Codeforces Round 1037 (Div. 3)

A. Only One Digit

1 second, 256 megabytes

You are given an integer x. You need to find the smallest non-negative integer y such that the numbers x and y share at least one common digit. In other words, there must exist a decimal digit d that appears in both the representation of the number x and the number y.

Input

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

The first line of each test case contains one integer x $(1 \le x \le 1000)$.

Output

For each test case, output one integer y — the minimum non-negative number that satisfies the condition.

input
```
5
6
96
78
122
696
```
output
```
6
6
7
1
6
```

In the first test case, the numbers 6 and 6 share the common digit '6'. Moreover, there is no natural number smaller than this that shares a common digit.

In the second test case, the numbers 6 and 96 share the common digit '6'.

B. No Casino in the Mountains

1 second, 256 megabytes

You are given an array a of n numbers and a number k. The value a_i describes the weather on the i-th day: if it rains on the i-th day, then $a_i = 1$; otherwise, if the weather is good on the i-th day, then $a_i = 0$.

Jean wants to visit as many peaks as possible. One hike to a peak takes exactly k days, and during each of these days, the weather must be good $(a_i = 0)$. That is, formally, he can start a hike on day i only if all $a_j = 0$ for all j $(i \le j \le i + k - 1)$.

After each hike, before starting the next one, Jean must take a break of at least one day, meaning that on the day following a hike, he cannot go on another hike.

Find the maximum number of peaks that Jean can visit.

Input

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

The first line of each test case contains two integers n and k ($1 \le n \le 10^5, 1 \le k \le n$).

The second line contains n numbers a_i ($a_i \in \{0, 1\}$), where a_i denotes the weather on the i-th day.

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

Output

For each test case, output a single integer: the maximum number of hikes that Jean can make.

input
```
5
5 1
0 1 0 0 0
7 3
0 0 0 0 0 0 0
3 1
1 1 1
4 2
0 1 0 1
6 2
0 0 1 0 0 0
```
output
```
3
2
0
0
2
```

In the first sample:

- Day 1 — good weather, Jean goes on a hike. $(a_1 = 0)$
- Day 2 — mandatory break.
- Day 3 — again good weather, Jean goes on the second hike. ($a_3 = 0)$
- Day 4 — break.
- Day 5 — good weather, third hike. $(a_5 = 0)$

Thus, Jean can make **3 hikes**, alternating each with a mandatory day of rest.
In the second sample:

- From day 1 to day 3 — three days of good weather, Jean goes on a hike. $(a_1 = a_2 = a_3 = 0)$
- Day 4 — mandatory break.
- From day 5 to day 7 — again three days of good weather, Jean goes on the second hike. $(a_5 = a_6 = a_7 = 0)$

In total, Jean makes **2 hikes**.
In the third sample:

- There are no days with good weather ($a_i = 1$ for all i)

Jean cannot make any hikes. **Answer: 0**

C. I Will Definitely Make It

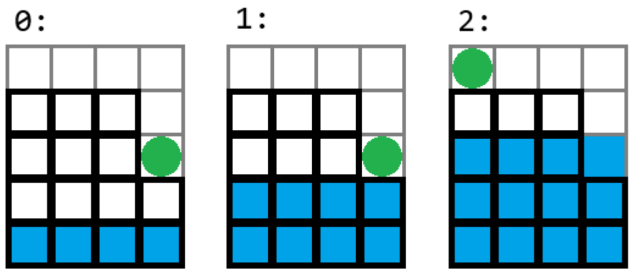
1 second, 256 megabytes

You are given n towers, numbered from 1 to n. Tower i has a height of h_i. At time 0, you are on the tower with index k, and the current water level is 1.

Every second, the water level rises by 1 unit. At any moment, if the water level becomes **strictly greater** than the height of the tower you are on, you perish.

You have a magical ability: at moment x, you can start teleporting from tower i to tower j, which will take $|h_i - h_j|$ seconds. That is, until moment $x + |h_i - h_j|$, you will be on tower i, and at moment $x + |h_i - h_j|$, you will move to tower j. You can start a new teleportation at the same moment you just arrived at tower j.

For example, if $n = k = 4$, $h = [4, 4, 4, 2]$, then if you start teleporting from tower 4 to tower 1 at moment 0, the movement will look as follows:



Note that if the height of tower 1 were 5, you would not be able to teleport to it immediately, as you would be submerged at moment 2.

Your goal is to reach any tower with the maximum height before the water covers you.

Determine if this is possible.

Input

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

The first line of each test case contains two integers n and k ($1 \le k \le n \le 10^5$) — the number of towers and the index of the tower you are initially on.

The second line contains n integers $h_1, h_2, \ldots, h_n$ ($1 \le h_i \le 10^9$) — the heights of the towers.

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

Output

For each test case, output one line: "YES", if you can reach the tower with the maximum height before the water covers you, or "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.

input

```
5
5 3
3 2 1 4 5
3 1
1 3 4
4 4
4 4 4 2
6 2
2 3 6 9 1 2
4 2
1 2 5 6
```

output

```
YES
NO
YES
YES
NO
```

In the first test case, the only possible path is: $3 \to 2 \to 1 \to 4 \to 5$.

In the second test case, regardless of the order, it will not be possible to reach the tallest tower.

In the third test case, one of the possible paths is: $4 \to 1$.

D. This Is the Last Time

2 seconds, 256 megabytes

You are given n casinos, numbered from 1 to n. Each casino is described by three integers: l_i, r_i, and $real_i$ ($l_i \le real_i \le r_i$). You initially have k coins.

You can play at casino i only if the current number of coins x satisfies $l_i \le x \le r_i$. After playing, your number of coins becomes $real_i$.

You can visit the casinos in any order and are not required to visit all of them. Each casino can be visited no more than once.

Your task is to find the maximum final number of coins you can obtain.

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 two integers n and k ($1 \le n \le 10^5$, $0 \le k \le 10^9$) — the number of casinos and the initial number of coins.

This is followed by n lines. In the i-th line, there are three integers l_i, r_i, $real_i$ ($0 \le l_i \le real_i \le r_i \le 10^9$) — the parameters of the i-th casino.

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

Output

For each test case, output a single integer — the maximum number of coins you can obtain after optimally choosing the order of visiting the casinos.

input

```
5
3 1
2 3 3
1 2 2
3 10 10
1 0
1 2 2
1 2
1 2 2
2 2
1 3 2
2 4 4
2 5
1 10 5
3 6 5
```

output

```
10
0
2
4
5
```

In the first test case, you can first play at the 2-nd casino. After that, you will have 2 coins. Then you can play at the 1-st casino, and the number of coins will increase to 3. Finally, after playing at the 3-rd casino, you will have 10 coins — this is the maximum possible amount.

In the second test case, you have no money, so you cannot earn more.

In the fourth test case, it is beneficial to play at the 2-nd casino right away and earn 4 coins.

E. G-C-D, Unlucky!

2 seconds, 256 megabytes

Two arrays p and s of length n are given, where p is the prefix GCD* of some array a, and s is the suffix GCD of the same array a. In other words, if the array a existed, then for each $1 \le i \le n$, the following equalities would hold both:

- $p_i = \gcd(a_1, a_2, \ldots, a_i)$
- $s_i = \gcd(a_i, a_{i+1}, \ldots, a_n)$.

Determine whether there exists such an array a for which the given arrays p and s can be obtained.

*$\gcd(x, y)$ denotes the greatest common divisor (GCD) of integers x and y.

Input

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

Each test case consists of three lines:

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

The second line of each test case contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 10^9$) — the elements of the array.

The third line of each test case contains n integers $s_1, s_2, \ldots, s_n$ ($1 \le s_i \le 10^9$) — the elements of the array.

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

Output

For each test case, output "Yes" (without quotes) if there exists an array a for which the given arrays p and s can be obtained, and "No" (without quotes) 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.

input

```
5
6
72 24 3 3 3 3
3 3 3 6 12 144
3
1 2 3
4 5 6
5
125 125 125 25 25
25 25 25 25 75
4
123 421 282 251
125 1981 239 223
3
124 521 125
125 121 121
```

output

```
YES
NO
YES
NO
NO
```

For the first test case, a possible array is: $[72,\ 24,\ 3,\ 6,\ 12,\ 144]$.

For the second test case, it can be shown that such arrays do not exist.

For the third test case, there exists an array: $[125,\ 125,\ 125,\ 25,\ 75]$.

The problem statement has recently been changed. View the changes. ×

F. 1-1-1, Free Tree!

4 seconds, 256 megabytes

Given a tree* with n vertices numbered from 1 to n. Each vertex has an initial color a_i.

Each edge of the tree is defined by three numbers: u_i, v_i, and c_i, where u_i and v_i are the endpoints of the edge, and c_i is the edge parameter. The cost of the edge is defined as follows: if the colors of vertices u_i and v_i are the same, the cost is 0; otherwise, the cost is c_i.

You are also given q queries. Each query has the form: repaint vertex v to color x. The queries depend on each other (after each query, the color change is preserved). After each query, you need to output the sum of the costs of all edges in the tree.

*A tree is a connected graph without cycles.

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 q ($1 \le n, q \le 2 \cdot 10^5$) — the number of vertices and the number of queries, respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le n)$, where the i-th number specifies the initial color of vertex i.

The next $n-1$ lines describe the edges of the tree. Each line contains three integers u, v, and c, denoting an edge between vertices u and v with parameter c $(1 \le u, v \le n, 1 \le c \le 10^9)$.

The following q lines contain the queries. Each query contains two integers v and x — repaint vertex v to color x $(1 \le v, x \le n)$.

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

Output

For each query, output a single integer on a separate line — the sum of the costs of all edges in the tree after applying the corresponding query.

input

```
4
1 1
1
1 1
2 3
1 1
1 2 10
1 2
2 2
1 1
5 4
1 2 1 2 3
1 2 5
2 3 3
2 4 4
4 5 7
3 2
5 2
1 2
2 3
4 3
1 1 2 2
1 2 2
2 3 6
2 4 8
3 1
4 1
2 2
```

output

```
0
10
0
10
12
5
0
12
8
0
16
```

First test: $n = 1$, one vertex — no edges. Query: repaint a_1 to 1, the sum of costs is 0.

Second test: $n = 2$, edge $1 - 2$ ($c = 10$). Queries:

- $a_1 = 2$: colors $[2, 1]$, cost is 10;
- $a_2 = 2$: colors $[2, 2]$, cost 0;
- $a_1 = 1$: colors $[1, 2]$, cost 10.

Third test: $n = 5$, edges: $1 - 2\ (c = 5)$, $2 - 3\ (c = 3)$, $2 - 4\ (c = 4)$, $4 - 5\ (c = 7)$. Initial colors $[1, 2, 1, 2, 3]$. Queries:

$a_3 = 2 \to [1, 2, 2, 2, 3]$: edges $1 - 2\ (c = 5)$ and $4 - 5\ (c = 7)$ give 12;

$a_5 = 2 \to [1, 2, 2, 2, 2]$: edge $1 - 2\ (c = 5)$, cost 5;

$a_1 = 2 \to [2, 2, 2, 2, 2]$: cost is 0;

$a_2 = 3 \to [2, 3, 2, 2, 2]$: edges $1 - 2\ (5)$, $2 - 3\ (3)$, $2 - 4\ (4)$ give 12.

G1. Big Wins! (easy version)

4 seconds, 256 megabytes

This is the easy version of the problem. The difference between the versions is that in this version $a_i \leq \min(n, 100)$.

You are given an array of n integers $a_1, a_2, \ldots, a_n$.

Your task is to find a subarray $a[l, r]$ (a continuous sequence of elements $a_l, a_{l+1}, \ldots, a_r$) for which the value of the expression $\operatorname{med}(a[l, r]) - \min(a[l, r])$ is maximized.

Here:

- med — the median of the subarray, which is the element at position $\left\lceil \frac{k+1}{2} \right\rceil$ after sorting the subarray, where k is its length;
- $\min$ — the minimum element of this subarray.

For example, consider the array $a = [1, 4, 1, 5, 3, 3]$ and choose the subarray $a[2, 5] = [4, 1, 5, 3]$. In sorted form, it looks like $[1, 3, 4, 5]$.

- $\operatorname{med}(a[2, 5]) = 4$, since $\left\lceil \frac{4+1}{2} \right\rceil =$ the third element in the sorted subarray is 4;
- $\min(a[2, 5]) = 1$, since the minimum element is 1.

In this example, the value $\operatorname{med} - \min = 4 - 1 = 3$.

Input

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

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

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq \min(n, 100)$) — 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 a single integer — the maximum possible value of $\operatorname{med} - \min$ among all subarrays of the array.

input

```
5
5
3 2 5 3 1
4
4 1 1 3
7
6 1 3 4 6 2 7
4
4 2 3 1
5
1 2 3 4 5
```

output

```
3
3
5
2
2
```

In the first example, consider the array: $a = [3,\ 2,\ 5,\ 3,\ 1]$ you can choose the subarray $a[2,\ 3]$, which consists of the elements $[2,\ 5]$.

- The length of the subarray is 2.
- The median is the element at position $\left\lceil \frac{3}{2} \right\rceil = 2$ in the sorted subarray. After sorting, we get $[2,\ 5]$, $\operatorname{med} = 5$.
- The minimum element of the subarray: $\min = 2$.

Therefore, $\operatorname{med} - \min = 5 - 2 = 3$, which is the maximum answer.

In the second test, the array: $a = [4,\ 1,\ 1,\ 3]$ you can choose the subarray $a[1,\ 2]$, which consists of the elements $[4,\ 1]$.

- The length of the subarray is 2.
- The median is the element at position $\left\lceil \frac{3}{2} \right\rceil = 2$ in the sorted subarray. After sorting, we get $[1,\ 4]$, $\text{med} = 4$.
- The minimum element of the subarray: $\min = 1$.

Therefore, $\text{med} - \min = 4 - 1 = 3$.

It can be proven that both of these subarrays are optimal and yield the maximum value of the expression $\text{med} - \min$.

G2. Big Wins! (hard version)

4 seconds, 256 megabytes

This is the hard version of the problem. The difference between the versions is that in this version $a_i \le n$.

You are given an array of n integers $a_1, a_2, \ldots, a_n$.

Your task is to find a subarray $a[l, r]$ (a continuous sequence of elements $a_l, a_{l+1}, \ldots, a_r$) for which the value of the expression $\text{med}(a[l, r]) - \min(a[l, r])$ is maximized.

Here:

- med is the median of the subarray, that is, the element at position $\left\lceil \frac{k+1}{2} \right\rceil$ after sorting the subarray, where k is its length;
- $\min$ is the minimum element of this subarray.

For example, consider the array $a = [1, 4, 1, 5, 3, 3]$ and choose the subarray $a[2, 5] = [4, 1, 5, 3]$. In sorted form, it looks like $[1, 3, 4, 5]$.

- $\text{med}(a[2, 5]) = 4$, since $\left\lceil \frac{4+1}{2} \right\rceil =$ the third element in the sorted subarray is 4;
- $\min(a[2, 5]) = 1$, since the minimum element is 1.

In this example, the value $\text{med} - \min = 4 - 1 = 3$.

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 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 $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — 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 possible value of $\text{med} - \min$ among all subarrays of the array.

input
```
5
5
3 2 5 3 1
4
4 1 1 3
7
6 1 3 4 6 2 7
4
4 2 3 1
5
1 2 3 4 5
```

output
```
3
3
5
2
2
```

In the first example, consider the array: $a = [3,\ 2,\ 5,\ 3,\ 1]$ you can choose the subarray $a[2,\ 3]$, that is, the elements $[2,\ 5]$.

- The length of the subarray is 2.
- The median is the element at position $\left\lceil \frac{3}{2} \right\rceil = 2$ in the sorted subarray. After sorting, we get $[2,\ 5]$, $\text{med} = 5$.
- The minimum element of the subarray: $\min = 2$.

Therefore, $\text{med} - \min = 5 - 2 = 3$, which is the maximum answer.

In the second test, the array: $a = [4,\ 1,\ 1,\ 3]$ you can choose the subarray $a[1,\ 2]$, that is, the elements $[4,\ 1]$.

- The length of the subarray is 2.
- The median is the element at position $\left\lceil \frac{3}{2} \right\rceil = 2$ in the sorted subarray. After sorting, we get $[1,\ 4]$, $\text{med} = 4$.
- The minimum element of the subarray: $\min = 1$.

Therefore, $\text{med} - \min = 4 - 1 = 3$.

It can be proven that both of these subarrays are optimal and yield the maximum value of the expression $\text{med} - \min$.