ICPC Chennai Regionals 2025-26

A - All Distinct

You are given an array a of length n and an integer k. The array a contains at least one occurrence of each integer from 1 to k. All the elements of a are in the range $[1, k]$.
We call the array a *good* if it contains at least one subarray of length k such that all its elements are distinct. Formally, a is called *good* if and only if there exists an integer i such that:

- $1 \leq i \leq n - k + 1$; and
- The elements $\{a_i, a_{i+1}, \ldots, a_{i+k-1}\}$ are pairwise distinct.

Your task is to count the number of pairs (l, r) satisfying the following conditions:

- $1 \leq l \leq r \leq n$; and
- It is possible to shuffle the subarray $a[l, r]$ so that the array a becomes *good*. That is, if you are allowed to arrange the elements $[a_l, a_{l+1}, \ldots, a_r]$ freely among positions $\{l, l + 1, \ldots, r\}$, it must be possible to make the array a *good*.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains two space-separated integers n and k — the length of the array and its maximum element.
  - The second line contains n space-separated integers $a_1, \ldots, a_n$.

Output Format

- For each test case, output a single integer — the number of pairs (l, r) such that reshuffling $a[l, r]$ makes the array a *good*.

Constraints

- $1 \leq t \leq 10^4$
- $1 \leq k \leq n \leq 10^6$
- $1 \leq a_i \leq k$
- The array a contains at least one occurrence of each integer from 1 to k.
- The sum of n over all test cases won't exceed 10^6.

Sample Input 1

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

Sample Output 1

```
6
15
8
8
```

Explanation

- **Test case** 1**:** The array is initially *good*, so trivially any pair (l, r) is valid.
- **Test case** 4**:** The valid pairs are as follows:
  - $(1, 2), (1, 3), (1, 4), (1, 5)$
  - $(2, 4), (2, 5)$
  - $(3, 4), (3, 5)$

B - Connect 3

You are given a tree consisting of n vertices. The vertices are numbered from 1 to n. Each vertex i has a value c_i associated with it, where $c_i \in \{1, 2, 3\}$. It is guaranteed that each value appears at least once.
You can perform the following operation any number of times:

- Choose two distinct vertices i and j $(1 \leq i, j \leq n)$.
- Swap the values c_i and c_j.

Your goal is to reach a configuration where, for each $k \in \{1, 2, 3\}$, the set of vertices with value k forms a connected set. A set of vertices S forms a connected set if for any two vertices $u, v \in S$, the (unique) simple path between u and v in the tree consists only of vertices that also belong to S.
Find the minimum number of operations required to achieve this goal. If it is impossible to make all three values connected, output -1.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of multiple lines of input:
  - The first line of each test case contains a single integer n — the number of vertices in the tree.
  - The second line contains n space-separated integers $c_1, \ldots, c_n$, denoting the initial values of the vertices.
  - The next $n - 1$ lines describe the edges. The i-th of these lines contains two space-separated integers u_i and v_i, denoting an edge between u_i and v_i.

Output Format

- For each test case, output a single integer — the minimum number of swaps needed. If it is impossible, output -1.

Constraints

- $1 \leq t \leq 10^5$
- $2 \leq n \leq 5 \cdot 10^5$
- $c_i \in \{1, 2, 3\}$
- Every value in $\{1, 2, 3\}$ appears at least once.
- The input edges describe a tree on $\{1, 2, \ldots, n\}$.
- The sum of n over all test cases won't exceed $5 \cdot 10^5$.

Sample Input 1

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

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

Sample Output 1

```
0
1
-1
2
```

Explanation

- **Test case 1:** No swaps are needed.
- **Test case 2:** One valid solution is to swap the values of vertices 4 and 5, so that the value array is $[3, 2, 1, 3, 2]$. Now,
  - Value 1: the vertex set is $\{3\}$, which is connected.
  - Value 2: the vertex set is $\{2, 5\}$, which is connected.
  - Value 3: the vertex set is $\{1, 4\}$, which is connected.

  So, one swap is optimal. Note that the initial configuration is invalid because vertices 1 and 5 have value 3, but $\{1, 5\}$ is not a connected subset.
- **Test case 3:** It can be shown that no sequence of swaps can result in every value being connected.

C - Deletion Minimization

An array c of length m is said to be *k-good* if it can be rearranged to get an array b such that

$$\gcd(|b_1 - b_2|, |b_2 - b_3|, \ldots, |b_{m-1} - b_m|) = k$$

That is, the greatest common divisors of all adjacent differences of b must equal k. Here, $|x|$ denotes the absolute value of x. For example, $|1| = 1$, $|-2| = 2$, $|0| = 0$.
Further, all arrays of length 1 are considered to be *k-good* for every k.

You are given an array a of length n. You are also given an integer k.
You would like the array a to become *k-good*. Find the minimum number of elements that must be deleted from a for it to become *k-good*.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains two space-separated integers n and k — the length of the array a and the target GCD.
  - The second line contains n space-separated integers $a_1, \ldots, a_n$.

Output Format

- For each test case, output a single integer — the minimum number of elements you should delete from a so that the remaining array is *k-good*.

Constraints

- $1 \leq t \leq 10^4$
- $2 \leq n \leq 2 \cdot 10^5$
- $1 \leq k \leq 10^{18}$
- $1 \leq a_i \leq 10^{18}$
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

```
3
4 1
1 2 3 4
3 10
100 10 1
2 5
100 100
```

Sample Output 1

```
0
2
1
```

Explanation

- **Test case 1:** The given array is already 1-*good*, because without rearranging we obtain $\gcd(|1-2|, |2-3|, |3-4|) = 1$. So, no deletions are needed.
- **Test case 2:** The only way to make the array 10-*good* is to delete two elements to end up with a length-1 array (which is by definition 10-*good*).

D - Game of Three

Alice and Bob are playing a game with **three** positive integers a_1, a_2, and a_3. The players take turns, with Alice moving first.
On their turn, the current player can perform the following operation:

- Choose two distinct indices i and j ($i \neq j$) such that both $a_i > 0$ and $a_j > 0$.
- Calculate $d = |a_i - a_j|$. Here $|x|$ denotes the absolute value of x. For example, $|1| = 1$, $|-2| = 2$, $|0| = 0$.
- Set both $a_i = d$ and $a_j = d$.

The game ends when a player cannot perform an operation. This happens when there are fewer than two positive integers among $[a_1, a_2, a_3]$.
The player who first cannot perform an operation **wins** the game. In other words, the **last** player to make a valid move loses.
Determine whether Alice has a winning strategy.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of a single line of input, containing three space-separated integers a_1, a_2 and a_3.

Output Format

- For each test case, output `Yes` if Alice has a winning strategy, and `No` otherwise.
- You can output the answer in any case (upper or lower). For example, the strings `yEs`, `yes`, `Yes`, and `YES` will be recognized as positive responses.

Constraints

- $1 \leq t \leq 10^5$
- $1 \leq a_1, a_2, a_3 \leq 10^9$

Sample Input 1

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

Sample Output 1

```
No
No
Yes
Yes
Yes
No
```

Explanation

- **Test case 1:** Alice is forced to choose two ones on her move. So, on Bob's turn the values will always be $[0, 0, 1]$, and he cannot make a move — thus Bob always wins.
- **Test case 3:** Alice has a winning strategy as follows:
  - On her first move, she chooses $i = 1$ and $j = 3$. This does the following:
    - $d = |a_1 - a_3| = 2$ is the difference.
    - Set $a_1 = a_3 = 2$.
    - The values are now $[2, 2, 2]$.
  - Bob now has to play with $[2, 2, 2]$. No matter what he does, the set of values after his move will be $[0, 0, 2]$, so Alice cannot make a move and wins.

E - Mex and Max

You are given a set S, consisting of n **distinct** non-negative integers.
In one move, you can perform exactly one of the following two operations:

- Insert $\text{mex}(S)$ into S.
- Delete $\max(S)$ from S. This can only be done if S is not empty.

Here, $\text{mex}(S)$ denotes the minimum excluded value of S, which is the smallest non-negative integer not present in S. For example, $\text{mex}(\{0, 1, 3, 1\}) = 2$ and $\text{mex}(\{1, 2, 5\}) = 0$.
Given S and an integer k, find the number of distinct sets that are reachable after performing **at most** k moves, where you can choose which operation to perform in each move.
Two sets are considered distinct if one of them contains an element that the other does not.
Since this value might be large, you only need to find it modulo 998 244 353.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains two space-separated integers n and k — the size of the set S and the number of allowed operations.
  - The second line contains n space-separated integers $s_1, \ldots, s_n$, denoting the initial elements of the set.

Output Format

- For each test case, output a single integer — the number of distinct states reachable after at most k moves, modulo 998 244 353.

Constraints

- $1 \leq t \leq 10^4$
- $1 \leq n \leq 2 \cdot 10^5$
- $1 \leq k \leq 10^9$
- $0 \leq s_i \leq 10^9$
- $s_i \neq s_j$ for $i \neq j$
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

```
4
3 1
2 0 1
4 2
3 4 5 9
3 3
0 1 7
2 1000000000
1 5
```

Sample Output 1

```
3
6
9
1755655
```

Explanation

- **Test case 1:** $k = 1$, so we can perform at most one operation. The reachable sets are as follows:
  - $\{0, 1, 2\}$, by not performing any operation
  - $\{0, 1\}$, by deleting the maximum element with one move.
  - $\{0, 1, 2, 3\}$, by inserting the mex with one move.

F - Minimum Divides Maximum

You are given an integer n.
You want to find a permutation[†] p of the integers $\{1, 2, \ldots, n\}$ such that the number of indices i ($1 \leq i \leq n$) where $\max(p_1, p_2, \ldots, p_i)$ is divisible by $\min(p_1, p_2, \ldots, p_i)$ is as small as possible.
Print the minimum value possible if you choose p optimally.
[†] A permutation of the integers $\{1, 2, \ldots, n\}$ is an array of length n that contains every integer from 1 to n exactly once each.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of a single line of input, containing one integer n.

Output Format

- For each test case, output on a new line the minimum number of indices i such that $\min(p_1, \ldots, p_i)$ divides $\max(p_1, \ldots, p_i)$, if you choose p optimally.

Constraints

- $1 \leq t, n \leq 5000$

Sample Input 1

```
3
7
2
1
```

Sample Output 1

```
2
2
1
```

Explanation

- **Test case 1:** Consider $p = \{7, 2, 3, 6, 4, 5, 1\}$. The maximum in $p_1, p_2, \ldots, p_i$ is divisible by the minimum in $p_1, p_2, \ldots, p_i$, only for $i = 1$ and $i = 7$.

G - Palindrome and Permutation

Suppose you have an array b of length m. Your goal is to make b a palindrome. The array b of length m is said to be a palindrome if $b_i = b_{m+1-i}$ for every $1 \leq i \leq m$.
To achieve this, you will choose a permutation p of $\{1, 2, \ldots, m\}$ and perform the following process using it:

- In the i-th step ($1 \leq i \leq m$), you must choose **exactly one** index j ($1 \leq j \leq m$) and set $a_j = p_i$.

Note that it is allowed to choose the same index j in different steps, if you wish to do so.
The process ends either after all m indices have been processed, or immediately after b becomes a palindrome for the first time.
Define $f(b, p)$ to be the minimum number of steps after which b can become a palindrome, if you choose the indices to operate on optimally. In particular,

- If b is already a palindrome initially, $f(b, p) = 0$.
- If it is impossible to make b a palindrome even after processing every element of p, we define $f(b, p) = m + 1$.

You are given an array a of length n. Calculate the sum of $f(a, p)$ over all possible permutations p of $\{1, 2, \ldots, n\}$.
The answer might be large, so output it modulo 998 244 353.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains a single integer n — the length of the array a.
  - The second line contains n space-separated integers $a_1, \ldots, a_n$.

Output Format

- For each test case, output a single integer — the sum of $f(a, p)$ across all p, modulo 998 244 353.

Constraints

- $1 \leq t \leq 10^5$
- $1 \leq n \leq 5000$
- $1 \leq a_i \leq n$
- The sum of n^2 over all test cases won't exceed 5000^2.

Sample Input 1

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

Sample Output 1

```
8
0
5040
31363200
```

Explanation

- **Test case 1:** We have the following:
  - For $p = [1, 2, 3]$ and $p = [1, 3, 2]$, the minimum number of moves needed is 1 since a can be turned into $[1, 2, 1]$ on the first move.
  - For $p = [3, 1, 2]$ and $p = [3, 2, 1]$, the minimum number of moves needed is again 1 since a can be turned into $[3, 2, 3]$ on the first move.
  - For $p = [2, 1, 3]$ and $p = [2, 3, 1]$, the minimum number of moves needed is 2. One move is clearly not enough, and two moves will suffice as follows:
    - For $p = [2, 1, 3]$, choose index 2 for the first move and index 3 for the second move.
    - For $p = [2, 3, 1]$, choose index 2 for the first move and index 1 for the second move.
- **Test case 2:** The array is already a palindrome, so $f(a, p) = 0$ for all p.
- **Test case 3:** It can be proved that a can never be made a palindrome no matter what p is. So, $f(a, p) = n+1 = 7$ for all p, and the answer is $7 \cdot 6! = 5040$.

H - Robin Hood

You are given an integer array a of length n.
You can perform the following operation any number of times:

- Choose an element x that is currently in the array.
- Remove one copy of x from the array.
- For every remaining element y in the array such that $y < x$, increase the value of y by 1.

For example, if the array is initially $[1, 2, 4, 4, 5]$ and you choose $x = 4$, the array turns into $[2, 3, 4, 5]$.
Find the **minimum** number of moves required to obtain an array that has all its elements equal.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains a single integer n — the length of the array.
  - The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$.

Output Format

- For each test case, output a single integer — the minimum number of moves needed to make all remaining elements equal.

Constraints

- $1 \leq t \leq 10^5$
- $1 \leq n \leq 2 \cdot 10^5$
- $1 \leq a_i \leq 10^9$
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

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

Sample Output 1

```
3
0
1
4
1
```

Explanation

- **Test case 1:** After making any three moves, we'll be left with a single element — so all remaining elements are clearly equal. It can be proven that it's impossible to achieve this in less than three moves.
- **Test case 2:** All elements are already equal.
- **Test case 3:** Choose $x = 4$, which will turn the array into $[8, 8]$. All elements are now equal.

I - Set Merging

You are given an integer n. You have n sets, denoted $s_1, s_2, \ldots, s_n$. Initially, $s_i = \{i\}$, that is, the i-th set contains the single element i. Your goal is to merge all the sets into a single set.
To achieve your goal, you can perform the following operation:

- Select two indices i and j such that $i \neq j$, $s_i \neq \emptyset$, and $s_j \neq \emptyset$. That is, choose two distinct non-empty sets.
- If this is the k-th operation you're performing, create a new set s_{n+k}.
- Set $s_{n+k} = s_i \cup s_j$, that is, the new set will be equal to the union of the two chosen sets.
- This operation has a cost of $\min(|s_i|, |s_j|)$, where $|s_i|$ denotes the size of set s_i.
- Then, set $s_i = \emptyset$ and $s_j = \emptyset$.

Note that the number of non-empty sets reduces by 1 in each operation. So, after exactly $n - 1$ operations, you will have exactly one non-empty set (which will be s_{2n-1}), and the process will end.
You are also given an integer k. Determine whether it is possible to achieve a cost of **exactly** k after performing $n-1$ operations. If it is possible, you also need to output a valid sequence of operations — see the output format below for how.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of a single line of input, containing two space-separated integers n and k — the number of initial sets, and the required merge cost, respectively.

Output Format

- For each test case:
  - If it is impossible to achieve a score of exactly k, output a single integer -1.
  - Otherwise, output $n - 1$ lines. The i-th of these lines should contain two **distinct** integers x_i and y_i ($1 \leq x_i, y_i \leq n + i - 1$) — denoting the indices of the sets you want to merge in the i-th operation. Note that the initial sets are indexed 1 to n, and the new set created in the k-th operation is assigned index $n+k$. **Once sets with indices x_i and y_i are merged, they are considered deleted and cannot be selected again in future operations.**
- If there are multiple valid sequences, print any of them.

Constraints

- $1 \leq t \leq 10^5$
- $1 \leq n \leq 2 \cdot 10^5$
- $1 \leq k \leq 10^{12}$
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

```
4
5 3
5 4
4 4
3 43
```

Sample Output 1

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

Explanation

- **Test case** 1**:** It can be proven that it is impossible to achieve a cost of 3 after performing 4 operations.
- **Test case** 2**:**
  - In the first operation, we select sets 1 and 2. They both have size equal to 1, so the cost of the operation is 1, and the set 6 is created of size 2.
  - In the second operation, we select sets 3 and 6. The set 3 has size 1 and set 6 has size 2, so the cost of the operation is 1, and the set 7 is created of size 3.
  - In the third operation, we select sets 4 and 7. The set 4 has size 1 and set 7 has size 3, so the cost of the operation is 1, and the set 8 is created of size 4.
  - In the fourth operation, we select sets 5 and 8. The set 5 has size 1 and set 8 has size 4, so the cost of the operation is 1, and the set 9 is created of size 5.
- **Test case** 3**:**
  - In the first operation, we select sets 1 and 2. They both have size equal to 1, so the cost of the operation is 1, and the set 5 is created of size 2.
  - In the second operation, we select sets 3 and 4. They both have size equal to 1, so the cost of the operation is 1, and the set 6 is created of size 2.
  - In the third operation, we select sets 5 and 6. They both have size equal to 2, so the cost of the operation is 2, and the set 7 is created of size 4.

J - Vertex Separation

Consider an undirected graph. A pair of vertices (u, v) are said to be *separated* if the following condition holds:

- There exist two simple paths, both from u to v, whose lengths differ by *at least* 2. Note that a simple path is a path where all vertices are distinct.

A vertex u is said to be *separable* if there exists at least one other vertex v such that the pair (u, v) is separated.

You are given three integers n, m, and k. Construct any **simple connected** undirected graph with n vertices and m edges, such that it has **exactly** k separable vertices. If no such graph exists, print -1.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of a single line of input, containing three space-separated integers n, m, and k — the required graph parameters.

Output Format

- For each test case:
  - If it is impossible to construct such a graph, output a single integer -1.
  - Otherwise, output m lines. The i-th of these lines should contain two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \neq v_i$) — the endpoints of the i-th edge. The graph must be **simple** (i.e. no self-loops and no repeated edges), **connected**, and have exactly k separable vertices.
- If there are multiple valid graphs, any of them will be accepted.

Constraints

- $1 \le t \le 10^5$
- $1 \le n \le 2 \cdot 10^5$
- $n - 1 \le m \le \min\left(2 \cdot 10^5, \dfrac{n \cdot (n-1)}{2}\right)$
- $0 \le k \le n$
- The sum of n and the sum of m over all test cases each won't exceed $2 \cdot 10^5$.

Sample Input 1

```
3
2 1 1
3 2 0
4 5 4
```

Sample Output 1

```
-1
1 2
2 3
1 2
2 3
3 4
1 4
1 3
```

Explanation

- **Test case 1:** There's only one possible graph with $n = 2$ and $m = 1$ (which is just a single edge), and it has 0 separable vertices.
- **Test case 2:** We want 0 separable vertices, which the given graph attains.
- **Test case 3:** All vertices are separable. For instance, vertex 1 is separable because of the pair $(1, 2)$, where the two paths $1 \to 2$ and $1 \to 4 \to 3 \to 2$ both exist and differ in length by 2.

K - X Makes the Difference

You are given an integer n.
We call a positive integer x *good* with respect to n, if there exists a permutation[†] p of the integers $\{1, 2, \ldots, n\}$ such that

$$|p_i - p_{n+1-i}| = x$$

for *all* $1 \leq i \leq n$. Here, $|d|$ denotes the absolute value of d. For example, $|1| = 1$, $|-2| = 2$, $|0| = 0$.
Count the number of integers that are *good* with respect to the given integer n.
[†] A permutation of the integers $\{1, 2, \ldots, n\}$ is an array of length n that contains every integer from 1 to n exactly once each.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of a single line of input, containing one integer n.

Output Format

- For each test case, output a single integer — the number of *good* integers x.

Constraints

- $1 \leq t \leq 10^5$
- $1 \leq n \leq 2 \cdot 10^5$
- There is **no constraint** on the sum of n across all tests.

Sample Input 1

```
3
2
3
4
```

Sample Output 1

```
1
0
2
```

Explanation

- **Test case 1:** The only good integer with respect to $n = 2$ is $x = 1$. Both $[1, 2]$ and $[2, 1]$ are permutations that make $x = 1$ good.
- **Test case 2:** It can be proved that no integer is good with respect to $n = 3$.