Question1-(Hash set or Hash map Can be used to solve)

Problem Statement: Alice is a 10-year-old girl who has a sweet tooth and loves collecting and eating various types of candies. Over time, she has gathered a collection of candies, where each candy belongs to a specific type, such as chocolate, lollipop, gummy bear, or hard candy. This collection is represented as an array `candyType`, where each element indicates the type of a specific candy. For instance, `candyType = [1, 1, 2, 2, 3, 3]` implies there are 6 candies, with three different types.

After a routine health check-up, Alice's doctor advises her to reduce her candy intake due to concerns about her weight and overall health. The doctor recommends that she should consume no more than half of the total candies she owns. However, Alice loves variety and wishes to enjoy as many different types of candies as possible within the doctor's restriction.

As a developer working on a child wellness app, your task is to implement a function that helps Alice determine the maximum number of different candy types she can eat, given that she is allowed to eat only half of her total candies.

Design and implement an efficient solution to support healthy and mindful eating in children.

Constraints:

n == candyType.length
2 <= n <= 10^4
n is even.
-10^5 <= candyType[i] <= 10^5

Example 1:

Input: candyType = [1,1,2,2,3,3]
Output: 3
Explanation: Alice can only eat 6 / 2 = 3 candies. Since there are only 3 types, she can eat one of each type.
Example 2:

Input: candyType = [1,1,2,3]
Output: 2
Explanation: Alice can only eat 4 / 2 = 2 candies. Whether she eats types [1,2], [1,3], or [2,3], she still can only eat 2 different types.
Example 3:

Input: candyType = [6,6,6,6]
Output: 1
Explanation: Alice can only eat 4 / 2 = 2 candies. Even though she can eat 2 candies, she only has 1 type.

Input0:
1 1 2 3 5 7 8
Output0:
3

Input1:

1 2 2 4
Output1:
2

Input2:
2 6 7 1 10 1 2 4
Output2:
4

Input3:
2 6 7 1
Output3:
2

Question2:(Easy Greedy)

Problem Statement: You are a caregiver at a community center organizing a treat distribution event for children. Each child has a specific preference for cookie size, reflecting how much of a treat they need to feel satisfied. These preferences are described as a "greed factor" — the minimum cookie size a child will accept to be content. You have a collection of cookies in various sizes, and you want to distribute them as fairly and efficiently as possible. However, there’s a constraint: each child can receive at most one cookie, and each cookie can be given to only one child.

Your objective is to maximize the number of content children, meaning you want to assign cookies such that as many children as possible receive a cookie that meets or exceeds their minimum requirement.

Write a function that takes two lists as input:

- `g` — a list of greed factors for each child
- `s` — a list of available cookie sizes

Return the maximum number of children who can be content after the distribution. This assessment tests your ability to think algorithmically, work with greedy strategies, and implement efficient solutions to real-world allocation problems.

Example 1:

Input: g = [1,2,3], s = [1,1]

Output: 1

Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3.

And even though you have 2 cookies, since their size is both 1, you could only make the

child whose greed factor is 1 content.

You need to output 1.

Example 2:

Input: g = [1,2], s = [1,2,3]

Output: 2

Explanation: You have 2 children and 3 cookies. The greed factors of 2 children are 1, 2.

You have 3 cookies and their sizes are big enough to gratify all of the children,

You need to output 2.

Constraints:

1 <= g.length <= 3 * 10^4

0 <= s.length <= 3 * 10^4

1 <= g[i], s[j] <= 2^31 - 1

Input0:

1 2 4 5

1 2 3

Output0:

2

Input1:

1 2 4 5

1 2 4

Output1:

3

Input2:

2 5 7

2 3 5 7

Output2:

3

Input3:

2 5 7 8 1

2 3

Output3:

2

Input4:

1 1 1 2

1 2

Output4:

2

Question 3:(Medium level sliding window)
Problem Statement: A retail analytics company wants to analyze customer behavior based on the variety of products purchased in a single shopping session. Each shopping session is recorded as a sequence of product IDs in the order they were added to the cart. The company is interested in identifying shopping patterns where a customer picks exactly k distinct products in a continuous stretch of the session. For example, if a session is recorded as [101, 202, 303, 101, 202], then there are sub-segments with exactly 3 different products. Your task is to develop an algorithm that, given a list of product IDs representing a shopping session and an integer k, returns the number of contiguous sub-segments (subarrays) that

contain exactly k distinct products. This will help the company better understand focused shopping behaviors and improve product placement strategies in their online store.

Constraints:
1 <= nums.length <= 2 * 10^4
1 <= nums[i], k <= nums.length

Example 1:
Input: nums = [1,2,1,2,3], k = 2
Output: 7
Explanation: Subarrays formed with exactly 2 different integers: [1,2], [2,1], [1,2], [2,3], [1,2,1], [2,1,2], [1,2,1,2]

Example 2:
Input: nums = [1,2,1,3,4], k = 3
Output: 3
Explanation: Subarrays formed with exactly 3 different integers: [1,2,1,3], [2,1,3], [1,3,4].

Input0:
1 2 1 3 4 5
3
Output0:
4

Input1:
1 2 1
3
Output1:
0

Input2:
1 2 1 4 8 9 10
2
Output2:
7

Input3:
4 5 4 4 5
2
Output3:
9