第 28 届全国信息学奥林匹克冬令营

NOI Winter Camp 2011

竞赛时间：2011 年 1 月 24 日上午 8:00-13:00

题目名称	最大 XOR 和路径	拼点游戏	关系挖掘
目录	xor	joy	relation
可执行文件名	xor	joy	无
输入文件名	xor.in	joy.in	relation1.in~ relation10.in
输出文件名	xor.out	joy.out	relation1.out~ relation10.out
每个测试点时限	1 秒	3 秒	N/A
测试点数目	10	10	10
每个测试点分值	10	10	10
是否有部分分	否	是	是
题目类型	传统	传统	提交答案
附加文件	N/A	N/A	checker

提交源程序须加后缀

对于 Pascal 语言	xor.pas	joy.pas	N/A
对于 C 语言	xor.c	joy.c	N/A
对于 C++ 语言	xor.cpp	joy.cpp	N/A

注意：最终测试时，所有编译命令均不打开任何优化开关。

最大 XOR 和路径

【问题描述】

XOR（异或）是一种二元逻辑运算，其运算结果当且仅当两个输入的布尔值不相等时才为真，否则为假。XOR 运算的真值表如下（1 表示真，0 表示假）：

输入		输出
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

而两个非负整数的 XOR 是指将它们表示成二进制数，再在对应的二进制位进行 XOR 运算。

譬如 12 XOR 9 的计算过程如下：

$$12=(1100)_2 \quad 9=(1001)_2$$

$$\begin{array}{rr} & 1\ 1\ 0\ 0 \\ \text{XOR} & 1\ 0\ 0\ 1 \\ \hline & 0\ 1\ 0\ 1 \end{array}$$

$$(0101)_2=5$$

故 12 XOR 9 = 5。

容易验证，XOR 运算满足交换律与结合律，故计算若干个数的 XOR 时，不同的计算顺序不会对运算结果造成影响。从而，可以定义 K 个非负整数 $A_1, A_2 \ldots A_{K-1}, A_K$ 的 XOR 和为

$$A_1 \text{ XOR } A_2 \text{ XOR } \ldots \text{ XOR } A_{K-1} \text{ XOR } A_K$$

考虑一个边权为非负整数的无向连通图，节点编号为 1 到 N，试求出一条从 1 号节点到 N 号节点的路径，使得路径上经过的边的权值的 XOR 和最大。

路径可以重复经过某些点或边，当一条边在路径中出现了多次时，其权值在计算 XOR 和时也要被计算相应多的次数，具体见样例。

【输入格式】

输入文件 xor.in 的第一行包含两个整数 N 和 M，表示该无向图中点的数目与边的数目。

接下来 M 行描述 M 条边，每行三个整数 S_i，T_i，D_i，表示 S_i 与 T_i 之间存在一条权值为 D_i 的无向边。

图中可能有重边或自环。

【输出格式】

输出文件 xor.out 仅包含一个整数，表示最大的 XOR 和（十进制结果）。

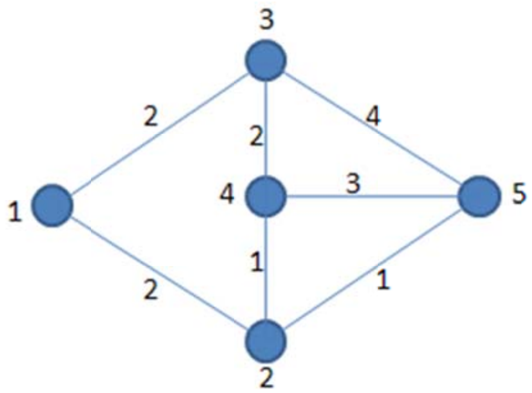
【样例输入】

```
5 7
1 2 2
1 3 2
2 4 1
2 5 1
4 5 3
5 3 4
4 3 2
```

【样例输出】

```
6
```

【样例说明】



如图，路径$1 \to 2 \to 4 \to 3 \to 5 \to 2 \to 4 \to 5$对应的 XOR 和为

$$2 \text{ XOR } 1 \text{ XOR } 2 \text{ XOR } 4 \text{ XOR } 1 \text{ XOR } 1 \text{ XOR } 3 = 6$$

当然，一条边数更少的路径$1 \to 3 \to 5$对应的 XOR 和也是$2 \text{ XOR } 4 = 6$。

【数据规模】

对于 20%的数据，$N \le 100$，$\mathrm{M} \le 1000$，$D_i \le 10^4$；

对于 50%的数据，$N \le 1000$，$\mathrm{M} \le 10000$，$D_i \le 10^{18}$；

对于 70%的数据，$N \le 5000$，$\mathrm{M} \le 50000$，$D_i \le 10^{18}$；

对于 100%的数据，$N \le 50000$，$M \le 100000$，$D_i \le 10^{18}$。

拼点游戏

【问题描述】

小 W 和小 Y 都很喜欢玩一种“拼点游戏”。游戏中两个人分别通过某种操作产生一个数作为自己的“点数”，点数大的一方获胜。“拼点游戏”的规则如下：

1、游戏开始时，给定一个包含n个元素的正整数序列$U = (u_1, u_2, \dots, u_n)$。

2、定义 U 的一个<u>下标序列</u>$I = (i_1, i_2, \dots, i_m)$是满足$1 \le i_1 < i_2 < \cdots < i_m \le n$的一个整数序列($m$ 可以为 0，即序列可以为空)，并且其对应 U 的子序列为$V = (u_{i_1}, u_{i_2}, \dots, u_{i_m})$。

3、定义下标序列$I = (i_1, i_2, \dots, i_m)$对应的点数$D(I)$为

$$D(I) = \sum_{p=1}^{m} u_{i_p} * (-1)^p$$

4、进行游戏时两人分别选择一个下标序列，谁选择的下标序列对应的点数$D(I)$大，谁就获胜。

然而在每次游戏中，小 W 总是能准确无误的算出点数最大的最优下标序列。为了让游戏更加具有竞技性，他们制定了下列额外规则：

Ex1. 小 W 可以选择一个非空区间$[l, r]$，并将$u_l, u_{l+1}, \dots, u_r$同时增加一个整数 c，产生的新序列将取代原序列 U。

Ex2. 当他们对于当前的 U 序列进行一次“拼点游戏”时，允许小 Y 在小 W 选出最优下标序列$I_W = (i_1, i_2, \dots, i_m)$之后，对$I_W$进行任意次修改操作。每次修改操作规则如下：

(1) 任意选择一个正整数 k 满足$2k + 1 \le m$，以及两个<u>非负整数</u> z_1，z_2 满足$i_{2k} + z_1 < i_{2k+1} - z_2$；

(2) 将i_{2k}修改为$i_{2k} + z_1$，将i_{2k+1}修改为$i_{2k+1} - z_2$。

若小 W 选出的下标序列I_W经过小 Y 若干次修改操作之后所对应的点数小于等于 0，则小 Y 获胜。

现在给出小 W 所进行的 Ex1 操作的信息，请你对于每一次“拼点游戏”，帮助他们计算：

a) 小 W 一开始所能选出的最优下标序列对应的点数是多少？
b) 小 Y 最少需要进行几次修改操作才能获胜？即使得$D(I_W) \le 0$。

【输入格式】

输入文件 joy.in 的第一行包含一个正整数 T，表示测试数据的组数。接下来为 T 组数据。

每一组数据的第一行包含两个整数 n 和 q，分别表示 U 中的元素个数和事件个数。

接下来的一行，包含 n 个用一个空格隔开的正整数，第 i 个整数为初始的序列中第 i 个元素 u_i。

接下来 q 行，每行代表一个事件（按事件发生顺序输入）。每行的第一个数非 0 即 1，表示这个事件的类型。

若为 0 ：在 0 之后还有三个整数 l，r 和 c（这四个数之间均有一个空格），表示小 W 将$u_l, u_{l+1}, \ldots, u_r$增加 c；

若为 1 ：表示两人进行了一次“拼点游戏”，你需要输出相应的结果。

输入数据保证序列 U 中的所有元素总是正整数。

【输出格式】

输出文件为 joy.out。

对于每一组测试数据，依次对每一次“拼点游戏”输出一行包含两个由一个空格隔开的整数 $D_{\max}$ 和 X，其中

$D_{\max}$ 为对于当前序列 U，小 W 所能选出的最优下标序列所对应的点数；

X 表示小 Y 最少需要进行几次修改操作才能获胜。如果小 Y 不论多少次操作都无法获胜，则输出-1。

数据保证最优下标序列总是唯一的。

【评分标准】

一个测试点包含多组测试数据，对于该测试点：

如果所有的 $D_{\max}$ 均正确但某个 X 不正确，则可以得到 3 分；

如果所有的 X 均正确但某个 $D_{\max}$ 不正确，则可以得到 7 分；

如果所有回答均正确，则可以得到 10 分。

【样例输入】

```
2
5 9
9 10 7 6 8
1
0 4 5 2
0 3 5 4
1
0 2 5 -2
0 3 5 -3
0 4 5 -2
0 5 5 -4
1
4 3
2 4 3 5
1
0 3 3 3
1
```

【样例输出】

```
3 1
5 -1
0 0
4 -1
4 -1
```

【样例说明】

输入数据包含两组测试数据。

在第一组测试数据中：

第一次“拼点游戏”时，最优下标序列为(1,2,4,5)，小 Y 只需要进行一次修改操作：选择 k=1，以及非负整数 z_1=1，z_2=0。这样经过修改操作之后下标序列将变为(1,3,4,5)，小 Y 获胜。

第三次“拼点游戏”时，序列 U 为(9,8,6,5,3)，小 W 所选择的最优下标序列为空序列，所产生的点数为 0。在这种情况下，小 Y 无法进行任何修改操作（也无需进行任何修改操作），此时小 Y 已经直接获胜。

【数据规模】

对于 10%的数据满足 n，$q \le 13$；

对于 30%的数据满足 n，$q \le 1000$；

对于另外 20%的数据满足 T=1 且 $n \le 40000$；

对于 100%的数据满足 $T \le 3$ 且 n，$q \le 10^5$，同时初始序列 U 满足 $0 < u_i < 2^{31}$，$|c| < 10^5$。

关系挖掘

【问题描述】

有人说："世界上任何两个人最多只需要通过 6 个人就能建立联系"。小 B 对此非常感兴趣，于是着手开展了一些社会网络中关系挖掘的研究工作。

现在小 B 得到了一个包含 N 个人的社会网络数据，该网络中有 M 条关系信息。一条关系信息可以表示为(a_i, b_i, w_i)，表示 a_i 与 b_i 两个人之间存在关系，且紧密度为 w_i（$w_i>0$）。现在小 B 希望挑选其中的 K（$K\leq N$）个人作为研究对象，为使研究工作具有较高的可信度，小 B 希望这 K 个人之间的关系紧密度之和尽量大。

该问题可抽象为：给定一个带权无向图 $G=(V, E)$和整数 K，目标是求出点集 V 的一个子集 S，使得$|S| = K$，且使下式最大化

$$\sum_{(a_i,b_i,w_i)\in E\text{且}a_i\in S\text{且}b_i\in S} w_i$$

【输入格式】

这是一道提交答案的试题，在你的目录下有 10 个输入文件 relation*.in。

输入文件的第一行为三个整数 N，M 和 K，分别表示给定的社会网络中的点数（人数）、边数（关系数目）以及需要选择的点数（人数）K。

接下来 M 行，每行三个正整数 a_i，b_i，w_i，表示一条边（关系）。所有的点（人）按 1 到 N 依次编号。

【输出格式】

对于每一个输入文件，在目录下给出对应的输出文件 relation*.out。

输出文件应包含 K 行，每行一个整数，表示选出的 K 个人的编号。

【样例输入】

```
3 2 2
1 2 3
1 3 5
```

【样例输出】

```
1
3
```

【评分标准】

对于每个测试点，如果你没有输出或者输出不合法则得 0 分。

对于每个测试点，我们设有四个评分参数 m_1、m_2、m_3 和 m_4。假设选手选出的 K 个人之间关系紧密度之和为 c，

若 $c>m_1$，得 12 分；

若 $c=m_1$，得 10 分；

若 $m_1>c\geq m_2$，得 8 分；

若 $m_2>c\geq m_3$，得 5 分；

若 $m_3>c\geq m_4$，得 3 分；

若 $c>0$，得 1 分；

否则，得 0 分。

【如何测试你的输出】

在你的目录下有一个名为 checker 的程序可以用来检查你的输出，你可以在终端中使用以下命令来检查你的输出：

./checker　*N*

其中 N 为测试点的编号，例如，要测试第 3 个测试点可以使用

./checker　3

该程序会检测你的输出是否合法。如果方案合法，程序还会给出该方案的紧密度之和。