Problem A. Queen Placement

Input file:	`standard input` or `input.txt`
Output file:	`standard output` or `output.txt`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Zakhar loves playing chess and programming. Therefore, more than anything in the world, Zakhar loves programming problems related to chess. Unfortunately, he has already solved all the classic chess problems, and now he has to come up with and solve new ones.

Zakhar has a chessboard, which is a square grid of m rows and m columns, as well as n queens. The rows of the grid are numbered with integers from 1 to m from top to bottom, and the columns from left to right.

Zakhar is trying to solve the problem of placing queens on the board: he needs to place the queens in such a way that no queen attacks another queen. Recall that a chess queen attacks all cells located in the same row, the same column, and on the same diagonals as the queen itself. Zakhar has n queens, and he plans to place the i-th queen in the cell with coordinates (x_i, y_i), where x_i is the column number and y_i is the row number. It is guaranteed that all the coordinates of the queens are pairwise distinct.

Unfortunately, Zakhar has not learned how to check the placement for correctness. He is interested in q segments of queens with numbers from l_i to r_i inclusive. Help Zakhar and determine for each of these segments whether there exists a pair of queens that attack each other. Formally, the placement of queens from the segment $[l_i, r_i]$ is considered correct if there are no two queens with numbers j and k such that $l_i \le j < k \le r_i$ and the j-th queen attacks the k-th queen.

Input

The first line contains two integers n and m ($2 \le n \le 200\,000$, $2 \le m \le 10^9$) — the number of queens and the size of the board, respectively.

The next n lines describe the positions of the queens. The i-th line contains two integers x_i and y_i ($1 \le x_i, y_i \le m$) — the coordinates of the cell where the i-th queen will be placed.

The following line contains one integer q ($1 \le q \le 10^6$) — the number of queries.

In the i-th of the next q lines, there are two integers l_i and r_i ($1 \le l_i \le r_i \le n$) — the boundaries of the segment for the i-th query.

It is guaranteed that all the coordinates of the queens are pairwise distinct.

Output

In q lines, output the answers to the queries.

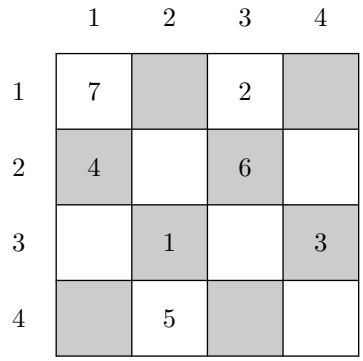
If the placement of queens from the segment $[l_i, r_i]$ is correct (i.e., there are no pairs of queens with numbers from the segment that attack each other), output "Yes"(without quotes) in the i-th line; otherwise, output "No"(without quotes).

Example

standard input	standard output
7 4 3 2 1 3 3 4 2 1 4 2 2 3 1 1 8 1 2 1 3 2 5 3 5 3 6 4 7 5 7 7 7	Yes No Yes Yes No No Yes Yes

Note

The picture shows a chessboard for an example. The cells indicate the numbers of the corresponding queens.



In the first query, queens 1 and 2 do not attack each other.

In the second query, queens 1 and 3 attack each other.

In the third query, queens 2 to 5 do not attack each other.

In the fourth query, queens 3 to 5 do not attack each other.

In the fifth query, queens 3 and 6, as well as 4 and 6, attack each other.

In the sixth query, queens 4 and 6, as well as 4 and 7, attack each other.

In the seventh query, queens 5 to 7 do not attack each other.

In the eighth query, there is only one queen present.

Scoring

The tests for this problem consist of seven groups. Points for each group are awarded only if all tests of the group and all tests of some of the previous groups are passed. Note that passing the tests from the statement is not required for some groups.

Group	Points	Constraints			Required	Comment
		n	m	q		
0	0	–	–	–	–	Samples
1	7	–	–	–	–	For all queens $x_i = 1$
2	12	$n \leq 100$	$m \leq 100\,000$	$q \leq 100$	0	
3	16	$n \leq 5\,000$	$m \leq 100\,000$	$q \leq 5\,000$	0, 2	
4	23	$n \leq 100\,000$	$m \leq 100\,000$	$q \leq 100\,000$	0, 2, 3	
5	17	–	$m \leq 100\,000$	–	–	In all queries $l_i = 1$
6	10	–	–	–	5	In all queries $l_i = 1$
7	15	–	–	–	0 – 6	

Problem B. History

Input file:	`standard input` or `input.txt`
Output file:	`standard output` or `output.txt`
Time limit:	1 second
Memory limit:	1024 megabytes

Traditionally, few students attended history classes, so the teacher decided to change the system and introduce mandatory group projects for students, which can only be completed in person during classes. Now all n students, numbered from 1 to n, attend the classes. The teacher knows that the knowledge level of the i-th student is a_i.

To complete the group project, students need to form pairs. To make this not too simple, the teacher has set the following requirement: two people with numbers $i \neq j$ can be paired if either $a_i + a_j = S$, or $a_i \oplus a_j = X$, where $\oplus$ denotes the bitwise exclusive OR (XOR) operation.

Help the students determine if it is possible to form pairs in such a way that the teacher's requirements are satisfied.

Input

The first line contains three integers n, S, and X ($2 \le n \le 500\,000$, $0 \le S, X < 2^{30}$, n is **even**) — the number of students and the required sum or XOR values in a pair, respectively.

The next line contains n integers $a_1,\ a_2,\ \ldots,\ a_n$ ($0 \le a_i < 2^{30}$) — the knowledge levels of the students.

Output

If it is possible to form pairs, then in the first line output "Yes"(without quotes).

In the next $\frac{n}{2}$ lines, output the formed pairs themselves, where the i-th line should contain two integers c_i, d_i ($1 \le c_i, d_i \le n, c_i \neq d_i$), indicating that students with numbers c_i and d_i should be paired together. Each student number, which is described by a number from 1 to n, must appear among the pairs exactly once.

If it is not possible to form pairs that satisfy all conditions, output "No"(without quotes) in a single line.

Examples

standard input	standard output
6 7 0 1 2 9 9 5 6	Yes 1 6 2 5 4 3
4 6 2 1 5 2 3	No

Note

We will show that the pairing in the first example is correct.

- $a_1 + a_6 = 1 + 6 = 7$
- $a_2 + a_5 = 2 + 5 = 7$
- $a_4 \oplus a_3 = 9 \oplus 9 = 0$

Scoring

The tests for this problem consist of eight groups. Points for each group are awarded only if all tests of the group and all tests of some of the previous groups are passed. Note that passing the tests from

the statement is not required for some groups. **Offline checking** means that the results of testing your solution on this group will only be available after the competition ends.

Group	Points	Constraints	Required	Comment
		n		
0	0	–	–	Samples
1	9	$n \le 20$	0	
2	15	$n \le 100$	0 – 1	
3	7	–	–	$S \le 1,\ a_i \ge 1$
4	17	–	–	$X = 0$
5	10	$n \le 2\,000$	0 – 2	
6	21	–	–	All numbers in a are distinct
7	11	$n \le 100\,000$	0 – 2, 5	
8	10	–	0 – 7	**Offline checking**

Problem C. Jelly Candies

Input file:	`standard input` or `input.txt`
Output file:	`standard output` or `output.txt`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Petya loves jelly candies very much. There are n stores selling jelly candies, and the i-th store sells an infinite number of jelly candies with tastiness a_i. Petya only eats jelly candies, so his friend Sasha is responsibly monitoring his diet.

Every day, there are two types of events:

1. In the stores numbered from l to r, the tastiness of the jelly candies increases by x:

$$a_i := a_i + x \quad \text{for all } i \in [l, r].$$

2. Purchase of jelly candies. Petya examines the stores in the segment numbered from l to r in order. In each store, Petya can either buy exactly one jelly candy or skip and buy nothing.

   Let's say Petya bought jelly candies from stores with numbers $l \le i_1 < i_2 < \ldots < i_m \le r$. We consider the tastiness of the jelly candies in the order of purchase, denoting $b_j := a_{i_j}$. Among all possible ways to purchase jelly candies, Petya chooses the one where the sequence b is lexicographically maximal.

   After the purchase, Sasha wants to know the value of b_k (that is, the tastiness of the k-th jelly candy that Petya will buy), or to find out that the length of the chosen sequence is less than k.

Help Sasha figure out Petya's diet!

Recall that the sequence $(b_1, b_2, \ldots, b_m)$ is lexicographically greater than the sequence $(c_1, c_2, \ldots, c_k)$ if there exists a position i such that $b_i > c_i$ and for all $j < i$, $b_j = c_j$, or if $(b_1, \ldots, b_k)$ is a prefix of $(c_1, \ldots, c_k)$ and $m > k$.

Input

The first line contains two integers n and q ($1 \le n, q \le 500\,000$) — the number of stores and the number of days.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the initial tastiness of the jelly candies.

The next q lines contain descriptions of the queries. First, the number t_i is given ($1 \le t \le 2$) — the type of the i-th query.

If $t = 1$, then three numbers l_i, r_i, and x_i follow ($1 \le l_i \le r_i \le n, 1 \le x_i \le 10^9$) — the increase in tastiness of the jelly candies in stores numbered $l_i, l_i + 1, \ldots, r_i$ by the number x_i.

If $t = 2$, then three numbers l_i, r_i, and k_i follow ($1 \le l_i \le r_i \le n, 1 \le k_i \le n$) — Petya examines the jelly candies in stores $l_i, l_i + 1, \ldots, r_i$, and Sasha wants to know what tastiness the k_i-th jelly candy Petya will buy has.

Output

For each query of the second type, in case Petya bought fewer than k jelly candies, print -1. Otherwise, print the tastiness of the k-th jelly candy Petya bought.

Examples

standard input	standard output
5 5 1 3 2 3 2 2 1 5 3 2 3 5 1 1 3 3 2 2 2 5 2 2 1 5 4	2 3 3 -1
5 6 5 2 5 5 2 1 3 5 10 2 2 2 1 2 3 5 1 1 2 3 11 2 3 4 1 2 2 4 1	2 15 26 26

Note

Consider the first example.

The initial tastiness of the jelly candies is $[1, 3, 2, 3, 2]$.

In the first query, Petya buys jelly candies from the entire array. The lexicographically maximal sequence b that he can obtain is $[3, 3, 2]$. We are asked for the third element of this sequence, which is two.

In the second query, Petya buys jelly candies from the segment $[3, 5]$. The lexicographically maximal sequence b that he can obtain is $[3, 2]$. We are asked for the first element of this sequence, which is three.

Then the tastiness in the third store increases by 2, and the tastiness becomes $[1, 3, 4, 3, 2]$.

Next, Petya buys jelly candies from the segment $[2, 5]$. The lexicographically maximal sequence b that he can obtain is $[4, 3, 2]$. We are asked for the second element of this sequence, which is three.

In the last query, Petya buys jelly candies from the entire array. The lexicographically maximal sequence b that he can obtain is $[4, 3, 2]$. We are asked for the fourth element of the sequence, and we output -1 because it does not exist.

Scoring

The tests for this problem consist of seven groups. Points for each group are awarded only if all tests in the group and all tests in some of the previous groups are passed. Note that passing the tests from the statement is not required for some groups. **Offline checking** means that the results of testing your solution on this group will only be available after the competition ends.

Let m be the number of jelly candies that Petya will buy in the current query.

Group	Points	Constraints		Required	Comment
		n	q		
0	0	–	–	–	Samples
1	8	$n \le 100$	$q \le 100$	0	
2	16	$n \le 300\,000$	$q \le 300\,000$	0	It is guaranteed that in the second type queries $k \le 50$
3	15	$n \le 300\,000$	$q \le 300\,000$		No changes, in the second type queries $k \in \{m, m+1\}$
4	21	–	–	3	No changes
5	14	$n \le 400\,000$	$q \le 400\,000$	3	In the second type queries $k \in \{m, m+1\}$
6	11	$n \le 300\,000$	$q \le 300\,000$	0, 1, 2, 3	
7	15	–	–	0 – 6	**Offline checking**

Problem D. Rectangular Apartment

Input file:	`standard input` or `input.txt`
Output file:	`standard output` or `output.txt`
Time limit:	1 second
Memory limit:	1024 megabytes

Recently, the turtle rented an apartment. By a happy coincidence, it turned out to be rectangular in shape. The apartment is divided into $n \times m$ squares of the same size: n rows with m squares in each. The rows are numbered from top to bottom, and the columns from left to right. We denote the square in the i-th row and j-th column as (i, j).

In some places in the apartment, there is furniture. The description of the apartment is given by a matrix a of size $n \times m$:

- If $a_{i,j} = \texttt{\#}$, then the square (i, j) is occupied by furniture.
- If $a_{i,j} = \texttt{.}$, then the square (i, j) is free.

The turtle has been training hard and has learned to move not only to the right or down but also up. To consolidate her success in mastering this new movement, she decided to do exercises every morning. The exercise is described by a string s and proceeds as follows:

1. The turtle stands in the square (i, j).
2. Then for each i from 1 to $|s|$, the turtle moves to another square. Suppose she is currently at (x, y), then:
   - If $s_i = \texttt{D}$, the turtle moves to $(x + 1, y)$.
   - If $s_i = \texttt{R}$, the turtle moves to $(x, y + 1)$.
   - If $s_i = \texttt{U}$, the turtle moves to $(x - 1, y)$.

Naturally, during the exercise, the turtle cannot go outside the boundaries of the apartment or stand in a square occupied by furniture. Thus, if at any moment the turtle tries to move to a square that does not exist or is occupied by furniture, the exercise fails. The square from which the turtle starts the exercise must also be free.

Help the turtle find the number of squares (i, j) from which she can complete her exercise fully.

Input

The first line contains two integers n and m ($2 \le n, m \le 500$) — the dimensions of the apartment.

The second line contains the string s ($1 \le |s| \le 2nm$, $s_i \in \{\texttt{D}, \texttt{R}, \texttt{U}\}$) — the description of the exercise.

The i-th of the following n lines contains $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$ ($a_{i,j} \in \{\texttt{\#}, \texttt{.}\}$) — the description of the turtle's apartment.

Output

Output a single number — the count of squares (i, j) from which the turtle can complete her exercise fully.

Examples

standard input	standard output
5 6 RDUUR .#.... .....# .#.#.. ....#....	3
4 2 RR	0

Note

In the first example, the turtle can complete the exercise starting from the squares $(2, 2)$, $(2, 4)$, and $(4, 4)$.

If the turtle starts the exercise in the square $(2, 2)$, her path looks like this: $(2, 2) \to (2, 3) \to (3, 3) \to (2, 3) \to (1, 3) \to (1, 4)$.

Scoring

The tests for this problem consist of five groups. Points for each group are awarded only if all tests of the group and all tests of some of the previous groups are passed. Note that passing the tests from the statement is not required for some groups.

Group	Points	Constraints	Required	Comments
0	0	–	–	Samples
1	17	$n, m \leq 50$	0	
2	14	$s_i = \texttt{R}$	–	
3	19	$s_i \in \{\texttt{D}, \texttt{U}\}$	–	
4	23	$s_i \in \{\texttt{R}, \texttt{D}\}$	2	
5	27	–	0 – 4	

Problem E. Simple Problem

Input file:	`standard input` or `input.txt`
Output file:	`standard output` or `output.txt`
Time limit:	1 second
Memory limit:	1024 megabytes

Given an undirected tree with n vertices. Each vertex v has a non-negative integer a_v recorded ($0 \le a_v < 2^k$).

A set of vertices is called good if the bitwise OR of the values a of the vertices in this set equals $2^k - 1$.

The cost of a set is defined as the maximum of the pairwise distances between the vertices in the set, where the distance between vertices is the number of edges on the unique simple path between them.

You need to find the minimum cost of a good set or state that such a set does not exist.

Input

The first line contains two integers n and k ($2 \le n \le 100\,000$, $1 \le k \le 20$) — the number of vertices in the tree and the number k, respectively.

The second line contains n integers a_i ($0 \le a_i < 2^k$) — the values of the vertices.

The next $n - 1$ lines describe the edges of the tree.

The i-th of them contains two integers v_i and u_i ($1 \le v_i, u_i \le n$) — the indices of the vertices connected by the i-th edge.

Output

If there does not exist a good set, output -1. Otherwise, output the minimum cost of a good set in a single line.

Examples

standard input	standard output
5 3 1 2 6 0 4 1 2 2 3 1 4 3 5	2
3 3 0 1 2 1 2 2 3	-1

Note

In the first example, you can choose the set of vertices $\{1, 2, 3\}$.

In the second example, the maximum OR that can be obtained is 3.

Scoring

The tests for this problem consist of nine groups. Points for each group are awarded only if all tests of the group and all tests of some previous groups are passed. Note that passing the tests from the statement is not required for some groups. **Offline checking** means that the results of testing your solution on this group will only be available after the competition ends.

Group	Points	Constraints		Required	Comment
		n	k		
0	0	–	–	–	Samples
1	12	$n \le 15$	–	0	
2	9	$n \le 1\,000$	–	–	$v_i = i$, $u_i = i + 1$
3	14	–	–	2	$v_i = i$, $u_i = i + 1$
4	6	–	$k = 1$	–	–
5	10	$n \le 1\,000$	$k = 2$	–	–
6	12	–	$k = 2$	5	–
7	9	$n \le 100$	$k \le 5$	0	–
8	16	$n \le 1\,000$	–	0 – 2, 5, 7	–
9	12	–	–	0 – 8	**Offline checking**

Problem F. Minimums on Arcs

Input file:	`standard input` or `input.txt`
Output file:	`standard output` or `output.txt`
Time limit:	3 seconds
Memory limit:	1024 megabytes

This problem can only be solved in the C++ programming language.

On a circle, there are n distinct numbers from 1 to n, with the number p_i written at the i-th vertex when traversing counterclockwise. For each pair of values x and y, there are two ways to get from x to y — going clockwise or counterclockwise. In this problem, n is always odd, so these two paths will have different lengths.

Let $f(x, y)$ be the minimum value we encounter on the way if we travel from number x to number y via the shortest path.

For example, if $p = [1, 5, 3, 2, 7, 4, 6]$, then the values 5 and 7 are connected by two paths, and the values of the vertices on these paths are, respectively, $[5, 3, 2, 7]$ and $[5, 1, 6, 4, 7]$. The shortest path is the first one mentioned, so $f(5, 7) = \min\{5, 3, 2, 7\} = 2$.

We say that two permutations p and q are *equivalent* if the value of the function f for these permutations is the same for all x and y. You are given a permutation p, and you need to guess any permutation q that is equivalent to it by making queries to the function $f(x, y)$.

Solution Format

This is an unusual problem. It has a testing format with a grader, where you need to implement only the function `guess` with the solution. This function will be called by the testing program of the jury (the grader), and the returned value of the function will be accepted as the solution to the problem.

In particular, this means that the code you submit **must not input or output any data**. Your code **must not** contain a function `main`. If necessary, you can implement any number of helper functions, structures, classes, and global variables, but all the code of your solution must be in one file.

You must implement the following function:

```
std::vector<int> guess(int n);
```

The function `guess` takes as input the number n — the length of the permutation.

In the implementation of the function `guess`, you can use the function `min_value`, which is implemented by the grader. This function takes two values x and y as input and returns $f(x, y)$ as the result. If an incorrect query is made or the number of queries is exhausted, the program will automatically terminate.

To access the function `min_value` in your solution, the first line of your code must include the header file with the following line:

```
#include "checkpoint.h"
```

The definition of the function `min_value` in the header file is as follows:

```
int min_value(int x, int y);
```

All parameters (values x, y, as well as the permutation q you return) are given in **1-indexing**.

Your function `guess` must determine the unknown permutation p, up to the specified equivalence, using calls to the function `min_value`. That is, if the jury has chosen the permutation p, and the function `guess` returns any permutation q that is equivalent to it, then that answer is considered correct.

During one run of the grader, **the jury may make several calls to the function** `guess`, in which case the function `min_value` will return the values $f(i, j)$ defined for the currently chosen permutation p within the specific call to the function `guess`.

The grader is not adaptive, meaning that the permutations p are fixed in advance and do not depend

on the implementation of the function `guess`.

Testing

You are provided with a solution template `checkpoint.cpp`, as well as a header file `checkpoint.h`, containing the definitions of the functions `min_value` and `guess`.

For convenience in testing, you are provided with a grader — the file `grader.cpp`. This file implements input reading from the standard input stream, calls the function `guess`, and outputs the returned value of the function `guess` to the standard output stream. In the testing system, these grader files may differ.

To compile your code `checkpoint.cpp` in C++, use the command

```
g++ -std=c++20 grader.cpp checkpoint.cpp -o grader
```

After executing this command, an executable file named `grader` or `grader.exe` will be created, depending on your operating system. You can run this file to input tests in the specified format.

If you encounter difficulties compiling via commands, for local testing, you can copy the implementation of the function `guess` into the file `grader.cpp` and run the file `grader.cpp`. However, before submitting your solution to the testing system, you need to leave only the implementation of the function `guess`, remembering to include the header file at the beginning of the code.

Input

The grader reads the test in the following format:

The first line contains the number t ($1 \le t \le 30\,000$) — the number of input data sets.

The first line of each input data set contains an **odd** number n ($1 \le n \le 30\,000$) — the length of the permutation.

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

Output

The grader outputs the results of the function `guess` — the guessed permutation for each input data set.

In the file `grader.cpp`, there is a variable `verbose`, which is initially set to 0. By increasing its value, the grader will provide more detailed information about your solution and its queries.

Example

standard input	standard output
2 3 1 2 3 5 1 4 2 3 5	queries count => 3 queries count => 10

Scoring

Tests for this problem consist of four groups.

Points for each of the first three groups are awarded only if all tests in the group are passed and all tests in some of the previous groups. The score for the last group equals the minimum of the scores obtained for each test in the fourth group.

For each test, let N be the sum of n across all input data sets, and Q be the limit on the total number of calls to the function `min_value` across all input data sets.

Group	Points	Constraints		Required	Comment
		N	Q		
0	0	–	$Q = 4950$	–	Samples
1	10	$N \leq 100$	$Q = 4950$		$p_{2i-1} = \frac{n-1}{2} + i$ for $1 \leq i \leq \frac{n+1}{2}$
2	20	$N \leq 100$	$Q = 4950$	0, 1	
3	20	$N \leq 1\,000$	$Q = 100\,000$	0, 1, 2	
4	60	$N \leq 30\,000$	$Q = 1\,000\,000$	–	

The last group is scored according to the formula. If you made x queries in the test, your score will be:

$$\text{score} = \min\left(60,\ \left\lfloor 95 \cdot \left(1 - \frac{x}{10^6}\right)\right\rfloor\right)$$

Your score for the last group equals the minimum of the scores obtained on the tests in the fourth group.

Problem G. «Titanic»

Input file:	`standard input` or `input.txt`
Output file:	`standard output` or `output.txt`
Time limit:	1 second
Memory limit:	1024 megabytes

Andrey loves slot machines very much, and this time he found a new one— the «Titanic» slot machine. It has n lifeboats located on a plane, with the i-th lifeboat positioned at the coordinates (x_i, y_i). There is also a rescue boat that, after starting the slot machine, begins to move at a constant speed from point A to point B.

The rescue boat has a hook that can shoot perpendicularly to the boat's movement in both directions. The hook shoots at an incredibly fast speed, and if a lifeboat is in the path of the hook, it catches it and starts pulling it towards the boat. The hook retracts slowly—if the distance between your boat and the lifeboat at the moment of shooting and catching was d, the rescue boat will travel a distance of d while the hook is retracting. While the hook is retracting, you cannot shoot to save other lifeboats.

Initially, your score of saved lifeboats is 0. The moment the hook is fully retracted, your score increases by 1, and you can shoot again. When the boat reaches point B, the game ends. If the hook manages to retract exactly at that moment, the saved lifeboat is counted; if the hook does not manage to retract fully, the lifeboat is not considered saved, and the score does not increase.

Andrey wondered if there was a catch in the machines and wanted to calculate the maximum score he could achieve. Unfortunately, Andrey decided to find this out through trial and error, so help him—calculate the maximum score that can be achieved in the machine before Andrey spends all your tokens!

Input

Each test consists of several sets of input data. The first line contains a single integer t ($1 \le t \le 10\,000$) — the number of input data sets. The following describes the input data sets.

The first line of each data set contains an integer n ($1 \le n \le 200\,000$) — the number of lifeboats in the machine.

The next n lines follow. The i-th of them contains two integers x_i, y_i ($-10^9 \le x_i, y_i \le 10^9$) — the coordinates of the i-th lifeboat.

In the penultimate line of each data set, there are two integers a_x, a_y ($-10^9 \le a_x, a_y \le 10^9$) — the coordinates of point A.

In the last line of each data set, there are two integers b_x, b_y ($-10^9 \le b_x, b_y \le 10^9$) — the coordinates of point B.

It is guaranteed that points A and B do not coincide. It is guaranteed that all lifeboats in each input data set are located at pairwise distinct points.

Let N denote the sum of n across all input data sets. It is guaranteed that N does not exceed 200 000.

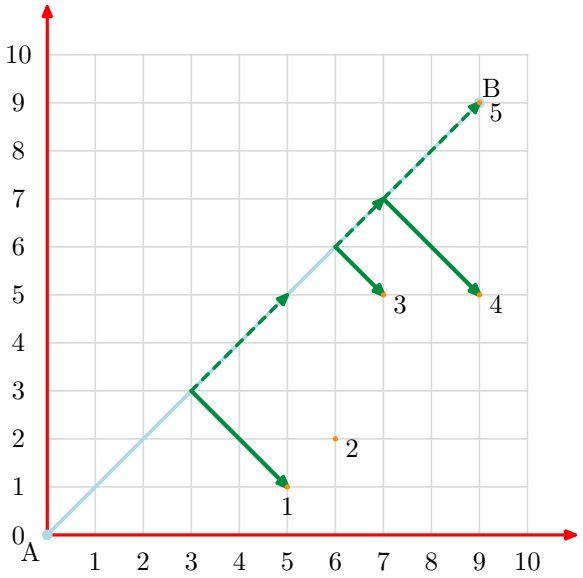
Output

For each set of input data, output the maximum score that can be achieved in the described «Titanic» slot machine.

Example

standard input	standard output
2 5 5 1 6 2 7 5 9 5 9 9 0 0 9 9 2 0 1 -1 5 0 0 -5 0	4 1

Note



In the first test case, our boat is moving from point $(0, 0)$ to point $(9, 9)$. We pick up the 1st lifeboat while at $(3, 3)$ and pull it until we reach $(5, 5)$. Next, at $(6, 6)$, we catch the 3rd lifeboat and pull it until we reach $(7, 7)$. Then we immediately catch the 4th lifeboat and pull it to $(9, 9)$. At the final point, we pick up the 5th lifeboat and catch it immediately. After that, the machine stops working, and we have a score of 4.

In the second test case, our boat is moving from point $(0, 0)$ to point $(-5, 0)$. At point $(0, 0)$, we catch the 1st lifeboat and pull it towards us until we reach $(-1, 0)$; after that, we can try to catch the 2nd lifeboat, but we won't have enough distance to pull it towards us—the machine will end the game earlier, so the final score is 1.

Scoring

The tests for this problem consist of five groups. Points for each group are awarded only if all tests of the group and all tests of some of the previous groups are passed. Note that passing the tests from the statement is not required for some groups.

Group	Points	Constraints	Required	Comment
		N		
0	0	–	–	Samples
1	21	$N \le 4000$	–	$a_y = b_y = 0$
2	20	$N \le 4000$	0, 1	
3	32	$N \le 100\,000$	1	$a_y = b_y = 0$
4	14	$N \le 100\,000$	0–3	
5	13	–	0–4	

Problem H. Playing Go

Input file:	`standard input` or `input.txt`
Output file:	`standard output` or `output.txt`
Time limit:	4 seconds
Memory limit:	1024 megabytes

Given n points on a plane. Some of them are white, and some are black. No two points coincide. No three points lie on the same line.

You are required to move one of the white points a distance of $\leq r$ in such a way as to maximize the area of the convex hull of the set of points that you will have after this operation.

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \leq t \leq 1\,000$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and r ($3 \leq n \leq 2\,000$, $1 \leq r \leq 10^9$) — the number of points and the maximum distance for moving a point.

The next n lines describe the points. The i-th line contains three integers x_i, y_i, and c_i ($-10^9 \leq x_i, y_i \leq 10^9$, $1 \leq c_i \leq 2$) — the coordinates and color of point i (1 — white, 2 — black).

It is guaranteed that the sum of n over all test cases in each test does not exceed $2\,000$.

Output

For each test case, output a single number on a separate line — the maximum area of the convex hull of this set of points, if any of the white points can be moved by no more than r.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}.

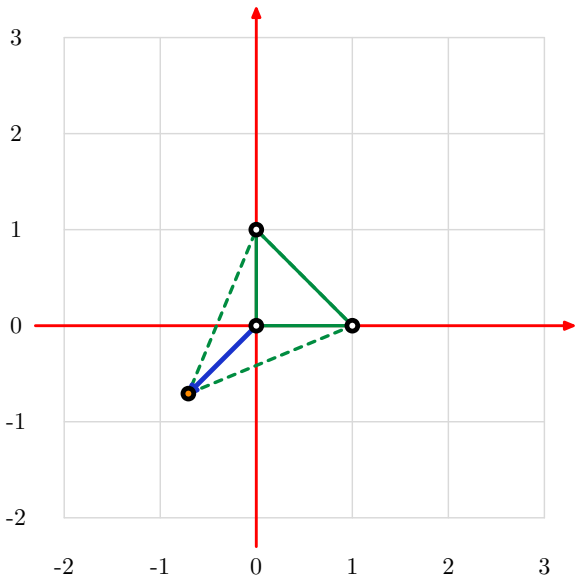
Formally, let your answer be a, and the jury's answer be b. Your answer will be accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \leq 10^{-6}$.

Examples

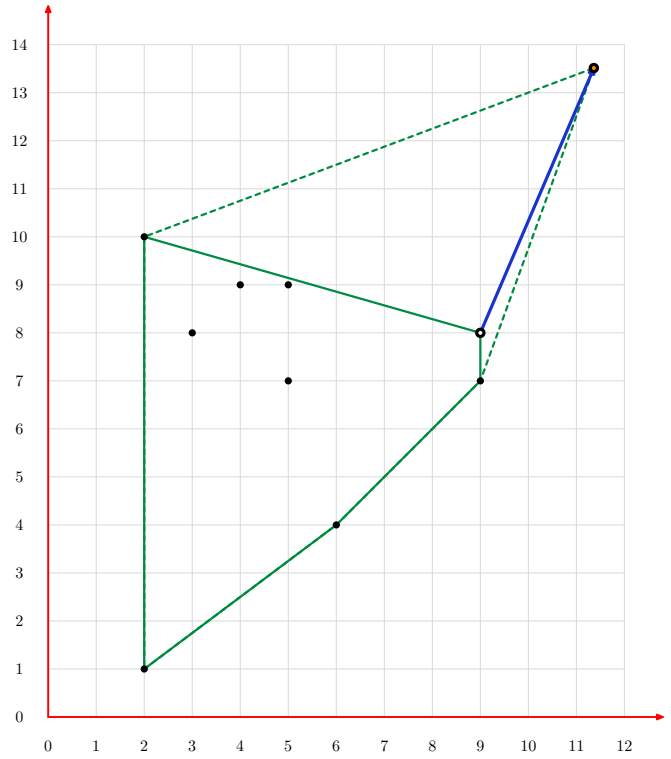
standard input	standard output
1 3 1 0 0 1 1 0 1 0 1 1	1.20710678118655
1 9 6 2 10 2 5 7 2 9 8 1 6 4 2 4 9 2 2 1 2 9 7 2 5 9 2 3 8 2	59.34731931759172
1 9 1 8 7 1 1 9 1 3 9 1 4 2 1 7 4 1 10 5 1 3 7 1 4 4 1 7 6 1	37.42442890089805

Note

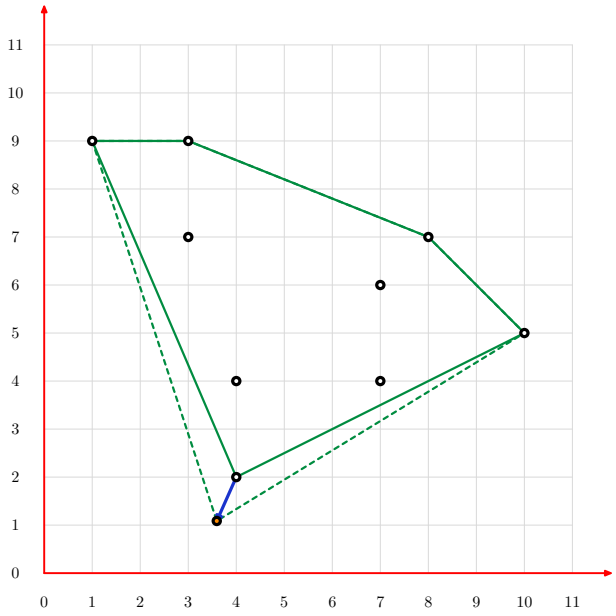
Below are images for the examples from the statement. The dashed line indicates the edges of the new convex hull, the solid line indicates the edges of the old convex hull and the common edges. The blue arrow shows the optimal movement of the white point.



Example 1



Example 2



Example 3

Scoring

The tests for this problem consist of nine groups. Points for each group are awarded only if all tests in the group and all tests in some of the previous groups are passed. Note that passing the tests from the statement may not be required for some groups. **Offline checking** means that the results of testing your solution on this group will only be available after the competition ends.

Let $\sum n$ be the sum of n over all test cases of this test.

Group	Points	Constraints $\sum n$	Required	Comment
0	0	–	–	Samples
1	11	$\sum n \le 3$	0	
2	18	$\sum n \le 30$	0, 1	
3	6	$\sum n \le 50$	0, 1, 2	
4	13	$\sum n \le 200$	–	All points on the convex hull are black
5	14	$\sum n \le 200$	0 – 4	
6	8	$\sum n \le 500$	4	All points on the convex hull are black
7	7	$\sum n \le 500$	0 – 6	
8	12	$\sum n \le 2000$	4, 6	All points on the convex hull are black. **Offline checking**
9	11	$\sum n \le 2000$	0 – 8	**Offline checking**

Problem I. Moving

Input file:	`standard input` or `input.txt`
Output file:	`standard output` or `output.txt`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Consider a city represented as a table with n rows and m columns. At the intersection of each row and each column, there is a house.

In house (i, j) at the intersection of row i and column j, there initially lived $a_{(i,j)}$ people. The following year, each person moved from their house to some other house (or possibly stayed in the same house). It is known that the following year, house (i, j) had $b_{(i,j)}$ people living in it.

You need to output the minimum number x such that people could have moved in such a way that the distance between each person's original and final house did not exceed x. The distance between cells (x_1, y_1) and (x_2, y_2) is defined as $|x_1 - x_2| + |y_1 - y_2|$.

Input

Each test consists of several test cases. The first line contains one integer t ($1 \le t \le 100$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n \le 3$, $1 \le m \le 100\,000$) — the number of rows and columns in the table, respectively.

The next n lines describe the initial number of residents in the houses. The i-th line contains m integers $a_{(i,1)}, a_{(i,2)}, \ldots a_{(i,m)}$ ($0 \le a_{(i,j)} \le 10^9$) — the initial number of people in each house.

The following n lines describe the number of residents in the houses after the move. The i-th line contains m integers $b_{(i,1)}, b_{(i,2)}, \ldots b_{(i,m)}$ ($0 \le b_{(i,j)} \le 10^9$) — the number of people in each house after the move.

It is guaranteed that the sum of the values $a_{(i,j)}$ equals the sum of the values $b_{(i,j)}$.

Let M be the sum of m over all test cases. It is guaranteed that M does not exceed $100\,000$.

Output

For each test case, output the minimum number x such that people could have moved so that the distance between each person's original and final house did not exceed x.

Example

standard input	standard output
1 2 5 0 4 0 4 0 0 0 0 0 0 1 1 1 1 1 0 1 1 1 0	2

Note

In the example, people from house $(1, 2)$ move to $(1, 1), (1, 2), (1, 3), (2, 2)$, and people from house $(1, 4)$ move to $(1, 4), (1, 5), (2, 3), (2, 4)$. The maximum distance is two.

Scoring

The tests for this problem consist of twelve groups. Points for each group are awarded only if all tests in the group and all tests in some of the previous groups are passed. Note that passing the tests from the statement may not be required for some groups. **Offline checking** means that the results of testing your solution on this group will only be available after the competition ends.

Let S be the total number of people in the city (the sum of the elements of either table), A be the number of non-zero values a_{ij}, and B be the number of non-zero values b_{ij}.

Group	Points	Constraints				Required	Comment
		M	S	A, B	n		
0	0	–	–	–	–	–	Samples
1	8	–	$S \le 7$	–	–	–	
2	9	–	$S \le 50$	–	–	1	
3	8	–	–	$A, B \le 13$	–	1	
4	7	–	–	$A \le 13$	–	1, 3	
5	6	$m \le 50$, $M \le 5\,000$	–	–	–	–	
6	10	$M \le 5\,000$	–	–	–	5	
7	8	$M \le 50\,000$	–	–	$n \le 1$	—	
8	11	$M \le 50\,000$	–	–	$n \le 2$	7	
9	5	$M \le 50\,000$	–	–	–	—	The answer does not exceed 2
10	6	$M \le 50\,000$	–	–	–	9	The answer does not exceed 3
11	12	$M \le 50\,000$	–	–	–	5–10	
12	10	—	–	–	–	1–11	**Offline checking**