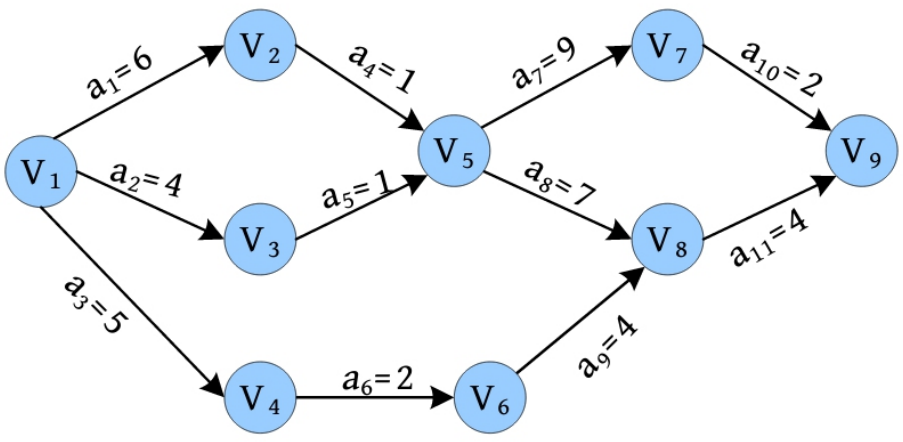
SDUTOJ2498

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）。

输入样例	输出样例
9 11 1 2 6 1 3 4 1 4 5 2 5 1 3 5 1 4 6 2 5 7 9 5 8 7 6 8 4 8 9 4 7 9 2	18 1 2 2 5 5 7 7 9

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。

输入样例	输出样例
5 2 1 2 1 3 4 1	2

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 P_i（0≤P_i≤100）和 P_i 个先决条件。

输出：单行输出完成所有家务所需的最少时间。

输入样例

```
7
5 0
1 1 1
3 1 2
6 1 1
1 2 2 4
8 2 2 4
4 3 3 5 6
```

输出样例

```
23
```

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 A_i、B_i（1≤i≤M），表示从城市 A_i 可以直飞到城市 B_i。

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。

输入样例

```
2
3
0 70 90
4
1 2
1 3
2 4
3 4
3
0 90 70
4
1 2
1 3
2 4
3 4
```

输出样例

```
CASE 1#
points : 90
circuit : 1->3->1

CASE 2#
points : 90
circuit : 1->2->1
```

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。

输入样例

```
5
0 1 2
-60 1 3
-60 1 4
20 1 5
0 0
5
0 1 2
20 1 3
-60 1 4
-60 1 5
0 0
5
0 1 2
21 1 3
-60 1 4
-60 1 5
0 0
5
0 1 2
20 2 1 3
-60 1 4
-60 1 5
0 0
-1
```

输出样例

```
hopeless
hopeless
winnable
winnable
```