Self-Practice No. : **6**

Topics Covered : **Array, String, Arraylist, Priority Queue, Control Flow Statements, Bit Manipulation, Treeset, Hashset**

Date : **21-05-2024**

Solve the following problems

Q No.	Question Detail	Level
1	**Sequential Digits** **Problem Statement** : An integer has sequential digits if and only if each digit in the number is one more than the previous digit. Return a sorted list of all the integers in the range [low, high] inclusive that have sequential digits. **Example 1:** **Input:** low = 100, high = 300 **Output**: [123,234] **Example 2:** **Input:** low = 1000, high = 13000 **Output**: [1234,2345,3456,4567,5678,6789,12345] **Constraints:** 10 <= low <= high <= 10^9	Medium
2	**Top K Frequent Elements** **Problem Statement** : Given an integer array nums and an integer k, return the k most frequent elements. You may return the answer in any order **Example 1:**	Medium

	Input: nums = [1,1,1,2,2,3], k = 2 **Output**: [1,2] **Example 2:** **Input:** nums = [1], k = 1 **Output:** [1] **Constraints:** 1 <= nums.length <= 10^5 -10^4 <= nums[i] <= 10^4 k is in the range [1, the number of unique elements in the array]. It is guaranteed that the answer is unique.	
3	**Dot Product of Two Sparse Vectors** **Problem Statement** : Given two sparse vectors, compute their dot product. Implement class SparseVector: SparseVector(nums) Initializes the object with the vector nums dotProduct(vec) Compute the dot product between the instance of SparseVector and vec A sparse vector is a vector that has mostly zero values, you should store the sparse vector efficiently and compute the dot product between two SparseVector. **Example 1:** **Input**: nums1 = [1,0,0,2,3], nums2 = [0,3,0,4,0] **Output:** 8 **Explanation**: v1 = SparseVector(nums1) , v2 = SparseVector(nums2) v1.dotProduct(v2) = 1*0 + 0*3 + 0*0 + 2*4 + 3*0 = 8 **Example 2:** **Input:** nums1 = [0,1,0,0,0], nums2 = [0,0,0,0,2] **Output:** 0	Medium

	Explanation: v1 = SparseVector(nums1) , v2 = SparseVector(nums2) v1.dotProduct(v2) = 0*0 + 1*0 + 0*0 + 0*0 + 0*2 = 0 **Example 3:** **Input:** nums1 = [0,1,0,0,2,0,0], nums2 = [1,0,0,0,3,0,4] **Output:** 6 **Constraints:** n == nums1.length == nums2.length 1 <= n <= 10^5 0 <= nums1[i], nums2[i] <= 100	
4	**Contiguous Array** **Problem Statement** : Given a binary array nums, return the maximum length of a contiguous subarray with an equal number of 0 and 1. **Example 1:** **Input:** nums = [0,1] **Output:** 2 **Explanation:** [0, 1] is the longest contiguous subarray with an equal number of 0 and 1. **Example 2:** **Input:** nums = [0,1,0] **Output:** 2 **Explanation:** [0, 1] (or [1, 0]) is a longest contiguous subarray with equal number of 0 and 1. **Constraints:** • 1 <= nums.length <= 10^5 • nums[i] is either 0 or 1.	Medium

5	**Gray Code**	Medium

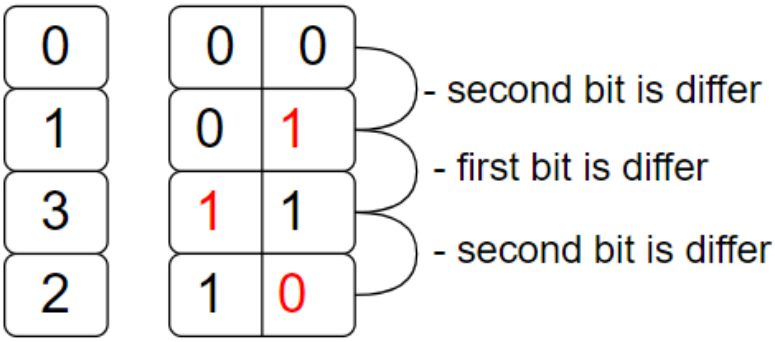
Problem statement : Given a number 'grayNumber'. Find the gray code sequence.

Conditions for a gray code sequence :

1. Gray code sequence contains numbers from 0 to 2^'grayNumber'-1 in bit/binary form.
2. Two consecutive gray code sequence numbers only differ by 1 bit.
3. Gray code sequence must start with 0.

Example :

Given 'grayNumber' : 2.



As depicted from above image, the gray code sequence is 0,1,3,2.

Note :

1. The output sequence must contain the decimal representation of numbers instead of the binary form.
2. There can be multiple answers print anyone.

Sample Input 1 :

2

2

3

Sample Output 1 :

Valid

Valid

Explanation For Sample Input 1 :

For first test case,

Given 'grayNumber' : 2

Bits representation of a number from 0 to 2^2-1 = 3 is " 00, 01, 10, 11".

But we have arranged these bits in such a way that two consecutive bits only differ by 1.

Only one possible way is "00, 01, 11, 10 ".

Hence return value of every number " 00 - 0 , 01 - 1, 11 - 3, 10 - 2".

Sequence : {0, 1, 3, 2}.

For second test case,

Given 'grayNumber' : 3

Bits representation of a number from 0 to 2^3-1 =7 is "000, 001, 010, 011, 100, 101, 110, 111".

But we have arranged these bits in such a way that two consecutive bits only differ by 1.

One of the possible ways is "000, 001, 011, 111, 101, 100, 110, 010".

Hence return value of every number "000 - 0, 001 - 1, 011 - 3, 111 - 7, 101 - 5, 100 - 4, 110 - 6, 010 - 2 ".

Sequence : {0, 1, 3, 7, 5, 4, 6, 2}.

One another possible sequence can be : {0, 1, 3, 2, 6, 7, 5, 4}.

Sample Input 2 :

2

4

1

Sample Output 2 :

Valid

Valid

Explanation For Sample Input 2 :

For first test case,

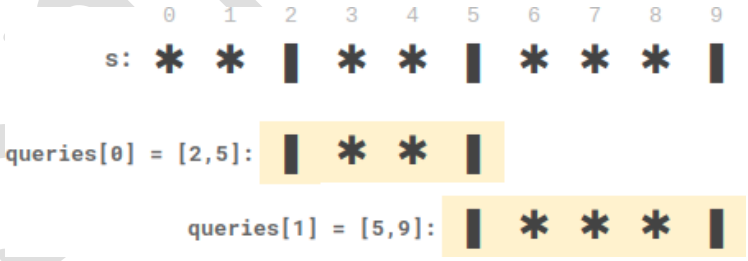
Given 'grayNumber' : 4

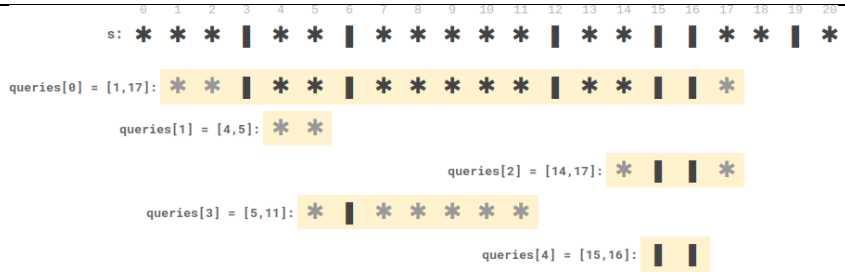
Sequence : {0, 1, 3, 2, 6, 7, 5, 4, 12, 13, 15, 14, 10, 11, 9, 8}.

For second test case,

Given 'grayNumber' : 1

Sequence : {0, 1}.

	Constraints : 1 <= T <= 2 0 <= grayNumber <= 15	
6	**Plates Between Candles** **Problem statement** : There is a long table with a line of plates and candles arranged on top of it. You are given a 0-indexed string s consisting of characters '*' and '|' only, where a '*' represents a plate and a '|' represents a candle. You are also given a 0-indexed 2D integer array queries where queries[i] = [$left_i$, $right_i$] denotes the substring s[$left_i$...$right_i$] (inclusive). For each query, you need to find the number of plates between candles that are in the substring. A plate is considered between candles if there is at least one candle to its left and at least one candle to its right in the substring. • For example, s = "||**||**|*", and a query [3, 8] denotes the substring "*||**|". The number of plates between candles in this substring is 2, as each of the two plates has at least one candle in the substring to its left and right. Return *an integer array* answer *where* answer[i] *is the answer to the* i^{th} *query*. **Example 1:**  **Input:** s = "**|**|***|", queries = [[2,5],[5,9]] **Output**: [2,3] **Explanation:** - queries[0] has two plates between candles. - queries[1] has three plates between candles. **Example 2:**	Medium



Input: s = "***|**|*****|**||**|*", queries = [[1,17],[4,5],[14,17],[5,11],[15,16]]

Output: [9,0,0,0,0]

Explanation:

- queries[0] has nine plates between candles.
- The other queries have zero plates between candles.

Constraints:

- 3 <= s.length <= 10^5
- s consists of '*' and '|' characters.
- 1 <= queries.length <= 10^5
- queries[i].length == 2
- 0 <= $left_i$ <= $right_i$ < s.length

7	**Task Scheduler** **Problem statement** : You are given an array of CPU tasks, each represented by letters A to Z, and a cooling time, n. Each cycle or interval allows the completion of one task. Tasks can be completed in any order, but there's a constraint: identical tasks must be separated by at least n intervals due to cooling time. Return the minimum number of intervals required to complete all tasks. **Example 1:** **Input:** tasks = ["A","A","A","B","B","B"], n = 2 **Output:** 8 **Explanation:** A possible sequence is: A -> B -> idle -> A -> B -> idle -> A -> B. After completing task A, you must wait two cycles before doing A again. The same applies to task B. In the 3rd interval, neither	Medium

	A nor B can be done, so you idle. By the 4th cycle, you can do A again as 2 intervals have passed. **Example 2:** **Input:** tasks = ["A","C","A","B","D","B"], n = 1 **Output:** 6 **Explanation:** A possible sequence is: A -> B -> C -> D -> A -> B. With a cooling interval of 1, you can repeat a task after just one other task. **Example 3:** **Input:** tasks = ["A","A","A", "B","B","B"], n = 3 **Output:** 10 **Explanation:** A possible sequence is: A -> B -> idle -> idle -> A -> B -> idle -> idle -> A -> B. There are only two types of tasks, A and B, which need to be separated by 3 intervals. This leads to idling twice between repetitions of these tasks. **Constraints:** 1 <= tasks.length <= 10^4 tasks[i] is an uppercase English letter. 0 <= n <= 100	
8	**Repeated DNA Sequences** **Problem statement** : The DNA sequence is composed of a series of nucleotides abbreviated as 'A', 'C', 'G', and 'T'. For example, "ACGAATTCCG" is a DNA sequence. When studying DNA, it is useful to identify repeated sequences within the DNA. Given a string s that represents a DNA sequence, return all the 10-letter-long sequences (substrings) that occur more than once in a DNA molecule. You may return the answer in any order.	Medium

	Example 1: **Input:** s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT" **Output**: ["AAAAACCCCC","CCCCCAAAAA"] **Example 2:** **Input**: s = "AAAAAAAAAAAAA" **Output**: ["AAAAAAAAAA"] **Constraints:** 1 <= s.length <= 10^5 s[i] is either 'A', 'C', 'G', or 'T'.	
9	**Find All Anagrams in a String** **Problem statement** : Given two strings s and p, return an array of all the start indices of p's anagrams in s. You may return the answer in any order. An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. **Example 1:** **Input**: s = "cbaebabacd", p = "abc" **Output**: [0,6] **Explanation:** The substring with start index = 0 is "cba", which is an anagram of "abc". The substring with start index = 6 is "bac", which is an anagram of "abc". **Example 2:** **Input**: s = "abab", p = "ab" **Output:** [0,1,2] **Explanation:** The substring with start index = 0 is "ab", which is an anagram of "ab".	Medium

	The substring with start index = 1 is "ba", which is an anagram of "ab". The substring with start index = 2 is "ab", which is an anagram of "ab". **Constraints:** 1 <= s.length, p.length <= 3 * 10^4 s and p consist of lowercase English letters.	
10	**Divide String** **Problem statement :** Bob has been given a string 'WORD' containing lower case alphabets. Bob wants to know all the strings by dividing 'WORD' into 'N' strings of equal length. For Example: For 'WORD' = "abcdefgh", 'N' = 2. Following are the 2 strings of length 4. "abcd" "efgh" Can you help Bob to get all the strings from 'WORD' by dividing them into equal parts? **Sample Input 1:** 2 asdfghjkl 3 codingninjas 5 **Sample Output 1:** asd fgh jkl **Explanation For Sample Output 1:** **For the first test case:** Given 'WORD' = "asdfghjkl" can be divided into 3 strings each of length 3. Following are the possible strings of length 3. 1. "asd" 2. "fgh" 3. "jkl" **For the second test case:**	Medium

Given 'WORD' = "codingninjas", it is impossible to divide this 'WORD' into 5 strings of equal length.

So we return an empty array/list.

Sample Input 2:

2

a 1

code 4

Sample Output 2:

a

c o d e

Constraints:

1 <= 'T' <= 100

'WORD' = Lower case english alphabet

1 <= |WORD| <= 2000

1 <= 'N' <= |WORD|