Educational Codeforces Round 76 (Rated for Div. 2)

A. Two Rival Students

1 second, 256 megabytes

You are the gym teacher in the school.

There are n students in the row. And there are two rivalling students among them. The first one is in position a, the second in position b. Positions are numbered from 1 to n from left to right.

Since they are rivals, you want to maximize the distance between them. If students are in positions p and s respectively, then distance between them is $|p - s|$.

You can do the following operation at most x times: choose two **adjacent (neighbouring)** students and swap them.

Calculate the maximum distance between two rivalling students after at most x swaps.

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of test cases.

The only line of each test case contains four integers n, x, a and b ($2 \le n \le 100$, $0 \le x \le 100$, $1 \le a, b \le n$, $a \ne b$) — the number of students in the row, the number of swaps which you can do, and positions of first and second rivaling students respectively.

Output

For each test case print one integer — the maximum distance between two rivaling students which you can obtain.

input
3 5 1 3 2 100 33 100 1 6 0 2 3

output
2 99 1

In the first test case you can swap students in positions 3 and 4. And then the distance between the rivals is equal to $|4 - 2| = 2$.

In the second test case you don't have to swap students.

In the third test case you can't swap students.

B. Magic Stick

1 second, 256 megabytes

Recently Petya walked in the forest and found a magic stick.

Since Petya really likes numbers, the first thing he learned was spells for changing numbers. So far, he knows only two spells that can be applied to a **positive** integer:

1. If the chosen number a is even, then the spell will turn it into $\frac{3a}{2}$;
2. If the chosen number a is greater than one, then the spell will turn it into $a - 1$.

Note that if the number is even and greater than one, then Petya can choose which spell to apply.

Petya now has only one number x. He wants to know if his favorite number y can be obtained from x using the spells he knows. The spells can be used any number of times in any order. It is not required to use spells, Petya can leave x as it is.

Input

The first line contains single integer T ($1 \le T \le 10^4$) — the number of test cases. Each test case consists of two lines.

The first line of each test case contains two integers x and y ($1 \le x, y \le 10^9$) — the current number and the number that Petya wants to get.

Output

For the i-th test case print the answer on it — YES if Petya can get the number y from the number x using known spells, and NO otherwise.

You may print every letter in any case you want (so, for example, the strings yEs, yes, Yes and YES will all be recognized as positive answer).

input
7 2 3 1 1 3 6 6 8 1 2 4 1 31235 6578234

output
YES YES NO YES NO YES YES

C. Dominated Subarray

2 seconds, 256 megabytes

Let's call an array t *dominated* by value v in the next situation.

At first, array t should have at least 2 elements. Now, let's calculate number of occurrences of each number num in t and define it as $occ(num)$. Then t is dominated (by v) if (and only if) $occ(v) > occ(v')$ for any other number v'. For example, arrays $[1, 2, 3, 4, 5, 2]$, $[11, 11]$ and $[3, 2, 3, 2, 3]$ are dominated (by 2, 11 and 3 respectevitely) but arrays $[3]$, $[1, 2]$ and $[3, 3, 2, 2, 1]$ are not.

Small remark: since any array can be dominated only by one number, we can not specify this number and just say that array is either dominated or not.

You are given array $a_1, a_2, \dots, a_n$. Calculate its shortest dominated subarray or say that there are no such subarrays.

The subarray of a is a contiguous part of the array a, i. e. the array $a_i, a_{i+1}, \dots, a_j$ for some $1 \le i \le j \le n$.

Input

The first line contains single integer T ($1 \le T \le 1000$) — the number of test cases. Each test case consists of two lines.

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

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

It's guaranteed that the total length of all arrays in one test doesn't exceed $2 \cdot 10^5$.

Output
Print T integers — one per test case. For each test case print the only integer — the length of the shortest dominated subarray, or -1 if there are no such subarrays.

input
4 1 1 6 1 2 3 4 5 1 9 4 1 2 4 5 4 3 2 1 4 3 3 3 3
output
-1 6 3 2

In the first test case, there are no subarrays of length at least 2, so the answer is -1.

In the second test case, the whole array is dominated (by 1) and it's the only dominated subarray.

In the third test case, the subarray a_4, a_5, a_6 is the shortest dominated subarray.

In the fourth test case, all subarrays of length more than one are dominated.

D. Yet Another Monster Killing Problem

2 seconds, 256 megabytes

You play a computer game. In this game, you lead a party of m heroes, and you have to clear a dungeon with n monsters. Each monster is characterized by its power a_i. Each hero is characterized by his power p_i and endurance s_i.

The heroes clear the dungeon day by day. In the beginning of each day, you choose a hero (exactly one) who is going to enter the dungeon this day.

When the hero enters the dungeon, he is challenged by the first monster which was not defeated during the previous days (so, if the heroes have already defeated k monsters, the hero fights with the monster $k + 1$). When the hero fights the monster, there are two possible outcomes:

- if the monster's power is strictly greater than the hero's power, the hero retreats from the dungeon. The current day ends;
- otherwise, the monster is defeated.

After defeating a monster, the hero either continues fighting with the next monster or leaves the dungeon. He leaves the dungeon either if he has already defeated the number of monsters equal to his endurance during this day (so, the i-th hero cannot defeat more than s_i monsters during each day), or if all monsters are defeated — otherwise, he fights with the next monster. When the hero leaves the dungeon, the current day ends.

Your goal is to defeat the last monster. What is the minimum number of days that you need to achieve your goal? Each day you have to use exactly one hero; it is possible that some heroes don't fight the monsters at all. Each hero can be used arbitrary number of times.

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

The first line of each test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of monsters in the dungeon.

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

The third line contains one integer m ($1 \le m \le 2 \cdot 10^5$) — the number of heroes in your party.

Then m lines follow, each describing a hero. Each line contains two integers p_i and s_i ($1 \le p_i \le 10^9$, $1 \le s_i \le n$) — the power and the endurance of the i-th hero.

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

Output
For each test case print one integer — the minimum number of days you have to spend to defeat all of the monsters (or -1 if it is impossible).

input
2 6 2 3 11 14 1 8 2 3 2 100 1 5 3 5 100 2 3 2 30 5 90 1
output
5 -1

E. The Contest

2 seconds, 512 megabytes

A team of three programmers is going to play a contest. The contest consists of n problems, numbered from 1 to n. Each problem is printed on a separate sheet of paper. The participants have decided to divide the problem statements into three parts: the first programmer took some prefix of the statements (some number of first paper sheets), the third contestant took some suffix of the statements (some number of last paper sheets), and the second contestant took all remaining problems. But something went wrong — the statements were printed in the wrong order, so the contestants have received the problems in some random order.

The first contestant has received problems $a_{1,1}, a_{1,2}, \ldots, a_{1,k_1}$. The second one has received problems $a_{2,1}, a_{2,2}, \ldots, a_{2,k_2}$. The third one has received all remaining problems ($a_{3,1}, a_{3,2}, \ldots, a_{3,k_3}$).

The contestants don't want to play the contest before they redistribute the statements. They want to redistribute them so that the first contestant receives some prefix of the problemset, the third contestant receives some suffix of the problemset, and the second contestant receives all the remaining problems.

During one move, some contestant may give one of their problems to other contestant. What is the minimum number of moves required to redistribute the problems?

It is possible that after redistribution some participant (or even two of them) will not have any problems.

Input
The first line contains three integers k_1, k_2 and k_3 ($1 \le k_1, k_2, k_3 \le 2 \cdot 10^5, k_1 + k_2 + k_3 \le 2 \cdot 10^5$) — the number of problems initially taken by the first, the second and the third participant, respectively.

The second line contains k_1 integers $a_{1,1}, a_{1,2}, \ldots, a_{1,k_1}$ — the problems initially taken by the first participant.

The third line contains k_2 integers $a_{2,1}, a_{2,2}, \ldots, a_{2,k_2}$ — the problems initially taken by the second participant.

The fourth line contains k_3 integers $a_{3,1}, a_{3,2}, \ldots, a_{3,k_3}$ — the problems initially taken by the third participant.

It is guaranteed that no problem has been taken by two (or three) participants, and each integer $a_{i,j}$ meets the condition $1 \le a_{i,j} \le n$, where $n = k_1 + k_2 + k_3$.

Output
Print one integer — the minimum number of moves required to redistribute the problems so that the first participant gets the prefix of the problemset, the third participant gets the suffix of the problemset, and the second participant gets all of the remaining problems.

input
```
2 1 2
3 1
4
2 5
```
output
```
1
```

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

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

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

In the first example the third contestant should give the problem 2 to the first contestant, so the first contestant has 3 first problems, the third contestant has 1 last problem, and the second contestant has 1 remaining problem.

In the second example the distribution of problems is already valid: the first contestant has 3 first problems, the third contestant has 1 last problem, and the second contestant has 2 remaining problems.

The best course of action in the third example is to give all problems to the third contestant.

The best course of action in the fourth example is to give all problems to the second contestant.

F. Make Them Similar

4 seconds, 1024 megabytes

Let's call two numbers *similar* if their binary representations contain the same number of digits equal to 1. For example:

- 2 and 4 are similar (binary representations are 10 and 100);
- 1337 and 4213 are similar (binary representations are 10100111001 and 1000001110101);
- 3 and 2 are not similar (binary representations are 11 and 10);
- 42 and 13 are similar (binary representations 101010 and 1101).

You are given an array of n integers a_1, a_2, ..., a_n. You may choose a non-negative integer x, and then get another array of n integers b_1, b_2, ..., b_n, where $b_i = a_i \oplus x$ ($\oplus$ denotes bitwise XOR).

Is it possible to obtain an array b where all numbers are similar to each other?

Input
The first line contains one integer n ($2 \le n \le 100$).

The second line contains n integers a_1, a_2, ..., a_n ($0 \le a_i \le 2^{30} - 1$).

Output
If it is impossible to choose x so that all elements in the resulting array are similar to each other, print one integer -1.

Otherwise, print **any** non-negative integer not exceeding $2^{30} - 1$ that can be used as x so that all elements in the resulting array are similar.

input
```
2
7 2
```
output
```
1
```

input
```
4
3 17 6 0
```
output
```
5
```

input
```
3
1 2 3
```
output
```
-1
```

input
3 43 12 12

output
1073709057

G. Divisor Set

5 seconds, 512 megabytes

You are given an integer x represented as a product of n its *prime* divisors $p_1 \cdot p_2, \cdot \ldots \cdot p_n$. Let S be the set of *all* positive integer divisors of x (including 1 and x itself).

We call a set of integers D *good* if (and only if) there is no pair $a \in D$, $b \in D$ such that $a \neq b$ and a divides b.

Find a *good* subset of S with maximum possible size. Since the answer can be large, print the size of the subset modulo 998244353.

Input

The first line contains the single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of prime divisors in representation of x.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($2 \leq p_i \leq 3 \cdot 10^6$) — the prime factorization of x.

Output

Print the maximum possible size of a *good* subset modulo 998244353.

input
3 2999999 43 2999957

output
3

input
6 2 3 2 3 2 2

output
3

In the first sample, $x = 2999999 \cdot 43 \cdot 2999957$ and one of the maximum good subsets is $\{43, 2999957, 2999999\}$.

In the second sample, $x = 2 \cdot 3 \cdot 2 \cdot 3 \cdot 2 \cdot 2 = 144$ and one of the maximum good subsets is $\{9, 12, 16\}$.