Problem A
Rearranging a Sequence

Input: Standard Input
Time Limit: 2 seconds

You are given an ordered sequence of integers, $(1, 2, 3, \ldots, n)$. Then, a number of requests will be given. Each request specifies an integer in the sequence. You need to move the specified integer to the head of the sequence, leaving the order of the rest untouched. Your task is to find the order of the elements in the sequence after following all the requests successively.

Input

The input consists of a single test case of the following form.

$$
\begin{aligned}
&n\ m \\
&e_1 \\
&\vdots \\
&e_m
\end{aligned}
$$

The integer n is the length of the sequence ($1 \leq n \leq 200000$). The integer m is the number of requests ($1 \leq m \leq 100000$). The following m lines are the requests, namely $e_1, \ldots, e_m$, one per line. Each request e_i ($1 \leq i \leq m$) is an integer between 1 and n, inclusive, designating the element to move. Note that, the integers designate the integers themselves to move, not their positions in the sequence.

Output

Output the sequence after processing all the requests. Its elements are to be output, one per line, in the order in the sequence.

Sample Input 1	**Sample Output 1**
5 3 4 2 5	5 2 4 1 3

Sample Input 2	**Sample Output 2**
10 8 1 4 7 3 4 10 1 3	3 1 10 4 7 2 5 6 8 9

In Sample Input 1, the initial sequence is $(1, 2, 3, 4, 5)$. The first request is to move the integer 4 to the head, that is, to change the sequence to $(4, 1, 2, 3, 5)$. The next request to move the integer 2 to the head makes the sequence $(2, 4, 1, 3, 5)$. Finally, 5 is moved to the head, resulting in $(5, 2, 4, 1, 3)$.

Problem B
Distribution Center

Input: Standard Input
Time Limit: 3 seconds

The factory of the Impractically Complicated Products Corporation has many manufacturing lines and the same number of corresponding storage rooms. The same number of conveyor lanes are laid out in parallel to transfer goods from manufacturing lines directly to the corresponding storage rooms. Now, they plan to install a number of robot arms here and there between pairs of adjacent conveyor lanes so that goods in one of the lanes can be picked up and released down on the other, and also in the opposite way. This should allow mixing up goods from different manufacturing lines to the storage rooms.

Depending on the positions of robot arms, the goods from each of the manufacturing lines can only be delivered to some of the storage rooms. Your task is to find the number of manufacturing lines from which goods can be transferred to each of the storage rooms, given the number of conveyor lanes and positions of robot arms.

Input

The input consists of a single test case, formatted as follows.

$$
\begin{aligned}
&n\ m \\
&x_1\ y_1 \\
&\vdots \\
&x_m\ y_m
\end{aligned}
$$

An integer n ($2 \leq n \leq 200000$) in the first line is the number of conveyor lanes. The lanes are numbered from 1 to n, and two lanes with their numbers differing with 1 are adjacent. All of them start from the position $x = 0$ and end at $x = 100000$. The other integer m ($1 \leq m < 100000$) is the number of robot arms.

The following m lines indicate the positions of the robot arms by two integers x_i ($0 < x_i < 100000$) and y_i ($1 \leq y_i < n$). Here, x_i is the x-coordinate of the i-th robot arm, which can pick goods on either the lane y_i or the lane $y_i + 1$ at position $x = x_i$, and then release them on the other at the same x-coordinate.

You can assume that positions of no two robot arms have the same x-coordinate, that is, $x_i \neq x_j$ for any $i \neq j$.

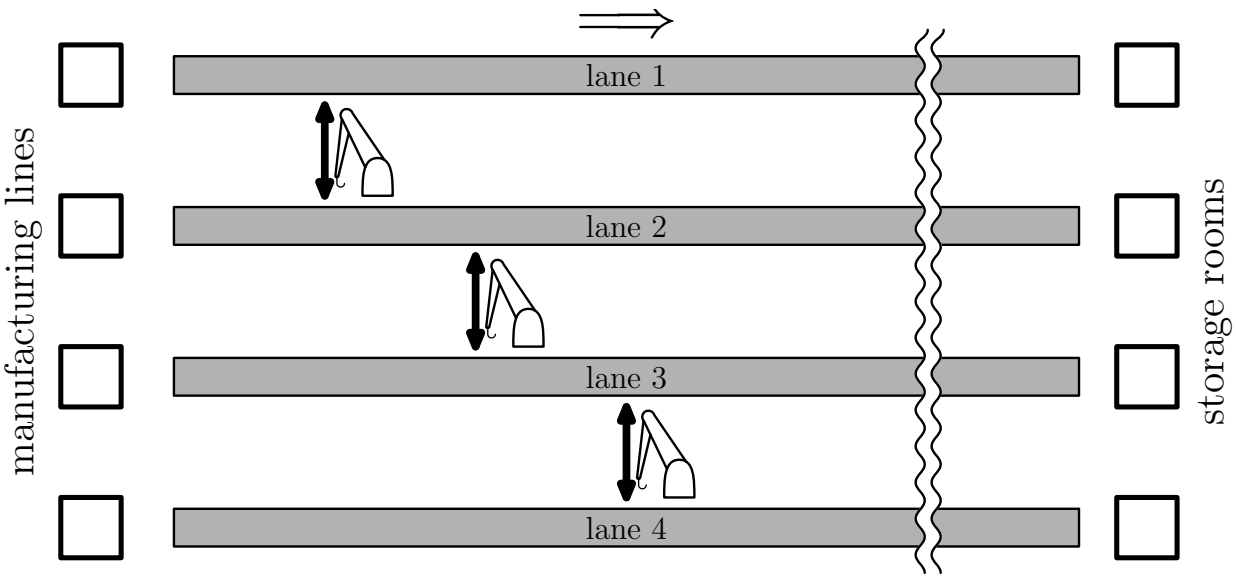

Figure C.1. Illustration of Sample Input 1

Output

Output n integers separated by a space in one line. The i-th integer is the number of the manufacturing lines from which the storage room connected to the conveyor lane i can accept goods.

Sample Input 1	**Sample Output 1**
4 3 1000 1 2000 2 3000 3	2 3 4 4

Sample Input 2	**Sample Output 2**
4 3 1 1 3 2 2 3	2 4 4 2

Problem C
Hidden Anagrams

Input: Standard Input
Time Limit: 10 seconds

An *anagram* is a word or a phrase that is formed by rearranging the letters of another. For instance, by rearranging the letters of "William Shakespeare," we can have its anagrams "I am a weakish speller," "I'll make a wise phrase," and so on. Note that when A is an anagram of B, B is an anagram of A.

In the above examples, differences in letter cases are ignored, and word spaces and punctuation symbols are freely inserted and/or removed. These rules are common but not applied here; only exact matching of the letters is considered.

For two strings s_1 and s_2 of letters, if a substring s'_1 of s_1 is an anagram of a substring s'_2 of s_2, we call s'_1 a *hidden anagram* of the two strings, s_1 and s_2. Of course, s'_2 is also a *hidden anagram* of them.

Your task is to write a program that, for given two strings, computes the length of the longest hidden anagrams of them.

Suppose, for instance, that "anagram" and "grandmother" are given. Their substrings "nagr" and "gran" are hidden anagrams since by moving letters you can have one from the other. They are the longest since any substrings of "grandmother" of lengths five or more must contain "d" or "o" that "anagram" does not. In this case, therefore, the length of the longest hidden anagrams is four. Note that a substring must be a sequence of letters occurring *consecutively* in the original string and so "nagrm" and "granm" are not hidden anagrams.

Input

The input consists of a single test case in two lines.

s_1
s_2

s_1 and s_2 are strings consisting of lowercase letters (`a` through `z`) and their lengths are between 1 and 4000, inclusive.

Output

Output the length of the longest hidden anagrams of s_1 and s_2. If there are no hidden anagrams, print a zero.

Sample Input 1	Sample Output 1
anagram grandmother	4

Sample Input 2	Sample Output 2
williamshakespeare iamaweakishspeller	18

Sample Input 3	Sample Output 3
aaaaaaaabbbbbbbb xxxxxababababxxxxxabab	6

Sample Input 4	Sample Output 4
abababacdcdcd efefefghghghghgh	0

Problem D
Placing Medals on a Binary Tree

Input: Standard Input
Time Limit: 4 seconds

You have drawn a chart of a perfect binary tree, like one shown in Figure G.1. The figure shows a finite tree, but, if needed, you can add more nodes beneath the leaves, making the tree arbitrarily deeper.

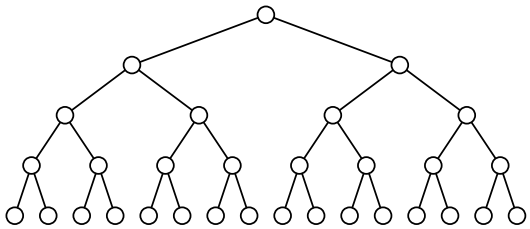

Figure G.1. A Perfect Binary Tree Chart

Tree nodes are associated with their depths, defined recursively. The root has the depth of zero, and the child nodes of a node of depth d have their depths $d + 1$.

You also have a pile of a certain number of medals, each engraved with some number. You want to know whether the medals can be placed on the tree chart satisfying the following conditions.

- A medal engraved with d should be on a node of depth d.
- One tree node can accommodate at most one medal.
- The path to the root from a node with a medal should not pass through another node with a medal.

You have to place medals satisfying the above conditions, one by one, starting from the top of the pile down to its bottom. If there exists no placement of a medal satisfying the conditions, you have to throw it away and simply proceed to the next medal.

You may have choices to place medals on different nodes. You want to find the best placement. When there are two or more placements satisfying the rule, one that places a medal upper in the pile is better. For example, when there are two placements of four medal, one that places only the top and the 2nd medal, and the other that places the top, the 3rd, and the 4th medal, the former is better.

In Sample Input 1, you have a pile of six medals engraved with 2, 3, 1, 1, 4, and 2 again respectively, from top to bottom.

- The first medal engraved with 2 can be placed, as shown in Figure G.2 (A).
- Then the second medal engraved with 3 may be placed , as shown in Figure G.2 (B).
- The third medal engraved with 1 cannot be placed if the second medal were placed as stated above, because both of the two nodes of depth 1 are along the path to the root from nodes already with a medal. Replacing the second medal satisfying the placement conditions, however, enables a placement shown in Figure G.2 (C).
- The fourth medal, again engraved with 1, cannot be placed with any replacements of the three medals already placed satisfying the conditions. This medal is thus thrown away.
- The fifth medal engraved with 4 can be placed as shown in of Figure G.2 (D).
- The last medal engraved with 2 cannot be placed on any of the nodes with whatever replacements.

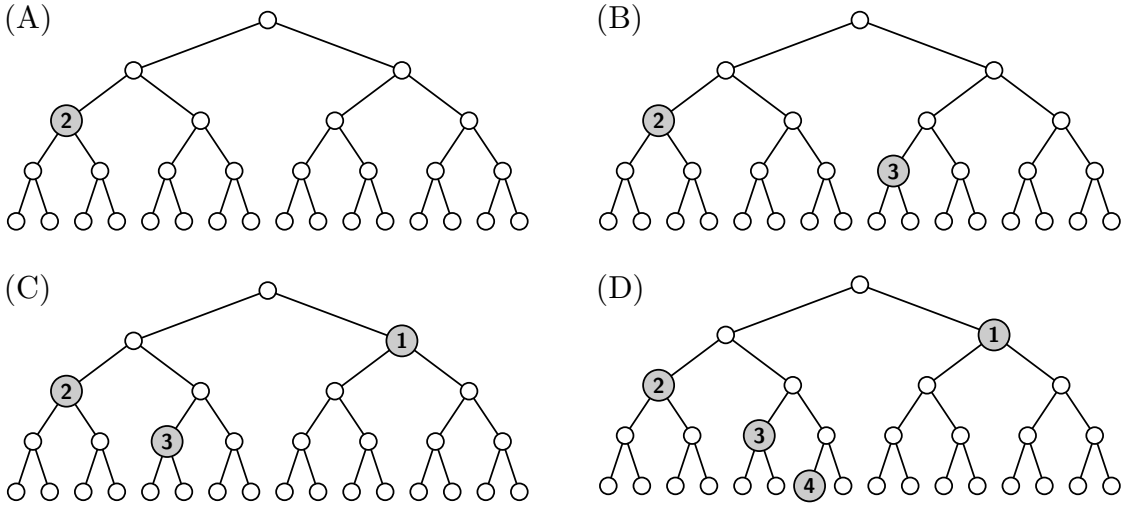

Figure G.2. Medal Placements

Input

The input consists of a single test case in the format below.

$$
\begin{aligned}
&n\\
&x_1\\
&\vdots\\
&x_n
\end{aligned}
$$

The first line has n, an integer representing the number of medals ($1 \leq n \leq 5 \times 10^5$). The following n lines represent the positive integers engraved on the medals. The i-th line of which has an integer x_i ($1 \leq x_i \leq 10^9$) engraved on the i-th medal of the pile from the top.

Output

When the best placement is chosen, for each i from 1 through n, output `Yes` in a line if the i-th medal is placed; otherwise, output `No` in a line.

Sample Input 1

```
6
2
3
1
1
4
2
```

Sample Output 1

```
Yes
Yes
Yes
No
Yes
No
```

Sample Input 2

```
10
4
4
4
4
4
4
4
4
1
4
```

Sample Output 2

```
Yes
Yes
Yes
Yes
Yes
Yes
Yes
Yes
Yes
No
```

Problem E
Animal Companion in Maze

Input: Standard Input
Time Limit: 2 seconds

George, your pet monkey, has escaped, slipping the leash!

George is hopping around in a maze-like building with many rooms. The doors of the rooms, if any, lead directly to an adjacent room, not through corridors. Some of the doors, however, are one-way: they can be opened only from one of their two sides.

He repeats randomly picking a door he can open and moving to the room through it. You are chasing him but he is so quick that you cannot catch him easily. He never returns immediately to the room he just has come from through the same door, believing that you are behind him. If any other doors lead to the room he has just left, however, he may pick that door and go back.

If he cannot open any doors except one through which he came from, voilà, you can catch him there eventually.

You know how rooms of the building are connected with doors, but you don't know in which room George currently is.

It takes one unit of time for George to move to an adjacent room through a door.

Write a program that computes how long it may take before George will be confined in a room. You have to find the longest time, considering all the possibilities of the room George is in initially, and all the possibilities of his choices of doors to go through.

Note that, depending on the room organization, George may have possibilities to continue hopping around forever without being caught.

Doors may be on the ceilings or the floors of rooms; the connection of the rooms may not be drawn as a planar graph.

Input

The input consists of a single test case, in the following format.

$$
\begin{array}{l}
n\ m \\
x_1\ y_1\ w_1 \\
\vdots \\
x_m\ y_m\ w_m
\end{array}
$$

The first line contains two integers n ($2 \leq n \leq 100000$) and m ($1 \leq m \leq 100000$), the number of rooms and doors, respectively. Next m lines contain the information of doors. The i-th line of these contains three integers x_i, y_i and w_i ($1 \leq x_i \leq n, 1 \leq y_i \leq n$, $x_i \neq y_i$, $w_i = 1$ or 2), meaning that the i-th door connects two rooms numbered x_i and y_i, and it is one-way from x_i to y_i if $w_i = 1$, two-way if $w_i = 2$.

Output

Output the maximum number of time units after which George will be confined in a room. If George has possibilities to continue hopping around forever, output "`Infinite`".

Sample Input 1

```
2 1
1 2 2
```

Sample Output 1

```
1
```

Sample Input 2

```
2 2
1 2 1
2 1 1
```

Sample Output 2

```
Infinite
```

Sample Input 3

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

Sample Output 3

```
4
```

Sample Input 4

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

Sample Output 4

```
1
```

Problem F
Black and White Boxes

Input: Standard Input
Time Limit: 2 seconds

Alice and Bob play the following game.

1. There are a number of straight piles of boxes. The boxes have the same size and are painted either black or white.
2. Two players, namely Alice and Bob, take their turns alternately. Who to play first is decided by a fair random draw.
3. In Alice's turn, she selects a black box in one of the piles, and removes the box together with all the boxes above it, if any. If no black box to remove is left, she loses the game.
4. In Bob's turn, he selects a white box in one of the piles, and removes the box together with all the boxes above it, if any. If no white box to remove is left, he loses the game.

Given an initial configuration of piles and who plays first, the game is a *definite perfect information game.* In such a game, one of the players has sure win provided he or she plays best. The draw for the first player, thus, essentially decides the winner.

In fact, this seemingly boring property is common with many popular games, such as chess. The chess game, however, is complicated enough to prevent thorough analyses, even by supercomputers, which leaves us rooms to enjoy playing.

This game of box piles, however, is not as complicated. The best plays may be more easily found. Thus, initial configurations should be fair, that is, giving both players chances to win. A configuration in which one player can always win, regardless of who plays first, is undesirable.

You are asked to arrange an initial configuration for this game by picking a number of piles from the given candidate set. As more complicated configuration makes the game more enjoyable, you are expected to find the configuration with the maximum number of boxes among fair ones.

Input

The input consists of a single test case, formatted as follows.

$$
\begin{aligned}
&n \\
&p_1 \\
&\vdots \\
&p_n
\end{aligned}
$$

A positive integer n (≤ 40) is the number of candidate piles. Each p_i is a string of characters `B` and `W`, representing the i-th candidate pile. `B` and `W` mean black and white boxes, respectively. They appear in the order in the pile, from bottom to top. The number of boxes in a candidate pile does not exceed 40.

Output

Output in a line the maximum possible number of boxes in a fair initial configuration consisting of some of the candidate piles. If only the empty configuration is fair, output a zero.

Sample Input 1

```
4
B
W
WB
WB
```

Sample Output 1

```
5
```

Sample Input 2

```
6
B
W
WB
WB
BWW
BWW
```

Sample Output 2

```
10
```