CCF CSP 计算机软件能力认证

CCF CSP

第 31 次认证

时间：2023 年 9 月 17 日 13:30 ~ 17:30

题目名称	坐标变换（其一）	坐标变换（其二）	梯度求解	阴阳龙	阻击
题目类型	传统型	传统型	传统型	传统型	传统型
输入	标准输入	标准输入	标准输入	标准输入	标准输入
输出	标准输出	标准输出	标准输出	标准输出	标准输出
每个测试点时限	1.0 秒	2.0 秒	1.0 秒	2.0 秒	2.0 秒
内存限制	512 MiB	512 MiB	512 MiB	1024 MiB	512 MiB
子任务数目	10	20	10	4	25
测试点是否等分	是	是	是	否	是

坐标变换（其一）（sum）

【题目描述】

对于平面直角坐标系上的坐标 (x, y)，小 P 定义了一个包含 n 个操作的序列 $T = (t_1, t_2, \cdots, t_n)$。其中每个操作 t_i（$1 \le i \le n$）包含两个参数 dx_i 和 dy_i，表示将坐标 (x, y) 平移至 $(x + dx_i, y + dy_i)$ 处。

现给定 m 个初始坐标，试计算对每个坐标 (x_j, y_j)（$1 \le j \le m$）依次进行 T 中 n 个操作后的最终坐标。

【输入格式】

从标准输入读入数据。

输入共 $n + m + 1$ 行。

输入的第一行包含空格分隔的两个正整数 n 和 m，分别表示操作和初始坐标个数。

接下来 n 行依次输入 n 个操作，其中第 i（$1 \le i \le n$）行包含空格分隔的两个整数 dx_i、dy_i。

接下来 m 行依次输入 m 个坐标，其中第 j（$1 \le j \le m$）行包含空格分隔的两个整数 x_j、y_j。

【输出格式】

输出到标准输出。

输出共 m 行，其中第 j（$1 \le j \le m$）行包含空格分隔的两个整数，表示初始坐标 (x_j, y_j) 经过 n 个操作后的位置。

【样例输入】

```
3 2
10 10
0 0
10 -20
1 -1
0 0
```

【样例输出】

```
21 -11
20 -10
```

【样例解释】

第一个坐标 $(1,-1)$ 经过三次操作后变为 $(21,-11)$；第二个坐标 $(0,0)$ 经过三次操作后变为 $(20,-10)$。

【子任务】

全部的测试数据满足：$n,m \leq 100$，所有输入数据（x,y,dx,dy）均为整数且绝对值不超过 10^5。

坐标变换（其二）（product）

【题目描述】

对于平面直角坐标系上的坐标 (x, y)，小 P 定义了如下两种操作：

1. 拉伸 k 倍：横坐标 x 变为 kx，纵坐标 y 变为 ky；
2. 旋转 θ：将坐标 (x, y) 绕坐标原点 $(0, 0)$ 逆时针旋转 θ 弧度（$0 \le \theta < 2\pi$）。易知旋转后的横坐标为 $x\cos\theta - y\sin\theta$，纵坐标为 $x\sin\theta + y\cos\theta$。

设定好了包含 n 个操作的序列 $(t_1, t_2, \cdots, t_n)$ 后，小 P 又定义了如下查询：

- `i j x y`：坐标 (x, y) 经过操作 $t_i, \cdots, t_j$（$1 \le i \le j \le n$）后的新坐标。

对于给定的操作序列，试计算 m 个查询的结果。

【输入格式】

从标准输入读入数据。

输入共 $n + m + 1$ 行。

输入的第一行包含空格分隔的两个正整数 n 和 m，分别表示操作和查询个数。

接下来 n 行依次输入 n 个操作，每行包含空格分隔的一个整数（操作类型）和一个实数（k 或 θ），形如 1 k（表示拉伸 k 倍）或 2 θ（表示旋转 θ）。

接下来 m 行依次输入 m 个查询，每行包含空格分隔的四个整数 i、j、x 和 y，含义如前文所述。

【输出格式】

输出到标准输出。

输出共 m 行，每行包含空格分隔的两个实数，表示对应查询的结果。

【样例输入】

```
10 5
2 0.59
2 4.956
1 0.997
1 1.364
1 1.242
1 0.82
2 2.824
1 0.716
2 0.178
```

```
2 4.094
1 6 -953188 -946637
1 9 969538 848081
4 7 -114758 522223
1 9 -535079 601597
8 8 159430 -511187
```

【样例输出】

```
-1858706.758 -83259.993
-1261428.46 201113.678
-75099.123 -738950.159
-119179.897 -789457.532
114151.88 -366009.892
```

【样例解释】

第五个查询仅对输入坐标使用了操作八：拉伸 0.716 倍。

横坐标：$159430 \times 0.716 = 114151.88$

纵坐标：$-511187 \times 0.716 = -366009.892$

由于具体计算方式不同，程序输出结果可能与真实值有微小差异，样例输出仅保留了三位小数。

【子任务】

80% 的测试数据满足：$n, m \leq 1000$；

全部的测试数据满足：

- $n, m \leq 10^5$；
- 输入的坐标均为整数且绝对值不超过 10^6；
- 单个拉伸操作的系数 $k \in [0.5, 2]$；
- 任意操作区间 $t_i, \cdots, t_j$（$1 \leq i \leq j \leq n$）内拉伸系数 k 的乘积在 $[0.001, 1000]$ 范围内。

【评分方式】

如果你输出的浮点数与参考结果相比，满足绝对误差不大于 0.1，则该测试点满分，否则不得分。

【提示】

- C/C++：建议使用 `double` 类型存储浮点数，并使用 `scanf("%lf", &x);` 进行输入，`printf("%f", x);` 输出，也可以使用 `cin` 和 `cout` 输入输出浮点数；`#include <math.h>` 后可使用三角函数 `cos()` 和 `sin()`。
- Python：直接使用 `print(x)` 即可输出浮点数 `x`；`from math import cos, sin` 后可使用相应三角函数。
- Java：建议使用 `double` 类型存储浮点数，可以使用 `System.out.print(x);` 进行输出；可使用 `Math.cos()` 和 `Math.sin()` 调用三角函数。

梯度求解（gradient）

【题目背景】

西西艾弗岛运营公司近期在大力推广智能化市政管理系统。这套系统是由西西艾弗岛信息中心研发的。它的主要目的是，通过详细评估岛上各处的市政设施的状况，来指导市政设施的维护和更新。这套系统的核心是一套智能化的传感器网络，它能够自动地对岛上的市政设施进行评估。对市政设施的维护是需要一定成本的，而年久失修的市政设施也可能给岛上的居民造成损失。为了能够平衡成本和收益，信息中心研发了一款数学模型，描述这些变量和损益之间的复杂数学关系。要想得到最优化的成本，就要依靠梯度下降算法来求解。

梯度下降算法中，求解函数在一点处对某一自变量的偏导数是十分重要的。小 C 负责实现这个功能，但是具体的技术实现，他还是一头雾水，希望你来帮助他完成这个任务。

【题目描述】

设被求算的函数 $u = f(x_1, x_2, \ldots, x_n)$，本题目要求你求出 u 对 x_i 在 $(a_1, a_2, \ldots, a_n)$ 处的偏导数 $\frac{\partial u}{\partial x_i}(a_1, a_2, \ldots, a_n)$。

求算多元函数在一点处对某一自变量的偏导数的方法是：将函数的该自变量视为单一自变量，其余自变量认为是常数，运用一元函数求导的方法求出该偏导数表达式，再代入被求算的点的坐标即可。

例如，要求算 $u = x_1 \cdot x_1 \cdot x_2$ 对 x_1 在 $(1, 2)$ 处的偏导数，可以将 x_2 视为常数，依次应用求导公式。先应用乘法的求导公式：$(x_1 \cdot (x_1 \cdot x_2))' = x_1'(x_1 \cdot x_2) + x_1(x_1 \cdot x_2)'$；再应用常数与变量相乘的求导公式，得到 $x_1' \cdot x_1 \cdot x_2 + x_1 \cdot x_2 \cdot x_1'$；最后应用公式 $x' = 1$ 得到 $1 \cdot x_1 \cdot x_2 + x_1 \cdot x_2 \cdot 1$。整理得 $\frac{\partial u}{\partial x_1} = 2x_2 \cdot x_1$。再代入 $(1, 2)$ 得到 $\frac{\partial u}{\partial x_1}(1, 2) = 4$。

常见的求导公式有：

- $c' = 0$（c是常数）
- $x' = 1$
- $(u + v)' = u' + v'$
- $(cu)' = cu'$（c是常数）
- $(u - v)' = u' - v'$
- $(uv)' = u'v + uv'$

本题目中，你需要求解的函数 f 仅由常数、自变量和它们的加法、减法、乘法组成。且为程序识读方便，函数表达式已经被整理为逆波兰式（后缀表达式）的形式。例如，$x_1 \cdot x_1 \cdot x_2$ 的逆波兰式为 `x1 x1 * x2 *`。逆波兰式即为表达式树的后序遍历的结果。若要从逆波兰式还原原始计算算式，可以按照这一方法进行：假设存在一个空栈 S，依次读取逆波兰式的每一个元素，若读取到的是变量或常量，则将其压入 S 中；若读取到的是计算符号，则从 S 中取出两个元素，进行相应运算，再将结果压入 S 中。最后，

若 S 中存在唯一的元素，则该表达式合法，其值即为该元素的值。例如对于逆波兰式 `x1 x1 * x2 *`，按上述方法读取，栈 S 的变化情况依次为（左侧是栈底，右侧是栈顶）：

1. x_1；
2. x_1，x_1；
3. $(x_1 \cdot x_1)$；
4. $(x_1 \cdot x_1)$，x_2；
5. $((x_1 \cdot x_1) \cdot x_2)$。

【输入格式】

从标准输入读入数据。

输入的第一行是由空格分隔的两个正整数 n、m，分别表示要求解函数中所含自变量的个数和要求解的偏导数的个数。

输入的第二行是一个逆波兰式，表示要求解的函数 f。其中，每个元素用一个空格分隔，每个元素可能是：

- 一个自变量 x_i，用字符 `x` 后接一个正整数表示，表示第 i 个自变量，其中 $i = 1, 2, \ldots, n$。例如，`x1` 表示第一个自变量 x_1。
- 一个整常数，用十进制整数表示，其值在 -10^5 到 10^5 之间。
- 一个运算符，用 `+` 表示加法，`-` 表示减法，`*` 表示乘法。

输入的第三行到第 $m + 2$ 行，每行有 $n + 1$ 个用空格分隔的整数。其中第一个整数是要求偏导数的自变量的编号 $i = 1, 2, \ldots, n$，随后的整数是要求算的点的坐标 $a_1, a_2, \ldots, a_n$。输入数据保证，对于所有的 $i = 1, 2, \ldots, n$，a_i 都在 -10^5 到 10^5 之间。

【输出格式】

输出到标准输出。

输出 m 行，每行一个整数，表示对应的偏导数对 $10^9 + 7$ 取模的结果。即若结果为 y，输出为 k，则保证存在整数 t，满足 $y = k + t \cdot (10^9 + 7)$ 且 $0 \le k < 10^9 + 7$。

【样例 1 输入】

```
2 2
x1 x1 x1 * x2 + *
1 2 3
2 3 4
```

【样例 1 输出】

```
15
3
```

【样例 1 解释】

读取逆波兰式，可得被求导的式子是：$u = x_1 \cdot (x_1 \cdot x_1 + x_2)$，即 $u = x_1^3 + x_1 x_2$。
对 x_1 求偏导得 $\frac{\partial u}{\partial x_1} = 3x_1^2 + x_2$。代入 $(2, 3)$ 得到 $\frac{\partial u}{\partial x_1}(2, 3) = 15$。
对 x_2 求偏导得 $\frac{\partial u}{\partial x_2} = x_1$。代入 $(3, 4)$ 得到 $\frac{\partial u}{\partial x_2}(3, 4) = 3$。

【样例 2 输入】

```
3 5
x2 x2 * x2 * 0 + -100000 -100000 * x2 * -
3 100000 100000 100000
2 0 0 0
2 0 -1 0
2 0 1 0
2 0 100000 0
```

【样例 2 输出】

```
0
70
73
73
999999867
```

【样例 2 解释】

读取逆波兰式，可得被求导的式子是：$u = x_2 \cdot x_2 \cdot x_2 + 0 - (-10^5) \cdot (-10^5) \cdot x_2$，即 $u = x_2^3 - 10^{10} x_2$。
因为 u 中实际上不含 x_1 和 x_3，对这两者求偏导结果均为 0。
对 x_2 求偏导得 $\frac{\partial u}{\partial x_2} = 3x_2^2 - 10^{10}$。

【子任务】

测试点	n	m	表达式的性质
1, 2	$= 1$	≤ 100	仅含有 1 个元素
3, 4			仅含有一个运算符
5, 6	≤ 10		含有不超过 120 个元素，且不含乘法
7, 8			含有不超过 120 个元素
9, 10	≤ 100		

【提示】

C++ 中可以使用 `std::getline(std::cin, str)` 读入字符串直到行尾。

当计算整数 n 对 M 的模时，若 n 为负数，需要注意将结果调整至区间 $[0, M)$ 内。

阴阳龙（dragon）

【题目描述】

西西艾弗岛的下方是一个庞大的遗迹群，神兽“阴阳龙”栖居在这个遗迹群中。

为了得到这件宝物，西西艾弗遗迹探索有限公司（以下简称“公司”）派遣了 p 名员工前往遗迹群，这些员工依次编号为 1 到 p。

遗迹可以视为一个大小为 $n \times m$ 的网格，左下角坐标 $(1,1)$，右上角坐标 (n,m)。初始时，第 i 名员工所在的位置是 (x_i, y_i)。保证所有员工初始所在的位置两两不同。

作为神兽，阴阳龙有着特殊之处。当其在 $\mathbf{p} = (u,v)$ 位置以强度 $t \in [1,7]$ 现身时，会导致遗迹群的环境发生阴和阳的变转，从而导致在遗迹中的人的位置发生变化。

具体来说，阴阳龙首先观察右、右上、上、左上、左、左下、下和右下这八个方向，并在这八个方向找到和阴阳龙“距离”最近的员工（不包括 $\mathbf{p}$）的“距离”。其中，垂直和水平方向的“距离”是指员工和阴阳龙连线的长度；斜线方向的“距离”是指员工和阴阳龙连线在水平方向上投影的长度。设想从阴阳龙的位置同时出发，分别向这 8 个方向前进，每一单位时间运动 1 个“距离”。如果在某一时刻，在某一方向刚好遇到一位员工，则此时前进的距离即被记为 k；否则，如果在某一时刻，在某一方向上刚好到达遗迹的边界，但是在此之前任何方向上都没有遇到员工，则令 $k = 0$。形式化描述上述确定 k 的方法是：

记 $\mathbf{d}_0$ 到 $\mathbf{d}_7$ 依次为向量 $(1,0),(1,1),(0,1),(-1,1),(-1,0),(-1,-1),(0,-1),(1,-1)$，令：

$$K_1 = \left\{k \in \mathbb{N}^+ \mid \exists i \in [0,7], j \in [1,p], \text{s.t.}(x_j, y_j) = \mathbf{p} + k\mathbf{d}_i\right\}$$

$$K_2 = \left\{k \in \mathbb{N}^+ \mid \forall i \in [0,7], (\mathbf{p} + k\mathbf{d}_i) \in [1,n] \times [1,m]\right\}$$

其中：

- (x_i, y_i) 为第 i 名员工在此次阴阳龙现身前的位置（这个位置可能和其初始位置不同，但为了方便起见，我们使用同一个记号）；
- K_1 为所有员工到阴阳龙距离组成的集合；
- K_2 为从阴阳龙出发直至在某一方向抵达边界所包括全部的距离组成的集合。

若 $K = K_1 \bigcap K_2 = \emptyset$，则令 $k = 0$；否则令 $k = \min K > 0$。

例如，参考下图中的例子，其中左下角为 $(1,1)$，右上角为 $(7,7)$，共有 8 名员工，位置如图。

若 $\mathbf{p} = (4,4)$，那么员工 1 刚好在阴阳龙所在位置，不计入；员工 3 不在阴阳龙的 8 个方向上，不计入；员工 2、4、5、6 与阴阳龙“距离”是 2；员工 7、8、9 与阴阳龙“距离”是 3，因此有 $K_1 = \{2,3\}$。由于与阴阳龙“距离”为 3 就到达了遗迹的边界，所以有 $K_2 = \{1,2,3\}$。因此 $k = 2$。

若 $\mathbf{p} = (2,2)$，那么员工 2、3、7、8、9 都不在阴阳龙的 8 个方向上，不计入；员工 1、6 与阴阳龙的“距离”是 2；员工 4、5 与阴阳龙的“距离”是 4，因此有 $K_1 = \{2,4\}$。

由于与阴阳龙“距离”为 1 时，就在向下、向左、向左下三个方向上到达了遗迹的边界，所以有 $K_2 = \{1\}$。因此 $k = 0$。

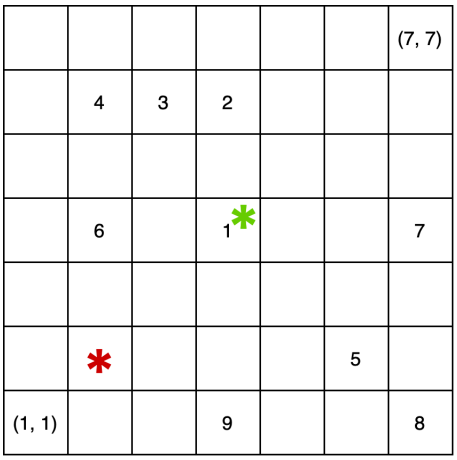

图 1: 变化前各员工位置

如果 $k > 0$，则将八个方向上的距离为 k 的位置上的员工以 $\mathbf{p}$ 为中心逆时针旋转 t 倍的八分之一个圆周的角度。形式化地：

- 若 $k = 0$，则什么也不会发生。
- 否则，$\forall i \in [0, 7]$，若 $\mathbf{p} + k\mathbf{d}_i$ 位置上有员工，那么其该员工会被移动到 $\mathbf{p} + k\mathbf{d}_{(i+t) \bmod 8}$。

易知在所有员工移动结束后，每个位置上仍至多有一个员工。例如，在上图所示的例子中取 $\mathbf{p} = (4, 4), t = 1$，则变化后各员工所在位置如下图所示。

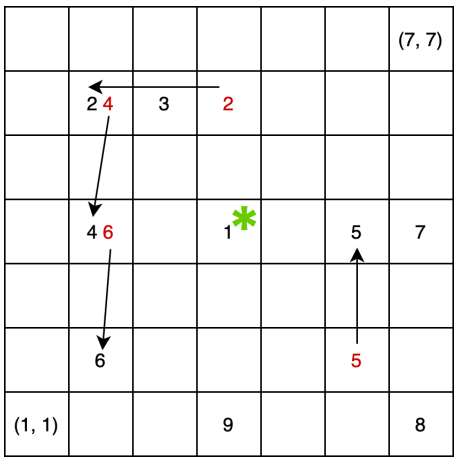

图 2: 变化后各员工位置

在全部员工进入遗迹群后，西西艾弗遗迹探索有限公司总共探测到 q 次阴阳龙的现身。很不幸的是，由于来自东方神秘力量的干扰，这 q 次阴阳龙的现身后，西西艾弗遗迹探索有限公司失去了所有员工的位置信息，因此他希望你帮他计算出所有员工的位置。

【输入格式】

从标准输入读入数据。

第一行四个正整数 n, m, p, q；

接下来 p 行，第 i 行两个正整数 (x_i, y_i) 表示第 i 名员工的初始位置。

保证所有员工初始所在的位置两两不同。

接下来 q 行，第 i 行三个正整数 u_i, v_i, t_i 表示西西艾弗遗迹探索有限公司探测到的第 i 次阴阳龙现身的位置和强度。

【输出格式】

输出到标准输出。

为了减少输出量，设 q 次阴阳龙的现身后所有员工的位置为 $(x_1, y_1), \ldots, (x_p, y_p)$，则你只需要输出：

$$\bigoplus_{i=1}^{p} i \times x_i + y_i$$

其中 $\bigoplus$ 表示按位异或，即 C/C++ 中的 ^ 运算符。

【样例 1 输入】

```
3 3 9 1
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
2 2 1
```

【样例 1 输出】

```
20
```

【样例 1 解释】

阴阳龙现身前，每个员工所在的位置如下：

```
3 6 9
2 5 8
1 4 7
```

阴阳龙现身一次后，每个员工所在位置如下：

```
6 9 8
3 5 7
2 1 4
```

【数据范围】

子任务编号	$n \leq$	$m \leq$	$p \leq$	$q \leq$	子任务分值
1	10^3	10^3	10^5	10^5	40
2	10^9	10^9	10^3	10^3	15
3	10^5	10^5	10^5	10^5	25
4	10^9	10^9			20

对于全部数据：

$1 \leq n, m \leq 10^9, 1 \leq p, q \leq 1 \times 10^5, 1 \leq x_i, u \leq n, 1 \leq y_i, v \leq m, 1 \leq t_i \leq 7$。

保证所有员工初始所在的位置两两不同。

阻击（block）

【题目描述】

上回提到，西西艾弗岛下方有一个庞大的遗迹群，栖息着一种名为“阴阳龙”的神兽。然而隔壁的狄迪吉岛盯上了西西艾弗岛，决定发动一场战争，试图从遗迹群中掠夺有价值的宝物。由此，西西艾弗岛不得不陷入一场漫长的阻击战中，史称“阴阳龙阻击战”。

狄迪吉岛拥有胜过西西艾弗岛的科技实力和武装水平，西西艾弗岛很快发现形势不妙：全歼敌军似乎是不可能的，唯一的策略是凭借主场作战的优势和人海战术，尽可能给敌军带来损失，当敌军发现发动进攻的损失明显超过收益时，就会无趣而归。

具体而言，西西艾弗岛共有 n 座城市，有 $n-1$ 条道路连接这些城市，使得所有城市之间均可以通过道路互相到达。容易发现，任意两座城市之间都有唯一一条不经过重复城市的路径。

由于缺乏城市巷战的实力，西西艾弗岛决定将防御重心放在道路上。在每条道路上均派遣了一定的军队防守，当敌军经过时对其发动阻击。虽然由于实力的差距，并不能阻止敌军通过道路，但仍然可以对敌军造成一定的损失。

然而，敌军具有更强的科技，可以趁机对道路附近的遗迹进行探索，并掠夺其中的宝物——这也正是敌军发动战争的意义所在。如此，敌军通过一条道路时，“发掘宝物的收益”w 和“受到阻击的损失”b 两个值是独立的。

西西艾弗岛事先在狄迪吉岛中安插了一系列间谍，得到的情报消息如下：敌军将选择西西艾弗岛的两座城市作为进攻的“起点”和“终点”，并派遣军队在进攻起点城市登陆，沿两座城市间唯一的路径进攻至终点城市。同时，间谍还背负着另外一个重要的使命：影响敌军对于起点和终点城市的决策，使得敌军受到的总损失尽可能大，其中“总损失”定义为敌军经过的每条道路上的“受到阻击的损失”减去“发掘宝物的收益”之和，即 总损失 $=\sum_{e\text{是路径上的每条边}}(b_e-w_e)$。

此外，遗迹中宝物的价值与所处的环境属性密切相关，而阴阳龙的“现身”会使得环境的阴阳属性发生变化，这会使得敌军通过现身位置处的某一条道路时“发掘宝物的收益”w 发生变化。

这样的“阴阳龙现身”事件共会发生 m 次，你的任务就是帮助间谍计算出在所有事件前及每次事件后，敌军对于起点和终点城市的决策应当怎样改变，以最大化敌军的总损失。

【输入格式】

从标准输入读入数据。

第 1 行，两个非负整数 n,m，分别表示西西艾弗岛的城市数和“阴阳龙现身”事件数。

接下来 $n-1$ 行，每行 4 个非负整数 u_i, v_i, w_i, b_i，表示第 i 条道路连接城市 u_i 和 v_i，敌军在这条道路上“发掘宝物的收益”为 w_i，“受到阻击的损失”为 b_i。

接下来 m 行，每行 2 个非负整数 x_i, y_i，表示一次“阴阳龙现身”事件，使得第 x_i 条道路的“发掘宝物的收益”变为 y_i。

【输出格式】

输出到标准输出。

输出 $m+1$ 行，每行一个非负整数，分别表示在所有事件前及每次事件后，对敌军造成的最大总损失。

【样例 1 输入】

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

【样例 1 输出】

```
0
1
3
4
```

【样例 1 解释】

在最初，由于敌人攻打每一条道路都会有正收益，因此间谍最好的策略就是将进攻起点和终点选为同一座城市，这样敌军的总损失为 0。

第 1 次事件后，间谍可以将进攻起点和终点分别选在城市 3 和 4，这样敌军的总损失为 $3-2=1$。

第 2 次事件后，间谍可以将进攻起点和终点分别选在城市 4 和 5，这样敌军的总损失为 $(3-2)+(5-3)=3$。

第 3 次事件后，间谍可以将进攻起点和终点分别选在城市 1 和 5，这样敌军的总损失为 $(4-1)+(1-2)+(5-3)=4$。

【数据范围】

对于所有测试数据保证：$2 \le n \le 10^5, 0 \le m \le 10^5, 1 \le u_i, v_i \le n, 1 \le x_i \le n-1, 0 \le w_i, b_i, y_i \le 10^9$。

测试点编号	$n \le$	$m \le$	特殊性质
1	20	20	无
2	300	300	
$3 \sim 4$	3000	3000	A
$5 \sim 6$			B
$7 \sim 9$			无
10	10^5	0	A
11			B
12			无
$13 \sim 15$		10^5	A
$16 \sim 18$			B
$19 \sim 21$			C
$22 \sim 25$			无

特殊性质 A：$u_i = i, v_i = i+1$。

特殊性质 B：$0 \le w_i, y_i \le 10^8 \le b_i$。

特殊性质 C：保证任意两座城市均可在经过不超过 100 条道路的前提下互相到达。