Problem A. Vacant Seat

输入文件：*standard input* 输出文件：*standard output* 时间限制：2 秒
内存限制：256 MB

这是一个交互问题。

令$N \geq 3$为奇数。

有N个座位围成一圈。座位编号为 0 到 $N-1$。对于每个i $(0 \leq i \leq N-2)$，座位i和座位$i+1$是相邻的。另外，座位$N-1$和座位0相邻。

每个座位要么空着，要么被男人或女人占据。然而，相邻的两个座位上没有两个同性别的人占据。因为N是大于1的奇数，所以可以证明至少有一个空座位。

给定了 N，但没有给出座位的状态。您的目标是正确猜测任一空座位的 ID 号。为此，您可以重复发送以下查询：

选择一个整数 i $(0 \leq i \leq N-1)$。如果座位i为空，则问题解决。否则，您会收到座位上人员的性别通知i。

通过发送最多 20 个查询来猜测空座位的 ID 号。

Interaction Protocol

第一行给出一个整数 N — 座位数（N 是奇数，$3 \leq N \leq 99\,999$）。

之后，您开始发送查询。查询由一个整数 i $(0 \leq i \leq N-1)$ 组成 - 座位数。不要忘记通过行尾字符结束查询并在每次查询后刷新标准输出。

交互器的响应是三个可能的答案之一：“`Vacant`”、“`Male`”和“`Female`”。其中每一个都意味着座位 i 是空的，分别被男性和女性占据。

当您收到“`Vacant`”答案时，立即终止程序。如果您发送的查询超过 20 个，您将收到错误答案的判决。

Example

standard input	standard output
3 Male Female Vacant	 0 1 2

Note

样本中，$N = 3$ 和座位 0、1、2 分别为男性、女性和空置。

Problem B. Colorful Doors

输入文件：*standard input* 输出文件
：*standard output* 时间限制：2 秒
内存限制：256 MB

有一座桥连接河的左岸和右岸。这座桥上的不同位置放置了 $2N$ 扇门，并涂上了不同的颜色。门的颜色由 1 到 N 的整数表示。对于每个 k $(1 \leq k \leq N)$，正好有两扇漆成颜色 k 的门。

斯努克决定从左岸过桥到右岸。他会继续向右走，但这样做时会发生以下事件：

当 Snuke 触摸一扇涂有颜色 k $(1 \leq k \leq N)$ 的门时，他传送到另一扇涂有颜色 k 的门的右侧。

可以证明他最终会到达正确的银行。

对于每个i $(1 \leq i \leq 2N-1)$，从左侧起第i个门和第$(i+1)$个门之间的部分将被称为部分i。过桥后，Snuke 为每个 i $(1 \leq i \leq 2N-1)$ 记录他是否走过第 i 部分。这条记录以长度为 $2N-1$ 的字符串 s 的形式提供给您。对于每个 i $(1 \leq i \leq 2N-1)$，如果 Snuke 走过了 i 部分，则 s 中的第 i 个字符是“1”；否则，第 i 个字符是“0”。

确定是否存在与记录一致的门布置。如果存在，则构造 CT
一种这样的安排。

Input

输入按以下格式给出：

N

s

约束：

N 是整数，$(1 \leq N \leq 10^5)$，s由‘0’和‘1’组成，$|s| = 2N-1$。

Output

如果没有与记录一致的门排列，则打印“No”。如果存在这样的排列，请在第一行打印“Yes”，然后在第二行打印一个这样的排列，格式如下：$c_1\ c_2\ \ldots\ c_{2N}$。这里，对于每个 i $(1 \leq i \leq 2N)$，c_i 是从左边算起第 i 个门的颜色。

Examples

standard input	standard output
2 010	Yes 1 1 2 2
2 001	No
3 10110	Yes 1 3 2 1 2 3
3 10101	No
6 00111011100	Yes 1 6 1 2 3 4 4 2 3 5 6 5

Problem C. Construct Point

输入文件：*standard input* 输出文件
：*standard output* 时间限制：2 秒
内存限制：256 MB

您有 Q 个三角形，编号为 1 到 Q。

第i个三角形的顶点坐标按逆时针顺序依次为(x_{1_i}, y_{1_i})、(x_{2_i}, y_{2_i})和(x_{3_i}, y_{3_i})。这里，x_{1_i}、x_{2_i}、x_{3_i}、y_{1_i}、y_{2_i}和y_{3_i}都是整数。

对于每个三角形，判断其内部（不包括边界）是否存在包含网格点。如果存在，则构造一个这样的点。

Input

输入按以下格式给出：

Q

$x_{11}\, y_{1_1}\ x_{2_1}\ y_{2_1}\ x_{3_1}\ y_{3_1 2_2}\ y_{1_2}\ x_{2_2}\ y_{2_2}\ x_{3_2}\ y_{3_2}$

x

...

$x_{1_Q}\ y1_Q\ x2_Q\ y2_Q\ x3_Q\ y$
3_Q 约束：

所有输入值均为整数，$1 \leq Q \leq 10\,000$、$0 \leq x_{1_i}, x_{2_i}, x_{3_i}, y_{1_i}, y_{2_i}, y_{3_i} \leq 10^9$、$(x_{1_i}, y_{1_i})$、$(x_{2_i}, y_{2_i})$ 和 d $(x3_i, y_{3_i})$ 按逆时针顺序列出，三角形是非退化的。

Output

输出应包含 Q 行。

在第i行中，如果三角形i内部（不包括边界）不包含网格点，则打印“-1 -1”。如果存在，请选择一个这样的网格点，然后打印其 x 坐标和 y 坐标，中间留有一个空格。

Example

standard input	standard output
4 1 7 3 5 5 7 1 4 1 2 5 4 6 1 7 1 7 6 11 3 11 4 8 5	3 6 2 3 -1 -1 10 4

Problem D. Knapsack and Queries

输入文件：*standard input* 输出文件：*standard output* 时间限制：10 秒
内存限制：1024 MB

首先，给出正整数 MOD。
你有一个背包，一开始是空的。
您必须执行 Q 查询。

- 在每个查询中，您必须首先执行 **ADD** 或 `REMOVE` 操作，然后执行 `FIND`。
- 对于 **ADD** 运算，您将获得正整数 w 和 v。您将重量为 w 且值为 v 的 cookie 放入背包中。
- 对于**REMOVE**操作，你从背包中取出重量最小的饼干并吃掉它。
- 在每次 **ADD** 或 **REMOVE** 执行 **FIND** 操作后：给定正整数 l 和 r，您回答以下问题：您能否从背包中选择 cookie，以便 $l \le (X \bmod MOD) \le r$ (X 是所选 cookie) 的重量之和？如果不能，则输出-1。否则输出所选cookie值的最大总和。

Input

输入按以下格式给出：

MOD
Q
$t'_1\ w'_1\ v'_1\ l'_1\ r'_1$
$t'_2\ w'_2\ v'_2\ l'_2\ r'_1$
2
...
$t'_Q\ w'_Q\ v'_Q\ l'_Q\ r'_Q$

约束：

$0 \le t'_i, w'_i, v'_i, l'_i, r'1 \le 2^{30} - 1$, $2 \le MOD \le 500$, $1 \le Q \le 100\ 000$。
查询被加密，如下所述。解密$t'_i, w'_i, v'_i, l'_i, r'_i$即可得到$t_i, w_i, v_i, l_i, r_i$。
您可以假设 $1 \le w_i, v_i \le 10^9$, $0 \le l_i \le r_i \le MOD - 1$, $t_i = 1$ 用于 **ADD+FIND** 查询，$t_i = 2$ 用于 **REMOVE+FIND** 查询（在本例中为 $w_i = v_i = 0$），由 **ADD** 给出的 cookie 比之前 **ADD** 添加的任何 cookie 都为 *heavier*，并且在执行时**REMOVE**，背包不是空的。

我们用C++11（或更高版本）、Java、D、C#准备了解密代码。使用类 `Crypto` 进行解密。类 `Crypto` 的代码及其使用示例可以从 `http://opentrains.mipt.ru/~ejudge/crypto.zip` 上传，适用于 C++11（或更高版本）、Java、D、C#。

以下是 C++ 的示例：

```
#include <cstdint> //uint8_t, uint32_t

class Crypto {
public:
    Crypto() {
```

```
        sm = cnt = 0;
        seed();
    }

    int decode(int z) {
        z ^= next();
        z ^= (next() << 8);
        z ^= (next() << 16);
        z ^= (next() << 22);
        return z;
    }

    void query(long long z) {
        const long long B = 425481007;
        const long long MD = 1000000007;
        cnt++;
        sm = ((sm * B % MD + z) % MD + MD) % MD;
        seed();
    }
private:
    long long sm;
    int cnt;

    uint8_t data[256];
    int I, J;

    void swap_data(int i, int j) {
        uint8_t tmp = data[i];
        data[i] = data[j];
        data[j] = tmp;
    }

    void seed() {
        uint8_t key[8];
        for (int i = 0; i < 4; i++) {
            key[i] = (sm >> (i * 8));
        }
        for (int i = 0; i < 4; i++) {
            key[i+4] = (cnt >> (i * 8));
        }

        for (int i = 0; i < 256; i++) {
            data[i] = i;
        }
        I = J = 0;

        int j = 0;
        for (int i = 0; i < 256; i++) {
            j = (j + data[i] + key[i%8]) % 256;
            swap_data(i, j);
        }
    }
```

```
    uint8_t next() {
        I = (I+1) % 256;
        J = (J + data[I]) % 256;
        swap_data(I, J);
        return data[(data[I] + data[J]) % 256];
    }
};
```

解密过程按以下方式进行:

- 首先，创建 `class Crypto` 的实例。
- 对于每个查询，首先按 t', w', v', l', r' 的顺序调用 `decode` 函数。返回值为t, w, v, l, r。然后执行查询并使用 **FIND** 的结果调用 `query` 函数。

示例 C++ 代码:

```
#include <cstdio>
#include <cstdlib>
#include <cstdint> //uint8_t, uint32_t

class Crypto {
    ...
};

int main() {
    int MOD, Q;
    scanf("%d %d", &MOD, &Q);
    Crypto c;
    for (int i = 0; i < Q; i++) {
        int t, w, v, l, r;
        scanf("%d %d %d %d %d", &t, &w, &v, &l, &r);
        t = c.decode(t);
        w = c.decode(w);
        v = c.decode(v);
        l = c.decode(l);
        r = c.decode(r);
        if (t == 1) {
            (add candy(w, v))
        } else {
            (delete candy)
        }
        long long ans = (answer for query(l, r));
        c.query(ans);
        printf("%lld\n", ans);
    }
}
```

请注意，类 `Crypto` 大约需要 200 时间来处理 $Q = 100\,000$。

Output

对于每个查询，打印 **FIND** 操作的结果。

Examples

standard input	standard output
10 7 281614559 249378726 433981056 466775634 683612866 727071329 787572584 591471796 328464426 757737734 279580343 240336097 538846427 808491898 224313807 222498984 42804452 371605808 667115067 791865961 68683864 1045549765 515479514 1067782238 349547144 907343711 381772625 149003422 879314974 953881571 883899098 700164610 414212891 752949213 972845634	10 0 -1 21 -1 11 111

standard input	standard output
7 20 281614559 249378726 433981094 466775639 683612870 59536386 999828879 241246766 434670565 174365647 172060134 848462699 857413429 182122460 807914643 808426426 600772095 829463884 974102196 354283529 370037909 1024921880 664216868 194331103 140834169 917331875 242953442 205247688 335469789 1055568137 823475244 641321246 617915164 160300810 1073617378 892669150 939175632 904628449 606339993 1059849410 829170894 436718235 288920513 228195002 55212938 772189413 373108543 94133155 610930061 513937768 986619331 175674265 812546186 865335970 605634588 880196843 1071068047 723408215 587598264 380801783 393196081 141080294 584230885 135343295 661927186 5740819 967233824 22597607 888639499 467454437 365679801 515258603 989059385 962028117 761163096 357270919 737051059 569528959 935653628 70506031 869282414 947492121 280522456 96822010 856514221 155948699 826430734 291243254 381421299 617876780 980891674 833928389 1048677341 522527723 223764850 50617939 963598173 281959650 499436870 47455938	0 134 90 158 -1 22 238 269 179 189 121 53 41 41 -1 58 -1 84 -1 149

Note

解码样本1的结果:

```
10
7
1 5 10 5 5
2 0 0 0 9
1 7 10 2 4
1 12 11 9 9
```

```
2 0 0 1 1
1 22 10 2 3
1 32 100 4 4
```

解码样本2的结果:

```
7
20
1 5 44 0 1
1 11 90 0 3
2 0 0 3 4
1 18 68 1 6
1 25 32 2 3
1 31 22 2 3
1 32 26 1 5
1 36 31 3 6
2 0 0 2 5
1 43 10 3 6
2 0 0 5 6
2 0 0 3 4
2 0 0 2 4
2 0 0 1 5
2 0 0 3 5
1 49 48 0 4
2 0 0 1 5
1 50 36 0 6
1 56 48 3 5
1 59 17 3 5
```

Problem E. XorTree

输入文件：*standard input* 输出文件：*standard output* 时间限制：2 秒
内存限制：256 MB

给你一棵具有 N 个顶点的树。顶点编号为 0 到 $N-1$，边编号为 1 到 $N-1$。边 i 连接顶点 x_i 和 y_i，并且具有值 a_i。您可以多次执行以下操作：选择一条简单路径和一个非负整数x，然后对于属于该路径的每条边e，通过执行a_e来更改a_e：$= a_e \oplus x$ ($\oplus$表示XOR)。

您的目标是所有边 e 的 $a_e =$ 为 0。找出实现该目标所需的最少操作数。

Input

输入按以下格式给出：

N

$x_1\ y_1\ a_1$
$x_2\ y_2\ a_2$

...

$x_{N-1}\ y_{N-1}\ a_{N-1}$

约束：

$2 \leq N \leq 10^5$, $0 \leq x_i, y_i \leq N-1$, $0 \leq a_i \leq 15$. 给定的图是一棵树，所有输入值都是整数。

Output

找出实现目标所需的最少操作次数。

Examples

standard input	standard output
5 0 1 1 0 2 3 0 3 6 3 4 4	3
2 1 0 0	0

Note

在示例1中，可以通过三个操作来实现该目标，如下：首先，选择连接Vertex 1, 2和$x = 1$的路径，然后选择连接Vertex 2, 3和$x = 2$的路径；最后，选择连接Vertex 0, 4和$x = 4$的路径。

Problem F. Antennas On Tree

输入文件：*standard input* 输出文件
：*standard output* 时间限制：2 秒
内存限制：256 MB

我们有一棵具有 N 个顶点的树。顶点编号为 0 到 $N-1$，第 i 条边 ($0 \leq i < N-1$) 连接顶点 a_i 和 b_i。对于每对顶点 u 和 v ($0 \leq u, v < N$)，我们将距离 $d(u,v)$ 定义为路径 u-v 上的边数。

预计其中一个顶点将被来自外太空的外星人入侵。 Snuke 希望在入侵发生时立即识别该顶点。为此，他决定在一些顶点安装天线。

首先，他决定天线的数量，K ($1 \leq K \leq N$)。然后，他选择 K 个不同的顶点 $x_0, x_1, \ldots, x_{K-1}$，在其上分别安装天线 $0, 1, \ldots, K-1$。如果顶点v被外星人入侵，天线k ($0 \leq k < K$)将输出距离$d(x_k, v)$。根据这些 K 输出，Snuke 将识别被入侵的顶点。因此，为了识别被入侵的顶点，无论入侵哪一个，都必须满足以下条件：对于每个顶点 u ($0 \leq u < N$)，考虑向量 $(d(x_0,u), \ldots, d(x_{K-1},u))$。这些 N 向量是不同的。

当条件满足时，求天线数量 K 的最小值。

Input

输入按以下格式给出：

N
$\ldots$
a_0 b_0 a_1 b_1 a_{N-2} b_{N-2} 约束： $2 \leq N \leq 10^5$, $0 \leq a_i, b_i < N$，给定的图是一棵树。

Output

当条件满足时，打印天线数量 K 的最小值。

Examples

standard input	standard output
5 0 1 0 2 0 3 3 4	2
2 0 1	1
10 2 8 6 0 4 1 7 6 2 3 8 6 6 9 2 4 5 8	3

Note

在示例 1 中，在顶点 1 和 3 上安装天线。然后，以下五个向量是不同的:

- $(d(1, 0), d(3, 0)) = (1, 1)$
- $(d(1, 1), d(3, 1)) = (0, 2)$
- $(d(1, 2), d(3, 2)) = (2, 2)$
- $(d(1, 3), d(3, 3)) = (2, 0)$
- $(d(1, 4), d(3, 4)) = (3, 1)$

在示例 2 中，可能的解决方案之一是在顶点 0 上安装天线。在示例 3 中，可能的解决方案之一是在顶点 0、4、9 上安装天线。

Problem G. Rectangles

输入文件：*standard input* 输出文件
：*standard output* 时间限制：2 秒
内存限制：256 MB

我们有一个大小为 $A \times B \times C$ 的长方体，分为 $1 \times 1 \times 1$ 个小立方体。小立方体的坐标从 (0, 0, 0) 到 ($A-1, B-1, C-1$)。

令p、q和r为整数。考虑以下一组 abc 小立方体：

$\{(\ (p+i)\ \bmod A$、$(q+j) \bmod B$、$(r+k) \bmod C\ ) \mid i$、j 和 k 是满足 $0 \le i <$ 的整数 a，，，，$0 \le j < b, 0 \le k < c\ \}$

可以使用一些整数 p、q 和 r 以上述格式表示的一组小立方体称为 d 大小为 $a \times b \times c$ 的 *torus cuboid*。

求满足以下条件的大小为 $a \times b \times c$ 的环面长方体的组数：

- 该集合中没有两个圆环长方体有交集。
- 该集合中所有环面长方体的并集是维度为 $A \times B \times C$ 的整个长方体。

由于答案可能太大，因此打印它对 $10^9 + 7$ 取模。

Input

输入按以下格式给出：

$a\ b\ c\ A\ B\ C$

约束：

$1 \le a < A \le 100, 1 \le b < B \le 100, 1 \le c < C \le 100$，所有输入值均为整数。

Output

打印满足条件的大小为 $a \times b \times c$ 的圆环长方体组的数量，模 $10^9 +$ 7。

Examples

standard input	standard output
1 1 1 2 2 2	1
2 2 2 4 4 4	744
2 3 4 6 7 8	0
2 3 4 98 99 100	471975164

Problem H. Generalized Insertion Sort

输入文件：*standard input* 输出文件
：*standard output* 时间限制：2 秒
内存限制：256 MB

给你一棵有 N 个顶点的有根树。顶点编号为 0, 1, . . . , $N-1$。根为顶点 0，顶点 i $(i = 1, 2, \ldots, N-1)$ 的父节点为顶点 p_i。

最初，在顶点i中写入整数a_i。这里，$(a_0, a_1, \ldots, a_{N-1})$ 是 (0, 1, . . . , $N-1$) 的排列。

您最多可以执行以下操作 25 000 次。目标是让价值写入 在顶点 i 等于 i。

- 选择一个顶点并将其命名为 v。考虑连接顶点 0 和 v 的路径。
- 旋转路径上写入的值。也就是说，对于路径上的每条边（i, p_i），将顶点p_i中写入的值替换为在此操作之前写入顶点i (中的值，并将v中写入的值替换为顶点0（在此操作之前）中写入的值。
- 您可以选择顶点 0，在这种情况下该操作不执行任何操作。

Input

输入按以下格式给出:

N

p_1 $p2$ ... p_{N-1} ... a_{N-1}

a_0 a_1
约束:

$2 \le N \le 2000$, $0 \le p_i \le i-1$, $(a_0, a_1, \ldots, a_{N-1})$ 是 (0, 1, . . . , $N-1$) 的排列。

Output

在第一行中，打印操作数 Q。在第二行到第 $(Q+1)$ 行中，按顺序打印所选的顶点。

Examples

standard input	standard output
5 0 1 2 3 2 4 0 1 3	2 3 4
5 0 1 2 2 4 3 1 2 0	3 4 3 1

Note

在样本 1 中，第一次操作后，写入顶点 0, 1, . . . , 4 的值为 4, 0, 1, 2, 3。在样本 2 中，第一次操作后，写入顶点 0, 1, . . . , 4 的值为 3, 1, 0, 2, 4、第二次操作后，顶点0, 1, . . . , 4写入的值为1, 0, 2, 3, 4。

Problem I. ADD, DIV, MAX

输入文件：*standard input* 输出文件
：*standard output* 时间限制：5 秒
内存限制：256 MB

给定一个整数序列 $a_0, a_1, \ldots, a_{N-1}$。
您必须执行 Q 查询，每个查询都是以下查询之一：

- **ADD** $t = 0$、`l r x`：对于介于 l 和 r 之间的每个 i，$a_i = a_i + x$。
- **DIV** $t = 1$、`l r x`：对于l和r之间的每个i，包括a_i =floor(a_i/x)，其中floor(y)是不大于y的最大整数。
- **MAX** $t = 2$. `l r x=0`：打印max($a_l, a_{l+1}, \ldots, a_r$)。

Input

输入按以下格式给出：
N Q
a_0 $a1$ ... a_{N-1}
t_1 l_1 r_1 x_1
t_2 l_2 r_2 x_2
$\ldots$
t_Q l_Q r_Q x_Q
约束：
所有输入值均为整数，$1 \le N, Q \le 200\,000$, $0 \le a_i \le 10^8$, $t_i = 0, 1, 2$, $0 \le l_i \le r_i \le N - 1$, $1 \le x_i \le 1000$ if $t_i \ne 2$, $x_i = 0$ if $t_i = 2$。

Output

对于每个 **MAX** 查询，打印 max($a_l, a_{l+1}, ..., a_r$)。

Examples

standard input	standard output
5 7 1 2 3 4 5 2 0 4 0 0 0 1 10 2 0 4 0 2 2 2 0 1 0 1 4 2 0 0 0 2 1 1 0	5 12 3 2 3

standard input	standard output
4 7 0 1 0 1 2 0 3 0 0 0 3 1 1 0 3 2 2 0 3 0 0 0 3 1 1 0 3 2 2 0 3 0	1 1 1
10 20 13 1 22 8 28 18 23 9 22 27 1 3 4 5 1 8 8 8 0 3 9 5 0 2 6 3 1 1 3 7 2 2 2 0 2 3 5 0 0 1 4 2 2 0 1 0 0 3 9 8 2 1 9 0 0 8 9 5 1 5 7 7 0 3 5 7 0 7 9 7 2 1 6 0 0 1 1 7 1 4 8 10 2 0 9 0 1 5 6 1	3 26 13 40 30 52

Note

对于样本 1， max(1, 2, 3, 4, 5) = 5 1, 2, 3, 4, 5 → 11 , 12, 3, 4, 5 max(11, 12, 3, 4, 5) = 12 最大(3) = 3 11, 12, 3, 4, 5 → 2, 3, 3, 4, 5 最大(2) = 2 最大(3) = 3

Problem J. Simple APSP Problem

输入文件：*standard input* 输出文件
：*standard output* 时间限制：3 秒
内存限制：256 MB

您将获得一个 $H \times W$ 网格。左上角的方块索引为 $(0, 0)$，第 th 处的方块 e
右下角的索引为 $(H-1, W-1)$。

N 方格 $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ 被漆成黑色，所有其他方格被漆成白色。

令白色方块 A 和 B 之间的最短距离为从 A **visiting only white squares** 到达 B 所需的最少移动次数，其中一个人可以一次移动到共享一侧（上、下、左或右）的相邻方块。

由于总共有 $H \times W - N$ 个白色方块，因此有 $C^2_{H \times W - N}$ 种方法可以选择其中两个白色方块。对于每种 $C^2_{H \times W - N}$ 方式，找到所选方格之间的最短距离，然后找到所有这些距离的总和，以 $1\,000\,000\,007 = 10^9 + 7$ 为模。

Input

输入按以下格式给出：

H W

N

x_1 y_1
x_2 y_2
$\ldots$

x_N y_N

约束：

$1 \le H, W \le 10^6, 1 \le N \le 30, 0 \le x_i \le H-1, 0 \le y_i \le W-1$. 如果是 $i \ne j$，则为 $x_i \ne x_j$ 或 $y_i \ne y_j$。保证至少有一个白色方块。对于每对白色方块A和B，有可能从仅访问白色方块的A到达B。

Output

打印以 $10^9 + 7$ 为模的最短距离之和。

Examples

standard input	standard output
2 3 1 1 1	20
2 3 1 1 2	16
3 3 1 1 1	64
4 4 4 0 1 1 1 2 1 2 2	268
1000000 1000000 1 0 0	333211937

Note

在示例 1 中，我们有e 下一个网格（‘.’表示白色方块，‘!’ - bl 确认方）：

```
...
.!.
```

我们将字母表分配给白色方块，如下所示。

```
ABC
D!E
```

所以我们得到（这里 $dist(A, B)$ 是 A 和 B 之间的最短距离）：

$dist(A, B) = 1$、$dist(A, C) = 2$、$dist(A, D) = 1$、$dist(A, E) = 3$、$dist(B, C) = 1$、$dist(B, D) = 2$、$dist(B, E) = 2$、$dist(C, D) = 3$、$dist(C, E) = 1$、$dist(D, E) = 4$，它们的总和为 20。

在示例 2 中，我们将字母分配给白色方块，如下所示。

```
ABC
DE!
```

所以我们得到：

$dist(A, B) = 1$、$dist(A, C) = 2$、$dist(A, D) = 1$、$dist(A, E) = 2$、$dist(B, C) = 1$、$dist(B, D) = 2$、$dist(B, E) = 1$、$dist(C, D) = 3$、$dist(C, E) = 2$、$dist(D, E) = 1$，它们的总和为 16。

Problem K. Forest Task

输入文件：*standard input* 输出文件：*standard output* 时间限制：2 秒
内存限制：256 MB

给您一个具有 N 个顶点和 M 个边的森林。顶点编号为 0 到 $N-1$。边以 (x_i, y_i) 格式给出，这意味着顶点 x_i 和 y_i 通过边连接。

每个顶点 i 被分配值 a_i。您想要在给定的森林中添加边，以便森林相互连接。要添加边，请选择两个不同的顶点 i 和 j，然后在 i 和 j 之间跨越一条边。此操作花费 $a_i + a_j$ 美元，之后顶点 i 和 j 都无法再次选择。

求使森林连通所需的最低总成本，如果不可能则打印“Impossible”。

Input

输入按以下格式给出：

$N\ M$

$a_0\ a_1 \dots\ a_{N-1}$
$x_1\ y_1$

...

$x_M\ y_M$

约束：

$1 \le N \le 100\,000$, $0 \le M \le N-1$, $1 \le a_i \le 10^9$, $0 \le x_i, y_i \le N-1$. 给定的图是一个森林。一个 二 输入值是整数。

Output

打印连接森林所需的最低总成本，如果不可能，则打印“Impossible”。

Examples

standard input	standard output
7 5 1 2 3 4 5 6 7 3 0 4 0 1 2 1 3 5 6	7
5 0 3 1 4 1 5	Impossible
1 0 5	0

Note

在示例 1 中，如果我们连接顶点 0 和 5，则图变为连通，成本为 $1 + 6 = 7$。在示例 2 中，我们无法使图连通。

在示例 3 中，无论我们是否做某事，图都是连通的。