A. k-Multiple Free Set

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A k-multiple free set is a set of integers where there is no pair of integers where one is equal to another integer multiplied by k. That is, there are no two integers x and y $(x < y)$ from the set, such that $y = x \cdot k$.

You're given a set of n distinct positive integers. Your task is to find the size of it's largest k-multiple free subset.

Input

The first line of the input contains two integers n and k $(1 \le n \le 10^5, 1 \le k \le 10^9)$. The next line contains a list of n distinct positive integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le 10^9)$.

All the numbers in the lines are separated by single spaces.

Output

On the only line of the output print the size of the largest k-multiple free subset of $\{a_1, a_2, ..., a_n\}$.

Sample test(s)

input

```
6 2
2 3 6 5 4 10
```

output

```
3
```

Note

In the sample input one of the possible maximum 2-multiple free subsets is {4, 5, 6}.

B. Zero Tree

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A *tree* is a graph with n vertices and exactly $n - 1$ edges; this graph should meet the following condition: there exists exactly one shortest (by number of edges) path between any pair of its vertices.

A *subtree* of a tree T is a tree with both vertices and edges as subsets of vertices and edges of T.

You're given a tree with n vertices. Consider its vertices numbered with integers from 1 to n. Additionally an integer is written on every vertex of this tree. Initially the integer written on the i-th vertex is equal to v_i. In one move you can apply the following operation:

1. Select the subtree of the given tree that includes the vertex with number 1.
2. Increase (or decrease) by one all the integers which are written on the vertices of that subtree.

Calculate the minimum number of moves that is required to make all the integers written on the vertices of the given tree equal to zero.

Input

The first line of the input contains n $(1 \le n \le 10^5)$. Each of the next $n - 1$ lines contains two integers a_i and b_i $(1 \le a_i, b_i \le n; a_i \ne b_i)$ indicating there's an edge between vertices a_i and b_i. It's guaranteed that the input graph is a tree.

The last line of the input contains a list of n space-separated integers $v_1, v_2, ..., v_n$ $(|v_i| \le 10^9)$.

Output

Print the minimum number of operations needed to solve the task.

Please, do not write the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Sample test(s)

input

```
3
1 2
1 3
1 -1 1
```

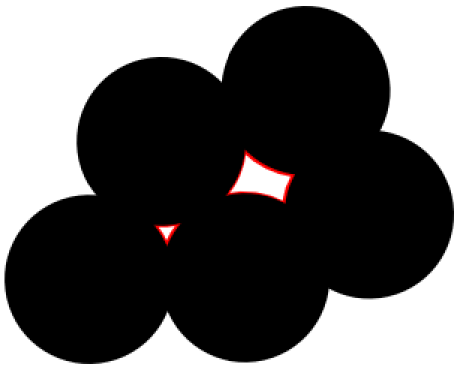
output

```
3
```

C. The Last Hole!

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Luyi has n circles on the plane. The i-th circle is centered at (x_i, y_i). At the time zero circles start to grow simultaneously. In other words, the radius of each circle at time t $(t > 0)$ is equal to t. The circles are drawn as black discs on an infinite white plane. So at each moment the plane consists of several black and white regions. Note that the circles may overlap while growing.



We define a *hole* as a closed, connected white region. For instance, the figure contains two holes shown by red border. During growing some holes may be created and it is easy to see that each created hole will disappear eventually. Luyi asks you to find moment of time such that the last hole disappears. In other words, you should find the first moment such that no hole can be seen after that.

Input

The first line of the input contains integer n $(1 \le n \le 100)$. Each of the next n lines contains two integers x_i and y_i $(-10^4 \le x_i, y_i \le 10^4)$, indicating the location of i-th circle.

It's guaranteed that no two circles are centered at the same point.

Output

Print the moment where the last hole disappears. If there exists no moment in which we can find holes print `-1`.

The answer will be considered correct if the absolute or relative error does not exceed 10^{-4}.

Sample test(s)

input

```
3
0 0
1 1
2 2
```

output

```
-1
```

input

```
4
0 0
0 2
2 2
2 0
```

output

```
1.414214
```

input

```
4
0 1
0 -1
-2 0
4 0
```

output

```
2.125000
```

D. Lovely Matrix

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Lenny had an $n \times m$ matrix of positive integers. He loved the matrix so much, because each row of the matrix was sorted in non-decreasing order. For the same reason he calls such matrices of integers *lovely*.

One day when Lenny was at school his little brother was playing with Lenny's matrix in his room. He erased some of the entries of the matrix and changed the order of some of its columns. When Lenny got back home he was very upset. Now Lenny wants to recover his matrix.

Help him to find an order for the columns of the matrix so that it's possible to fill in the erased entries of the matrix to achieve a lovely matrix again. Note, that you can fill the erased entries of the matrix with any integers.

Input

The first line of the input contains two positive integers n and m $(1 \le n \cdot m \le 10^5)$. Each of the next n lines contains m space-separated integers representing the matrix. An integer -1 shows an erased entry of the matrix. All other integers (each of them is between 0 and 10^9 inclusive) represent filled entries.

Output

If there exists no possible reordering of the columns print -1. Otherwise the output should contain m integers $p_1, p_2, \dots, p_m$ showing the sought permutation of columns. So, the first column of the lovely matrix will be p_1-th column of the initial matrix, the second column of the lovely matrix will be p_2-th column of the initial matrix and so on.

Sample test(s)

input

```
3 3
1 -1 -1
1 2 1
2 -1 1
```

output

```
3 1 2
```

input

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

output

```
1 3 2
```

input

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

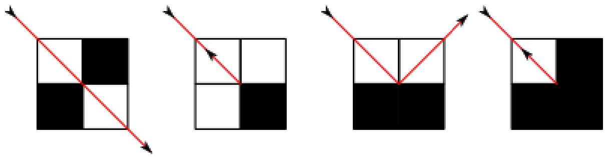
output

```
-1
```

E. Mirror Room

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Imagine an $n \times m$ grid with some blocked cells. The top left cell in the grid has coordinates $(1, 1)$ and the bottom right cell has coordinates (n, m). There are k blocked cells in the grid and others are empty. You flash a laser beam from the center of an empty cell (x_s, y_s) in one of the diagonal directions (i.e. north-east, north-west, south-east or south-west). If the beam hits a blocked cell or the border of the grid it will reflect. The behavior of the beam reflection in different situations is depicted in the figure below.



After a while the beam enters an infinite cycle. Count the number of empty cells that the beam goes through at least once. We consider that the beam goes through cell if it goes through its center.

Input

The first line of the input contains three integers n, m and k $(1 \le n, m \le 10^5, 0 \le k \le 10^5)$. Each of the next k lines contains two integers x_i and y_i $(1 \le x_i \le n, 1 \le y_i \le m)$ indicating the position of the i-th blocked cell.

The last line contains x_s, y_s $(1 \le x_s \le n, 1 \le y_s \le m)$ and the flash direction which is equal to "NE", "NW", "SE" or "SW". These strings denote directions $(-1, 1)$, $(-1, -1)$, $(1, 1)$, $(1, -1)$.

It's guaranteed that no two blocked cells have the same coordinates.

Output

In the only line of the output print the number of empty cells that the beam goes through at least once.

Please, do not write the %lld specifier to read or write 64-bit integers in C++. It is preferred to use the cin, cout streams or the %I64d specifier.

Sample test(s)

input

```
3 3 0
1 2 SW
```

output

```
6
```

input

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

output

```
14
```