DSA - Sec B Lab 5

A. Acquire The Kingdoms

1 second, 256 megabytes

NOTE: You are allowed to use only stdlib.h and stdio.h . Your submission will not be graded if there is use of any other library.

King John has decided to acquire the nearby kingdoms. He has gotten along with his troops to accomplish this mission. There are total of N cities. The path to each kingdom goes from city 1. The kingdoms are in the last city of each path from city 1. Along each path there are some cities where soldiers have been deployed to protect the kingdoms. So if i_{th} city has soldiers deployed, then it'll be denoted by $S_i = 1$, otherwise $S_i = 0$. Note that there can be soldiers deployed in the cities containing kingdoms. Unfortunately the King's troops have power to defend soldiers in at most K consecutive cities along the path. Your task is to tell the number of kingdoms that the King can conquer.

Constraints:

- $2 \leq N \leq 100000$
- $1 \leq K \leq N$

Input

The first line contains two spaced integers N and K where N is the number of cities. The second line contains N spaced integers where $S_i = 1$ or $S_i = 0$, denoting whether soldiers are deployed in the i_{th} city or not. Each of the following $N - 1$ lines contains 2 spaced integers u and v ($u \neq v$). denoting that city u and v are connected to each other.

NOTE: it is guaranteed that there is no cyclic paths.

Output

Print the number of kingdoms that the King can conquer.

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

Here we have 5 cities. The path to each city goes via the city 1. There are soldiers in city 1, 2, 4, 5. We start from city 1. The kingdom is present in city 5 only as there is no city that we can travel from city 5 further once we go from 1 -> 2 -> -> 3 -> 4 -> 5. In this path we have maximum of 2 consecutive cities that have soldiers deployed. Hence only 1 kingdom will be acquired.

B. Kone in Egypt

1 second, 256 megabytes

After exploring all the art galleries, Kone has now set off on an expedition to Egypt in order to find the pharaoh's hidden treasure. He is navigating inside the Scooby-Doo pyramid and after hours of hard work, finds the hidden treasure. But alas, as soon as he touches the treasure chest, a series of mummies come to life and run to capture Kone. It is upto you to help Kone escape the pyramid alive, or report that there's no way out and he too, shall become a mummy. The pyramid is represented as a 2D matrix. Each cell can be of the following type -

- '.' implies that Kone can freely move through this cell
- '#' implies that the cell is blocked
- '*M*' implies that a mummy is present in this cell (there can be multiple mummies)
- '*A*' implies Kone's initial location

Note that both Kone and mummies can move in following directions - up (U), down (D), left (L) and right (R). Kone can safely escape the pyramid if he reaches any free boundary cell without ever sharing a cell with any mummy. All the mummies are free to move in any way possible (in the available 4 directions).

Your task is to find out if it is possible for Kone to escape, even if the mummies chase directly after Kone. In other words, find if it is possible for Kone to escape if all the mummies move optimally.

If yes, print "YES" in the first line, followed by the number of moves in the second line, followed by the sequence of moves (in terms of U, D, L, R as a single string) in the third line (see example for more clarity).

If no, and Kone has to become a mummy, just print "NO"

Input

- the first line contains two integers m and n, the height and width of the pyramid ($1 \leq m, n \leq 1000$)
- then there are m lines with a string of length n in each line indicating the layout of the pyramid.

input
4 4 #.#M #.#. #A.. ####
output
YES 2 UU

input
4 4 #..M #.#. #A.. ####
output
NO

In test case-1, Kone can move up two times and escape the only mummy. In test case-2, the mummy can catch up to Kone whether he moves up or right.

C. Connective Cities

2 seconds, 1024 megabytes

NOTE: You are allowed to use only stdlib.h and stdio.h . Your submission will not be graded if there is use of any other library.

There are N cities which are connected in the form an **n-ary** tree. The city at the root of the tree is numbered as city 1. You will be given the list of cities which are interconnected to each other. Say **s-t** is the shortest path between city s and city t. We define the connectivity of the city as the distance of the shortest path be $D_(s-t) = d$. You will be given T test-cases where in each test-case contains 2 integers V and D. You need to compute the number of cities having connectivity D with the city 1 such that V lies in the shortest path from that particular city to the city 1.
Constraints:

- $1 \leq T \leq 200000$
- $2 \leq N \leq 200000$
- $2 \leq V \leq N$
- $0 \leq D \leq (N-1)$

Input
The first line contains 2 spaced integers N and T denoting the number of cities and number of test-cases. Each of the following $N-1$ lines contains 2 spaced integers u and v denoting the connection between city u, v. Each of the following T lines contains 2 spaced integers V and D as defined in the problem statement.

Output
Print the number of cities having connectivity D with the city 1 such that V lies in the shortest path from that particular city to the city 1.

input

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

output

```
0
1
0
```

D. Plants on Fire

1.5 seconds, 256 megabytes

You are given a rectangular land of dimension $m * n$ represented by a binary matrix, where 0 represents normal plants and 1 represents supernatural plants. A fire has started spreading from the boundaries of the land where normal plants are present. The fire cannot spread through supernatural plants, but it can spread horizontally and vertically to neighboring normal plants in one unit of time.

You are given a series of queries, each consisting of two numbers x and y. For each query, you need to determine the maximum time to pluck the fruits from the plant located at position (x, y) before the fire reaches it. If the plant does not catch fire, print "infinity".

Constraints

- $1 \leq m, n \leq 1000$
- $0 \leq x < m$
- $0 \leq y < n$
- $0 \leq q < 2 * 10^5$

Input

The first line of input consists of two integers m and n, separated by a space, representing the dimensions of the land (number of rows and columns).

Subsequently, there are m lines with n entries in each line, representing the binary 2D matrix.

The next line contains a single integer q, representing the number of queries.

The following q lines contain two space-separated integers x and y, representing the position of a plant for each query.

Output

For each query, print the maximum time in next line to pluck the fruits from the plant before the fire reaches it. If the plant does not catch fire, print "infinity".

input

```
6 5
0 0 0 0 0
0 0 1 0 0
0 1 0 1 0
1 0 0 1 0
0 1 1 0 0
0 0 0 0 0
6
0 0
1 1
2 2
3 3
4 4
5 0
```

output

```
1
2
infinity
infinity
1
```

```
1
1
```

input

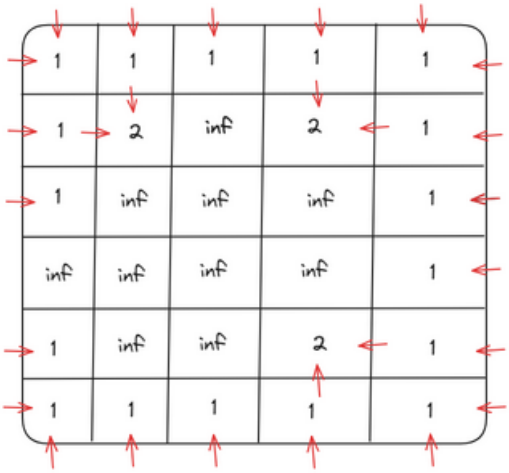
```
6 5
1 1 1 1 1
0 0 0 0 1
1 1 1 0 1
1 0 1 0 1
1 0 0 0 1
1 1 1 1 1
4
1 2
2 1
3 1
3 4
```

output

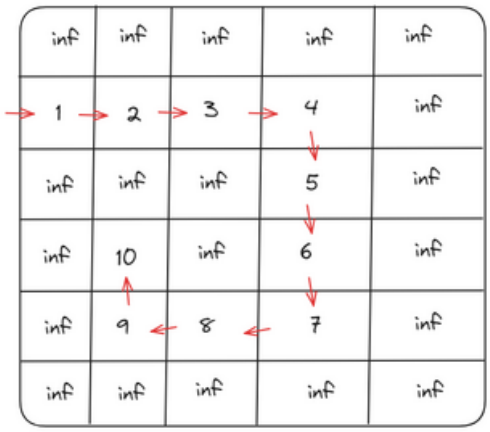
```
3
infinity
10
infinity
```

Red Arrow represents the direction of fire.

Visualisation of Testcase-1 Each cell value represent the time taken to catch fire.



Visualisation of Testcase-2



E. Treasure Collector

1 second, 256 megabytes

You are given an $n \times n$ matrix, representing a game board. The matrix contains four types of entries:

- An empty cell denoted by '.', where players can move freely.
- A player's position denoted by 'p'. There are two players, player 1 and player 2, initially located at different positions.
- A treasure position denoted by 't'. The goal is to find this treasure.
- A wall denoted by '#', which is an obstacle that players cannot pass through.

Both players can move in four directions: up, down, left, and right. Each movement to an adjacent cell (up, down, left, or right) takes exactly one unit of time. Players can move through empty cells but cannot move through walls.

Your task is to calculate the minimum time required for both players to find the treasure, assuming they collaborate optimally.

Input
The first line of input consists of a single integer n ($1 \leq n \leq 1000$), representing the size of the game board.

Subsequently,There are n lines which represent rows of matrix.

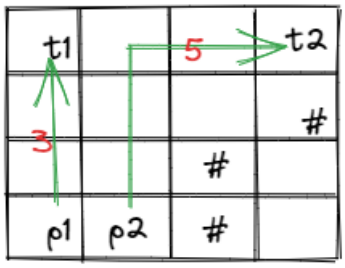
The strings contain characters '.', 'p', 't', and '#', separated by spaces.

Output
print the minimum time required for both players to find the treasure. If the treasure is unreachable, return -1.

input
4 t . . t . . . # . . # . p p # .
output
5

input
4 . . t t . p . . . . p . .
output
3

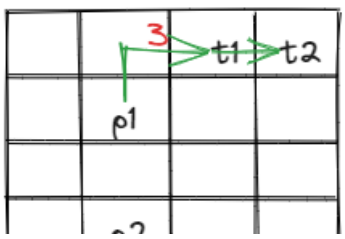
Visualisation of Testcase-1



if they work collaboratively it will takes 3 unit time to collect $t1$ by $p1$ and 5 unit time to collect $t2$ by $p2$.

So In total 5 unit time they can collect both treasures.

Visualisation of Testcase-2



$p1$ alone can collect both treasures in 3 unit time.
