A. Diamond Miner

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Diamond Miner is a game that is similar to Gold Miner, but there are n miners instead of 1 in this game.

The mining area can be described as a plane. The n miners can be regarded as n points **on the y-axis**. There are n diamond mines in the mining area. We can regard them as n points **on the x-axis**. For some reason, **no miners or diamond mines can be at the origin** (point $(0, 0)$).

Every miner should mine **exactly** one diamond mine. Every miner has a hook, which can be used to mine a diamond mine. If a miner at the point (a, b) uses his hook to mine a diamond mine at the point (c, d), he will spend $\sqrt{(a-c)^2 + (b-d)^2}$ energy to mine it (the distance between these points). The miners can't move or help each other.

The object of this game is to minimize **the sum of the energy** that miners spend. Can you find this minimum?

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the number of miners and mines.

Each of the next $2n$ lines contains two space-separated integers x ($-10^8 \le x \le 10^8$) and y ($-10^8 \le y \le 10^8$), which represent the point (x, y) to describe **a miner's or a diamond mine's** position. Either $x = 0$, meaning there is a miner at the point $(0, y)$, or $y = 0$, meaning there is a diamond mine at the point $(x, 0)$. There can be multiple miners or diamond mines at the same point.

It is guaranteed that no point is at the origin. It is guaranteed that the number of points on the x-axis is equal to n and the number of points on the y-axis is equal to n.

It's guaranteed that the sum of n for all test cases does not exceed 10^5.

Output

For each test case, print a single real number — the minimal sum of energy that should be spent.

Your answer is considered correct if its absolute or relative error does not exceed 10^{-9}.

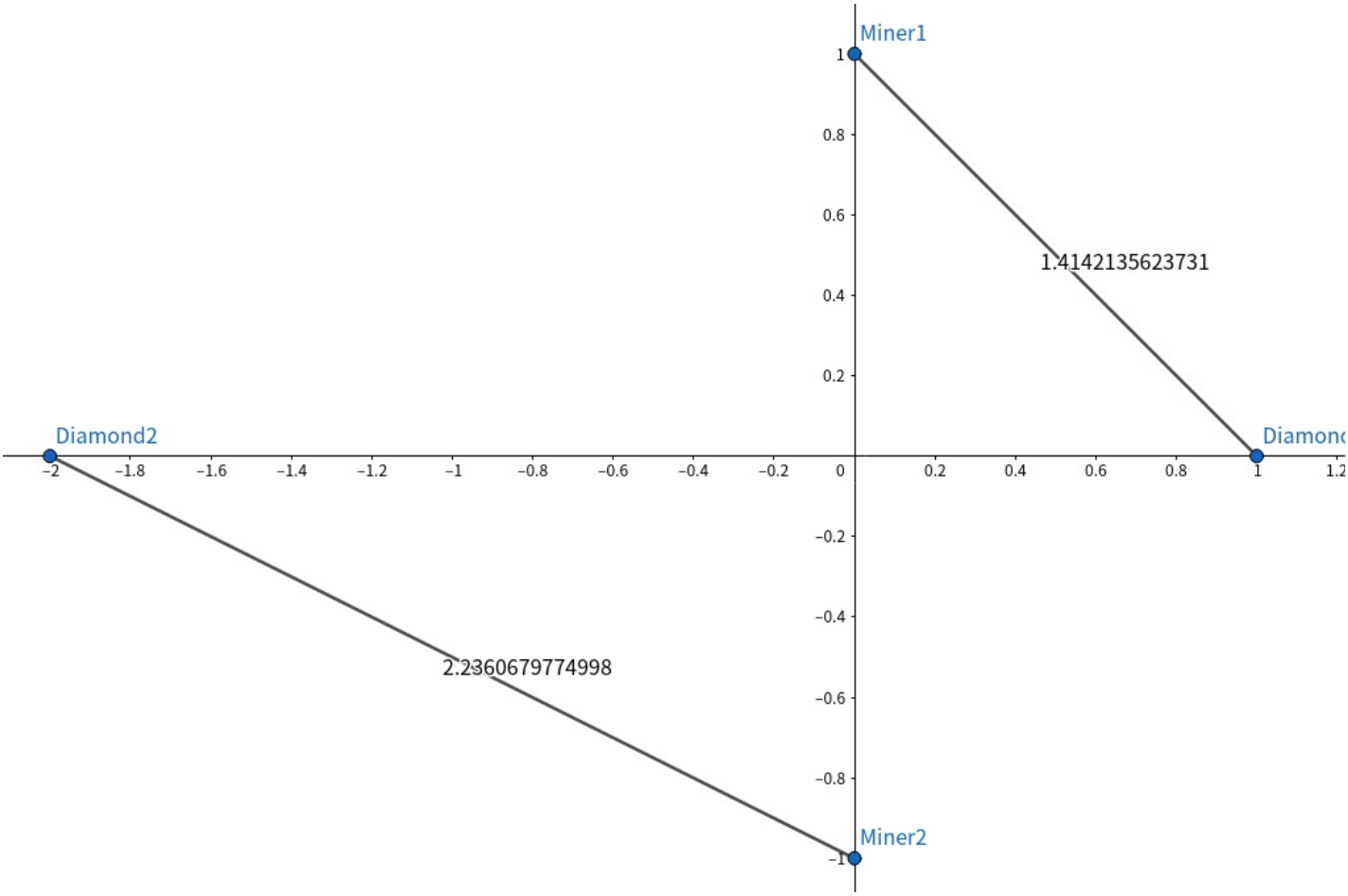
Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1, |b|)} \le 10^{-9}$.

Example

input
3 2 0 1 1 0 0 -1 -2 0 4 1 0 3 0 -5 0 6 0 0 3 0 1 0 2 0 4 5 3 0 0 4 0 -3 4 0 2 0 1 0 -3 0 0 -10 0 -2 0 -10
output
3.650281539872885 18.061819283610362 32.052255376143336

Note

In the first test case, the miners are at $(0, 1)$ and $(0, -1)$, while the diamond mines are at $(1, 0)$ and $(-2, 0)$. If you arrange the miners to get the diamond mines in the way, shown in the picture, you can get the sum of the energy $\sqrt{2} + \sqrt{5}$.



B. Let's Go Hiking

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

On a weekend, Qingshan suggests that she and her friend Daniel go hiking. Unfortunately, they are busy high school students, so they can only go hiking on scratch paper.

A permutation p is written from left to right on the paper. First Qingshan chooses an integer index x ($1 \le x \le n$) and tells it to Daniel. After that, Daniel chooses another integer index y ($1 \le y \le n$, $y \ne x$).

The game progresses turn by turn and as usual, Qingshan moves first. The rules follow:

- If it is Qingshan's turn, Qingshan must change x to such an index x' that $1 \le x' \le n$, $|x' - x| = 1$, $x' \ne y$, and $p_{x'} < p_x$ at the same time.
- If it is Daniel's turn, Daniel must change y to such an index y' that $1 \le y' \le n$, $|y' - y| = 1$, $y' \ne x$, and $p_{y'} > p_y$ at the same time.

The person who can't make her or his move loses, and the other wins. You, as Qingshan's fan, are asked to calculate the number of possible x to make Qingshan win in the case both players play optimally.

Input

The first line contains a single integer n ($2 \le n \le 10^5$) — the length of the permutation.

The second line contains n distinct integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$) — the permutation.

Output

Print the number of possible values of x that Qingshan can choose to make her win.

Examples

input
5 1 2 5 4 3
output
1

input
7 1 2 4 6 5 3 7
output
0

Note

In the first test case, Qingshan can only choose $x = 3$ to win, so the answer is 1.

In the second test case, if Qingshan will choose $x = 4$, Daniel can choose $y = 1$. In the first turn (Qingshan's) Qingshan chooses $x' = 3$ and changes x to 3. In the second turn (Daniel's) Daniel chooses $y' = 2$ and changes y to 2. Qingshan can't choose $x' = 2$ because $y = 2$ at this time. Then Qingshan loses.

C. Garden of the Sun

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are many sunflowers in the Garden of the Sun.

Garden of the Sun is a rectangular table with n rows and m columns, where the cells of the table are farmlands. All of the cells grow a sunflower on it. Unfortunately, one night, the lightning stroke some (possibly zero) cells, and sunflowers on those cells were burned into ashes. In other words, those cells struck by the lightning became empty. Magically, **any two empty cells have no common points** (neither edges nor corners).

Now the owner wants to remove some (possibly zero) sunflowers to reach the following two goals:

- When you are on an empty cell, you can walk to any other empty cell. In other words, those empty cells are connected.
- There is **exactly one** simple path between any two empty cells. In other words, there is no cycle among the empty cells.

You can walk from an empty cell to another if they share a common edge.

Could you please give the owner a solution that meets all her requirements?

Note that you are not allowed to plant sunflowers. You **don't need** to minimize the number of sunflowers you remove. It can be shown that the answer always exists.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains two integers n, m ($1 \le n, m \le 500$) — the number of rows and columns.

Each of the next n lines contains m characters. Each character is either 'X' or '.', representing an empty cell and a cell that grows a sunflower, respectively.

It is guaranteed that the sum of $n \cdot m$ for all test cases does not exceed $250\,000$.

Output

For each test case, print n lines. Each should contain m characters, representing one row of the table. Each character should be either 'X' or '.', representing an empty cell and a cell with a sunflower, respectively.

If there are multiple answers, you can print any. It can be shown that the answer always exists.

Example

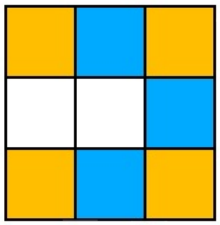
input
5 3 3 X.X ... X.X 4 4 .... .X.X .... .X.X 5 5 .X... ....X .X... ..... X.X.X 1 10 ....X.X.X. 2 2 .. ..

output
XXX ..X XXX XXXX .X.X .X.. .XXX .X... .XXXX .X... .X... XXXXX XXXXXXXXXX .. ..

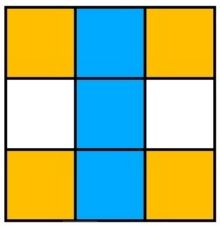
Note

Let's use (x, y) to describe the cell on x-th row and y-th column.

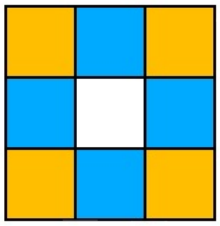
In the following pictures white, yellow, and blue cells stand for the cells that grow a sunflower, the cells lightning stroke, and the cells sunflower on which are removed, respectively.



In the first test case, one possible solution is to remove sunflowers on $(1, 2)$, $(2, 3)$ and $(3, 2)$.

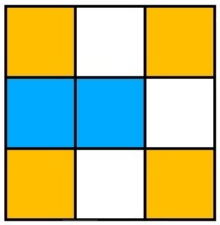


Another acceptable solution is to remove sunflowers on $(1, 2)$, $(2, 2)$ and $(3, 2)$.



This output is considered wrong because there are 2 simple paths between any pair of cells (there is a cycle). For example, there are 2 simple paths between $(1, 1)$ and $(3, 3)$.

1. $(1, 1) \to (1, 2) \to (1, 3) \to (2, 3) \to (3, 3)$
2. $(1, 1) \to (2, 1) \to (3, 1) \to (3, 2) \to (3, 3)$



This output is considered wrong because you can't walk from $(1, 1)$ to $(3, 3)$.

D. BFS Trees

time limit per test: 2.5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

We define a spanning tree of a graph to be a BFS tree *rooted at* vertex s if and only if for every node t the shortest distance between s and t in the graph is equal to the shortest distance between s and t in the spanning tree.

Given a graph, we define $f(x, y)$ to be the number of spanning trees of that graph that are BFS trees rooted at vertices x and y at the same time.

You are given an undirected connected graph with n vertices and m edges. Calculate $f(i, j)$ for all i, j by modulo $998\,244\,353$.

Input

The first line contains two integers n, m ($1 \leq n \leq 400$, $0 \leq m \leq 600$) — the number of vertices and the number of edges in the graph.

The i-th of the next m lines contains two integers a_i, b_i ($1 \leq a_i, b_i \leq n$, $a_i < b_i$), representing an edge connecting a_i and b_i.

It is guaranteed that all edges are distinct and the graph is connected.

Output

Print n lines, each consisting of n integers.

The integer printed in the row i and the column j should be $f(i, j) \bmod 998\,244\,353$.

Examples

input
4 4 1 2 2 3 3 4 1 4

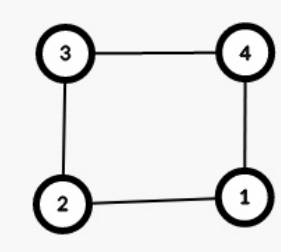
output
2 1 0 1 1 2 1 0 0 1 2 1 1 0 1 2

input
8 9 1 2 1 3 1 4 2 7 3 5 3 6 4 8 2 3 3 4

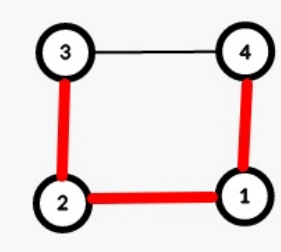
output
1 0 0 0 0 0 0 0 0 2 0 0 0 0 2 0 0 0 1 0 1 1 0 0 0 0 0 2 0 0 0 2 0 0 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 2 0 0 0 0 2 0 0 0 0 2 0 0 0 2

Note

The following picture describes the first example.



The tree with red edges is a BFS tree rooted at both 1 and 2.



Similarly, the BFS tree for other adjacent pairs of vertices can be generated in this way.

E. Qingshan and Daniel

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Qingshan and Daniel are going to play a card game. But it will be so boring if only two persons play this. So they will make n robots in total to play this game automatically. Robots made by Qingshan belong to the team 1, and robots made by Daniel belong to the team 2. Robot i belongs to team t_i. Before the game starts, a_i cards are given for robot i.

The rules for this card game are simple:

- Before the start, the robots are arranged in a circle in the order or their indices. The robots will discard cards in some order, in each step one robot discards a single card. When the game starts, robot 1 will discard one of its cards. After that, robots will follow the following rules:
- If robot i discards the card last, the nearest robot whose team is opposite from i's will discard the card next. In another word j will discard a card right after i, if and only if among all j that satisfy $t_i \neq t_j$, $dist(i, j)$ (definition is below) is minimum.
- The robot who has no cards should quit the game immediately. This robot won't be considered in the next steps.
- When no robot can discard the card next, the game ends.

We define the distance from robot x to robot y as $dist(x, y) = (y - x + n) \bmod n$. It is similar to the oriented distance on the circle.

For example, when $n = 5$, the distance from 1 to 3 is $dist(1, 3) = (3 - 1 + 5) \bmod 5 = 2$, the distance from 3 to 1 is $dist(3, 1) = (1 - 3 + 5) \bmod 5 = 3$.

Later, Qingshan finds out that it will take so much time to see how robots play. She wants to know the result as quickly as possible. You, as Qingshan's fan, are asked to calculate an array $[ans_1, ans_2, \ldots, ans_n]$ — ans_i is equal to the number of cards, that i-th robot will discard during the game. You need to hurry!

To avoid the large size of the input, the team and the number of cards of each robot will be generated in your code with some auxiliary arrays.

Input

The first line contains one integer n ($1 \le n \le 5 \cdot 10^6$) — the number of robots playing this game.

The second line contains one integer m ($1 \le m \le \min(n, 200\,000)$).

Each of the next m line contains four integers p_i, k_i, b_i, w_i ($1 \le p_i \le n$, $1 \le k_i \le 10^9 + 7$, $0 \le b_i, w_i < k_i$). It's guaranteed that $p_m = n$ and $p_{j-1} < p_j$ ($2 \le j \le m$).

Arrays a_j and t_j should be generated by the following pseudo code:

```
seed = 0
base = 0

function rnd():
    ret = seed
    seed = (seed * base + 233) mod 1000000007
    return ret

p[0] = 0
for i = 1 to m:
    seed = b[i]
    base = w[i]
    for j = p[i - 1] + 1 to p[i]:
        t[j] = (rnd() mod 2) + 1
        a[j] = (rnd() mod k[i]) + 1
```

Output

Print a single integer $\left(\prod_{i=1}^{n}((ans_i \oplus i^2) + 1)\right) \bmod 10^9 + 7$, where $\oplus$ denotes the bitwise XOR operation.

Examples

input
3 3 1 5 2 3 2 7 1 2 3 2 1 1
output
100

input
5000000 2 1919810 998244353 114514 19260817 5000000 233333333 623532 7175
output
800210675

input
1 1 1 1 0 0
output
1

Note

In the first test case $a = [5, 5, 1]$ and $t = [1, 2, 2]$.

The robot 1 discards the card first.

Then robot 2 discards the card next. Robot 3 doesn't discard the card next because $dist(1, 2) < dist(1, 3)$.

Then robot 1 discards the card next. Robot 3 doesn't discard the card next because $t_2 = t_3$.

If we write down the index of the robot who discards a card in time order, it will be the sequence $[1, 2, 1, 2, 1, 2, 1, 2]$. So robots 1, 2 and 3 discard 5, 5 and 0 cards, respectively. And the answer is $(((5 \oplus 1^2) + 1) \times ((5 \oplus 2^2) + 1) \times ((0 \oplus 3^2) + 1)) \bmod 10^9 + 7 = (5 \times 2 \times 10) \bmod 10^9 + 7 = 100$.

F. Squares

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n squares drawn from left to right on the floor. The i-th square has three integers p_i, a_i, b_i, written on it. The sequence $p_1, p_2, \ldots, p_n$ forms a permutation.

Each round you will start from the leftmost square 1 and jump to the right. If you are now on the i-th square, you can do one of the following two operations:

1. Jump to the $i+1$-th square and pay the cost a_i. If $i = n$, then you can end the round and pay the cost a_i.
2. Jump to the j-th square and pay the cost b_i, where j is the leftmost square that satisfies $j > i, p_j > p_i$. If there is no such j then you can end the round and pay the cost b_i.

There are q rounds in the game. To make the game more difficult, you need to maintain a square set S (initially it is empty). You **must** pass through these squares during the round (other squares can also be passed through). The square set S for the i-th round is obtained by adding or removing a square from the square set for the $(i-1)$-th round.

For each round find the minimum cost you should pay to end it.

Input

The first line contains two integers n, q ($1 \le n, q \le 2 \cdot 10^5$) — the number of squares and the number of rounds.

The second line contains n distinct integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$). It is guaranteed that the sequence $p_1, p_2, \ldots, p_n$ forms a permutation.

The third line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

The fourth line contains n integers $b_1, b_2, \ldots, b_n$ ($-10^9 \le b_i \le 10^9$).

Then q lines follow, i-th of them contains a single integer x_i ($1 \le x_i \le n$). If x_i was in the set S on the $(i-1)$-th round you should remove it, otherwise, you should add it.

Output

Print q lines, each of them should contain a single integer — the minimum cost you should pay to end the corresponding round.

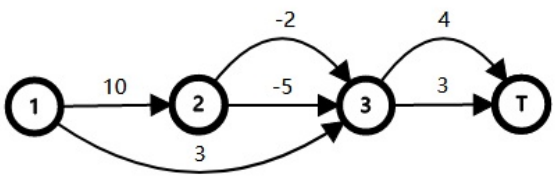
Examples

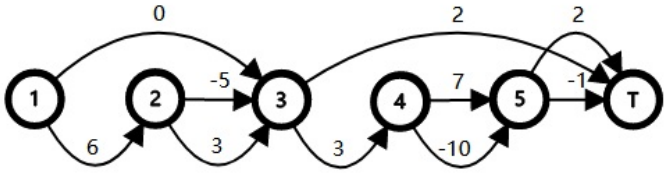
input
3 2 2 1 3 10 -5 4 3 -2 3 1 2
output
6 8

input
5 4 2 1 5 3 4 6 -5 3 -10 -1 0 3 2 7 2 1 2 3 2
output
-8 -7 -7 -8

Note

Let's consider the character T as the end of a round. Then we can draw two graphs for the first and the second test.





In the first round of the first test, the set that you must pass through is $\{1\}$. The path you can use is $1 \to 3 \to T$ and its cost is 6.

In the second round of the first test, the set that you must pass through is $\{1, 2\}$. The path you can use is $1 \to 2 \to 3 \to T$ and its cost is 8.
