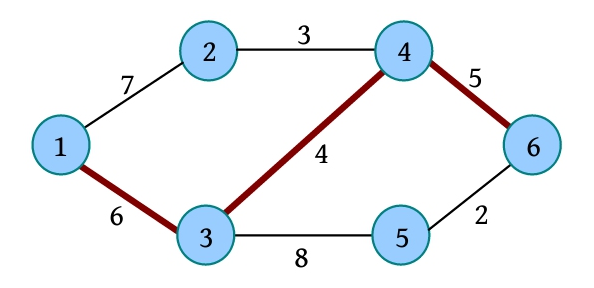
街道（具有重量限制）之间的交叉点编号为1～n。找到从1号（Hugo的地方）到n号（客户的地方）可以运输的最大重量。假设至少有一条路径，所有街道都是双向的。

**输入：**第1行包含测试用例数量。每个测试用例的第1行都包含n（1≤n≤1000）和m，分别表示街道交叉口的数量和街道的数量。以下m行，每行都包含3个整数（正数且不大于106），分别表示街道的开始、结束和承重。在每对交叉点之间最多有一条街道。

**输出：**对每个测试用例，输出都以包含“Scenario #i:”的行开头，其中i是从1开始的测试用例编号。然后单行输出可以运输给客户的最大承重。在测试用例之间有一个空行。



**题解：**本题要求找到一条通路，是最小边权最大的通路，该通路的最小边权即最大承重。如下图所示，从节节点1到节点6有3条通路，其中1-2-4-6的最小边权为3；1-3-4-6的最小边权为4；1-3-5-6的最小边权为2；最小边权最大的通路为1-3-4-6，该通路的最大承重为4，超过4则无法承受。



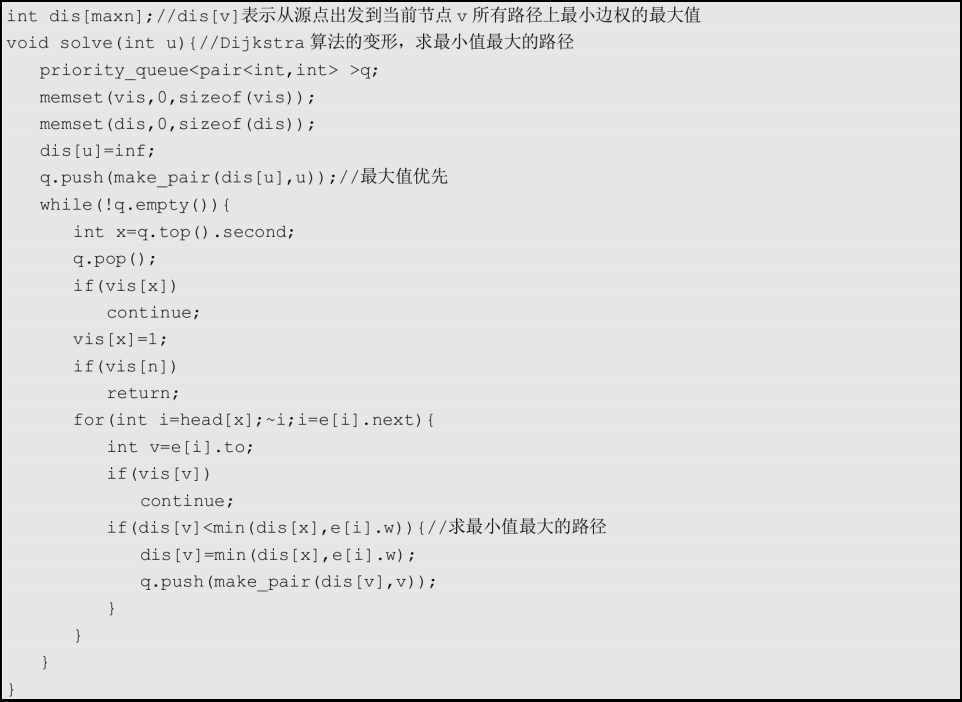
1. **算法设计**

（1）将所有街道都采用链式前向星存储，每个街道都是双向的。

（2）将Dijkstra算法的更新条件变形一下，改为最小值最大的更新。



1. **算法实现**



**POJ1860**

**题目描述（POJ1860）：**有几个货币兑换点，每个点只能兑换两种特定货币。可以有几个专门针对同一种货币的兑换点。每个兑换点都有自己的汇率，货币A到货币B的汇率是1A兑换B的数量。此外，每个交换点都有一些佣金，即必须为交换操作支付的金额。佣金始终以源货币收取。

例如，如果想在兑换点用100美元兑换俄罗斯卢布，而汇率为29.75，佣金为0.39，则将获得(100 - 0.39)×29.75=2963.3975RUR。

可以处理N种不同的货币。货币编号为1～N。对每个交换点都用6个数字来描述：整数A和B（交换的货币类型），以及RAB、CAB、RBA和CBA（分别表示交换A到B和B到A时的汇率和佣金）。

尼克有一些货币S，并想知道他是否能在一些交易所操作之后增加他的资本。当然，他最终想要换回货币S。在进进行操作时所有金额都必须是非负数。

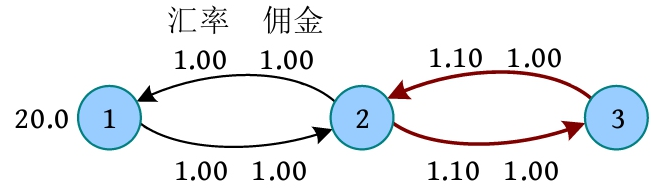
**输入：**输入的第1行包含4个数字：N表示货币类型的数量，M表示交换点的数量，S表示尼克拥有的货币类型，V表示他拥有的货币数量。以下M行，每行都包含6个数字，表示相应交换点的描述。数字由一个或多个空格分隔。1≤S≤N≤100，1≤M≤100，V是实数，0≤V≤103。汇率和佣金在小数点后至多有两位，10-2 ≤汇率≤102，0≤佣金≤102。

**输出：**如果尼克可以增加他的财富，则输出“YES”，在其他情况下输出“NO”。



**题解：**本题从当前货币出发，走一个回路，赚到一些钱。因为走过的边是双向的，因此能走过去就一定能够走回来。只需判断在图中是否有正环，即使这个正环不包含S也没关系，走一次正环就会多赚一些钱。

输入样例1，如下图所示，包含一个正环2 -3-2，每走一次就赚一些钱。



计算过程如下。

• 1-2：(20-1.00)×1.00=19.00。

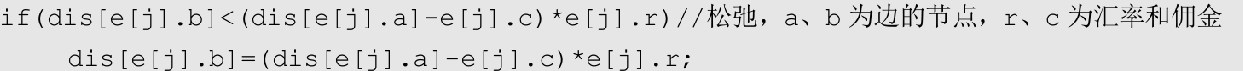
• 2-3、3-2：(19-1.00)×1.10=19.80、(19.8-1.00)×1.10=20.68。

• 2-3、3-2：(20.68-1.00)×1.10=21.648、(21.648-1.00)×1.10=22.7128。

• 2-1：(22.7128-1.00)×1.00=21.7128。

**1. 算法设计**

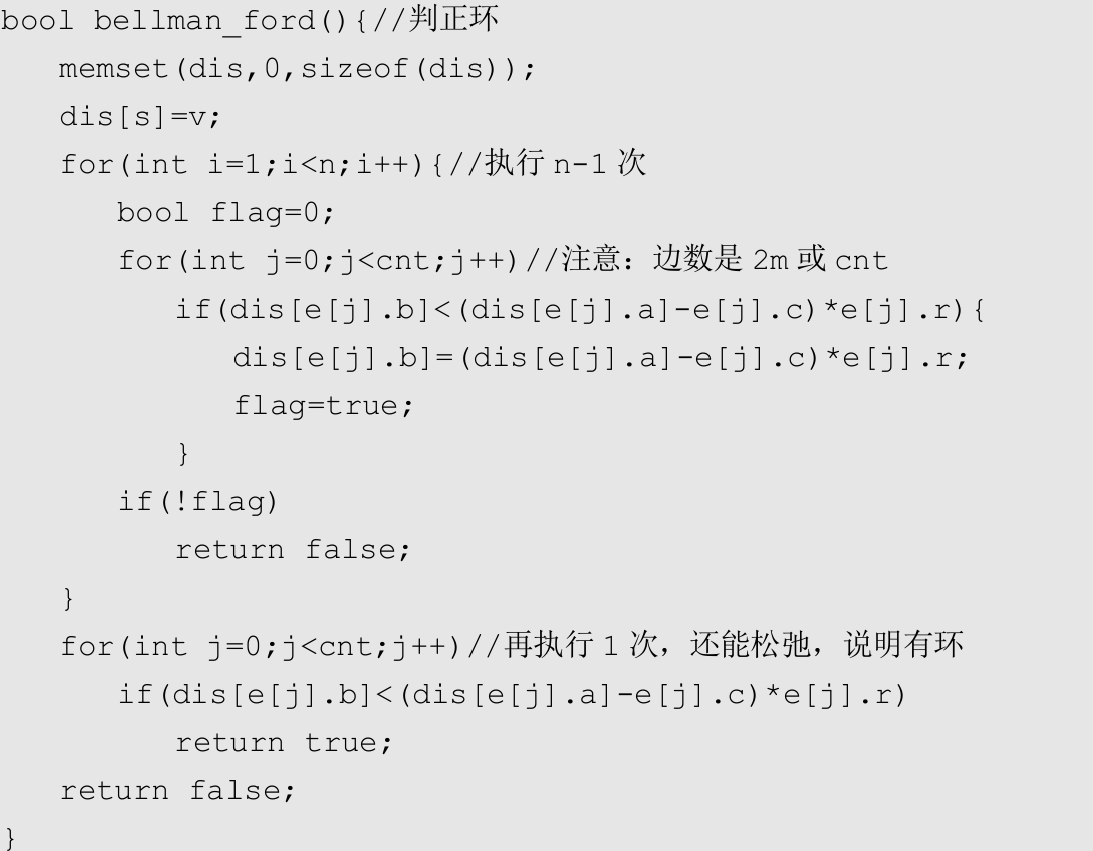
**（1）Bellman-Ford算法，判断正环**。用边松弛n-1次后，再执行一次，如果还可以松弛，则说明有环（是正环还是负环，主要取决于松弛条件）。注意：对双向边，边数是2m或使用边数计数器cnt。

****

**（2）SPFA算法，判断正环。**松弛时，若对一个节点访问n次，则存在环。

**（3）DFS深度优先搜索，判断正环。**若在松弛时访问到已遍历的节点，则存在环。

**2. 算法实现**

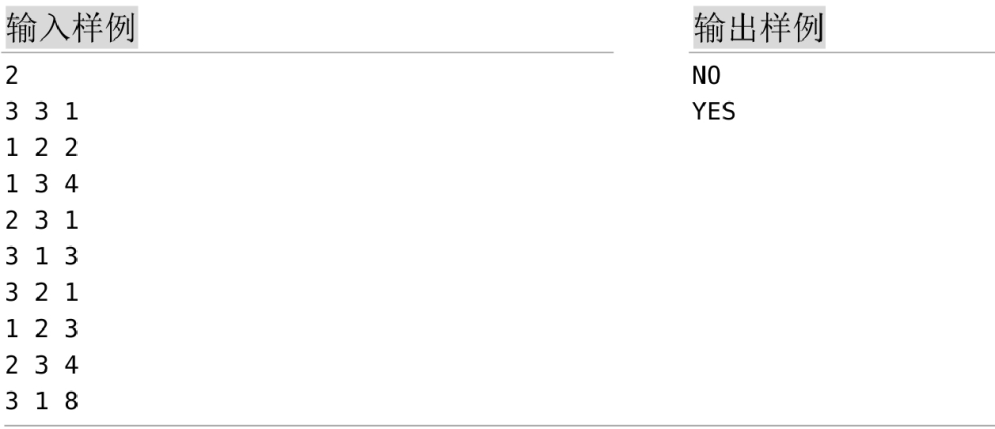


**POJ3259**

**题目描述（POJ3259）：**在探索许多农场时，约翰发现了一些令人惊奇的虫洞。虫洞是非常奇特的，因为它是一条单向路径，可以将人穿越到虫洞之前的某个时间！约翰想从某个地点开始，穿过一些路径和虫洞，并在他出发前的一一段时间返回起点，也许他将能够见到自己。

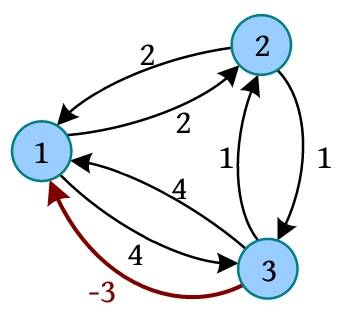
**输入：**第1行是单个整数F（1≤F≤5），表示农场的数量。每个农场的第1行有3个整数N、M、W，表示编号为1～N的N（1≤N≤500）块田、M（1≤M≤2500）条路径和W（1≤W≤200）个虫洞。第2～M+1行，每行都包含3个数字S、E、T，表示穿过S与E之间的路径（双向）需要T秒。两块田都可能有多个路径。第M+2～M+W+1行，每行都包含3个数字S、E、T，表示对从S到E的单向路径，旅行者将穿越T秒。没有路径需要超过10000秒的旅行时间，没有虫洞可以穿越超过10000秒。

**输出：**对于每个农场，如果约翰可以达到目标，则输出“YES”，否则输出“NO”。

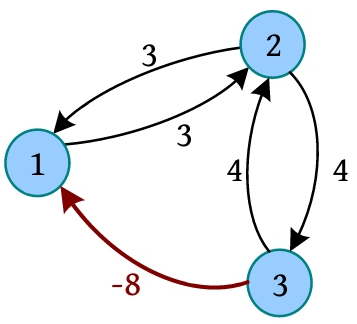


**提示：**对于农场1，约翰无法及时返回；对于农场2，约翰可以在1→2→3→1的周期内及时返回，在他离开前1秒返回他的起始位置。他可以从周期内的任何地方开始实现这一目标。

**题解：**根据输入样例1，如下图所示，约翰无法在他出发之前的时间返回。



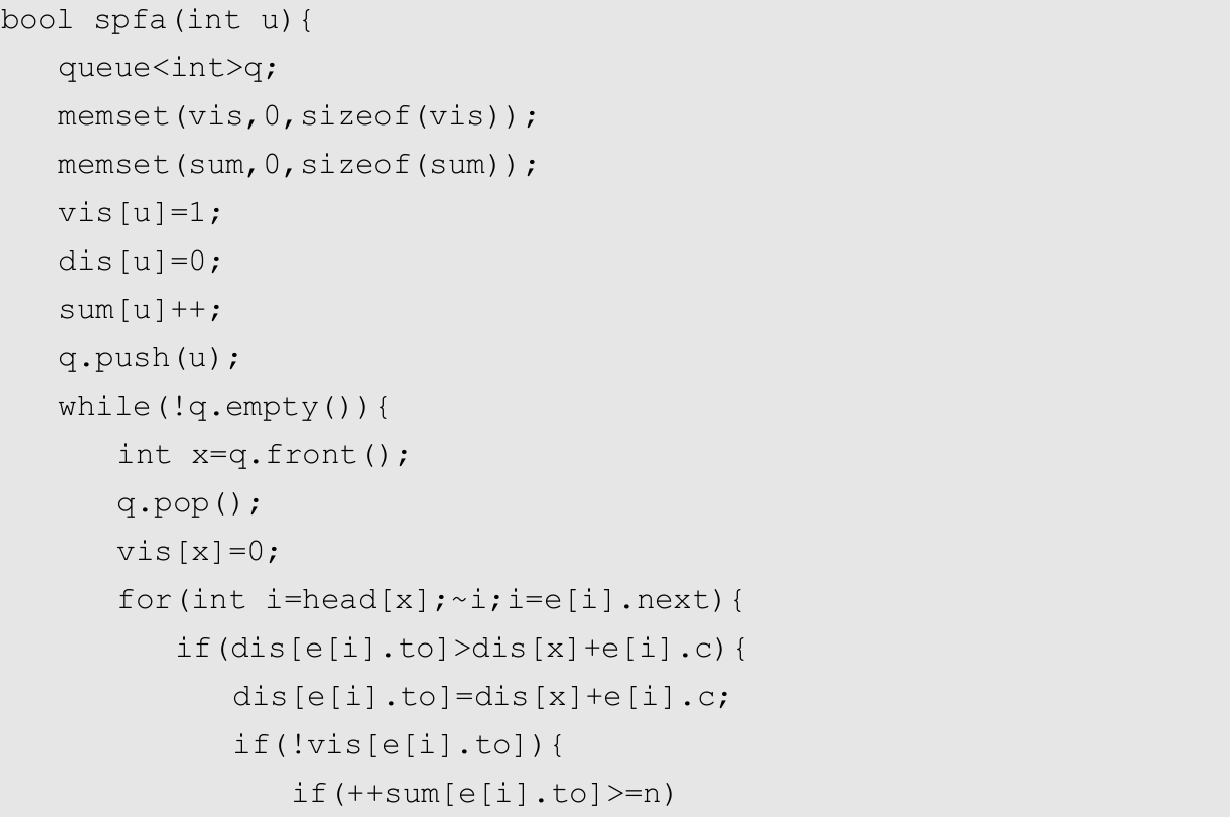
根据输入样例2，如下图所示。约翰可以在1→2→3→1的周期内及时返回，在他离开前1秒返回他的起始位置。他可以从周期内的任何地方开始实现这一目标。因为存在一个负环（边权之和为负）1→2→3→1，边权之和为-1。

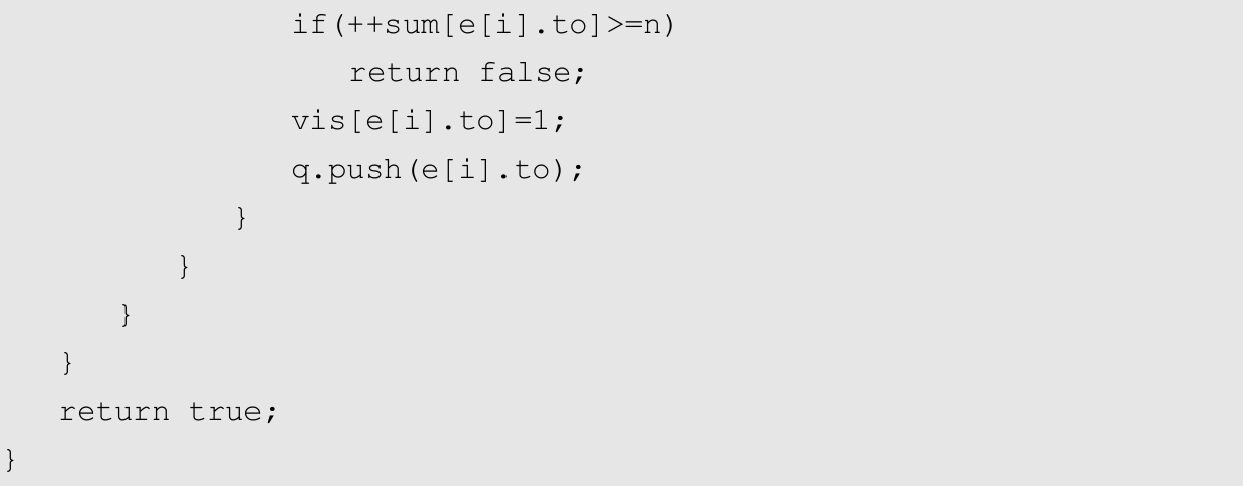
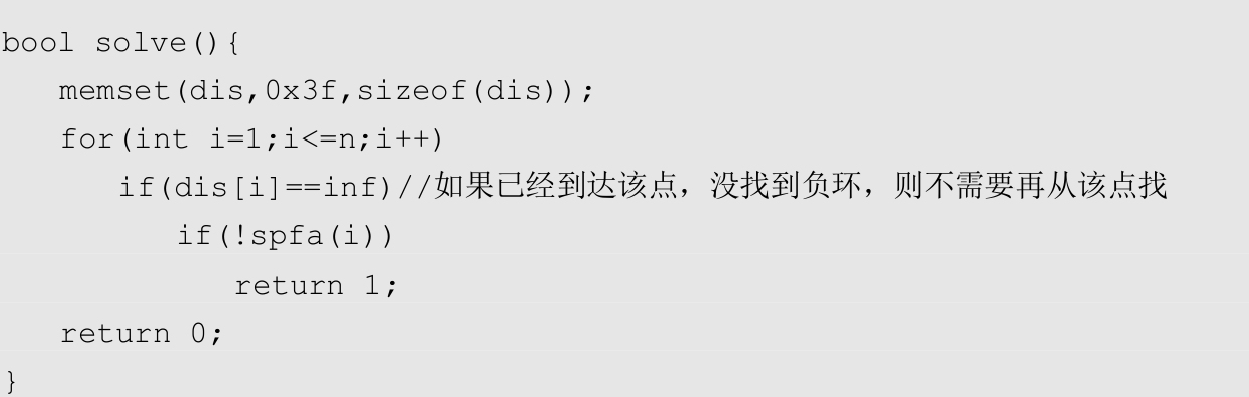


**1. 算法设计**

本题其实就是判断是否有负环，使用SPFA判断负环即可。注意：普通道路是双向的，虫洞是单向的，而且时间为负值。

**2. 算法实现**





**POJ3268**

**题目描述（POJ3268）：**母牛从N个农场中的任一个去参加盛大的母牛聚会，聚会地点在X号农场。共有M条单行道分别连接两个农场，且通过路i需要花Ti时间。每头母牛都必须参加宴会，并且在宴会结束时回到自己的领地，但是每头母牛都会选择时间最少的方案。来时的路和去时的路可能不一样，因为路是单向的。求所有的母牛中参加聚会来回的最长时间。

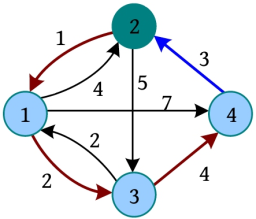
**输入：**第1行包含3个整数N、M和X。在第2～M+1行中，第i+1描述道路i，有3个整数：Ai、Bi和Ti，表示从Ai号农场到Bi号农场需要Ti时间。其中，1≤N≤1000，1≤X≤N，1≤M≤100 000，1≤Ti≤100。

**输出：**单行输出母牛必须花费的时间最大值。



**提示：**母牛从4号农场进入聚会地点（2号农场），再通过1号农场和3号农场返回，共计10个时间。

**题解：**根据输入样例，有4个农场、8条路，聚会地点在2号农场。母牛从4号农场出发，走一个回路4-2-1-3-4，共计10个时间，该时间是所有母牛中来回时间最长的，如下图所示。



**1. 算法设计**

因为母牛来回走的都是最短路径，所以先求每个节点从出发到聚会地点来回的最短路径之和，然后求最大值即可。

（1）从i号农场到聚会地点X，相当于在反向图中从X到i。

（2）从聚会地点X返回到i号农场，相当于在正向图中从X到i。

（3）创建正向图和反向图，都把X作为源点，分别调用SPFA算法求正向图、反向图中源点到其他各个点的最短时间dis[i]和rdis[i]，求最大和值。

**2. 算法实现**

因为正向图、反向图均要调用SPFA算法，因此将图的存储结构e[ ]、head[ ]及最短距离dis[ ]作为参数，调用时传参即可。

