A. Fashionable Array

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

In 2077, everything became fashionable among robots, even arrays...

We will call an array of integers a *fashionable* if $\min(a) + \max(a)$ is divisible by 2 without a remainder, where $\min(a)$ — the value of the minimum element of the array a, and $\max(a)$ — the value of the maximum element of the array a.

You are given an array of integers $a_1, a_2, \ldots, a_n$. In one operation, you can remove any element from this array. Your task is to determine the minimum number of operations required to make the array a *fashionable*.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$). The description of the test cases follows.

The first line of each test case contains one integer n ($1 \le n \le 50$) — the size of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 50$) — the elements of the array a.

Output

For each test case, output one integer — the minimum number of operations required to make the array a *fashionable*.

Standard Input	Standard Output
6 2 5 2 7 3 1 4 1 5 9 2 7 2 7 4 6 9 11 5 3 1 2 1 2 2 1 8 8 6 3 6 4 1 1 6	1 0 2 1 1 3

Note

In the first test case, at least one element needs to be removed since $\min(a) + \max(a) = 2 + 5 = 7$, and 7 is not divisible by 2. If any of the elements are removed, only one element will remain. Then $\max(a) + \min(a)$ will be divisible by 2.

In the second test case, nothing needs to be removed since $\min(a) + \max(a) = 1 + 9 = 10$, and 10 is divisible by 2.

In the third test case, you can remove the elements with values 2 and 4, then $\min(a) + \max(a) = 5 + 11 = 16$, and 16 is divisible by 2.

,

B. Down with Brackets

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

In 2077, robots decided to get rid of balanced bracket sequences once and for all!

A bracket sequence is called *balanced* if it can be constructed by the following formal grammar.

1. The empty sequence $\varnothing$ is balanced.
2. If the bracket sequence A is balanced, then (A) is also balanced.
3. If the bracket sequences A and B are balanced, then the concatenated sequence AB is also balanced.

You are the head of the department for combating balanced bracket sequences, and your main task is to determine which brackets you can destroy and which you cannot.

You are given a balanced bracket sequence represented by a string s, consisting of the characters (and). Since the robots' capabilities are not limitless, they can remove **exactly** one opening bracket and **exactly** one closing bracket from the string.

Your task is to determine whether the robots can delete such two brackets so that the string s is no longer a balanced bracket sequence.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

Each test case consists of a single string s ($2 \le |s| \le 2 \cdot 10^5$) — a sequence of the characters (and).

It is guaranteed that s is a balanced bracket sequence.

It is also guaranteed that the sum of $|s|$ across all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output "YES" if the robots can make the string stop being a balanced bracket sequence, and "NO" otherwise.

You may output each letter in any case (lowercase or uppercase). For example, the strings "yEs", "yes", "Yes", and "YES" will be accepted as a positive answer.

Standard Input	Standard Output
4 (()) (())()() () (())(())	NO YES NO YES

Note

In the first test case, it can be shown that the robots will not be able to break the correct bracket sequence.

In the second test case, one of the options for removing brackets is as follows:

`(())()()` $\rightarrow$ `((()))(`, which is not a correct bracket sequence.

In the fourth test case, one of the options for removal is as follows:

`(())(())` $\rightarrow$ `())(()`, which is not a correct bracket sequence.

,

C. Racing

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

In 2077, a sport called hobby-droning is gaining popularity among robots.

You already have a drone, and you want to win. For this, your drone needs to fly through a course with n obstacles.

The i-th obstacle is defined by two numbers l_i, r_i. Let the height of your drone at the i-th obstacle be h_i. Then the drone passes through this obstacle if $l_i \le h_i \le r_i$. Initially, the drone is on the ground, meaning $h_0 = 0$.

The flight program for the drone is represented by an array $d_1, d_2, \ldots, d_n$, where $h_i - h_{i-1} = d_i$, and $0 \le d_i \le 1$. This means that your drone either does not change height between obstacles or rises by 1. You already have a flight program, but some d_i in it are unknown and marked as -1. Replace the unknown d_i with numbers 0 and 1 to create a flight program that passes through the entire obstacle course, or report that it is impossible.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

In the first line of each test case, an integer n ($1 \le n \le 2 \cdot 10^5$) is given — the size of the array d.

In the second line of each test case, there are n integers $d_1, d_2, \ldots, d_n$ ($-1 \le d_i \le 1$) — the elements of the array d. $d_i = -1$ means that this d_i is unknown to you.

Next, there are n lines containing 2 integers l_i, r_i ($0 \le l_i \le r_i \le n$) — descriptions of the obstacles.

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

Output

For each test case, output n integers $d_1, d_2, \ldots, d_n$, if it is possible to correctly restore the array d, or -1 if it is not possible.

Standard Input	Standard Output
5 4 0 -1 -1 1 0 4 1 2 2 4 1 4 3 0 -1 -1 0 1 2 2 0 3 2	0 1 1 1 -1 -1 0 1 1 0 1 0 0 1 -1

-1 -1 0 0 2 2 8 -1 -1 1 -1 -1 0 0 -1 0 0 0 1 0 2 0 2 1 3 0 4 2 5 4 5 1 0 1 1	

Note

In the first test case, one possible answer is $d = [0, 1, 1, 1]$. The array h will be $[0, 0 + 1, 0 + 1 + 1, 0 + 1 + 1 + 1] = [0, 1, 2, 3]$. This array meets the conditions of the problem.

In the second test case, it can be proven that there is no suitable array d, so the answer is -1.

,

D. Fewer Batteries

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 256 megabytes

In 2077, when robots took over the world, they decided to compete in the following game.

There are n checkpoints, and the i-th checkpoint contains b_i batteries. Initially, the Robot starts at the 1-st checkpoint with no batteries and must reach the n-th checkpoint.

There are a total of m one-way passages between the checkpoints. The i-th passage allows movement from point s_i to point t_i ($s_i < t_i$), but not the other way. Additionally, the i-th passage can only be used if the robot has at least w_i charged batteries; otherwise, it will run out of power on the way.

When the robot arrives at point v, it can additionally take any number of batteries from 0 to b_v, inclusive. Moreover, it always carries all previously collected batteries, and at each checkpoint, it recharges all previously collected batteries.

Find the minimum number of batteries that the robot can have at the end of the journey, or report that it is impossible to reach from the first checkpoint to the last.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n, m ($2 \le n \le 2 \cdot 10^5, 0 \le m \le 3 \cdot 10^5$) — the number of checkpoints and the number of passages, respectively.

The second line contains n numbers b_i ($0 \le b_i \le 10^9$) — the number of batteries at the i-th checkpoint.

The next m lines contain three integers s_i, t_i, w_i ($1 \le s_i < t_i \le n, 1 \le w_i \le 10^9$) — the endpoints of the passage and the minimum number of batteries required to pass through it.

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

It is guaranteed that the sum of m does not exceed $3 \cdot 10^5$.

Output

For each test case, output the minimum number of batteries that you can have at the end of the journey, or -1 if it is impossible to reach point n.

Standard Input	Standard Output
4 3 3 2 0 0 1 2 1 2 3 1 1 3 2 5 6 2 2 5 0 1 1 2 2	1 4 -1 10

1 3 1 1 4 3 3 5 5 2 4 4 4 5 3 2 0 1 1 4 4 3 10 0 0 1 2 1 1 3 3 2 3 10 3 4 5	

Note

In the first test case, you need to take 1 battery at the starting point, then move to point 2, and then to point 3.

In the second test case, you need to take 2 batteries at the starting point, then move to point 2, take another 2 batteries, move to point 4, and then to point 5.

In the third test case, there is no path from point 1 to point n.

In the fourth test case, you need to take 1 battery at the starting point, then move to point 2, take another 9 batteries, move to point 3, and then to point 4.

,

E. Melody

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

In 2077, the robots that took over the world realized that human music wasn't that great, so they started composing their own.

To write music, the robots have a special musical instrument capable of producing n different sounds. Each sound is characterized by its volume and pitch. A sequence of sounds is called music. Music is considered `beautiful` if any two consecutive sounds differ either only in volume or only in pitch. Music is considered `boring` if the volume or pitch of any three consecutive sounds is the same.

You want to compose `beautiful`, **non**-`boring` music that contains each sound produced by your musical instrument exactly once.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

In the first line of each test case, there is a number n ($1 \le n \le 2 \cdot 10^5$) — the number of sounds that the musical instrument can produce.

Next, there are n lines, where the i-th line contains a pair of numbers v_i, p_i ($1 \le v_i,\ p_i \le 10^9$) — the volume and pitch of the i-th sound, respectively. It is guaranteed that among all n sounds, there are no duplicates, meaning for any $i \neq j$, at least one of the conditions $v_i \neq v_j$ or $p_i \neq p_j$ holds.

The sum of n across all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, if it is possible to compose such music, output "YES", and on the next line, output n numbers — the indices of the sounds in the order that forms beautiful non-boring music. Otherwise, output "NO".

You may output each letter in any case (lowercase or uppercase). For example, the strings "`yEs`", "`yes`", "`Yes`", and "YES" will be accepted as a positive answer.

Standard Input	Standard Output
5 4 179 239 179 179 239 179 239 239 3 1 1 2 1 3 1 1	YES 4 3 2 1 NO YES 1 NO YES 3 4 6 7 2 1 5

5 7 5 1 1 1 2 2 1 2 2 99 99 7 1 1 1 3 2 1 2 2 3 1 3 2 3 3	

Note

In the first test case, the music $(239, 239) - (239, 179) - (179, 179) - (179, 239)$ is suitable, contains all sounds, and all consecutive sounds differ either only in volume or only in pitch.

In the second test case, it can be shown that there is no suitable music with the given sounds.

,

F. Faculty

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	256 megabytes

In 2077, after the world was enslaved by robots, the robots decided to implement an educational reform, and now the operation of taking the modulus is only taught in the faculty of "Ancient World History". Here is one of the entrance tasks for this faculty:

We define the `beauty` of an array of positive integers b as the maximum $f(b_i, b_j)$ over all pairs $1 \le i, j \le n$, where $f(x, y) = (x \bmod y) + (y \bmod x)$.

Given an array of positive integers a of length n, output n numbers, where the i-th number ($1 \le i \le n$) is the `beauty` of the array $a_1, a_2, \ldots, a_i$.

$x \bmod y$ is the remainder of the division of x by y.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^6$) — the size of the array a.

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

It is guaranteed that the sum of n across all test cases does not exceed 10^6.

Output

For each test case, output n integers — the beauties of all prefixes of the array a.

Standard Input	Standard Output
2 5 3 1 4 1 5 7 5 11 11 4 2 1 10	0 1 4 4 5 0 6 6 7 7 7 11

Note

The beauty of the array 3 is 0.

The beauty of the array $3, 1$ is $f(3, 1) = 1$.

The beauty of the array $3, 1, 4$ is $f(3, 4) = 4$.

The beauty of the array $3, 1, 4, 1$ is $f(4, 3) = 4$.

The beauty of the array $3, 1, 4, 1, 5$ is $f(4, 5) = 5$.