LICT

19-20 April 2019

SUST Inter University Programming Contest 2019

Powered By CEFALO

Contest Platform Toph

B Play the Lottery

Input: Standard Input
Output: Standard Output

The statement of this problem is going to be short and sweet because the setter was busy with life and didn't get a lot of time to prepare it. You have gone to a fair in some shady part of USA and all of a sudden you find a tent conspicuously placed in the center of the fair. You go inside the tent and you find M boxes placed side by side with balls of different number written on them. The boxes are not transparent so you can't see the number on the balls of course.

The owner of the tent invites to play the game. "Luck is overrated. Play the game and win prizes!", he cries out. He explains to you that you will choose a number N at the start of the game and then select one ball from each box randomly and at the end if the sum of numbers on the balls that you collected is equal to the number N you win! You decide to watch the game for a while before playing to know a bit about what kind of balls are in each box. After watching for hours you have made a list of observations that you made about each box. For each box you have discovered the number of balls it has of each number. Given this information can you calculate the number of ways to make different values of N?

Input

The first line of input contains M($1 \leq M \leq 10^5$) - the number of boxes. The next M lines contain the information of each box. The first number of these lines contain X($0 < X < 10^5$) denoting the number of types of balls in it. The next X pairs contain information of the balls. The first number of each pair is the number written on this type of ball and the second one is the amount, c_i($0 < c_i < 10^3$) of this type of ball in the box. You may assume that for a single box, all the types will be unique (that is two different types of balls won't have the same number written on them). Also the total sum possible with the highest numbered balls from each box won't be greater than 10^5.

Output

For each possible sum that can be made with the balls, print the sum and the number of ways it can be made in each line. Output should be sorted by increasing value of sum. Do not print the sum that cannot be made or the the number of ways they can be made is 0 after modulo by 786433. Print the number of ways modulo 786433.

Sample Input	Output for Sample Input
2 2 2 2 1 1 2 3 3 1 1	2 1 3 2 4 3 5 6

C Childhood Game

Input: Standard Input
Output: Standard Output

About twenty years ago, when we had only 16MB RAMs and 5GB hard disks, playing games might seem like a difficult thing to do. But luckily, we did not require very high end hardware back then to play some amazingly engaging games. One of these games is called, the "Brick Buster".

The idea behind Brick Buster or similar games was like this: You have a bouncing ball and a tray inside of a rectangular grid. The grid is closed on the three sides: left, right and top. The bottom side is empty. When the ball keeps bouncing inside the grid, it hits different walls on the left, right or top. After hitting the walls, the ball just changes its direction, but keeps moving at its initial speed. But when it reaches the bottom of the grid, it falls off and the game ends. A tray floats near the bottom of the grid, which a player can move to left or right of the screen. The purpose of the player is to move his tray in such a way, that the ball does not fall off the grid. Additionally, there are some bricks near the top of the board, you have to break those bricks by hitting the bricks with the bouncing ball.

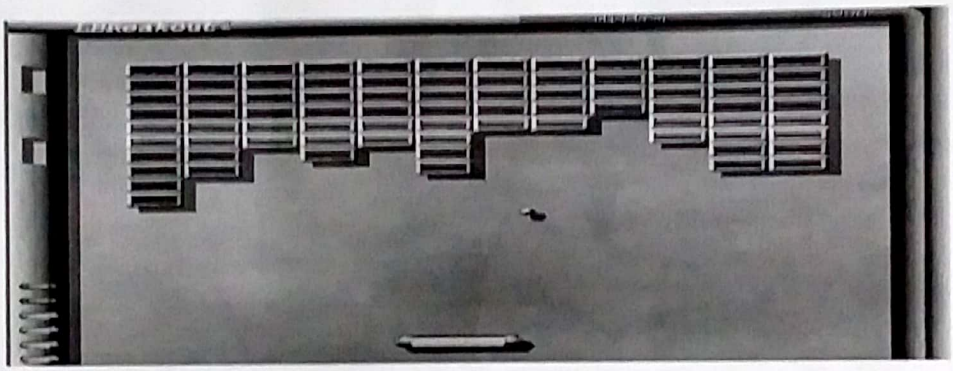


Figure: Picture of a brick breaking game named "Breakout". Image credit: Wilinckx

You are cranking up a new company and want to reproduce this game for portable devices. You thought to yourself, well, this is easy. But soon you figured, it is not. Time is running out, and meeting with your investor is coming near. As a result, you have decided to create an easier version of the game.

In your "easier" game, there are no bricks, no trays, not even an empty side at the bottom of the grid. You have a 500 * 500 board, where the upper-left corner has the coordinate (0, 0). There are just some moving balls, that keeps bouncing all over the board. Initially, every ball has a specific position and a fixed velocity. Whenever a ball hits the wall, it just changes its direction by the following rules:

1. If the ball hits the left or the right wall, then the direction towards the x axis changes to the opposite.
2. If the ball hits the top or the bottom wall, then the direction towards the y axis changes to the opposite.

A collision between two balls takes place when the distance between the center of two balls is less than or equal to the sum of their radius. When two balls collide, they change their direction by following these rules:

1. If two balls have been moving in the same direction in terms of x-axis, their movement towards the x-axis will not change after the collision. Otherwise, both of the balls' movement towards the x-axis will change to the opposite.
2. If two balls have been moving in the same direction in terms of y-axis, their movement towards the y-axis will not change after the collision. Otherwise, both of the balls' movement towards the y-axis will change to the opposite.

Now, given some balls, their initial velocity, you have to determine the position of the balls after some certain number of seconds.

You can apply the following assumptions for your calculations:

1. The game only renders one frame every second. The transition from one frame to the next is discrete and the information between two consecutive frames are lost.
2. If more than two balls collide with each other at the end of a second, then the collision must be processed sequentially in the lexicographical order of the balls. If three balls B1, B3 and B5 collide together at the end of a second, then the collisions must be processed in this order: (B1, B3), (B1, B5), (B3, B5). Here, 1, 3 and 5 are the indexes of the balls in their input sequence.
3. If a ball is in collision with the other balls and walls at the same time, then the collision with the balls will be calculated first.
4. Every ball starts moving at the 0-th second. After the end of a second, collisions will be calculated.

You can safely assume that initially no ball is in collision with other balls or the wall.

Input

Input starts with B ($1 \le B \le 50$), the number of balls. Then follow B lines, each having two integers X and Y ($6 \le X, Y \le 494$), the initial position of the balls. No balls will be at the same place in the beginning. For simplicity, assume that each of the balls is a circle with a radius of 5. Then follow another B lines, each with two numbers V_x and V_y ($|Vx|=1, |Vy|=1$), the original velocity towards the x axis and the y axis for each of the ball.

In the last line of input, you will be given an integer T (≤ 10000), the number of seconds from the start of the game, where you need to report the position of the balls.

Output

For each case, output B lines, one for each ball. In each line, you will have two real numbers X_i and Y_i, the position of the i'th ball (according to the input sequence) after T seconds. Your answer needs to be accurate for 2 decimal places.

Sample Input	Output for Sample Input
1 10 20 1 1 1	11.000000 21.000000

D — So Much Difference

Input: Standard Input
Output: Standard Output

Difference starts with a 'D'. This is the root of all dissimilarities and arguments and fights. Figuring out differences is just like figuring out your life. Sometimes they confuse you but in the end it is all worth it. If you can't find out the difference between things, or solve the difference between people, your life is as meaningless as you are. But before you can solve the difference between things you must find it out. That's why you have made a function that can figure out the difference between everything.

You are given a Tree. And some paths on that tree, you need to use your function to figure out the difference between those paths.

```
function Distance(x1, y1, x2, y2)
    Array1 = []
    Array2 = []

    For A1 = nodes on the unique path from x1 to y1
        Array1.push(V[A1])
    end For

    For A2 = nodes on the unique path from x2 to y2
        Array2.push(V[A2])
    end For

    NewArray1 = sort the elements inside Array1
    NewArray2 = sort the elements inside Array2

    len1 = element count in Array1
    len2 = element count in Array2
    mid1 = ceiling(len1 / 2)
    mid2 = ceiling(len2 / 2)
    maxFreq = 0

    For P = elements in Array1
        cnt = 0
        For Q = elements in Array1
            if P equals Q
                cnt = cnt + 1
            end if
        end For
       maxFreq = Maximum(maxFreq, cnt)
    end For
    //|A| denotes the absolute value of A
    difference = |NewArray1[mid1] - NewArray2[mid2]|

    return (maxFreq * difference)
end function
```

Note: All the Array index starts from 1

Input

The first line contains an integer **N** – number of nodes in the tree.

Then there will be a single line containing **N** integers **V[1], V[2], V[3], … V[N]**. **V[i]** is the value on the i^{th} node. Following **(N - 1)** lines will contain two integers **u** and **v**, means there is an edge from **u** to **v**. Edges are bidirectional.

Then there will be Q queries. Each line of query will contain four integers – **x1**, **y1**, **x2**, **y2**.

Constraints:

$1 \le N \le 10^5$

$1 \le u, v, x1, y1, x2, y2 \le N$

$1 \le Q \le 10^5$

$-10^8 \le V[i] \le 10^8$

Output

For each query output the answer of the Distance function in a single line. For better understanding check the samples.

Sample Input	Output for Sample Input
8 1 2 3 4 5 6 7 8 1 2 2 3 3 4 4 5 5 6 6 7 7 8 5 1 2 1 7 2 8 4 8 1 5 5 8 1 8 2 5 8 8 3 7	3 1 3 1 3

E Remember the Name

Input: Standard Input
Output: Standard Output

There's a famous song written by Mike Shinoda, which has these lines: **This is 10% luck, 20% skill, 15% concentrated power of will, 5% pleasure, 50% pain, and a 100% reason to remember the name...**

So, this problem is about Luck and Skill. You are moderating a competition. There are already **N** competitors. And there are **Q** new competitors yet to come. Each of them is very competitive. They follow the exact same procedure. Each of these contestants first comes to you, asks you exactly how many competitors are compatible with him/her (let's denote this number as **comp(q_i)** for the **i-th** new competitor), and then joins the competition. But there's a little problem with the new competitors. If the number of compatible competitors exceeds the tolerance factor (let T_i denote it) of a new competitor, then he/she thinks that there are too many compatible competitors and he/she doesn't join the competition.

Each competitor has a luck factor L_i and a skill factor S_i. The number **comp(q_i)** denotes the number of compatible competitors (who has already joined the competition before the **i-th** new competitor) with the **i-th** new competitor. Let's say the luck factor and skill factor of a competitor is **L** and **S**, respectively. Then he/she will be compatible with the **i-th** new competitor if, $(L-L_i)^2 + (S-S_i)^2 <= X_i^2$, where X_i is the satisfaction factor of the **i-th** new competitor.

Your job is to answer every new competitor that how many compatible competitors with them there are already and whether it's within the tolerance limit or not.

Input

The first line will contain two integers **N** and **Q**, the number of pre-existing competitors and the number of new competitors to join the competition. The next two lines both will contain **N** integers, the first one will contain the luck factors and the second line will contain the skill factors. The **i-th** integer of both lines denotes the **i-th** competitor.

The next **Q** lines will contain four integers L_i, S_i, X_i, and T_i each. **i-th** line denotes the luck factor, skill factor, satisfaction factor and tolerance factor of the **i-th** new competitor, respectively.

Constraints:

$1 <= N, Q, L_i, S_i, X_i <= 10^5$

$1 <= T_i <= 10^3$

Output

You will print **Q** lines that contain one integer each, where the **i-th** integer denotes the number of compatible competitors with the **i-th** new competitor. If the number of compatible competitors exceeds the tolerance factor for that competitor, print **"Too Many!!!"** without the quotes.

Sample Input	Output for Sample Input
7 6 0 2 2 0 1 0 0 0 0 2 2 1 0 0 1 0 1 7 0 1 2 6 0 0 2 7 2 2 2 6 1 1 5 10 1 1 5 10	5 6 Too Many!!! 4 10 Too Many!!!

F MaxFlow (II)

Input: Standard Input
Output: Standard Output

Given a connected undirected graph with **n** nodes and **m** edges and the capacity of each edge is **1**. **maxFlow(u, v)** defines the maximum flow from node **u** to node **v**. You have to find out how many pair of nodes ***(u, v)*** are there where ***u < v*** and ***maxFlow(u, v) = 2***.

Input

The first line of the input contains an integer **T**, denoting the number of test cases. For each test case, the first line will have two integers **n** and **m**, denoting the number of nodes and edges of the graph. The following **m** lines will contain two integers ***u*** and ***v***, denoting an undirected edge between ***u*** and ***v***. You can assume that the graph won't have any self loop.

Constraints:

$1 \le T \le 50$

$1 \le n \le 10^5$

$n - 1 \le m \le 7 \times 10^5$

$1 \le u, v \le n$

$\sum_{i}^{T} n_i \le 5 \times 10^5$

$\sum_{i}^{T} m_i \le 10^6$

Output

For each test case, print the case number and the result for that case.

Sample Input	Output for Sample Input
2 3 3 1 2 1 3 3 2 3 2 1 2 2 3	Case 1: 3 Case 2: 0

G APM

Input: Standard Input
Output: Standard Output

Package managers are common tools in software development. The main purpose of those tools are mainly to store new package library, retrieving and resolving package dependencies. ACM corporation is planning to create a simple package manager called APM(ACM Package Manager).

You will be requested to add a new package with a dependent package name list. If there is no conflict to resolve recursive dependencies and all dependent packages do exist in the APM server, then the requested package will be added to the server, otherwise not.

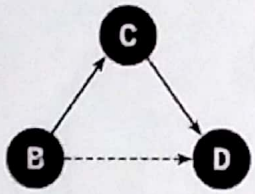
The APM package name formatting is: **package-id:version-number**

Example: acm-server:v1.0

Here, package-id = acm-server

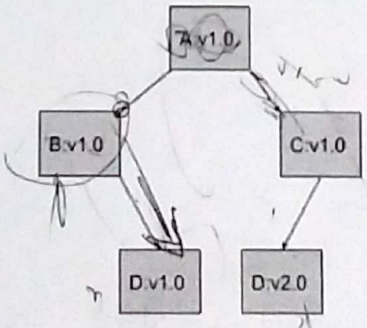
version-number = v1.0

APM follows transitive dependency rules to resolve dependencies. If package B depends on package C and C depends on D, then B indirectly depends on D.



While adding a new package, dependencies will be conflicted or error will occur if any of the following is true:

- if there's any directly or indirectly dependent package with same package id but different version number
- if any of the dependent package doesn't exist in APM Server



This is a conflicting case, because A:v1.0 wants to import different versions of D.

Input

You are given list of N (1 <= N <= 100) requests. Each request has a new package with unique name P (1 <= length(P) <= 100) and M (0 <= M <= 10) size of the dependent package name list. Then following M lines contain dependent package names. In a package name, package-id and version number is separated by a colon( ) and contains only alphanumeric characters. Initially APM server contains no package.

Output

For each request find out if there is any error or conflict to add new package. If so, output "Request {ID}: ERROR", otherwise output "Request {ID}: OK" and print list of all added package names so far including the current one. The list should be ordered lexicographically by package-id, and in case package-id is same, ordered lexicographically by version-number.

Sample Input

```
7
D:v1.0
0
D:v2.0
10
C:v1.0
0
D:v2.0
B:v1.0
1
D:v1.0
A:v1.0
2
B:v1.0
C:v1.0
E:v1.0
1
C:v1.0
E:v2.0
1
F:v1.0
```

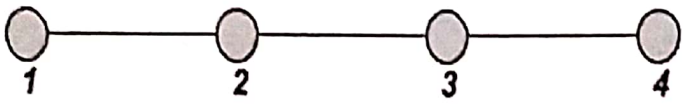
Output for Sample Input

```
Request 1: OK
D:v1.0
Request 2: OK
D:v1.0
D:v2.0
Request 3: OK
C:v1.0
D:v1.0
D:v2.0
Request 4: OK
B:v1.0
C:v1.0
D:v1.0
D:v2.0
Request 5: ERROR
Request 6: OK
B:v1.0
C:v1.0
D:v1.0
D:v2.0
E:v1.0
Request 7: ERROR
```

H T5 Racing

Input: Standard Input
Output: Standard Output

It is high time for the annual T5 racing and your team is planning to win the race.
In T5 racing the drivers need to complete at most ***X*** laps (lap means a portion of a trip). However, the racing track is different from F5 racing. The T5 racing track has ***N*** checkpoints and each checkpoint ***i*** has score point P_i associated with it. Checkpoints are connected by some bidirectional racing roads and there is no cycle in the T5 racing track.



T5 Racing Track

For example, there are four checkpoints in the picture and three racing roads connecting them. The rules of the T5 racing are very simple. At any point of time, if a driver is at checkpoint ***2,*** he/she can move to any of the directly connected checkpoint ***1 or 3*** to complete one lap.
The race will start at checkpoint 1 and the drivers can finish the race at any checkpoint. Two drivers can take two different paths to finished the race as long as they complete at most **X** laps.

A driver can add score point P_i to his/her scorecard when he/she enters checkpoint ***i*** for the first time. Drivers can enter the same checkpoint multiple times during the race but they will get the score only the first time. The highest scorer will win the race and there can be multiple winners.

Since you are the greatest T5 hacker on this Galaxy, and you already know the racing track map for the next race, now you have to tell what is the maximum score your team driver needs to win the race.

Input

Input starts with an integer ***T*** **(≤ 100)**, denoting the number of test cases.
Each case starts with two integers ***N*** **(1 ≤ *N* ≤ 100)** and ***X*** **(1 ≤ *X* ≤ 200)** as mentioned above. Next line contains ***N*** integers separated by spaces and the i'th integer of this line represents the score point P_i $(0 \le P_i \le 10^5)$ of the ***i'th*** checkpoint. Next ***N*-1** lines will contain two integers ***i*** and ***j***, meaning that there is a raceing road between checkpoint ***i*** and ***j***.

Output

For each case, print the case number and the maximum score point your team driver needs to win the race in a line.

Sample Input	Output for Sample Input
1 4 2 1 4 2 3 1 2 1 3 3 4	Case 1: 6

I Biswa and Morse Code

Input: Standard Input
Output: Standard Output

Biswa wanted to know what the heck Morse Code is and here's what he found out on the internet -

Morse code is a character encoding scheme used in telecommunication that encodes text characters as standardized sequences of two different signal durations called dots and dashes or dits and dahs. The International Morse Code encodes the 26 English letters A through Z, some non-English letters, the Arabic numerals and a small set of punctuation and procedural signals (prosigns). There is no distinction between upper and lower case letters. Each Morse code symbol is formed by a sequence of dots and dashes. The dot duration is the basic unit of time measurement in Morse code transmission. The duration of a dash is three times the duration of a dot. Each dot or dash within a character is followed by period of signal absence, called a space, equal to the dot duration. The letters of a word are separated by a space of duration equal to three dots, and the words are separated by a space equal to seven dots.

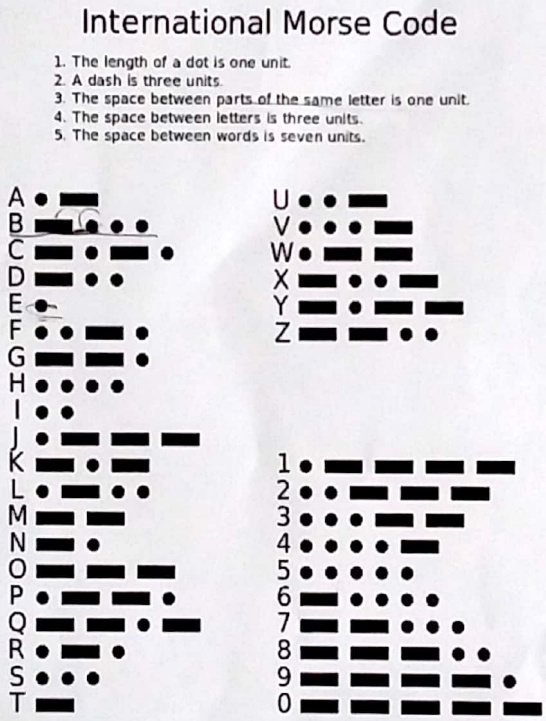
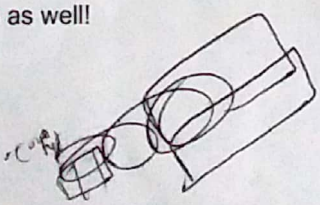


Fig. Chart of the International Morse code of 26 letters and 10 numerals.

Time to hear about the cool project Biswa has been working on lately. It's called - Online Love Guru. It's an online platform where users ask their love related questions anonymously. The system generates the response to the questions of the users using some "Machine Learning" approach and send back to them. This online platform is now one of the most used one on planet earth and guess what, the aliens want to use this as well!



But he wants to do something interesting for the aliens. He wants to send them the answer to their queries using morse code. He now has a list of answers that can cover all possible questions. And he wants to transmit each answer in most efficient way. For each question, he'll first send his morse code encoding for each character used in that answer and then transmit the answer using those morse code encodings. He firmly believes that it is potentially possible to find more efficient morse codes than the conventional ones for each of the answers in his list. The efficient morse codes will require less duration to transmit the answer.

Given the list of answers, can you write a program to generate the most efficient morse codes for each answer?

Constraints:

- Each answer contains one or more words
- Each word contains english letters ('a'-'z', 'A'-'Z') and numbers ('0' - '9')
- Each word contains no more than 20 characters
- Each answer contains no more than 100 words
- There is no distinction between upper and lower case letters
- Efficient morse code should also be prepared using dots and dashes and will have non-zero length
- All the letters and numbers will be present at least once in an answer

Input

First line of the input will specify the number of answers - **N** ($1 \le N \le 100$). Then there will be N subsequent lines with one answer each.

Output

Output should contain the proposed efficient morse codes for letters A to Z followed by the morse codes for numbers 0 to 9. If there's multiple set of most efficient morse codes, you can output anyone from there.

Sample Input

```
1
Pack my box with five dozen liquor jugs 0123456789
```

Output for Sample Input

```
-.-.
.--.
--.-
....-
-..-
.-
.-.-
..--
---
-
.....
-...
```

J Jumping Frog (II)

Input: Standard Input
Output: Standard Output

A frog wants to cross a river. There are some posts in the river. The frog can go from one post to another by jumping. All the posts are located in a single line and numbered from **1** to **N** and distance between **i-th** post and **(i+1)-th** post is **1**. Distance between the **1st** post and the starting position is **1** and distance between the **N^{th}** post and the other side of the river **1**. The frog can jump at most X_i distance from **i^{th}** post. Now you have to answer if there is any path that the frog can take to cross the river.

Input

Input starts with an integer **T (≤ 5)**, denoting the number of test cases.
Each case starts with two integers **N** and **Q** where **N** is the number of posts and **Q** is number of queries. Next line contains **N** integers separated by spaces and the **i'th** integer of this line represents the strength X_i of the **i'th** post. Next **Q** lines will contain two or three integers each. First number **t** represents the type of query.

- If **t = 1**, there will be 2 more integers **p**, **s** meaning that **p^{th}** post's strength changes to **s**.
- If **t = 2**, there will be one more integer **s** meaning that frogs can jump at most **s** distance from starting point.

Constraints:
$1 \le N \le 10^5$
$1 \le Q \le 10^5$
$0 \le X_i \le 10^9$
$1 \le t \le 2$
$1 \le p \le N$
$0 \le s \le 10^9$

Output

For each case, print the case number in one line and for each query type of 2 output **"yes"** if the frog can go to other side with current configuration or **"no"** otherwise in one line.

Sample Input	Output for Sample Input
1 5 11 0 0 1 5 3 2 1 2 2 2 3 2 4 2 5 1 4 0 2 1 2 5	Case 1: no no yes yes yes no yes no yes

2 3 1 1 4 2 1	

K Send the Love

Input: Standard Input
Output: Standard Output

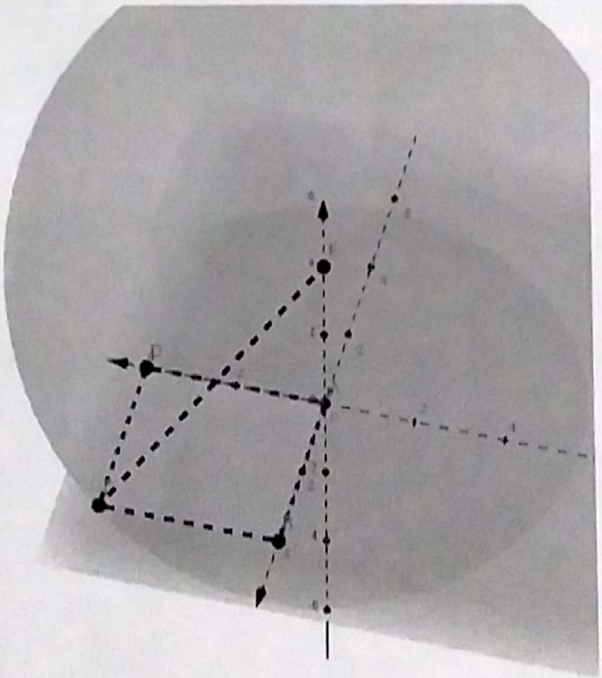
Yesterday **Akhtaruzzaman Mridha**(Akhtar mia) was stealing mango from a tree near Asad Gate. And suddenly the branch he was sitting on, broke, and he fell. But Luckily a strong girl named **Nadia Tasnim** was below the tree. She caught Akhter mia and saved him from breaking his neck. That was it for Akhter mia! Instant love! Next thing he could remember is he saying the girl "I Love You". Somehow Nadia found that very cute and accepted his Love. Now they are a couple. Sweet eh?

Since his childhood, Akhter mia has dreamt of sending love emoji to his girlfriend. But after Nadia came into his life, something really frustrating started to happen. As Nadia likes to roam around the city a lot, sometimes she goes out of network, which results in Akhtar mia not being able to send her any love emoji. Akhter mia was very sad that he cried for twenty whole days. Then he decided to put those tears back into his eyes and do something about the situation.

Being a very smart individual, Akhtar mia developed an app along with networking hardware called **Lovesphere**. Lovesphere has a spherical coverage area. We call two Lovespheres **connected** if the two Lovespheres **have an intersection between them**. Also, **connectivity is transitive** (If Lovesphere A and B are connected, and Lovesphere B and C are connected. Then Lovesphre A and C are also connected).

If two individuals use his app from the **same Lovesphere or from two connected Lovespheres**, they can send love emoji to each other without any problem.

But as the city mayor had a crush on Nadia, so he wanted Akhtar mia to fail. So he permitted him some weird coordinates in the sky, where Akhtar mia can set up his Lovesphere(s).



The city they live in has a rectangular shape with corners at (0,0), (x,0), (x,y), (0,y). Akhtar mia lives in his lab at (0,0) and never goes out. On the other hand, Nadia is always roaming around in the city. As none of them can fly, they always stay on the **ground and inside the city**.

Akhtar mia got permission for setting up **N** Lovespheres. Now Akhtar mia wants the radius of the coverage of all the Lovespheres to be same. He wants to know the minimum radius **r** he can set for all of the Lovespheres, such that, no matter wherever Nadia is, he can always send love emoji to her.

Input

The first line of the input will have two integers **x**, **y**, representing the corner of the rectangle opposite to the origin. Then next line will contain an integer **N** representing the number of Lovespheres he was permitted by the mayor. Then each of the next **N** lines will have three integers x_i, y_i, h_i representing the coordinates and the elevation where he is permitted to place Lovespheres.

Constraints:

$1 <= N <= 10^3$

$0 <= x, y <= 10^5$

$0 <= x_i <= x$

$0 <= y_i <= y$

$0 <= h_i <= 10^5$

Output

Print a single real number **r** (rounded to two digits after decimal) – the minimum radius he needs to set for all the Lovespheres.

Sample Input	Output for Sample Input
2 2 2 0 1 1 2 1 1	1.73

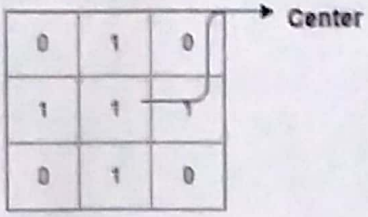
Sample Input	Output for Sample Input
2 2 2 0 0 1 2 2 1	2.24

L Game in a Binary Grid

Input: Standard Input
Output: Standard Output

Binary grid indicates to a grid that contains only 0s and 1s in its cells. You can find a plus(+) symbol in a binary grid.

Following grid contains a plus(+) symbol.



You can form a plus symbol by choosing some cells from a grid if the chosen cells satisfy all of the following conditions:

- Number of 1s in each of four directions must be same.
- Each cell containing 1 (except the center) must have 0 in both side. For the cells in horizontal line both side means adjacent upper and lower cell, for the cells in vertical line both side means adjacent left and right cell.

Weight of a plus symbol means the distance between the center and furthest cell containing 1. Weight of the above plus(+) symbol is 1. Whenever you make a plus(+) symbol, your objective is to maximize its weight. Same cell can be part of multiple plus symbol.

Following grid contains a plus symbol of weight 2.

0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	1	1	1	1	1	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Score of a grid is the sum of weight of all plus symbols in it.

Score of the following grid is 3 (weight of plus symbol centered in (3, 3) cell is 2 and weight of plus symbol centered in (6, 3) cell is 1)

0	0	1	0	0	1
0	0	1	0	0	1
1	1	1	1	1	0
0	0	1	0	0	0
0	0	1	0	0	0
0	1	1	1	0	0
0	0	1	0	0	0

Bob has received a binary grid from his friend as a birthday gift. But he has lost the data of this grid. He has become sad, because he didn't calculate the score of his grid. Only thing he can remember is the length and width of the grid. So he is interested in knowing the expected score of the grid if it is filled up randomly. You have a good friendship with Bob. So you want to help Bob by calculating the expected score of his grid.

You will be given the length and width of a binary grid. You task is to calculate the expected score of the grid if it is filled up randomly. Assume that 0s and 1s are distributed in the grid with equal probability.

Input

First line contains an integer T, denotes the number of test cases.
Then T line follows, Each line contains two integers N and M, denotes the length and width of the grid respectively.

Constraints:

$1 \le T \le 10^3$

$1 \le N, M \le 10^6$

Output

For each case, output the case number and the expected score in a single line.
If the answer is $\frac{P}{Q}$, then print it as $\left(P \times Q^{-1}\right) mod\ 1000000007$
Here, P is an integer greater than or equal to 0, Q is an integer greater than 0.
See the sample output for exact format.

Sample Input	Output for Sample Input
1 3 3	Case 1: 142578126