Problem Set 6 — Shortest Paths — 15-295 Spring 2020

A. Meeting Place

time limit per test: 1 second
memory limit per test: 256 megabytes

You and your friend live in a town with n houses connected by m bidirectional roads. You would like to pick a house where the two of you can meet for lunch every day, but you want to pick this spot fairly. It is guaranteed that the town is connected, ie. that you can reach any house from any other house.

You live at house 1, and your friend lives at house n. You would like to pick a meeting spot such that the distance from it to your house is as close as possible to the distance from it to your friend's house. Find the smallest possible difference in distances that you can achieve.

Input

The first line contains two integers, n ($2 \le n \le 10^5$), the number of houses in the town, and m ($0 \le m \le 10^5$), the number of roads.

The following m lines contain roads, specified by three space-separated integers u, v, w, where $1 \le u, v \le n$ denote the endpoints of the road, and w ($1 \le w \le 10^6$) is the distance to travel along that road.

Output

Print the smallest possible difference between the two distances.

Examples

input	output
4 4 1 2 4 2 4 6 1 3 1 3 4 4	2

input	output
2 1 1 2 1	1

Note

In the first sample test, the best place to meet is at house 2, which is a distance 4 from you, and a distance 6 from your friend. Even though meeting at house 3 would yield a smaller total distance, house 2 is fairer since the difference between distances is less.

It may be the case (as in sample test 2) that the best place to meet is at you or your friend's house.

B. Speedy Paths

time limit per test: 1 second
memory limit per test: 256 megabytes

You're given a $n \times m$ grid of **energy** values. We consider paths in the grid that move in the four compass directions. The length of the path from a cell to one of its neighbors is 1 meter.

Our initial speed is V meters per second. Each cell contains certain amount of energy. After going from a cell with energy A to another with energy B, our speed is multiplied by 2^{A-B}.

Compute the smallest amount of time it takes to go from the upper left corner to the lower right corner.

Input

The first line contains three integers, which are V, n and m. Each of the n lines contain m integers, which are the energy values of all cells.

It is guaranteed that $1 \leq V \leq 100$, $1 \leq n, m \leq 100$. The absolute values of all energy values is at most 25.

Output

Output a single real number which is the minimum time necessary to go from the upper left corner to the lower right corner of the grid.

Example

input	output
1 3 3 1 5 3 6 3 5 2 4 3	29.0000000000

Note

The solution is regarded as correct as long as it is has a relative or absolute error of at most 10^{-6}.

C. Removing Obstacles

time limit per test: 1 second
memory limit per test: 256 megabytes

You're given an $n \times m$ grid G. The cells of the grid are each either an obstacle (indicated by a `1`) or free (indicated by a `0`). You're also given an integer T which will be explained below.

You can take a walk in the grid by starting at some cell and making a sequence of moves each in one of the four compass directions (north, south, east or west). Such a path is not allowed to make use of any cell (including the first and last one) with an obstacle in it.

A pair of cells $C_1 = (x_1, y_1)$ and $C_2 = (x_2, y_2)$ is said to be **T-compatible in a grid G** if it's possible to change T (or fewer) obstacle cells to free cells in G, then walk in the modified grid from C_1 to C_2.

The distance between the two cells C_1 and C_2 is defined to be $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2)}$. The problem is to compute the distance between the pair of T-compatible cells with the largest distance.

Input

The first line contains three integers n, m and T. Each of the following n lines contains a string describing a row of the grid. The j-th char in the i-th row is 1 if the corresponding cell is an obstacle and 0 otherwise.

It is guaranteed that $1 \leq n, m, T \leq 30$.

Output

The output is a single floating point number. It's the distance between the pair of T-compatible cells with the largest distance. Any solution within 10^{-4} of the correct answer will be considered correct.

Example

input	output
3 3 1 001 001 001	2.8284271247

D. Around the Rectangles

time limit per test: 5 seconds

memory limit per test: 256 megabytes

The input to this problem is n disjoint axis-aligned rectangles, along with a starting point (sx, sy) and an ending point (ex, ey). All have integer coordinates. The goal is to get from the starting point to the ending point with as short a path as possible. The path of movement is restricted as follows:

- You can start in any of the four directions from (sx, xy).
- At all points in time the movement is in one of the four directions parallel to the coordinate axes.
- When moving through empty space, not in contact with any rectangle, the path must continue to go in the same direction until it hits a rectangle or (ex, ey).
- The path cannot enter the interior of any of the rectangles, but it can move along the boundaries of the rectangles.
- When in contact with a rectangle, the path can change direction any time, as long as its movement is axis-aligned, and it does not enter the interior of the rectangle.

The goal is to find the length of the shortest path (obeying the rules) from (sx, sy) to (ex, ey), or determine that such a path does not exist.

Input

The first line contains a single integer $1 \leq T \leq 20$, which is the number test cases. The following lines contain T test cases. Before each test case there is a blank line.

For each test case, the first line contains four integers, which are sx, sy, ex, ey. The second line contains a single integer $0 \leq n \leq 1000$, which is the number of axis-aligned rectangles. Each of the next n lines contains four integers, which are the coordinates of two opposite corners of one of the rectangles.

It is guaranteed that no two rectangles share any common point, including on their boundaries. It is also guaranteed that the $(sx, sy) \neq (ex, ey)$, all coordinates are in $[-10^9, 10^9]$, and all rectangles have strictly positive area.

Output

For each test case, output a single line with a single integer which is the shortest path length from the starting to the ending point. If there is no valid path, output "No Path".

Example

input	output
2 1 7 7 8 2 2 5 3 8 4 10 6 7 2 1 5 4 1 3 1 4 3	9 No Path

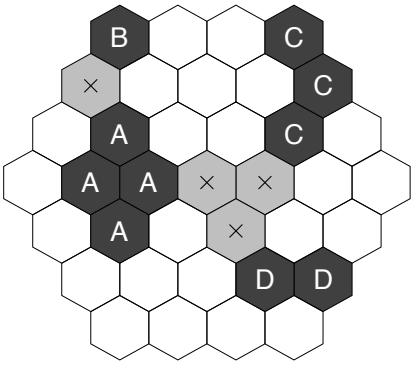
E. Connecting in a Hexagonal Grid

time limit per test: 1 second

memory limit per test: 256 megabytes

A civil engineer that has recently graduated from the Czech Technical University encountered an interesting problem and asked us for a help. The problem is more of economical than engineering nature. The engineer needs to connect several buildings with an infrastructure. Unfortunately, the investor is not the owner of all the land between these places. Therefore, some properties have to be bought first.

The land is divided into a regular "grid" of hexagonal parcels, each of them forms an independent unit and has the same value. Some of the parcels belong to the investor. These parcels form four connected areas, each containing one building to be connected with the others. Your task is to find the minimal number of parcels that must be acquired to connect the four given areas.



The whole land also has a hexagonal shape with six sides, each consisting of exactly H parcels. The above picture shows a land with $H = 4$, parcels with letters represent the four areas to be connected. In this case, it is necessary to buy four additional parcels. One of the possible solutions is marked by crosses.

Input Specification

The input contains several scenarios. Each scenario begins with an integer number H, which specifies the size of the land, $2 \leq H \leq 20$. Then there are $2.H - 1$ lines representing individual "rows" of the land (always oriented as in the picture). The lines contain one non-space character for each parcel. It means the first line will contain H characters, the second line $H + 1$, and so on. The longest line will be the middle one, with $2.H - 1$ characters. Then the "length" descends and the last line contains H parcels, again.

The character representing a parcel will be either a dot ("`.`") for the land that is not owned by the investor, or one of the uppercase letters "`A`", "`B`", "`C`", or "`D`". The areas of parcels occupied by the same letter will always be connected. It means that between any two parcels in the same area, there exists a path leading only through that area.

Beside the characters representing parcels, the lines may contain any number of spaces at any positions to improve "human readability" of the input. There is always at least one space between two letters (or the dots). After the land description, there will be one empty line and then the next scenario begins. The last scenario is followed by a line containing zero.

Output Specification

For each scenario, output one line with the sentence "`You have to buy` P `parcels.`", where P is the minimal number of parcels that must be acquired to make all four areas connected together.

Areas are considered *connected*, if it is possible to find a path between them that leads only through parcels that have been bought.

Sample Input

```
4
    B . . C
   . . . . C
  . A . . C .
 . A A . . . .
  . A . . . .
   . . . D D
    . . . .

0
```

Output for Sample Input

```
You have to buy 4 parcels.
```