Problem A. Astana Hard Metro

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

The Astana metro was built in an unusual way.

In the first year after the city was founded, one station was built in the city center, with number 1. In each following year v, the mayor chose one of the already passed years p, where $1 \le p < v$, and ordered a new station to be built with a track to the station that appeared in year p. Trains on this track can move only from the new station toward the center.

Thus, the metro always remained a tree, and from any station one can move only towards the central station, while from the central station there is nowhere to go. Stations that cannot be reached from other stations are called terminal.

Historically, the transfer complexity of station v is defined as the number of stations from which one can get to v by **one** track. However, the transfer complexity of a terminal station is equal to 1.

It has been n years since Astana was founded, so the metro already has n stations.

This year, the mayor announced that in all following years, the station to which a new track will be attached will be chosen uniformly at random among all stations already built.

Pupa is planning a tourist visit to Astana in m years, when there will already be $n + m$ stations. The mayor will build the new m stations in this random way. According to the sightseeing plan, Pupa will make one metro ride: she will choose uniformly at random one terminal station, travel towards the center, and get off at the central station. Pupa will also buy a paper guidebook with all stations and their transfer complexities. She will cross out every station she visits (including both the first and the last station on the path) from her guidebook.

After the ride, Pupa plans to multiply the transfer complexities of all stations that were not crossed out from the guidebook — this number will be the result of her visit. If she crosses all the stations, result of her visit will be equal to 1.

Help Pupa find the expected value of the result of her visit, modulo 998 244 353.

Input

The first line contains two integers n, m ($2 \le n \le 200, 1 \le m \le 200$) — current number of stations in the metro, and the number of years after which Pupa will come to Astana.

The second line contains the integers $p_2, \ldots, p_n$ ($1 \le p_i < i$) — the tracks destinations for all stations from 2 to n.

Output

Print a single number, the expected value of the result of Pupa visit after m years. Print the answer modulo 998 244 353.

Examples

standard input	standard output
4 1 1 1 1	1
12 2 1 1 2 2 3 3 4 5 6 4 5	413192725
2 77 1	253554867

Problem B. Beggar Robin

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

There are n people standing in a row. The i-th person has a_i coins.

Beggar Robin wants to help everyone become more equal. In one operation, he chooses two people and distributes their money equally between them. Formally, he chooses two positions i and j such that $1 \le i \le j \le n$, computes $x = \frac{a_i + a_j}{2}$, and then sets $a_i := x$ and $a_j := x$.

But there is one problem: coins are indivisible. If $a_i + a_j$ is odd, Robin cannot split one coin into two halves. In that case, Robin happily takes this one coin as his salary, and distributes the rest equally.

Robin is very hungry, so he wants to earn his salary (i.e. to take at least one coin) as soon as possible.

For an array $a_1, a_2, \ldots, a_n$ of positive integers, define $f(a_1, a_2, \ldots, a_n)$ as the minimum number of operations Robin needs to perform to earn his salary. If Robin can never earn his salary, then $f(a_1, a_2, \ldots, a_n) = 0$.

You are given an array $a_1, a_2, \ldots, a_n$. Your task is to calculate

$$\sum_{l=1}^{n} \sum_{r=l}^{n} f(a_l, \ldots, a_r)^2$$

modulo 998 244 353.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the numbers of coins the people have.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the sum of squared values of f over all subarrays of the given array, modulo 998 244 353.

Example

standard input	standard output
4 3 2 18 2 10 1 2 3 4 5 6 7 8 9 10 5 20 36 100 36 20 2 1000000000 999999999	25 45 174 1

Problem C. Cost of Conflicts

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	1024 megabytes

After the fall of the Empire, the galaxy fell into dark times of economic inequality and endless military conflicts. There are n planets in the galaxy. All protected hyperspace routes between planets, which were once used by civilian and transport ships, were forgotten.

The leaders of the Second Foundation, a secret organization whose goal is to restore the Empire, managed to make all planets agree to a truce. Their next step is to restore protected routes between the planets, so that it becomes possible to travel from any planet to any other planet.

For the convenience of planning, the Second Foundation numbered the planets from 1 to n in the order of increasing economic level. More precisely, the economic level of planet v is an integer a_v, and it is guaranteed that $a_1 < a_2 < \ldots < a_n$.

Before building new protected routes, the Foundation estimated the danger of every possible route. At first, the danger of a route was based only on the economic inequality between its endpoints. More precisely, for two different planets u and v, the initial danger of the route between them was equal to

$$\frac{\max(a_u, a_v)}{\min(a_u, a_v)}.$$

Since the planets are numbered in increasing order of economic level, for $u < v$ this value is equal to $\frac{a_v}{a_u}$.

However, one should not forget about the violent battles that took place between the planets after the fall of the Empire. In total, there were q military conflicts.

In the i-th conflict, the coalition of planets with numbers from L_i^1 to R_i^1 fought against the coalition of planets with numbers from L_i^2 to R_i^2. The intensity of this conflict was c_i.

The times after the fall of the Empire were very chaotic, so sometimes the two fighting coalitions could even have common planets.

The Second Foundation decided that these conflicts directly affect the danger of future protected routes, because after such wars the relations between planets remain bad for a long time. More precisely, for every pair of different planets u and v, the danger of the route between them is multiplied by c_i if one of the following conditions holds:

$$u \in [L_i^1, R_i^1] \text{ and } v \in [L_i^2, R_i^2],$$

or

$$u \in [L_i^2, R_i^2] \text{ and } v \in [L_i^1, R_i^1].$$

For one fixed conflict, the danger of the same route can be multiplied by c_i at most once, even if both conditions hold at the same time. However, different conflicts are independent. If several conflicts affect the same route, all corresponding multipliers are applied.

This task turned out to be difficult even for the geniuses of the Second Foundation, so they ask you for help. Find the minimum possible total danger of protected routes that is enough to connect all planets.

In other words, consider the complete undirected weighted graph on n planets. The weight of an edge is the final danger of the corresponding route. You have to find the weight of a minimum spanning tree of this graph.

If this value is greater than 10^9, then the galaxy is still far from being united. In this case, you should output 10^9 instead.

Input

The first line contains two integers n and q — the number of planets and the number of military conflicts ($1 \le n \le 10^5$, $0 \le q \le 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ — the economic levels of the planets ($1 \le a_v \le 10^7$, $a_1 < a_2 < \ldots < a_n$).

Each of the next q lines contains five integers $L_i^1,\ R_i^1,\ L_i^2,\ R_i^2,\ c_i$ describing the i-th military conflict ($1 \le L_i^1 \le R_i^1 \le n$, $1 \le L_i^2 \le R_i^2 \le n$, $1 \le c_i \le 10^5$).

Here $[L_i^1, R_i^1]$ is the first coalition, $[L_i^2, R_i^2]$ is the second coalition, and c_i is the intensity of the conflict.

Output

Output one real number — the minimum possible total danger of protected routes needed to connect all planets.

If this value is greater than 10^9, output 10^9 instead.

Your answer will be accepted if its absolute or relative error does not exceed 10^{-4}.

Examples

standard input	standard output
4 0 1 2 5 10	6.5
5 4 2 3 6 10 30 1 3 3 5 2 2 4 4 5 3 1 1 5 5 5 2 5 2 5 2	35.5
4 3 1 2 4 8 1 4 1 4 2 1 4 1 4 3 1 4 1 4 5	180
5 4 1 2 3 4 10000000 1 4 1 4 100000 1 2 5 5 100000 3 3 5 5 100000 4 4 5 5 100000	1000000000

Note

In the first sample, there are no conflicts, so all route dangers are equal to their initial values.

The initial dangers of all routes are:

	1	2	3	4
1	–	2	5	10
2	2	–	2.5	5
3	5	2.5	–	2
4	10	5	2	–

One optimal way to connect all planets is to choose the routes $(1, 2),\ (2, 3),\ (3, 4)$. Their total danger is $2 + 2.5 + 2 = 6.5$.

In the second sample, the final dangers of all routes are shown in the table below.

	1	2	3	4	5
1	–	1.5	6	10	150
2	1.5	–	8	40	120
3	6	8	–	20	60
4	10	40	20	–	18
5	150	120	60	18	–

Problem D. Digesting

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

Let's define *digesting* of an array $a_1, a_2, \ldots, a_n$ the following way:

- Choose any permutation of length n: $p_1, p_2, \ldots, p_n$ (an array consisting of integers from 1 to n, such that each of them appears exactly once).
- Simultaneously perform $a_i := a_i \bmod \max(a_{p_i}, 1)$.
- Choose any number a_i to kick from the array, concatenate others into **new** array $a_1, a_2, \ldots, a_{n-1}$. This array is the result of the digesting operation.

Any array that can be obtained by this procedure, can be the result of digesting operation.

You are given an array $a_1, a_2, \ldots, a_n$ consisting of integers from 1 to n. What is the maximum possible value of a_1 that can be achieved, after digesting this array $n - 1$ times in a row?

Input

Each test consists of multiple test cases. The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases. The following lines describe the test cases.

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of the array a.

It is guaranteed that the sum of the values of n for all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the maximum possible value of a_1 that can be achieved.

Example

standard input	standard output
1 3 3 2 2	1

Problem E. Easter Eggs

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

There are n initially empty buckets arranged in a line.

Every minute, Palych chooses one of the non-locked buckets uniformly at random and puts one easter egg into it! After that, you may lock any number of buckets, possibly zero. Once a bucket is locked, it remains locked forever.

You win as soon as the numbers of eggs in the buckets, from left to right, form a strictly increasing array.

You are not allowed to lock any buckets before the first egg is dropped. Also, if you lock all the buckets before you win — you lose, and this outcome must be avoided.

Your goal is to minimize the expected number of minutes until you win. Determine this minimum expected value.

Input

The first line contains one integer n ($2 \le n \le 100$) — the number of buckets.

Output

Output one real number — the minimum possible expected number of minutes until you win.

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

Example

standard input	standard output
2	2.0

Problem F. Fresh Cartridges

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

There is an n-dimensional **cyclic** arena for a robot: each coordinate is a residual modulo $p = 998\,244\,353$. Each cartridge contains a command — a shift vector of n residuals modulo p. Applying a cartridge once moves the robot exactly by the vector written in it (each coordinate is taken modulo p).

A remote control can hold exactly n cartridges. We call a remote control universal if, using only the cartridges installed in it and applying each of them an arbitrary number of times, the robot can get from any cell of the arena to any other one.

There are m cartridges in the warehouse. The i-th cartridge contains the shift vector v_i.

There are q independent warehouse inspection reports. In the j-th report, only the cartridges with indices from l_j to r_j remain fresh — they can be taken from the warehouse for free, while all other warehouse cartridges have expired and are prohibited from use.

Moreover, within each report, it is allowed to manufacture any number of new fresh cartridges. Each such cartridge costs one coin, and its shift vector can be chosen arbitrarily (including coincident vectors). Manufactured cartridges are not preserved between reports.

For each report, determine the minimum number of coins needed to assemble **two** universal remote controls from fresh cartridges. Each cartridge can be used at most once and only in one remote control.

Input

The first line contains one integer $1 \le t \le 1000$ — the number of test cases.

For each test case, the first line contains three integers n, m and q — the dimension of the arena, the number of cartridges in the warehouse and the number of inspection reports, respectively ($1 \le n \le 150,\ 1 \le m, q \le 3 \cdot 10^5$). It is additionally guaranteed that the sum of $n \cdot m$ and the sum of q over all test cases do not exceed $3 \cdot 10^5$.

Then m lines follow, each containing n integers $v_{i,1}, v_{i,2}, \ldots, v_{i,n}$ — the shift vector of the i-th cartridge ($0 \le v_{i,k} < p$).

Then q lines follow, each containing two integers l_j and r_j ($1 \le l_j \le r_j \le m$) — in the corresponding inspection report, only the cartridges with indices from l_j to r_j remain fresh.

Output

For each inspection report, output one integer — the minimum number of coins needed to assemble two universal remote controls from fresh cartridges.

Examples

standard input	standard output
1 1 3 4 0 1 2 1 1 2 2 3 3 1 3	2 1 1 0
1 2 4 3 1 0 0 1 0 1 0 1 1 2 1 3 1 4	2 1 1

Problem G. Game Show: XOR Edition

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

Famous game show returns with yet another season — now it is XOR edition! The show works like this: the contestant has a bag of numbers, initially empty. Then n rounds will happen in order. In each round, the contestant will be offered an integer, uniformly randomly chosen from $[0; 2^k - 1]$. Then the contestant can choose to either add it to their bag or ultimately skip it. After all n rounds, the contestant goes away with the prize equal to the XOR of the numbers in their bag. Note that the XOR of an empty bag is considered to be equal to zero.

The contestant knows n and k, and the fact that numbers are generated uniformly randomly from $[0; 2^k - 1]$, but does not know exactly what the numbers are equal to. Given that, the contestant optimizes their strategy to maximize the expected value of the prize, and also, for the great show, if in the optimal strategy taking or not taking the next number leads to the same expected value, the contestant will always choose to pick a number.

You are the producer of the show. You were given a list of integers that will be used for tomorrow's show, that are indeed uniformly randomly generated. You want to minimize the prize for the contestant, to cut the costs of a show. But changing the numbers themselves may raise too much suspicion; therefore, you only want to change the order of the numbers in which they will be presented to the contestant. Of course, you are doing this secretly, and no one, including the contestant, knows about such shenanigans; therefore, the contestant's strategy will remain the same. Please find any possible order of the numbers such that the contestant will make as little money as possible!

Input

First line of the input contains two integers k, n ($4 \le k \le 18, 10k \le n \le 2 \cdot 10^5$).

Next line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^k$). It is guaranteed that these integers are uniformly randomly generated.

This problem has 90 tests, with 6 tests for each value of $4 \le k \le 18$, where n values are distributed in arithmetic progression between $10k$ and $2 \cdot 10^5$, including both ends.

Output

Output n integers: rearranged array $a_1, a_2, \ldots, a_n$ that achieves a minimal value. If there are multiple answers, output any of them.

Example

standard input	standard output
4 40 4 12 3 5 9 7 0 6 13 10 12 14 14 7 2 8 3 4 2 3 4 9 13 4 13 6 10 2 0 10 4 9 15 13 10 8 6 7 14 3	***

Note

In order to fit on the page, several line breaks were added to sample input. The real inputs follow the input description.

We deliberately don't show you the answer for the first test. Follow output format.

Problem H. Hockey Hog

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

A hockey rink for hogs has the shape of a convex polygon A with vertices $A_1, \ldots, A_n$. At vertex A_i there is a net numbered i.

The rink lies entirely strictly inside another convex polygon B with vertices $B_1, \ldots, B_m$. Polygon B bounds a dangerous zone: a spectator is not allowed to be inside B, but may stand on its boundary or outside B.

The nets open toward the inside of the rink. From the outside, each net is protected by a small opaque fence consisting of two panels placed along the two sides of the rink incident to vertex A_i.

A spectator at point P can see a goal scored into net i if the ray starting at A_i and passing through P lies inside the interior angle of polygon A at vertex A_i, or on its boundary. Otherwise, the protective fence of this net blocks the view. The fence of net i can block the view only of net i and does not affect the visibility of the other nets.

Because of the strength and mass of the hogs, a net into which a goal is scored breaks immediately and is no longer used. The match ends when all nets are broken. The order in which the goals are scored is given by a permutation $p_1, p_2, \ldots, p_n$: the k-th goal is scored into net p_k.

Before the k-th goal, the spectator wants to change his location: choose one valid point P outside the interior of B, move to it, and remain there until the end of the match. Independently for each k from 1 to n, determine the maximum number of goals among the remaining ones (into nets $p_k, \ldots, p_n$) that the spectator will still be able to see after choosing the best possible point P.

Input

The first line contains one integer $1 \le T \le 10^5$ — the number of test cases. The descriptions of the T test cases follow.

The first line of each test case contains two integers n and m ($3 \le n, m \le 5 \cdot 10^5$) — the numbers of vertices of polygons A and B, respectively.

The next n lines each contain two integers x_i and y_i — the coordinates of vertex A_i. The vertices of polygon A are given in counterclockwise order.

The next m lines each contain two integers z_i and v_i — the coordinates of vertex B_i. The vertices of polygon B are given in counterclockwise order.

All vertex coordinates of the polygons are integers whose absolute values do not exceed 10^9. It is guaranteed that both polygons are strictly convex, and that polygon A lies entirely strictly inside polygon B.

The last line of each test case contains a permutation $p_1, p_2, \ldots, p_n$ of the numbers from 1 to n — the order of the nets into which the goals are scored.

It is guaranteed that the sum of n over all test cases does not exceed $5 \cdot 10^5$, and the sum of m over all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, output n integers: the k-th integer must be equal to the maximum number of future goals the spectator can see if they choose the best possible valid point P before the k-th goal.

Example

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

Note

Consider the second sample test case.

Before the first goal, the spectator can choose, for example, the point $(0, 10)$. From this point, the goals into nets 1 and 2 are visible, so the spectator can see 2 future goals.

Before the second goal, net 1 is already broken. The spectator can choose the point $(-3.5, 2.5)$, which lies on the boundary of polygon B. From this point, the goals into nets 2 and 3 are visible, so the spectator can again see 2 future goals.

Before the third goal, only nets 3 and 4 remain. There is no valid point from which both of these goals are visible at the same time. However, the spectator can choose, for example, the point $(3, 2)$ on the boundary of B and see the goal into net 4. Therefore, the answer for this moment is 1.

Before the fourth goal, only net 4 remains. The spectator can stay at point $(3, 2)$ and see the last goal. Therefore, the answer is also 1.

Thus, for the second sample test case, the answers are $2, 2, 1, 1$.

Problem I. Incident in the Bakery

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

In the royal bakery, the head chef decided to bake the most even pancake in the world. He baked a pancake in the shape of a strictly convex polygon A, whose vertices $A_1, A_2, \ldots, A_n$ were listed in counterclockwise order.

After that, the chef's assistant decided to "make the shape a little neater": on each side, he marked its midpoint, connected the marked points in order, and cut off all the dough outside the resulting polygon. The head chef didn't even have time to object before the assistant repeated the same thing once again.

Formally, polygon B was first obtained from polygon A:

$$B_i = \frac{A_i + A_{i+1}}{2},$$

and then polygon C was obtained from polygon B in the same way:

$$C_i = \frac{B_i + B_{i+1}}{2},$$

where $A_{n+1} = A_1$ and $B_{n+1} = B_1$.

Now only the final polygon C is left on the table. The head chef is sure that the assistant ruined everything and demands that the pancake be restored to the shape it had before the cuttings. Meanwhile, the assistant has already been arguing for three hours about whether such a pancake could have been baked at all.

You need to restore any suitable initial polygon A, or report that no such polygon exists.

It is guaranteed that if an answer exists, then there exists a suitable polygon A such that both coordinates of each of its vertices are integers, divisible by 4, and do not exceed 10^9 in absolute value.

Input

The first line contains a single integer t ($1 \le t \le 7500$) — the number of test cases. The descriptions of the test cases follow.

The first line of each test case contains a single integer n ($3 \le n \le 1000$) — the number of vertices of the polygon.

Each of the next n lines contains two integers x_i and y_i ($|x_i|, |y_i| \le 10^9$) — the coordinates of point C_i.

It is guaranteed that, for each test case, the points $C_1, C_2, \ldots, C_n$ form a strictly convex polygon and are listed in counterclockwise order.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, print the answer.

If there is no suitable initial polygon, print `NO`.

Otherwise, print `YES`. Then print n lines, the i-th of which should contain two coordinates of point A_i separated by a space. The coordinates of the points may be real numbers whose absolute values do not exceed $4 \cdot 10^9$. The printed points must form a strictly convex polygon in counterclockwise order. In addition, for every i the following inequality must hold:

$$[(A_{i+1} - A_i), (A_{i+2} - A_{i+1})] > 10^{-7},$$

where indices are taken modulo n, and

$$[V, U] = V_x U_y - V_y U_x.$$

Your answer is considered correct if, after applying the described operation twice to the polygon you printed, both coordinates of the i-th resulting point coincide with the coordinates of point C_i with absolute or relative error at most 10^{-6}.

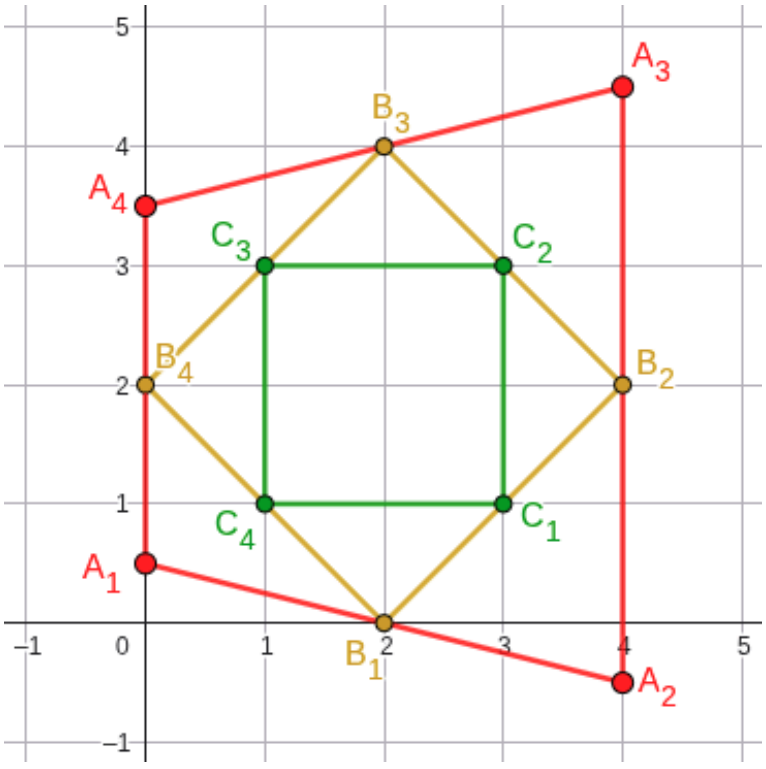
You can print `YES` and `NO` in any case. For example, the strings `yEs`, `yes`, `Yes`, and `YES` will be recognized as positive answers.

Example

standard input	standard output
3 3 5 4 4 5 3 3 4 3 1 3 3 1 3 1 1 6 0 0 2 0 4 1 3 3 1 4 0 2	YES 0 0 8 4 4 8 YES 0 0.5 4 -0.5 4 4.5 0 3.5 NO

Note

Illustration of the second test case: one of the possible polygons A that generates the required polygon C:



Problem J. Jaded Juice

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

Lupa is a big fan of transfusing some liquid from one container to another.

Lupa has n ordinary transparent bottles and one special transparent bottle. Each ordinary bottle has height w: its bottom is at level 0, and its top is at level w. The special bottle has height $2w$.

All bottles are initially filled with juice up to the same hidden integer level x, where $0 \le x \le w$. Lupa does not know the value of x.

The ordinary bottles are partially covered by dark labels. In the i-th ordinary bottle, the label covers all levels from l_i to w. Therefore, the juice level in this bottle is visible to Lupa if and only if it is strictly below l_i.

The special bottle has a different label: it covers all levels from 0 to w. Therefore, the juice level in the special bottle is visible to Lupa if and only if it is strictly above w.

Lupa wants to determine the hidden value x uniquely.

Lupa may perform pours. In one pour, she chooses one of the ordinary bottles and a non-negative integer amount p, then pours exactly p units of juice from this ordinary bottle into the special bottle. After each pour, Lupa observes all juice levels that are visible, according to the labels described above.

However, Lupa is not allowed to make a pour that might be impossible. More precisely, before choosing a pour, consider all values of x that are still consistent with every observation made so far. The pour of p units from the chosen bottle is allowed only if, for every such possible value of x, that bottle contains at least p units of juice.

Lupa's strategy may be adaptive: the next pour may depend on the observations obtained after the previous pours. She stops as soon as the observations uniquely determine the value of x.

Find the minimum number of pours that guarantees determining x, no matter what the hidden value x is.

Input

The first line contains two integers n and w ($1 \le n \le 10^6$, $2 \le w \le 10^{18}$) — the number of ordinary bottles and their height.

The second line contains n integers $l_1, l_2, \ldots, l_n$ ($1 \le l_i < w$) — the lower levels of the dark labels on the ordinary bottles.

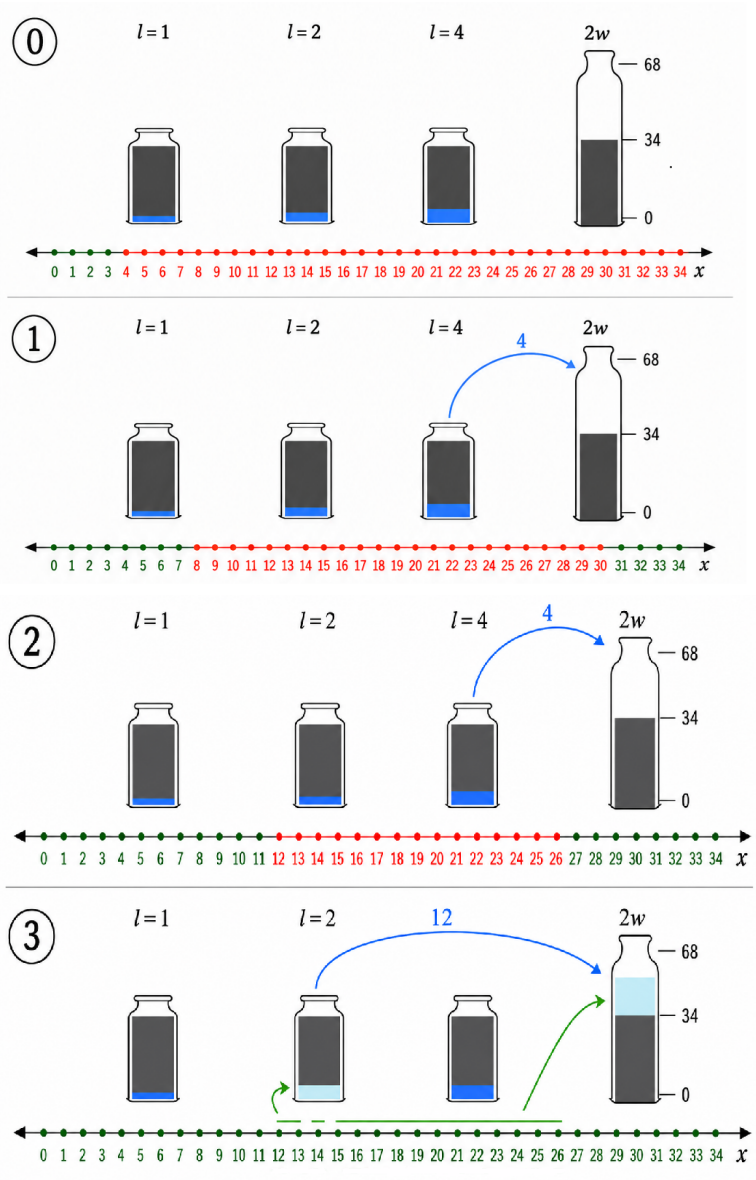
Output

Output one integer — the minimum number of pours required to guarantee determining x.

Example

standard input	standard output
3 34 1 2 4	3

Note



In the first test, consider the bottle with label 4. If $x < 4$, its juice level is visible before any pour, so x is already known. Therefore, assume that $4 \le x \le 34$.

Lupa first pours 4 units from the bottle with label 4 into the special bottle.

After this pour, if $x < 8$, then the level in this bottle is $x - 4 < 4$, so it is visible and determines x. If $x > 30$, then the level in the special bottle is $x + 4 > 34$, so it is visible and determines x. Therefore, only values $8 \le x \le 30$ may still be unknown.

Next, Lupa pours 4 more units from the same bottle.

After this pour, if $x < 12$, then the level in this bottle is $x - 8 < 4$, so it is visible and determines x. If $x > 26$, then the level in the special bottle is $x + 8 > 34$, so it is visible and determines x. Therefore, only values $12 \le x \le 26$ may still be unknown.

Finally, Lupa pours 12 units from the bottle with label 2.

After this pour, if $x < 14$, then the level in this bottle is $x - 12 < 2$, so it is visible and determines x. If $x > 14$, then the level in the special bottle is $x + 20 > 34$, so it is visible and determines x. The only remaining value is $x = 14$, so it is also determined.

Thus, after 3 pours, Lupa is guaranteed to determine x.

Problem K. Kitchen Knives

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

Little MS has a collection of n knives, and the i-th knife has a cutting power of a_i. She needs to cut a giant cake into pieces, but she can't decide which knives to choose from her collection for this very important job.

After some time, Little MS decided that she will choose a non-empty subset of knives uniformly at random (so every subset has a chance of $\frac{1}{2^n-1}$ to be chosen). Then she will use all the chosen knives in any order; a knife with cutting power of x will cut every current slice into x smaller ones. In the end, Little MS will end up with $a_{i_1} \cdot \ldots \cdot a_{i_k}$ slices, where $i_1, \ldots, i_k$ are the indices of the chosen knives.

Help Little MS determine if there is a number s such that the probability of Little MS ending up with exactly s slices of cake is **strictly greater** than 50%.

Input

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

The first line of each test case contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of knives.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the cutting power of the i-th knife.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output -1 if there is no suitable number s.

Otherwise, output a number s such that the probability that Little MS ends up with exactly s cake slices is strictly greater than 50%.

Example

standard input	standard output
4 2 2 3 2 100 100 3 2 2 2 4 1 1 1 1	-1 100 -1 1

Note

In the first test case, the possible products over all non-empty subsets of knives are 2, 3, and 6. Each of them appears once, so none of them is obtained with a probability strictly greater than 50%. Therefore, the answer is -1.

In the second test case, the possible products over all non-empty subsets of knives are 100, 100, and 10000. The product 100 appears in 2 out of 3 cases, so its probability is strictly greater than 50%. Therefore, the answer is 100.

In the third test case, the possible products over all non-empty subsets of knives are 2, 2, 2, 4, 4, 4, and 8. No product appears more than half of the time, so the answer is -1.

Problem L. Leave Triple & Link Tree

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

You are given n triplets of integers from 1 to n. It is guaranteed that each number appears in all triplets a total of 3 times, and that all numbers in each triplet are distinct.

You need to remove one of the triplets, and from all the remaining triplets, remove one number from each. As a result, you will have $n-1$ pairs of integers, which we will treat as an edges of a graph. It is required that the graph with n vertices from 1 to n and these $n-1$ edges is connected. Find a way to do this, or report that it is impossible. If there are multiple ways, find one that **maximizes** index of removed triple. If there are still multiple ways, you can find any of them.

Input

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

The first line of each test case contains a single integer n ($3 \le n \le 2 \cdot 10^5$) — the number of triplets.

Next n lines contain triples, i-th line contains 3 integers a_i, b_i, c_i ($1 \le a_i, b_i, c_i \le n$) — the numbers in a triple. It is guaranteed that $a_i \neq b_i, a_i \neq c_i, b_i \neq c_i$.

It is guaranteed that every number from 1 to n is included in exactly 3 triples.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print `NO` if it is impossible to perform the required operations. Otherwise, print `YES` in the first line. Then print n lines. In the i-th of these lines, describe the result for the i-th triplet.

If the i-th triplet is removed, print `-1 -1`. Otherwise, print two integers: the two numbers left in this triplet after removing one number from it in any order. If there are several valid answers, output one with the maximum possible index of the removed triplet. If there are still several valid answers, you may output any of them.

Example

standard input	standard output
2 9 1 2 3 9 8 7 6 8 7 5 1 4 6 9 2 3 5 7 2 4 6 4 3 1 9 5 8 6 1 3 2 1 3 2 1 3 2 4 5 6 4 5 6 4 5 6	YES 2 3 9 8 8 7 5 1 6 9 3 5 2 6 4 1 -1 -1 NO

Problem M. Minecraft Spiderweb

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

In a new Minecraft update, spiders have finally gotten the long-awaited ability to weave webs by hand. However, to preserve the spirit of the game, the developers allowed them to stretch threads only strictly horizontally or strictly vertically.

A spider has a rectangular area of size $W \times H$, which we will consider as the rectangle with corners $(0, 0)$, $(W, 0)$, (W, H), and $(0, H)$. On the boundary of this rectangle, n points are marked — places where the spider must definitely stretch its web.

The spider may draw any number of horizontal and vertical thread segments. The segments may be arbitrary — they may intersect, lie along the rectangle's boundary, or even hang in the air. The spider must weave the web so that all marked points lie in one connected component. In other words, it must be possible to get from any marked point to any other marked point by moving only along the drawn segments.

Naturally, the developers want to save on web texture. Therefore, you need to find the minimum possible total length of all threads drawn.

Input

The first line contains an integer t ($1 \le t \le 2 \cdot 10^5$) — the number of test cases. Then the test cases follow.

The first line of each test case contains three integers n, W, H ($1 \le n \le 10^6$, $1 \le W, H \le 10^9$) — the number of marked points and the rectangle dimensions.

The next n lines contain pairs of integers x_i, y_i ($0 \le x_i \le W$, $0 \le y_i \le H$) — the coordinates of the marked points.

It is guaranteed that all points are distinct and lie on the boundary of the rectangle. It is also guaranteed that the points are listed in counterclockwise order along the boundary of the rectangle, starting from the bottom-left corner $(0, 0)$. The sum of n over all test cases does not exceed 10^6.

Output

For each test case, output one number — the minimum possible total length of threads that can be stretched.

Example

standard input	standard output
3 1 2 1 1 0 2 1 2 0 0 1 2 3 2 2 1 0 2 1 0 1	0 3 3

Problem N. Native Niuean

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

After ending his international football career, Cristiano Ronaldo decided that he wants to become a citizen of Niue. To do so, he must pass an exam on Niuean traditions and culture. He answered all the questions easily except for the last one, so he needs your help. Can you help Ronaldo become a citizen of Niue? Here is the final question:

"For array $a_1, a_2, \ldots, a_n$ of positive integers we define it's `nativeness` as $\sum_{i=1}^{n} a_i \cdot f_i$, where $f_0 = 1, f_1 = 1$ and for $i \geq 2$: $f_i = a_{i-1} f_{i-1} + f_{i-2}$.

Also, we will call an array `niuean` if it is lexicographically smaller than it's reverse.

You are given an integer m. Please calculate how many `niuean` arrays have `nativeness` not bigger than m. Give your answer modulo 998 244 353."

Input

The first line contains one integer m ($1 \leq m \leq 10^{10}$).

Output

Output number of arrays modulo 998 244 353.

Examples

standard input	standard output
6	1
7	2
13	8
42	114
67	305
6767	3471737
676767	863282261
67676767	217404956
6767676767	691869515

Note