Day 76 coding Statement :

You are given **N** integers. In each step you can choose some **K** of the remaining numbers and delete them, if the following condition holds: Let the **K** numbers you've chosen be a_1, a_2, a_3, ..., a_K in sorted order. Then, for each i ≤ **K** - 1, a_{i+1} must be greater than or equal to a_i * **C**.

You are asked to calculate the maximum number of steps you can possibly make.

Input

- The first line of the input contains an integer **T**, denoting the number of test cases. The description of each testcase follows.
- The first line of each testcase contains three integers: **N**, **K**, and **C**
- The second line of each testcase contains the **N** initial numbers

Output

For each test case output the answer in a new line.

Sample Input

2

6 3 2

4 1 2 2 3 1

6 3 2

1 2 2 1 4 4

Sample Output

1

2

```
import java.util.*;
import java.lang.*;
import java.io.*;

public class Program {
	static boolean isPoss(int x, long[] arr, int k, int c) {
```

```
		ArrayList<ArrayList<Long>> list = new ArrayList<>();
		int cur = 0, n = arr.length;
		for (int i = 0; i < x; i++) {
			list.add(new ArrayList<Long>());
		}
		for (int i = 0; i < n; i++) {
			cur = cur % x;
			int sz = list.get(cur).size() - 1;
			if (sz < 0 || list.get(cur).get(sz) * c <= arr[i]) {
				list.get(cur).add(arr[i]);
				cur = (cur + 1) % x;
			}
		}
		if (list.get(x - 1).size() >= k)
			return true;

		return false;
	}

	static long divset(long[] arr, int k, int c) {
		int n = arr.length;
		int l = 1, r = n;// To avoid zero x
		int res = 0;
		Arrays.sort(arr);
		while (l <= r) {
			int mid = l + (r - l) / 2;

			if (isPoss(mid, arr, k, c)) {
				l = mid + 1;
				res = mid;
			} else
				r = mid - 1;
		}
		return res;
	}

	public static void main(String[] args) throws java.lang.Exception {
		BufferedReader bf = new BufferedReader(new
InputStreamReader(System.in));
		int t = Integer.parseInt(bf.readLine());
		StringBuffer str = new StringBuffer("");
		while (t-- > 0) {
			String s[] = bf.readLine().trim().split("\\s+");
			int n = Integer.parseInt(s[0]);
			int k = Integer.parseInt(s[1]);
			int c = Integer.parseInt(s[2]);
			long arr[] = new long[n];
			s = bf.readLine().trim().split("\\s+");
			for (int i = 0; i < n; i++)
				arr[i] = Long.parseLong(s[i]);

			str.append(divset(arr, k, c) + "\n");
		}
		System.out.println(str);
	}
```

```
}
```