A. Race

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

Alice and Bob participate in a game TV show. When the game starts, the prize will be dropped to a certain point, and whoever gets to it first will get the prize.

Alice decided that she would start running from point a. Bob, however, has not yet chosen his starting position.

Bob knows that the prize could drop either at point x or at point y. He also knows that he can reach the prize faster than Alice if the distance from his starting position to the prize is **strictly less** than the distance from Alice's starting position to the prize. The distance between any two points c and d is calculated as $|c - d|$.

Your task is to determine whether Bob can choose an integer point that is guarantee to get to the prize faster, regardless of where it appears (at point x or y). Bob can choose any integer point, except for a (in particular, he can choose to start in point x, point y, or any other point, but not a).

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

The only line of each test case contains three integers a, x, y ($1 \le a, x, y \le 100$). Points a, x, and y are pairwise distinct.

Output

For each test case, print "YES" (case insensitive) if Bob can choose an integer point that is guarantee to get to the prize faster, regardless of where it appears. Otherwise, print "NO" (case insensitive).

Standard Input	Standard Output
3 1 3 4 5 3 1 3 1 5	YES YES NO

Note

In the first example, Bob can choose point 4. If the prize will be at point x, then Bob's distance is $|4 - 3| = 1$ and Alice's distance is $|1 - 3| = 2$. If the prize will be at point y, then Bob's distance is $|4 - 4| = 0$ and Alice's distance is $|1 - 4| = 3$.

In the second example, Bob can choose point 2. If the prize will be at point x, then Bob's distance is $|2 - 3| = 1$ and Alice's distance is $|5 - 3| = 2$. If the prize will be at point y, then Bob's distance is $|2 - 1| = 1$ and Alice's distance is $|5 - 1| = 4$.

In the third example, Bob cannot choose a point to guarantee his victory.

,

B. Shrinking Array

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

Let's call an array b *beautiful* if it consists of at least two elements and there exists a position i such that $|b_i - b_{i+1}| \leq 1$ (where $|x|$ is the absolute value of x).

You are given an array a, and as long as it consists of at least two elements, you can perform the following operation:

1. Choose two adjacent positions i and $i + 1$ in the array a.
2. Choose an integer x such that $\min(a_i, a_{i+1}) \leq x \leq \max(a_i, a_{i+1})$.
3. Remove the numbers a_i and a_{i+1} from the array, and insert the number x in their place. Obviously, the size of the array will decrease by 1.

Calculate the minimum number of operations required to make the array beautiful, or report that it is impossible.

Input

The first line contains one integer t ($1 \leq t \leq 200$) — the number of test cases.

The first line of each test case contains one integer n ($2 \leq n \leq 1000$) — the size of the array a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^6$) — the array a itself.

Output

For each test case, output one integer — the minimum number of operations needed to make the array a beautiful, or -1 if it is impossible to make it beautiful.

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

Note

In the first test case, the given array is already beautiful, as $|a_2 - a_3| = |3 - 3| = 0$.

In the second test case, it is impossible to make the array beautiful, as applying the operation would reduce its size to less than two.

In the third test case, you can, for example, choose a_1 and a_2 and replace them with the number 2. The resulting array $[2, 3, 7]$ is beautiful.

In the fourth test case, you can, for example, choose a_2 and a_3 and replace them with the number 3. The resulting array $[1, 3, 2]$ is beautiful.

,

C. Coloring Game

Input file: standard input
Output file: standard output
Time limit: 2.5 seconds
Memory limit: 256 megabytes

Alice and Bob are playing a game using an integer array a of size n.

Initially, all elements of the array are colorless. First, Alice chooses 3 elements and colors them red. Then Bob chooses any element and colors it blue (if it was red — recolor it). Alice wins if the sum of the red elements is strictly greater than the value of the blue element.

Your task is to calculate the number of ways that Alice can choose 3 elements in order to win regardless of Bob's actions.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains a single integer n ($3 \le n \le 5000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_1 \le a_2 \le \cdots \le a_n \le 10^5$).

Additional constraint on the input: the sum of n over all test cases doesn't exceed 5000.

Output

For each test case, print a single integer — the number of ways that Alice can choose 3 elements in order to win regardless of Bob's actions.

Standard Input	Standard Output
6 3 1 2 3 4 1 1 2 4 5 7 7 7 7 7 5 1 1 2 2 4 6 2 3 3 4 5 5 5 1 1 1 1 3	0 0 10 2 16 0

Note

In the first two test cases, no matter which three elements Alice chooses, Bob will be able to paint one element blue so that Alice does not win.

In the third test case, Alice can choose any three elements. If Bob colors one of the red elements, the sum of red elements will be 14, and the sum of blue elements will be 7. If Bob chooses an uncolored element, the sum of red elements will be 21, and the sum of blue elements will be 7.

In the fourth test case, Alice can choose either the 1-st, 3-rd and 4-th element, or the 2-nd, 3-rd and 4-th element.

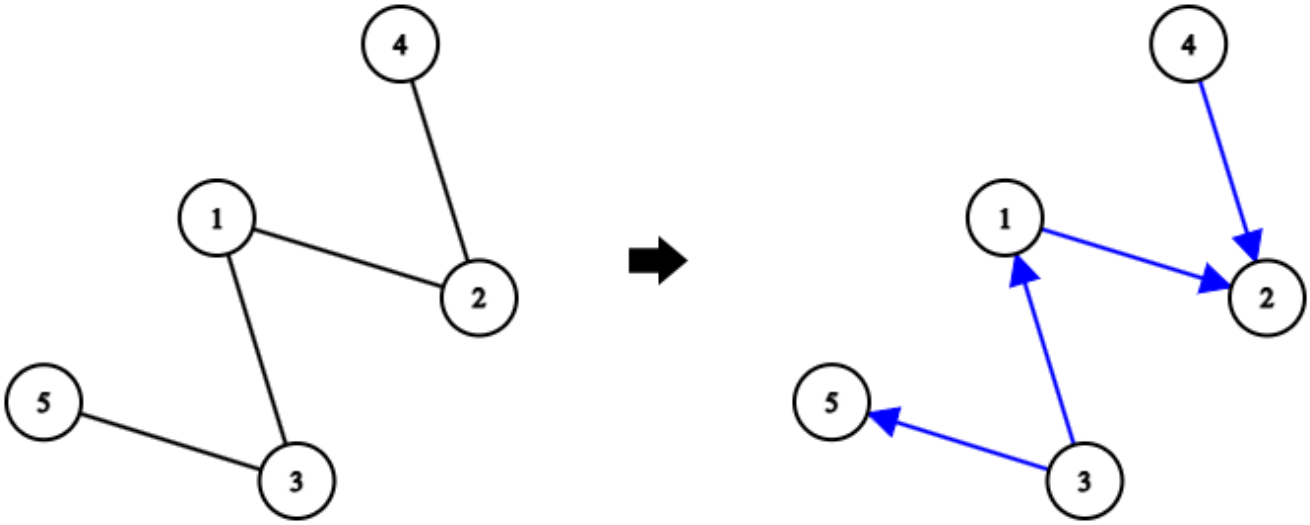
,

D. Reachability and Tree

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

Let u and v be two distinct vertices in a directed graph. Let's call the ordered pair (u, v) *good* if there exists a path from vertex u to vertex v along the edges of the graph.

You are given an undirected tree with n vertices and $n - 1$ edges. Determine whether it is possible to assign a direction to each edge of this tree so that the number of good pairs in it is **exactly** n. If it is possible, print any way to direct the edges resulting in exactly n good pairs.



One possible directed version of the tree for the first test case.

Input

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

The first line of each test case contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the number of vertices in the tree.

The next $n - 1$ lines describe the edges. The i-th line contains two integers u_i and v_i ($1 \le u_i, v_i \le n$; $u_i \ne v_i$) — the vertices connected by the i-th edge.

It is guaranteed that the edges in each test case form an undirected tree and that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print "NO" (case-insensitive) if it is impossible to direct all edges of the tree and obtain exactly n good pairs of vertices.

Otherwise, print "YES" (case-insensitive) and then print $n - 1$ pairs of integers u_i and v_i separated by spaces — the edges directed from u_i to v_i.

The edges can be printed in any order. If there are multiple answers, output any.

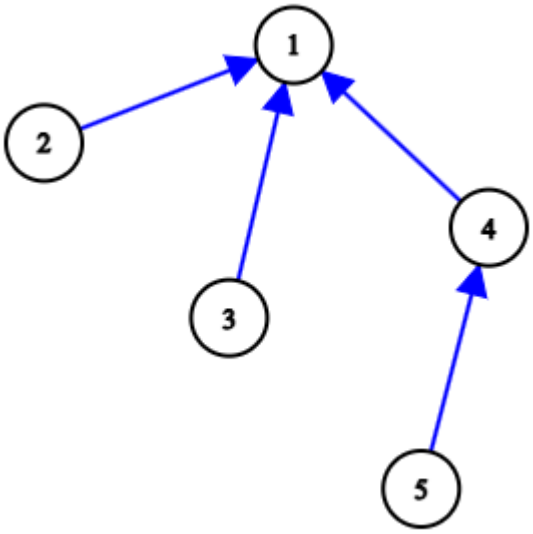
Standard Input	Standard Output

4 5 1 2 2 4 1 3 3 5 5 1 2 1 3 1 4 4 5 2 2 1 4 3 1 1 2 2 4	YES 1 2 3 1 3 5 4 2 YES 2 1 3 1 4 1 5 4 NO YES 1 3 2 1 2 4

Note

The tree from the first test case and its possible directed version are shown in the legend above. In this version, there are exactly 5 good pairs of vertices: $(3, 5)$, $(3, 1)$, $(3, 2)$, $(1, 2)$, and $(4, 2)$.

One possible directed version of the tree from the second test case is shown below:



In the presented answer, there are exactly 5 good pairs of vertices: $(2, 1)$, $(3, 1)$, $(4, 1)$, $(5, 4)$, and $(5, 1)$.

In the third test case, there are only two directed pairs of vertices, but for any direction of the edge, only one pair will be good.

,

E. Tree Colorings

```
Input file:     standard input
Output file:    standard output
Time limit:     4 seconds
Memory limit:   256 megabytes
```

Consider a rooted undirected tree. Each vertex can be colored blue, green, or yellow. A coloring is called *beautiful* if it meets these conditions:

- the root of the tree is green;
- if you consider **all blue and green vertices**, they are reachable from each other without passing through any **yellow** vertices;
- if you consider **all yellow and green vertices**, they are reachable from each other without passing through any **blue** vertices;

You are given an integer m. Your task is to calculate the minimum number of vertices in a tree with **exactly** m *beautiful* colorings.

Input

The first line contains a single integer ($1 \le t \le 10^5$) — the number of test cases.

The only line of each test case contains a single integer m ($1 \le m \le 5 \cdot 10^5$).

Output

For each test case, print a single integer — the minimum number of vertices in a tree with **exactly** m *beautiful* colorings. If such a tree does not exist, print -1.

Standard Input	Standard Output
5 1 3 5 7 9	1 2 3 4 3

Note

In the following notes, let g describe green color, b be blue, and y be yellow.

In the first example, consider a simple tree with just 1 vertex. This tree has exactly 1 beautiful coloring: the root is green.

In the second example, consider a simple tree with 2 vertices with a root at the 1-st vertex. There are exactly 3 beautiful colorings: $[g, g]$, $[g, b]$ and $[g, y]$.

In the third example, consider a bamboo tree with 3 vertices with a root at the 1-st vertex. There are exactly 5 beautiful colorings: $[g, g, g]$, $[g, g, b]$, $[g, g, y]$, $[g, b, b]$ and $[g, y, y]$.

In the fifth example, consider a tree with 3 vertices with a root at the 1-st vertex, and the other 2 vertices connected to it. There are exactly 9 beautiful colorings: $[g, g, g]$, $[g, g, b]$, $[g, g, y]$, $[g, b, g]$, $[g, b, b]$, $[g, b, y]$, $[g, y, g]$, $[g, y, b]$ and $[g, y, y]$.

,

F. Variables and Operations

Input file: standard input
Output file: standard output
Time limit: 5 seconds
Memory limit: 512 megabytes

There are n variables; let's denote the value of the i-th variable as a_i.

There are also m operations which will be applied to these variables; the i-th operation is described by three integers x_i, y_i, z_i. When the i-th operation is applied, the variable x_i gets assigned the following value: $\min(a_{x_i}, a_{y_i} + z_i)$.

Every operation will be applied **exactly once**, but their order is not fixed; they can be applied in any order.

Let's call a sequence of initial variable values $a_1, a_2, \ldots, a_n$ **stable**, if no matter in which order we apply operations, the resulting values will be the same. If the resulting value of the i-th variable depends on the order of operations, then the sequence of initial variable values is called i-**unstable**.

You have to process q queries. In each query, you will be given initial values $a_1, a_2, \ldots, a_n$ and an integer k; before applying the operations, you can at most k times choose a variable and decrease it by 1. For every variable i, you have to independently determine if it is possible to transform the given values into an i-unstable sequence.

Input

The first line contains two integers n and m ($2 \le n \le 500$; $1 \le m \le 4 \cdot 10^5$) — the number of variables and operations, respectively.

Then, m lines follow. The i-th of them contains three integers x_i, y_i, z_i ($1 \le x_i, y_i \le n$; $x_i \ne y_i$; $0 \le z_i \le 10^5$) — the description of the i-th operation.

The next line contains one integer q ($1 \le q \le 1000$) — the number of queries.

Each query consists of two lines:

- the first line contains one integer k ($0 \le k \le 10^9$) — the maximum number of times you can choose a variable and decrease it by 1;
- the second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the initial values of the variables.

Output

For each query, print a string of n zeroes and/or ones. The i-th character should be `1` if it is possible to obtain an i-unstable sequence, or `0` otherwise.

Standard Input	Standard Output
4 5 2 1 10 3 2 5 1 4 8 1 2 6 3 1 17	0000 0000 0110

3 0 20 0 15 5 10 20 0 15 5 30 20 0 15 5	
3 5 1 2 100 1 2 10 1 3 5 1 2 100 3 2 5 1 1000000000 0 0 0	000
3 4 2 3 5 1 2 0 3 1 4 1 3 4 10 5 7 5 3 2 5 7 0 1 1 1 1 5 3 0 1 0 5 3 5 5 6 0 4 5 1 5 6 1 7 7 2 1 1 6 6 4 7 7 2	000 000 000 001 000 001 001 000 000 000

Note

Consider the first example. If the initial variable values are $[20, 0, 15, 5]$, the resulting values will be $[6, 0, 5, 5]$ with any order of operations. Decreasing the variables 10 times is not enough. However, if we can

apply no more than 30 changes, we can decrease the 1-st variable by 2, and the 4-th variable by 25, we get initial values equal to $[18, 0, 15, -20]$, and this sequence is 2-unstable and 3-unstable:

- if you apply the operations in the order they are given, you will get $[-12, 0, 5, -20]$;
- however, if you apply the operations in order $[3, 2, 4, 1, 5]$, you will get $[-12, -2, 5, -20]$;
- and if you apply the operations in order $[3, 4, 5, 1, 2]$, you will get $[-12, -2, 3, -20]$.