# Infinity Champions Phase 4_Week 1 PS 1

**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

**Question 4:(Easy Sum of sub sets)**

**Problem Statement:** A gaming company is analyzing character performance in its latest strategy game. Each character (hero) has a strength value represented by an integer. To simulate various battle scenarios, the game engine needs to compute the overall combat effectiveness of all possible teams that can be formed. The power of a team is calculated using the formula: (maximum strength in the team)² × (minimum strength in the team).

Given the array of hero strengths, your task is to develop a system that computes the total power of all possible non-empty combinations of heroes (teams). Since the number of combinations and resulting total power can be extremely large, the final result should be returned modulo 10^9 + 7. This metric will be used by the game developers to balance team dynamics, improve matchmaking algorithms, and adjust the difficulty levels based on team composition diversity.

Constraints:

1 <= nums.length <= 10^5

1 <= nums[i] <= 10^9

Example 1:

Input: nums = [2,1,4]

Output: 141

Explanation:

1st group: [2] has power = 22 * 2 = 8.

2nd group: [1] has power = 12 * 1 = 1.

3rd group: [4] has power = 42 * 4 = 64.

4th group: [2,1] has power = 22 * 1 = 4.

5th group: [2,4] has power = 42 * 2 = 32.

6th group: [1,4] has power = 42 * 1 = 16.

7th group: [2,1,4] has power = 42 * 1 = 16.

The sum of powers of all groups is 8 + 1 + 64 + 4 + 32 + 16 + 16 = 141.

Example 2:

Input: nums = [1,1,1]

Output: 7

Explanation: A total of 7 groups are possible, and the power of each group will be 1. Therefore, the sum of the powers of all groups is 7.

Input0:
1 4 8
Output0:
977

Input1:
3 4 2
Output1:
229

Input2:
3 4 2 1
Output2:
316

Input3:
1 2
Output3:
13

**Question 5:(parity optimization-Completely uncovered topic)**

**Problem Statement:** A cybersecurity company is analyzing patterns in network traffic logs represented as character streams. Each log entry is a long string of characters where each character represents a type of request or signal. To detect anomalies or specific behavioral patterns, analysts want to study the imbalance between frequently and infrequently occurring signals within certain time windows. Specifically, for any continuous segment (substring) of the log with a minimum length of *k*, they want to find the maximum difference in frequency between two signal types, where one occurs an odd number of times and the other an even number of times.

- subs has a size of at least k.
- Character a has an odd frequency in subs.
- Character b has an even frequency in subs.
- Return the maximum difference.

This metric helps identify irregular communication bursts or suspicious traffic patterns that might indicate system misuse or threats. Given the character stream and the integer *k*, your task is to compute the highest such frequency difference. This insight will aid in enhancing the company's anomaly detection systems.

Constraints:

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

s consists only of digits '0' to '4'.

The input is generated that at least one substring has a character with an even frequency and a character with an odd frequency.

1 <= k <= s.length

Example 1:

Input: s = "12233", k = 4

Output: -1

Explanation:

For the substring "12233", the frequency of '1' is 1 and the frequency of '3' is 2. The difference is 1 - 2 = -1.

Example 2:

Input: s = "1122211", k = 3

Output: 1

Explanation:
For the substring "11222", the frequency of '2' is 3 and the frequency of '1' is 2. The difference is 3 - 2 = 1.

Example 3:
Input: s = "110", k = 3
Output: -1

Input0:
1223
4
Output0:
-1

Input1:
1122211
4
Output1:
1

Input2:
33222
4
Output2:
1

Input3:
1112
2
Output3:
-1

**Question 6:**

**Problem Statement:** You are working as a software engineer for a financial analytics company that builds tools to detect unusual market behavior. One of your tasks is to analyze a series of stock price changes over several days. These changes are represented as an array of integers, where each integer indicates the change in stock price for a given day—positive for a gain and negative for a loss.

To identify patterns of consistent risk, your team defines the "beauty" of a sliding window of days as the x-th smallest negative change in that period. If there are fewer than x negative changes in a window, the beauty is considered 0.

You are asked to write a function that, given an array of daily price changes, a window size k, and a rank x, returns a list of beauties for all subarrays of size k.

Your solution will help identify high-risk trends in historical price data.

Constraints:
n == nums.length
1 <= n <= 10^5
1 <= k <= n
1 <= x <= k
-50 <= nums[i] <= 50

Example 1:

Input: nums = [1,-1,-3,-2,3], k = 3, x = 2
Output: [-1,-2,-2]
Explanation: There are 3 subarrays with size k = 3.
The first subarray is [1, -1, -3] and the 2nd smallest negative integer is -1.
The second subarray is [-1, -3, -2] and the 2nd smallest negative integer is -2.
The third subarray is [-3, -2, 3] and the 2nd smallest negative integer is -2.
Example 2:

Input: nums = [-1,-2,-3,-4,-5], k = 2, x = 2
Output: [-1,-2,-3,-4]

Explanation: There are 4 subarrays with size k = 2.
For [-1, -2], the 2nd smallest negative integer is -1.
For [-2, -3], the 2nd smallest negative integer is -2.
For [-3, -4], the 2nd smallest negative integer is -3.
For [-4, -5], the 2nd smallest negative integer is -4.
Example 3:

Input: nums = [-3,1,2,-3,0,-3], k = 2, x = 1
Output: [-3,0,-3,-3,-3]
Explanation: There are 5 subarrays with size k = 2.
For [-3, 1], the 1st smallest negative integer is -3.
For [1, 2], there is no negative integer so the beauty is 0.
For [2, -3], the 1st smallest negative integer is -3.
For [-3, 0], the 1st smallest negative integer is -3.
For [0, -3], the 1st smallest negative integer is -3.

Input0:
1 -1 -3 -2 3 4
3
2
Output0:
-1 -2 -2 0

Input1:
1 -1 -3 -2 3 -4
3
2
Output1:
-1 -2 -2 -2

Input2:
-1 -3 -2 3 -4
3

3

Output2:

-1 0 0

Input3:

-1 -3 -2 3 -4

2

2

Output3:

-1 -2 0 0