Problem A: Double Chomp

Filename: `chomp2`
Time limit: `2 seconds`

Nom Chomsky the chain chomp regularly sneaks bites of food from Bowser's dinner table. All the food on the table is arranged on a line and Chomsky is able to eat any contiguous section in a single bite, but he can only manage up to two bites before being caught. Chomsky knows how tasty each dish is and wishes to maximize the combined tastiness of his two bites. However he must be careful because some dishes taste so bad that they have a negative tasty value! Find the maximum tastiness Chomsky can obtain with two or fewer bites.

Input

The first line of input contains a single positive integer, ***n*** (***n*** $\leq 10^5$), representing the number of food items on the table. The second line contains ***n*** space-separated positive integers. The ith integer, $\boldsymbol{t_i}$ ($-1000 \leq t_i \leq 1000$), on this line represents the tastiness of the ith food item from the left on the table.

Output

For each test case output the maximum sum of tastiness Nom Chomsky can enjoy if he takes two or fewer bits.

Samples

Input	Output

`1`
`-1`
	`0`

`2`
`1 -2`
	`1`

`11`
`1 -9 5 -1 2 -3 1 -4 3 -4 1`
	`9`

Solution for the third sample test case with bites bolded: `1 -9` **`5 -1 2`** `-3 1 -4` **`3`** `-4 1`

Problem B: Geography Assignment

Filename: geography

Time limit: `3 seconds`

Arnold is a student at the University of Carpathian Flamingoes. He is taking a geography class, and it's really hard. His teacher has asked him to determine which cities within a county can reach each other. However, some counties can have lots of cities, so Arnold wants you to write a program to help him find the answer. Also, there's a quirk of the country that the University is in: the roads only go in one direction!

Input

The first line of the input will contain two integers, ***n*** ($2 \le \mathbf{n} \le 500$) and ***m*** ($1 \le \boldsymbol{m} \le 1000$), the number of cities in the county and the number of roads in the county, respectively.

The next m lines will each describe a road.

Each line of these lines will contain two integers, ***a*** and ***b*** ($1 \le \boldsymbol{a}, \boldsymbol{b} \le \boldsymbol{n}, \boldsymbol{a} \ne \boldsymbol{b}$) indicating that there is a road from city a to city b. There may or may not be a road from city b to city a. Each road listed will be unique.

Output

Output the number of pairs of cities (a, b), with a ≠ b, such that it's possible to travel from city a to city b, directly or indirectly. Note that if it's possible to travel from city a to city b AND it's possible to travel from city b to city a, then these should add 2 to the count.

Samples

Input	Output

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

Explanation of Sample Input 1:

City 1 can visit cities 2 and 3. City 3 can visit city 2. City 4 can visit city 5. City 5 can visit city 4.

Problem C: Grid

Filename: `grid`
Time limit: `2 seconds`

You are on an ***n***x***m*** grid where each square on the grid has a digit on it. From a given square that has digit ***k*** on it, a move consists of jumping exactly ***k*** squares in one of the four cardinal directions. A move cannot go beyond the edges of the grid; it does not wrap. What is the minimum number of moves required to get from the top-left corner to the bottom-right corner?

Input

The first line of input contains two space-separated integers ***n*** and ***m*** (1 ≤ ***n***, ***m*** ≤ 500), indicating the size of the grid. It is guaranteed that at least one of ***n*** and ***m*** is greater than 1.

The next ***n*** lines will each consist of ***m*** digits, with no spaces, indicating the ***n***x***m*** grid. Each digit is between 0 and 9, inclusive.

The top-left corner of the grid will be the square corresponding to the first character in the first line of the test case. The bottom-right corner of the grid will be the square corresponding to the last character in the last line of the test case.

Output

Output a single integer on a line by itself representing the minimum number of moves required to get from the top-left corner of the grid to the bottom-right. If it isn't possible, output -1.

Samples

Input	Output

`2 2`
`11`
`11`
	`2`

`2 2`
`22`
`22`
	`-1`

`5 4`
`2120`
`1203`
`3113`
`1120`
	`6`

Problem D: Bodies of Water

Filename: `water`

Time limit: `1 second`

For those who don't like regular images, ASCII Maps Inc. has created maps that are fully printable ASCII characters. Each map is a rectangular grid of lowercase English letters, where each letter stands for various locations. In particular, 'w' stands for water and the other 25 letters represent various different land locations. For this problem, we are interested in counting the number of bodies of water on a given ASCII map. A body of water is a maximal set of contiguous grid squares on the ASCII map where each square in the body of water shares a boundary with at least one other square in the body of water. Thus, for two grid squares to be part of the same body of water, one must be above, below, to the left, or to the right of the other grid square.

Input

The first line of input consists of two space separated integers, ***r*** ($1 \le r \le 50$) and ***c*** ($1 \le c \le 50$), the number of rows and columns, respectively for the input map. The next ***r*** lines will each contain ***c*** lowercase English letters, representing the corresponding row of the input map.

Output

On a line by itself, output the number of bodies of water in the input map.

Samples

Input	Output

`5 6`
`waaaww`
`wawawc`
`bbbbwc`
`wwwwww`
`dddddd`
	`3`

`2 8`
`wxwxwxwx`
`xwxwxwxw`
	`8`

Problem E: Arch Enemies

Filename: `arch`
Time limit: `2 seconds`

Arup and Matt are both instructors for this camp. What you may not know is they are secretly ***arch enemies***^{TM}**!** That's right. In order to make this problem work, they have decided to foster a not-so-secret-anymore relationship of loathing, bitter detestation, empirically verifiable hatred, and mutual respect.

Given that they are now arch enemies, they need to build their secret labs for doing evil computer scientist things as far away from each other as possible. You will be given a layout of a sub city of Orlando with key points of interest (locations) connected by roads they are willing to travel down using their mathematics mobiles. (Arup's Accord of Mathematical Destruction has a license plate of EULER. Matt's DNE Machine is ironic.) Your task is to find two locations for their evil lairs that maximize the distance between the locations when taking the shortest possible driving route.

Input

The first line contains a single integer n ($3 \le n \le 10^5$), representing both the number of locations and roads.

The next n lines contains three integers a_i, b_i, and d_i ($1 \le a_i \neq b_i \le n$, $1 \le d_i \le 10^9$), representing a road between locations a_i and b_i of length d_i.

It will always be possible to reach any location from any other location in the layout.

Output

Output a single integer, the maximum distance the secret lairs can be from each other.

Samples

Input	Output

6
1 2 4
2 3 2
2 4 5
4 5 4
5 6 2
4 6 3
	13

6
1 2 800
2 3 30
3 4 4
2 5 21
5 4 8
4 6 900
	1729

Problem F: Cactus Customs

Filename: `customs`
Time limit: `1 second`

Gennady loves to travel; one might even call him a professional tourist. On most of his trips he will bring graphs with him to play with. Unfortunately for him, some of his graphs are cacti. He is worried that US customs will classify his cacti as a prohibited agricultural item and prevent him from bringing them into the country.

So he needs to identify which of his graphs are cacti. Sometimes the US will allow cacti into the country if they meet certain spanning tree limitations. Namely, the US requires that the number of spanning trees that can be formed from the given cactus does not exceed a certain amount. Therefore, he would like you to check if a given graph in his collection is a cactus and, if so, report the number of spanning trees of the cactus.

A cactus is a connected graph where each edge is involved in at most one cycle. A spanning tree is a subgraph of a given graph that is both a tree and every vertex of the original graph is included in the tree. Two spanning trees are considered different if they differ in edges they include.

Input

The first line contains two integers $\boldsymbol{n}$ and $\boldsymbol{m}$ ($1 \le \boldsymbol{n} \le 2 \cdot 10^4$, $1 \le \boldsymbol{m} \le 10^5$), the number of nodes and number of edges in the graph, respectively.

The next $\boldsymbol{m}$ lines contains two integers $\boldsymbol{a_i}$ and $\boldsymbol{b_i}$ ($1 \le \boldsymbol{a_i} \ne \boldsymbol{b_i} \le \boldsymbol{n}$), representing an edge between nodes $\boldsymbol{a_i}$ and $\boldsymbol{b_i}$.

Output

If the graph given is not a cactus, output "`safe`" without quotes. Otherwise output a single integer, the number of spanning trees of the given cactus. As this number can be quite large, output the result modulo 10^9+7.

Samples

Input	Output

14 15
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 3
2 9
9 10
10 11
11 12
12 13
13 10
2 14
	24

10 11
1 2
2 3
3 4
4 5
5 6
6 1
3 7
7 8
8 9
9 10
10 2
	safe

3 1 1 3	safe

Problem G: Balanced Cycles

Filename: `cycles`
Time limit: `2 seconds`

The wonderful thing about cycles is you always get back to where you started and things come full circle. The wonderful thing about trees is that when any edge is added to the graph, a cycle is formed. Edward would like to combine these wonderful things.

You are given some cities and roads between them. The layout of the city network forms a tree. That is, it is always possible to reach some city from some other through a sequence of roads, but there is always one unique simple path between any two cities. Each road has a color, either red or blue. You would like to add a single road to the network (also either red or blue) and create a balanced cycle (one where the number of red edges equals the number of blue edges). When adding a road you must ensure the road does not already exist between the two cities. Your task is to count the number of ways you select a pair of different cities and create balanced cycles.

Input

The first line contains one integer n ($1 \leq n \leq 10^5$), the number of cities.

The next $n-1$ lines contains two integers a_i and b_i ($1 \leq a_i \neq b_i \leq n$) and a letter c_i ($c_i \in \{r, b\}$), representing a road between cities a_i and b_i with color c_i. The letter 'r' represents a red road and the letter 'b' represents a blue road.

Output

Output the number of balanced cycles you can form.

Samples

Input	Output
6 1 2 r 2 3 b 2 4 r 4 6 b 4 5 b	4
7 1 2 r 2 3 b 3 4 r 4 5 b 5 6 r 6 7 b	6

Problem H: Colony Tunnel Upgrade

Filename: `upgrade`

Timelimit: `3 second`

Spapyur bought a 2-dimensional ant colony a while back. Over the course of several months, the ants of Spapyur's colony started digging a single hole, indexed 1, and proceeded to tunnel downwards to form ***n***-1 different junctions, indexed between 2 and ***n***. These ants only ever dig downwards to form another junction, and from any junction (other than the hole at the top), another junction is only ever built one centimeter below it in the vertical direction. Also, no two tunnels ever lead downwards to the same junction.

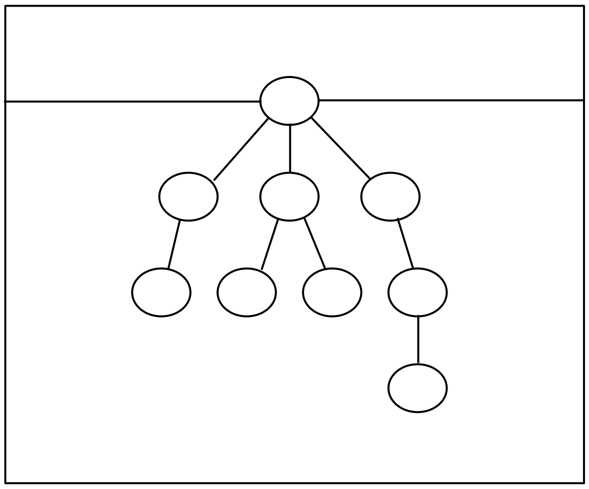

Figure 1: The ant colony from the second sample case.

Now, the leaders of the ant colony want to build horizontal tunnels connecting junctions of the same height. Since the ant colony is 2-dimensional, an ant digging from one junction horizontally to another will either dig into another junction or dig into the wall. The ants do not want to dig into the wall, and since the digging substrate is opaque, they ask Spapyur for help. Spapyur doesn't want to deal with this problem, so he asks you for help.

Input

The first line of input contains a single integer ***n*** ($2 \le n \le 10^5$), denoting the number of junctions, where the hole at the top counts as junction number 1. A line containing ***n***-1 space-separated integers, d_i ($1 \le i \le n-1$) follows. The i^{th} of these integers, d_i, specifies that junction d_i connects downwards to junction *i*+1. It is guaranteed for each junction that any junctions immediately below it will be ordered, from left to right, in increasing order of their indices.

Output

Output two lines. On the first line, output ***n*** space-separated integers, where the *i*th integer denotes the index of the junction immediately to the left of junction *i*, or -1 if such a junction does not exist. On the second line, output ***n*** space-separated integers, where the *i*th integer denotes the index of the junction immediately to the right of junction *i*, or -1 if such a junction does not exist.

Samples

Input	Output

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

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