A. Square Year

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

One can notice the following remarkable mathematical fact: the number 2025 can be represented as $(20 + 25)^2$.

You are given a year represented by a string s, consisting of exactly 4 characters. Thus, leading zeros are allowed in the year representation. For example, "0001", "0185", "1375" are valid year representations. You need to express it in the form $(a + b)^2$, where a and b are **non-negative integers**, or determine that it is impossible.

For example, if s = "0001", you can choose $a = 0$, $b = 1$, and write the year as $(0 + 1)^2 = 1$.

Input

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

The following lines describe the test cases.

The only line of each test case contains a string s, consisting of exactly 4 characters. Each character is a digit from 0 to 9.

Output

On a separate line for each test case, output:

- Two numbers a and b ($a, b \ge 0$) such that $(a + b)^2 = s$, if they exist. If there are multiple suitable pairs, you may output any of them.
- The number -1 otherwise.

Standard Input	Standard Output
5 0001 1001 1000 4900 2025	0 1 -1 -1 34 36 20 25

,

B. Not Quite a Palindromic String

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

Vlad found a binary string* s of even length n. He considers a pair of indices $(i, n - i + 1)$, where $1 \le i < n - i + 1$, to be good if $s_i = s_{n-i+1}$ holds true.

For example, in the string '010001' there is only 1 good pair, since $s_1 \neq s_6$, $s_2 \neq s_5$, and $s_3 = s_4$. In the string '0101' there are no good pairs.

Vlad loves palindromes, but not too much, so he wants to rearrange some characters of the string so that there are exactly k good pairs of indices.

Determine whether it is possible to rearrange the characters in the given string so that there are exactly k good pairs of indices $(i, n - i + 1)$.

*A string s is called binary if it consists only of the characters '0' and '1'

Input

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

The first line of each test case contains two integers n and k ($2 \le n \le 2 \cdot 10^5$, $0 \le k \le \frac{n}{2}$, n is even) — the length of the string and the required number of good pairs.

The second line of each test case contains a binary string s of length n.

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 "YES" if there is a way to rearrange the characters of the string so that there are exactly k good pairs, 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

6 6 2 000000 2 1 01 4 1 1011 10 2 1101011001 10 1 1101011001 2 1 11	NO NO YES NO YES YES

,

C. Need More Arrays

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

Given an array a and n integers. It is sorted in non-decreasing order, that is, $a_i \le a_{i+1}$ for all $1 \le i < n$.

You can remove any number of elements from the array (including the option of not removing any at all) without changing the order of the remaining elements. After the removals, the following will occur:

- a_1 is written to a new array;
- if $a_1 + 1 < a_2$, then a_2 is written to a new array; otherwise, a_2 is written to the same array as a_1;
- if $a_2 + 1 < a_3$, then a_3 is written to a new array; otherwise, a_3 is written to the same array as a_2;
- $\ldots$

For example, if $a = [1, 2, 4, 6]$, then:

- $a_1 = 1$ is written to the new array, resulting in arrays: $[1]$;
- $a_1 + 1 = 2$, so $a_2 = 2$ is added to the existing array, resulting in arrays: $[1, 2]$;
- $a_2 + 1 = 3$, so $a_3 = 4$ is written to a new array, resulting in arrays: $[1, 2]$ and $[4]$;
- $a_3 + 1 = 5$, so $a_4 = 6$ is written to a new array, resulting in arrays: $[1, 2]$, $[4]$, and $[6]$.

Your task is to remove elements in such a way that the described algorithm creates as many arrays as possible. If you remove all elements from the array, no new arrays will be created.

Input

The first line of input 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 ($1 \le n \le 2 \cdot 10^5$) — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$, $a_i \le a_{i+1}$) — the elements of the array.

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 one integer — the maximum number of arrays that can be obtained by removing any (possibly zero) number of elements.

Standard Input	Standard Output
6 6 1 2 3 4 5 6 3 1 2 3 4 1 2 2 4 1 2 3	3 2 2 1 3 1

1 4 8 2 1 1	

Note

In the first example, you can remove a_3 and a_5, then $a = [1, 2, 4, 6]$, the process of forming arrays for it is shown in the statement.

In the second example, you need to remove a_2, after which $a = [1, 3]$, and the arrays $[1]$ and $[3]$ will be written.

In the third example, no removals are needed; for $a = [1, 2, 2, 4]$, the arrays $[1, 2, 2]$ and $[4]$ will be written.

,

D. Come a Little Closer

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

The game field is a matrix of size $10^9 \times 10^9$, with a cell at the intersection of the a-th row and the b-th column denoted as (a, b).

There are n monsters on the game field, with the i-th monster located in the cell (x_i, y_i), while the other cells are empty. No more than one monster can occupy a single cell.

You can move one monster to any cell on the field that is not occupied by another monster **at most once** .

After that, you must select **one** rectangle on the field; all monsters within the selected rectangle will be destroyed. You must pay 1 coin for each cell in the selected rectangle.

Your task is to find the minimum number of coins required to destroy all the monsters.

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 number of monsters on the field.

The following n lines contain two integers x_i and y_i ($1 \le x_i, y_i \le 10^9$) — the coordinates of the cell with the i-th monster. All pairs (x_i, y_i) are distinct.

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

Output

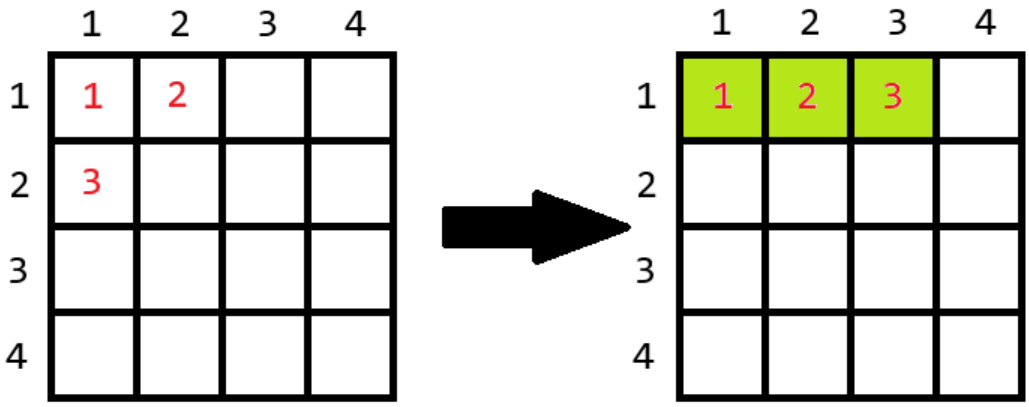
For each test case, output a single integer — the minimum cost to destroy all n monsters.

Standard Input	Standard Output
7 3 1 1 1 2 2 1 5 1 1 2 6 6 4 3 3 8 2 4 1 1 1 1000000000 1000000000 1 1000000000 1000000000 1	3 32 1000000000000000000 1 6 4 8

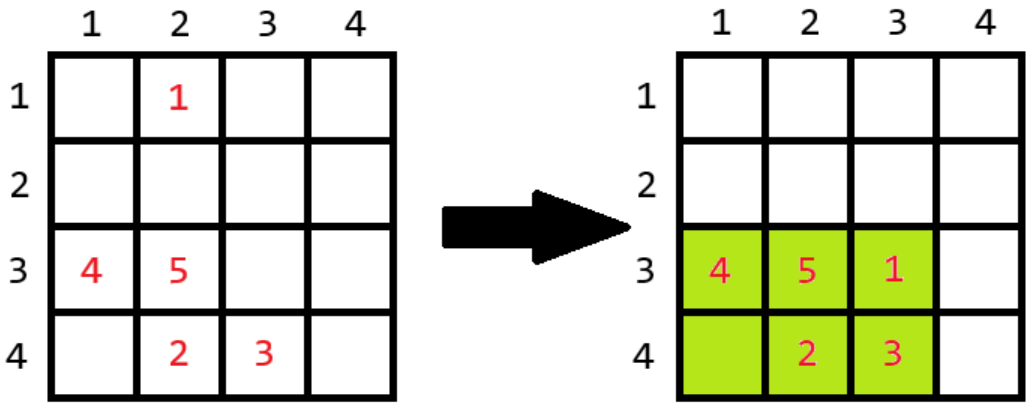
1 1 5 1 2 4 2 4 3 3 1 3 2 3 1 1 2 5 2 2 4 4 3 3 1 4 4 1 2	

Note

Below are examples of optimal moves, with the cells of the rectangle to be selected highlighted in green.



Required move for the first example.



Required move for the fifth example.

,

E. Kirei Attacks the Estate

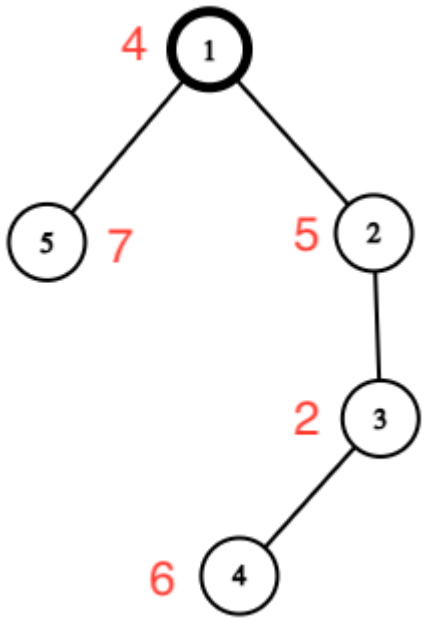
Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	256 megabytes

Once, Kirei stealthily infiltrated the trap-filled estate of the Ainzbern family but was discovered by Kiritugu's familiar. Assessing his strength, Kirei decided to retreat. The estate is represented as a tree with n vertices, with the **root** at vertex 1. Each vertex of the tree has a number a_i recorded, which represents the *danger* of vertex i. Recall that a tree is a connected undirected graph without cycles.

For a successful retreat, Kirei must compute the *threat* value for each vertex. The *threat* of a vertex is equal to the **maximum** *alternating* sum along the vertical path starting from that vertex. The *alternating* sum along the vertical path starting from vertex i is defined as $a_i - a_{p_i} + a_{p_{p_i}} - \ldots$, where p_i is the parent of vertex i on the path to the root (to vertex 1).

For example, in the tree below, vertex 4 has the following vertical paths:

- $[4]$ with an alternating sum of $a_4 = 6$;
- $[4, 3]$ with an alternating sum of $a_4 - a_3 = 6 - 2 = 4$;
- $[4, 3, 2]$ with an alternating sum of $a_4 - a_3 + a_2 = 6 - 2 + 5 = 9$;
- $[4, 3, 2, 1]$ with an alternating sum of $a_4 - a_3 + a_2 - a_1 = 6 - 2 + 5 - 4 = 5$.



The *dangers* of the vertices are indicated in red.

Help Kirei compute the *threat* values for all vertices and escape the estate.

Input

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

The following describes the test cases.

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

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the *dangers* of the vertices.

The next $n - 1$ lines contain the numbers v, u ($1 \le v, u \le n, v \ne u$) — the description of the edges of the tree.

It is guaranteed that the sum of n across all test cases does not exceed $2 \cdot 10^5$. It is also guaranteed that the given set of edges forms a tree.

Output

For each test case, output n integers — the *threat* of each vertex.

Standard Input	Standard Output
2 5 4 5 2 6 7 1 2 3 2 4 3 5 1 6 1000000000 500500500 900900900 9 404 800800800 3 4 5 1 2 5 1 6 6 4	4 5 2 9 7 1000000000 1500500096 1701701691 199199209 404 800800800

Note

The tree from the first test case is depicted in the statement, and the maximum *variable-sign* sums are achieved as follows:

1. $a_1 = 4$;
2. $a_2 = 5$;
3. $a_3 = 2$;
4. $a_4 - a_3 + a_2 = 6 - 2 + 5 = 9$;
5. $a_5 = 7$.

,

F. Small Operations

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

Given an integer x and an integer k. In one operation, you can perform one of two actions:

- choose an integer $1 \leq a \leq k$ and assign $x = x \cdot a$;
- choose an integer $1 \leq a \leq k$ and assign $x = \frac{x}{a}$, where the value of $\frac{x}{a}$ must be an integer.

Find the minimum number of operations required to make the number x equal to y, or determine that it is impossible.

Input

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

The only line of each test case contains three integers x, y and k ($1 \leq x, y, k \leq 10^6$).

It is guaranteed that the sum of x and the sum of y across all test cases does not exceed 10^8.

Output

For each test case, output -1 if it is impossible to achieve $x = y$ using the given operations, and the minimum number of required operations otherwise.

Standard Input	Standard Output
8 4 6 3 4 5 3 4 6 2 10 45 3 780 23 42 11 270 23 1 982800 13 1 6 2	2 -1 -1 3 3 3 6 -1

,

G. Build an Array

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 512 megabytes

Yesterday, Dima found an empty array and decided to add some integers to it. He can perform the following operation an unlimited number of times:

- add any integer to the left or right end of the array.
- then, as long as there is a pair of identical adjacent elements in the array, they will be replaced by their sum.

It can be shown that there can be at most one such pair in the array at the same time.

For example, if the array is $[3, 6, 4]$ and we add the number 3 to the left, the array will first become $[3, 3, 6, 4]$, then the first two elements will be replaced by 6, and the array will become $[6, 6, 4]$, and then — $[12, 4]$.

After performing the operation **exactly** k times, he thinks he has obtained an array a of length n, but he does not remember which operations he applied. Determine if there exists a sequence of k operations that could result in the given array a from an empty array, or determine that it is impossible.

Input

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

The first line of each test case description contains two integers n and k ($1 \le n \le 10^5$, $n \le k \le 10^6$) — the length of the resulting array and the number of operations.

The second line contains n integers a_i ($1 \le a_i \le 10^9$, $a_{i-1} \ne a_i$) — the elements of the resulting array.

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

Output

For each test case, if there is no suitable sequence of operations of length k, output "NO". Otherwise, output "YES".

You may output "YES" and "NO" in any case (for example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as a positive answer).

Standard Input	Standard Output
8 3 3 2 1 4 3 7 2 1 4 2 15 2 16 3 10 256 32 1 3 289	YES NO YES YES YES NO YES YES

768 96 1 3 290 768 96 1 5 7 5 1 6 3 10 4 6 6 8 5 10	