Problem A. Vacant Seat

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

This is an interactive problem.

Let $N \geq 3$ be an odd number.

There are N seats arranged in a circle. The seats are numbered 0 through $N-1$. For each i ($0 \leq i \leq N-2$), seat i and seat $i+1$ are adjacent. Also, seat $N-1$ and seat 0 are adjacent.

Each seat is either vacant, or occupied by a man or a woman. However, no two adjacent seats are occupied by two people of the same sex. It can be shown that there is at least one empty seat because N is an odd number greater than 1.

You are given N, but the states of the seats are not given. Your objective is to correctly guess the ID number of any one of the empty seats. To do so, you can repeatedly send the following query:

Choose an integer i ($0 \leq i \leq N-1$). If Seat i is empty, the problem is solved. Otherwise, you are notified of the sex of the person at seat i.

Guess the ID number of an empty seat by sending at most 20 queries.

Interaction Protocol

In the first line you are given one integer N — the number of seats (N is odd, $3 \leq N \leq 99\,999$).

After that, you start to send queries. A query consists of one integer i ($0 \leq i \leq N-1$) — number of seat. Do not forget to end the query by end-of-line character and to flush standard output after each query.

The responce of the interactor is one of three possible answers: "`Vacant`", "`Male`" and "`Female`". Each of these means that Seat i is empty, occupied by a man and occupied by a woman, respectively.

When you receive "`Vacant`" answer, immediately terminate the program. If you send more than 20 queries, you will receive the Wrong Answer verdict.

Example

standard input	standard output
3 Male Female Vacant	 0 1 2

Note

In the sample, $N = 3$, and Seat 0, 1, 2 are occupied by a man, occupied by a woman and vacant, respectively.

Problem B. Colorful Doors

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

There is a bridge that connects left and right banks of a river. There are $2N$ doors placed at different positions on this bridge, painted in some colors. The colors of the doors are represented by integers from 1 through N. For each k ($1 \leq k \leq N$), there are exactly two doors painted in color k.

Snuke decides to cross the bridge from the left bank to the right bank. He will keep on walking to the right, but the following event will happen while doing so:

At the moment Snuke touches a door painted in Color k ($1 \leq k \leq N$), he teleports to the right side of the other door painted in color k.

It can be shown that he will eventually get to the right bank.

For each i ($1 \leq i \leq 2N-1$), the section between the i-th and $(i+1)$-th doors from the left will be referred to as section i. After crossing the bridge, Snuke recorded whether or not he walked through Section i, for each i ($1 \leq i \leq 2N-1$). This record is given to you as a string s of length $2N-1$. For each i ($1 \leq i \leq 2N-1$), if Snuke walked through section i, the i-th character in s is '1'; otherwise, the i-th character is '0'.

Determine if there exists an arrangement of doors that is consistent with the record. If it exists, construct one such arrangement.

Input

Input is given in the following format:

N

s

Constraints:

N is integer, ($1 \leq N \leq 10^5$), s consists of '0' and '1', $|s| = 2N-1$.

Output

If there is no arrangement of doors that is consistent with the record, print "No". If there exists such an arrangement, print "Yes" in the first line, then print one such arrangement in the second line, in the following format: c_1 c_2 ... c_{2N}. Here, for each i ($1 \leq i \leq 2N$), c_i is the color of the i-th door from the left.

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

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You have Q triangles, numbered 1 through Q.

The coordinates of the vertices of the i-th triangle are (x_{1_i}, y_{1_i}), (x_{2_i}, y_{2_i}) and (x_{3_i}, y_{3_i}) in counterclockwise order. Here, x_{1_i}, x_{2_i}, x_{3_i}, y_{1_i}, y_{2_i} and y_{3_i} are all integers.

For each triangle, determine if there exists a grid point contained in its interior (excluding the boundary). If it exists, construct one such point.

Input

Input is given in the following format:

Q

$x_{1_1}\ y_{1_1}\ x_{2_1}\ y_{2_1}\ x_{3_1}\ y_{3_1}$

$x_{1_2}\ y_{1_2}\ x_{2_2}\ y_{2_2}\ x_{3_2}\ y_{3_2}$

...

$x_{1_Q}\ y_{1_Q}\ x_{2_Q}\ y_{2_Q}\ x_{3_Q}\ y_{3_Q}$

Constraints:

All input values are integers, $1 \le Q \le 10\,000$, $0 \le x_{1_i}, x_{2_i}, x_{3_i}, y_{1_i}, y_{2_i}, y_{3_i} \le 10^9$, (x_{1_i}, y_{1_i}), (x_{2_i}, y_{2_i}) and (x_{3_i}, y_{3_i}) are listed in counterclockwise order, the triangles are non-degenerate.

Output

Output should contain Q lines.

In the i-th line, if there is no grid point contained in the interior (excluding the boundary) of Triangle i, print "`-1 -1`". If it exists, choose one such grid point, then print its x-coordinate and y-coordinate with a space in between.

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

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	1024 mebibytes

First, you are given the positive integer MOD.

You have a knapsack, that is empty at first.

You have to perform Q queries.

- In each query, you have first to perform either **ADD** or `REMOVE` operation, and then perform `FIND`.
- For **ADD** operation, you are given positive integers w and v. You put the cookie with weight w and value v to the knapsack.
- For **REMOVE** operation, you take out the cookie with smallest weight from the knapsack and eat it.
- After each **ADD** or **REMOVE** you perform **FIND** operation: given positive integers l and r, you answer the follow question: can you choose cookies from that knapsack so that the $l \le (X \bmod MOD) \le r$ (X is sum of the weight of the selected cookies)?. If you can't, output -1. Otherwise output the maximum sum of the value of the selected cookies.

Input

Input is given in the following format:

MOD

Q

$t'_1\ w'_1\ v'_1\ l'_1\ r'_1$

$t'_2\ w'_2\ v'_2\ l'_2\ r'_2$

...

$t'_Q\ w'_Q\ v'_Q\ l'_Q\ r'_Q$

Constraints:

$0 \le t'_i, w'_i, v'_i, l'_i, r'_1 \le 2^{30} - 1$, $2 \le MOD \le 500$, $1 \le Q \le 100\,000$.

Queries are encrypted as described later. You can get t_i, w_i, v_i, l_i, r_i by decryption $t'_i, w'_i, v'_i, l'_i, r'_i$.

You may assume that $1 \le w_i, v_i \le 10^9$, $0 \le l_i \le r_i \le MOD - 1$, $t_i = 1$ for **ADD+FIND** query, $t_i = 2$ for **REMOVE+FIND** query (and in this case $w_i = v_i = 0$), that the cookie, which is given by **ADD**, is *heavier* than any cookies which added by the previous **ADD**, and that when executing **REMOVE**, the knapsack isn't empty.

We prepared the decryption code with C++11 (or later), Java, D, C#. Use class `Crypto` for decryption. The code of class `Crypto` and examples of its usage can be uploaded from `http://opentrains.mipt.ru/~ejudge/crypto.zip` for C++11 (or later), Java, D, C#.

Here is the example for C++:

```
#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];
    }
};
```

The decryption process works in the next way:

- First, you make the instance of `class Crypto`.
- For each query first call the `decode` function in order of t', w', v', l', r'. The return values are t, w, v, l, r. Then perform the query and call the `query` function with the result of the **FIND**.

The sample C++ code:

```
#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);
    }
}
```

Note that class `Crypto` consume time about 200 to process $Q = 100\,000$.

Output

For each query print the result of **FIND** operation.

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

The result of decoding Sample 1:

```
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
```

The result of decoding Sample 2:

```
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

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are given a tree with N vertices. The vertices are numbered 0 through $N-1$, and the edges are numbered 1 through $N-1$. Edge i connects vertex x_i and y_i, and has a value a_i. You can perform the following operation any number of times: choose a simple path and a non-negative integer x, then for each edge e that belongs to the path, change a_e by executing $a_e := a_e \oplus x$ ($\oplus$ denotes XOR).

Your objective is to have $a_e = 0$ for all edges e. Find the minimum number of operations required to achieve it.

Input

Input is given in the following format:

N

$x_1\ y_1\ a_1$

$x_2\ y_2\ a_2$

...

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

Constraints:

$2 \leq N \leq 10^5$, $0 \leq x_i, y_i \leq N-1$, $0 \leq a_i \leq 15$. The given graph is a tree, all input values are integers.

Output

Find the minimum number of operations required to achieve the objective.

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

In Sample 1, the objective can be achieved in three operations, as follows: first, choose the path connecting Vertex $1, 2$, and $x = 1$, then, choose the path connecting Vertex $2, 3$, and $x = 2$; lastly, choose the path connecting Vertex $0, 4$, and $x = 4$.

Problem F. Antennas On Tree

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

We have a tree with N vertices. The vertices are numbered 0 through $N-1$, and the i-th edge $(0 \leq i < N-1)$ connects vertex a_i and b_i. For each pair of vertices u and v $(0 \leq u, v < N)$, we define the distance $d(u, v)$ as the number of edges on the path u-v.

It is expected that one of the vertices will be invaded by aliens from outer space. Snuke wants to immediately identify that vertex when the invasion happens. To do so, he has decided to install an antenna on some vertices.

First, he decides the number of antennas, K $(1 \leq K \leq N)$. Then, he chooses K different vertices, $x_0, x_1, \ldots, x_{K-1}$, on which he installs antenna $0, 1, \ldots, K-1$ respectively. If vertex v is invaded by aliens, antenna k $(0 \leq k < K)$ will output the distance $d(x_k, v)$. Based on these K outputs, Snuke will identify the vertex that is invaded. Thus, in order to identify the invaded vertex no matter which one is invaded, the following condition must hold: for each vertex u $(0 \leq u < N)$, consider the vector $(d(x_0, u), \ldots, d(x_{K-1}, u))$. These N vectors are distinct.

Find the minumum value of K, the number of antennas, when the condition is satisfied.

Input

Input is given in the following format:

N

a_0 b_0

a_1 b_1

...

a_{N-2} b_{N-2}

Constraints:

$2 \leq N \leq 10^5$, $0 \leq a_i, b_i < N$, the given graph is a tree.

Output

Print the minumum value of K, the number of antennas, when the condition is satisfied.

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

In Sample 1, install an antenna on Vertex 1 and 3. Then, the following five vectors are distinct:

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

In Sample 2 one of posible solutions is to install an antenna on Vertex 0.

In Sample 3 one of possible solutions is to install an antenna on Vertex 0, 4, 9.

Problem G. Rectangles

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

We have a rectangular parallelepiped of size $A \times B \times C$, divided into $1 \times 1 \times 1$ small cubes. The small cubes have coordinates from $(0, 0, 0)$ through $(A - 1, B - 1, C - 1)$.

Let p, q and r be integers. Consider the following set of abc small cubes:

$\{(\ (p + i)\ \mod\ A,\ (q + j)\ \mod\ B,\ (r + k)\ \mod\ C\ ) \mid i,\ j$ and k are integers satisfying $0 \leq i < a$, $0 \leq j < b$, $0 \leq k < c\ \}$

A set of small cubes that can be expressed in the above format using some integers p, q and r, is called a *torus cuboid* of size $a \times b \times c$.

Find the number of the sets of torus cuboids of size $a \times b \times c$ that satisfy the following condition:

- No two torus cuboids in the set have intersection.
- The union of all torus cuboids in the set is the whole rectangular parallelepiped of dimensions $A \times B \times C$.

Since answer may be too big, print it modulo $10^9 + 7$.

Input

Input is given in the following format:

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

Constraints:

$1 \leq a < A \leq 100$, $1 \leq b < B \leq 100$, $1 \leq c < C \leq 100$, all input values are integers.

Output

Print the number of the sets of torus cuboids of size $a \times b \times c$ that satisfy the condition, modulo $10^9 + 7$.

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

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are given a rooted tree with N vertices. The vertices are numbered $0, 1, \ldots, N-1$. The root is vertex 0, and the parent of vertex i ($i = 1, 2, \ldots, N-1$) is Vertex p_i.

Initially, an integer a_i is written in vertex i. Here, $(a_0, a_1, \ldots, a_{N-1})$ is a permutation of $(0, 1, \ldots, N-1)$.

You can execute the following operation at most 25 000 times. The goal is to make the value written in vertex i equal to i.

- Choose a vertex and call it v. Consider the path connecting vertex 0 and v.
- Rotate the values written on the path. That is, For each edge (i, p_i) along the path, replace the value written in vertex p_i with the value written in vertex i (just before this operation), and replace the value of v with the value written in vertex 0 (just before this operation).
- You may choose vertex 0, in which case the operation does nothing.

Input

Input is given in the following format:

N

p_1 p_2 ... p_{N-1}

a_0 a_1 ... a_{N-1}

Constraints:

$2 \leq N \leq 2000$, $0 \leq p_i \leq i-1$, $(a_0, a_1, \ldots, a_{N-1})$ is a permutation of $(0, 1, \ldots, N-1)$.

Output

In the first line, print the number of operations, Q. In the second through $(Q+1)$-th lines, print the chosen vertices in order.

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

In Sample 1, after the first operation, the values written in vertex $0, 1, \ldots, 4$ are $4, 0, 1, 2, 3$.

In Sample 2, after the first operation, the values written in vertex $0, 1, \ldots, 4$ are $3, 1, 0, 2, 4$. After the second operation, the values written in vertex $0, 1, \ldots, 4$ are $1, 0, 2, 3, 4$.

Problem I. ADD, DIV, MAX

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

You are given an integer sequence $a_0, a_1, \ldots, a_{N-1}$.

You have to perform Q queries, each query is one of the following:

- **ADD** $t = 0$, `l r x`: for each i between l and r inclusively, $a_i = a_i + x$.
- **DIV** $t = 1$, `l r x`: for each i between l and r inclusively $a_i = \text{floor}(a_i/x)$, where $\text{floor}(y)$ is the biggest integer that is not greater than y.
- **MAX** $t = 2$. `l r x=0`: print $\max(a_l, a_{l+1}, \ldots, a_r)$.

Input

Input is given in the following format:

N Q

a_0 a_1 ... a_{N-1}

t_1 l_1 r_1 x_1

t_2 l_2 r_2 x_2

...

t_Q l_Q r_Q x_Q

Constraints:

All input values are integers, $1 \le N, Q \le 200\,000$, $0 \le a_i \le 10^8$, $t_i = 0, 1, 2$, $0 \le l_i \le r_i \le N - 1$, $1 \le x_i \le 1000$ if $t_i \ne 2$, $x_i = 0$ if $t_i = 2$.

Output

For each **MAX** query, print $\max(a_l, a_{l+1}, ..., a_r)$.

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

For Sample 1,

$\max(1, 2, 3, 4, 5) = 5$
$1, 2, 3, 4, 5 \to 11, 12, 3, 4, 5$
$\max(11, 12, 3, 4, 5) = 12$
$\max(3) = 3$
$11, 12, 3, 4, 5 \to 2, 3, 3, 4, 5$
$\max(2) = 2$
$\max(3) = 3$

Problem J. Simple APSP Problem

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

You are given an $H \times W$ grid. The square at the top-left corner is indexed $(0, 0)$, and the square at the bottom-right corner is indexed by $(H - 1, W - 1)$.

N squares $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ are painted black, and all other squares are painted white.

Let the shortest distance between white squares A and B be the minimum number of moves required to reach B from A **visiting only white squares**, where one can travel to an adjacent square sharing a side (up, down, left or right) in one move.

Since there are $H \times W - N$ white squares in total, there are $C^2_{H \times W - N}$ ways to choose two of the white squares. For each of these $C^2_{H \times W - N}$ ways, find the shortest distance between the chosen squares, then find the sum of all those distances, modulo $1\,000\,000\,007 = 10^9 + 7$.

Input

Input is given in the following format:

H W

N

x_1 y_1

x_2 y_2

...

x_N y_N

Constraints:

$1 \leq H, W \leq 10^6$, $1 \leq N \leq 30$, $0 \leq x_i \leq H - 1$, $0 \leq y_i \leq W - 1$. If $i \neq j$, then either $x_i \neq x_j$ or $y_i \neq y_j$. It is guaranteed that there is at least one white square. For every pair of white squares A and B, it is possible to reach B from A visiting only white squares.

Output

Print the sum of the shortest distances modulo $10^9 + 7$.

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

In Sample 1, we have the next grid ('.' denotes white square, '!' — black square):

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

We assign alphabet to white squares, like below.

```
ABC
D!E
```

So we get (here $dist(A, B)$ is the shortest distance between A and B):

$dist(A, B) = 1$, $dist(A, C) = 2$, $dist(A, D) = 1$, $dist(A, E) = 3$, $dist(B, C) = 1$, $dist(B, D) = 2$, $dist(B, E) = 2$, $dist(C, D) = 3$, $dist(C, E) = 1$, $dist(D, E) = 4$, and sum of those is 20.

In Sample 2, we assign alphabet to white squares, like below.

```
ABC
DE!
```

So we get:

$dist(A, B) = 1$, $dist(A, C) = 2$, $dist(A, D) = 1$, $dist(A, E) = 2$, $dist(B, C) = 1$, $dist(B, D) = 2$, $dist(B, E) = 1$, $dist(C, D) = 3$, $dist(C, E) = 2$, $dist(D, E) = 1$, and sum of those is 16.

Problem K. Forest Task

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are given a forest with N vertices and M edges. The vertices are numbered 0 through $N-1$. The edges are given in the format (x_i, y_i), which means that vertex x_i and y_i are connected by an edge.

Each vertex i is assigned value a_i. You want to add edges in the given forest so that the forest becomes connected. To add an edge, you choose two different vertices i and j, then span an edge between i and j. This operation costs $a_i + a_j$ dollars, and afterward neither vertex i nor j can be selected again.

Find the minimum total cost required to make the forest connected, or print "`Impossible`" if it is impossible.

Input

Input is given in the following format:

N M

a_0 a_1 ... a_{N-1}

x_1 y_1

...

x_M y_M

Constraints:

$1 \leq N \leq 100\,000$, $0 \leq M \leq N-1$, $1 \leq a_i \leq 10^9$, $0 \leq x_i, y_i \leq N-1$. The given graph is a forest. All input values are integers.

Output

Print the minimum total cost required to make the forest connected, or print "`Impossible`" if it is 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

In Sample 1, if we connect vertices 0 and 5, the graph become connected, and the cost is $1 + 6 = 7$.

In Sample 2, we can't make the graph connected.

In Sample 3, the graph is connected, regardless of whether we do something or not.