**Self-Practice No.** **: 7**

**Topics Covered** **: Array, Priority Queue, List, Hashmap, Treemap, Heap, LinkedHashMap**

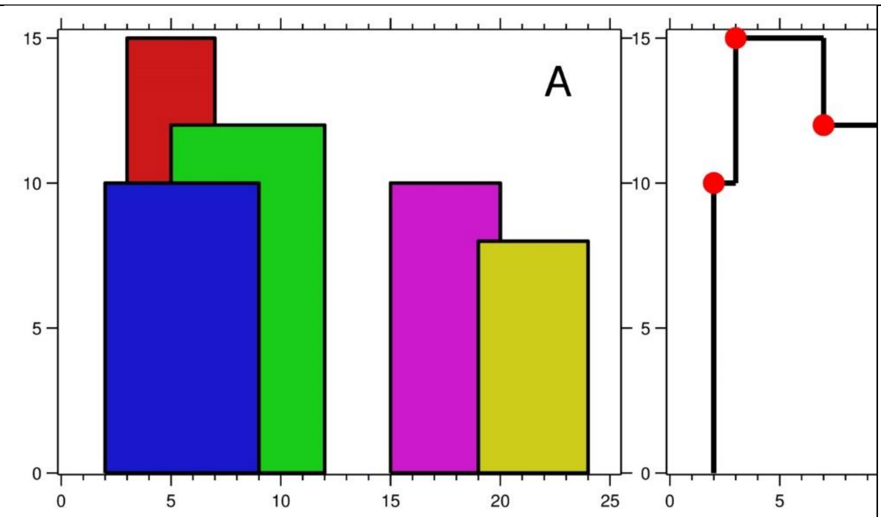
**Date** **: 22-05-2024**

**Solve the following problems**

| Q No. | Question Detail | Level |
|---|---|---|
| 1 | **Find Median from Data Stream**<br>**Problem statement :** The median is the middle value in an ordered integer list. If the size of the list is even, there is no middle value, and the median is the mean of the two middle values.<br>• For example, for arr = [2,3,4], the median is 3.<br>• For example, for arr = [2,3], the median is (2 + 3) / 2 = 2.5.<br>Implement the MedianFinder class:<br>• MedianFinder() initializes the MedianFinder object.<br>• void addNum(int num) adds the integer num from the data stream to the data structure.<br>• double findMedian() returns the median of all elements so far. Answers within $10^{-5}$ of the actual answer will be accepted.<br>**Example 1:**<br>Input<br>["MedianFinder", "addNum", "addNum", "findMedian", "addNum", "findMedian"]<br>[[], [1], [2], [], [3], []]<br>Output<br>[null, null, null, 1.5, null, 2.0]<br>Explanation<br>MedianFinder medianFinder = new MedianFinder();<br>medianFinder.addNum(1); // arr = [1]<br>medianFinder.addNum(2); // arr = [1, 2] | Hard |

| | | |
|---|---|---|
| | medianFinder.findMedian(); // return 1.5 (i.e., (1 + 2) / 2)<br>medianFinder.addNum(3);    // arr[1, 2, 3]<br>medianFinder.findMedian(); // return 2.0<br><br>**Constraints:**<br>• $-10^5$ <= num <= $10^5$<br>• There will be at least one element in the data structure before calling findMedian.<br>At most 5 * $10^4$ calls will be made to addNum and findMedian. | |
| 2 | **Strong Password Checker**<br><br>**Problem statement :** A password is considered strong if the below conditions are all met:<br>• It has at least 6 characters and at most 20 characters.<br>• It contains at least **one lowercase** letter, at least **one uppercase** letter, and at least **one digit**.<br>• It does not contain three repeating characters in a row (i.e., "B**aaa**bb0" is weak, but "B**aa**b**a**0" is strong).<br>Given a string password, return *the minimum number of steps required to make password strong. if password is already strong, return 0.*<br>In one step, you can:<br>• Insert one character to password,<br>• Delete one character from password, or<br>• Replace one character of password with another character.<br><br>**Example 1:**<br>**Input:** password = "a"<br>**Output:** 5<br>**Example 2:**<br>**Input:** password = "aA1"<br>**Output:** 3<br>**Example 3:**<br>**Input:** password = "1337C0d3"<br>**Output:** 0<br><br>**Constraints:** | Hard |

| | | |
|---|---|---|
| | - 1 <= password.length <= 50<br>password consists of letters, digits, dot '.' or exclamation mark '!'. | |
| **3** | **The Skyline Problem**<br>**Problem statement :** A city's **skyline** is the outer contour of the silhouette formed by all the buildings in that city when viewed from a distance. Given the locations and heights of all the buildings, return *the* ***skyline*** *formed by these buildings collectively*.<br>The geometric information of each building is given in the array buildings where buildings[i] = [left$_i$, right$_i$, height$_i$]:<br>- left$_i$ is the x coordinate of the left edge of the i$^{th}$ building.<br>- right$_i$ is the x coordinate of the right edge of the i$^{th}$ building.<br>- height$_i$ is the height of the i$^{th}$ building.<br>You may assume all buildings are perfect rectangles grounded on an absolutely flat surface at height 0.<br>The **skyline** should be represented as a list of "key points" **sorted by their x-coordinate** in the form $[[x_1,y_1],[x_2,y_2],...]$. Each key point is the left endpoint of some horizontal segment in the skyline except the last point in the list, which always has a y-coordinate 0 and is used to mark the skyline's termination where the rightmost building ends. Any ground between the leftmost and rightmost buildings should be part of the skyline's contour.<br>**Note:** There must be no consecutive horizontal lines of equal height in the output skyline. For instance, [...,[2 3],[4 5],[7 5],[11 5],[12 7],...] is not acceptable; the three lines of height 5 should be merged into one in the final output as such: [...,[2 3],[4 5],[12 7],...]<br>**Example 1:** | Hard |

| | | |
|---|---|---|
| | <br> <br>**Input:** buildings = [[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]<br>**Output:** [[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]<br>**Explanation:**<br>Figure A shows the buildings of the input.<br>Figure B shows the skyline formed by those buildings. The red points in figure B represent the key points in the output list.<br>**Example 2:**<br>**Input:** buildings = [[0,2,3],[2,5,3]]<br>**Output:** [[0,3],[5,0]]<br><br>**Constraints:**<br>• 1 <= buildings.length <= $10^4$<br>• 0 <= $left_i$ < $right_i$ <= $2^{31}$ - 1<br>• 1 <= $height_i$ <= $2^{31}$ - 1<br><br>buildings is sorted by $left_i$ in non-decreasing order. | |
| 4 | **Fruits and Baskets**<br><br>**Problem statement :** There are ***'n'*** fruit trees that are planted along a road. The trees are numbered from 0 to n-1. The type of fruit each tree bears is represented by an integer from 1 to 'n'.<br><br>A Haary is walking along that road. He has two baskets and wants to put the maximum number of fruits in them. The restriction is that each basket can have only one type of fruit. | Hard |

Harry  can start with any tree and end at any tree, but once he has started, he cannot skip a tree i.e if he picks fruit from the tree ***'i'***, then he has to pick fruit from tree 'i+1' before going to the tree 'i+2'. He will pick one fruit from each tree until he cannot, i.e, he will stop when he has to pick a fruit of the third type because only two different fruits can fill both baskets.

You are given an array ***'arr'***. The 'i'th integer in this array represents the type of fruit tree 'i' bears. Return the maximum number of fruits Ninja can put in both baskets after satisfying all the conditions.

**For Example:**

'arr' = [1, 2, 3]

Here, we have three different types of fruits. We can pick [1, 2] or [2, 3]. We can pick a maximum of two fruits.

Hence, we return 2.

**Sample Input 1:**

4

1 1 2 3

**Sample Output 1:**

3

**Explanation of Sample Input 1:**

There are four trees and the type of fruits in them are 1, 1, 2, 3 respectively.

One way is that Ninja can start picking fruits from tree 0. He picks one fruit from tree 0 and put it in the first basket, then he picks one fruit from tree 1 and put it in the first basket, then he picks one fruit from tree 2 and put it in the second basket, he cannot pick fruit from tree 3 because the first basket has the fruit of type 1 and second has the fruit of type 2 and type of fruit in tree-3 is 3.

Thus he has to stop there. The number of fruits he picks in this way is 3. We can show that this is the maximum possible number of fruits ninjas can pick.

| | | |
|---|---|---|
| | **Sample Input 2:**<br>4<br>1 2 3 4<br>**Sample Output 2:**<br>2<br>**Explanation of Sample Input 2:**<br>There are four trees, and each of them has different types of fruit. No matter from which tree Ninja starts picking fruits he can only collect 2 fruits.<br>**Constraints:**<br>1 <= n <= 10^4<br>1 <= arr[I] <= n<br>Where 'n' represents the number of trees. | |
| **5** | **Minimum Sorted Groups**<br>**Problem statement :** You are given an array 'ARR' containing 'N' integers.<br>You have a simple task, you need to split the elements of this array into different groups, inside each group the relative order between elements must be maintained.<br>You need to find the minimum number of groups that are required to be formed such that elements inside each group are sorted in ascending order.<br>**For Example :**<br>If 'N' = 7 and 'ARR' = { 1, 5, 2, 3, 4, 6, 7 }<br>Then, we can split the array elements into two groups: { 1, 2, 3, 4 } and { 5, 6, 7 }, this splitting is valid as it maintains the relative ordering of the elements of the original array and after splitting all the groups contain elements in sorted order. Therefore we will print 2.<br>Note that a group like { 1, 2, 3, 5 } can't be formed as it doesn't have relative ordering the same as the input array.<br>**Sample Input 1 :**<br>2<br>7<br>1 5 2 3 4 6 7 | Hard |

6

-1 0 2 3 4 6

**Sample Output 1 :**

2

1

**Explanation For Sample Input 1 :**

For test case 1 :

We will print 2 because:

We can split the array elements into two groups: { 1, 2, 3, 4 } and { 5, 6, 7 }, this splitting is valid as it maintains the relative ordering of the elements of the original array and after splitting all the groups contain elements in sorted order.

For test case 2 :

We will print 1 because:

The given array is itself sorted, so all the elements can be grouped together and this will result in the formation of a sorted group while maintaining the relative order of the array elements.

**Sample Input 2 :**

2

4

3 1 2 0

3

1 1 0

**Sample Output 2 :**

3

2

**Constraints :**

1 ≤ T ≤ 10

1 ≤ N ≤ 5000

-10^9 ≤ ARR[i] ≤ 10^9

**6** **Binary Matrix** Hard

**Problem statement :** You are given a matrix ‘MAT’ consisting of ‘N’ rows and ‘M’ columns. Let (i, j) represent the cell at the intersection of the ith row and the jth column. Each cell of the matrix ‘MAT’ has either integer 0 or 1. For each cell in 'MAT', you have to find the Manhattan distance of the nearest cell from this cell that has the integer 0. The nearest cell will be the cell having the minimum Manhattan distance from it.

Manhattan distance between two cells, (p1, q1) and (p2, q2) is |p1 - p2| + |q1 - q2|.

You should return a matrix consisting of 'N' rows and 'M' columns, where the cell (i, j) represents the Manhattan distance of the nearest cell from the cell (i, j) in 'MAT' that has integer 0.

**Note**

1. There is at least one cell having the integer 0 in the given matrix.

**Example:**

Consider the following 2*3 matrix 'MAT':

[0, 1, 1]
[0, 1, 1]

Here, the nearest cell having the integer 0 from the cell (0, 0) is the cell (0, 0) itself. The Manhattan distance between them is |0 - 0| + |0 - 0| = 0.

The nearest cell having the integer 0 from the cell (0, 1) is cell (0, 0). The Manhattan distance between them is |0 - 0| + |1 - 0| = 1.

The nearest cell having the integer 0 from the cell (0, 2) is cell (0, 0). The Manhattan distance between them is |0 - 0| + |2 - 0| = 2.

The nearest cell having the integer 0 from the cell (1, 0) is cell (1, 0) itself. The Manhattan distance between them is |1 - 1| + |0 - 0| = 0.

The nearest cell having the integer 0 from the cell (1, 1) is cell (1, 0). The Manhattan distance between them is |1 - 1| + |1 - 0| = 1.

The nearest cell having the integer 0 from the cell (1, 2) is cell (1, 0). The Manhattan distance between them is |1 - 1| + |2 - 0| = 2.

Thus we should return matrix:

[0, 1, 2]

[0, 1, 2]

**Sample Input 1:**

```
2
1 1
0
2 3
0 1 1
0 1 1
```

**Sample Output 1:**

```
0
0 1 2
0 1 2
```

**Explanation of Sample Input 1:**

Test case 1:

There is only one cell in the given matrix and that cell has integer 0, so the nearest cell from this cell having integer 0 is this cell itself. The Manhattan distance of the cell from itself will be 0.

Test case 2:

See the problem statement for an explanation.

**Sample Input 2:**

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

**Sample Output 2:**

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

| | | |
|---|---|---|
| | 0 1 1 0<br>**Constraints:**<br>1 <= T <= 50<br>1 <= N <= 100<br>1 <= M <= 100<br>0 <= MAT[i][j] <= 1<br>Where 'T' is the total number of test cases, 'N' and 'M' denote the number of rows and columns in the given matrix 'MAT', and MAT[i][j] is the element of the given matrix at cell (i, j). | |
| **7** | **Rearrange string**<br>**Problem statement :** You are given a string "S". Your task is to rearrange the characters of a string "S", such that it does not contain any two adjacent characters which are the same.<br>If it is possible to rearrange the string "S", then print any possible arrangement. else, print "not possible" without quotes.<br>For Example:<br>For a string "qaacde", This string has two same adjacent characters. So, one possible way to rearrange the string is "qacade". Now, this string does not have two adjacent characters that are the same.<br>**Sample Input 1 :**<br>2<br>coding<br>abaab<br>**Sample Output 1 :**<br>Yes<br>Yes<br>**Explanation :**<br>For the first test case, the given string is "coding". This string does not have two adjacent characters that are the same. So if we return "coding", then we will get an output as "Yes". | Hard |

| | | |
|---|---|---|
| | For the second test case, the given string is “abaab”. This string has two adjacent characters that are the same i.e. abaab,  So we can rearrange the string as “ababa”.Hence we return “ababa”, then we will get an output as "Yes”.<br>**Sample Input 2 :**<br>1<br>bbbbbb<br>**Sample Output 2 :**<br>Yes<br>**Explanation :**<br>For the first test case, the given string is “bbbbbb”. No matter how you rearrange characters of string , it will always remain the same as bbbbbb.So we return “not possible” as an answer, then we will get an output as ‘Yes”.<br>**Constraints :**<br>1 <= T <= 10<br>0 <= \|S\| <= 10^5 | |
| **8** | **Longest Duplicate Substring**<br>**Problem statement :** Given a string s, consider all duplicated substrings: (contiguous) substrings of s that occur 2 or more times. The occurrences may overlap.<br>Return any duplicated substring that has the longest possible length. If s does not have a duplicated substring, the answer is "".<br>**Example 1:**<br>Input: s = "banana"<br>Output: "ana"<br>**Example 2:**<br>Input: s = "abcd"<br>Output: ""<br>**Constraints:**<br>2 <= s.length <= 3 * 10^4 | Hard |

| | | |
|---|---|---|
| | s consists of lowercase English letters. | |
| **9** | **Guess the Word**<br>You are given an array of unique strings words where words[i] is six letters long. One word of words was chosen as a secret word. You are also given the helper object Master. You may call Master.guess(word) where word is a six-letter-long string, and it must be from words. Master.guess(word) returns:<br>• -1 if word is not from words, or<br>• an integer representing the number of exact matches (value and position) of your guess to the secret word.<br>There is a parameter allowedGuesses for each test case where allowedGuesses is the maximum number of times you can call Master.guess(word).<br>For each test case, you should call Master.guess with the secret word without exceeding the maximum number of allowed guesses. You will get:<br>• **"Either you took too many guesses, or you did not find the secret word."** if you called Master.guess more than allowedGuesses times or if you did not call Master.guess with the secret word, or<br>• **"You guessed the secret word correctly."** if you called Master.guess with the secret word with the number of calls to Master.guess less than or equal to allowedGuesses.<br>The test cases are generated such that you can guess the secret word with a reasonable strategy (other than using the bruteforce method).<br><br>**Example 1:**<br>**Input:** secret = "acckzz", words = ["acckzz","ccbazz","eiowzz","abcczz"], allowedGuesses = 10<br>**Output:** You guessed the secret word correctly.<br>**Explanation:**<br>master.guess("aaaaaa") returns -1, because "aaaaaa" is not in wordlist. | Hard |

| | | |
|---|---|---|
| | master.guess("acckzz") returns 6, because "acckzz" is secret and has all 6 matches.<br>master.guess("ccbazz") returns 3, because "ccbazz" has 3 matches.<br>master.guess("eiowzz") returns 2, because "eiowzz" has 2 matches.<br>master.guess("abcczz") returns 4, because "abcczz" has 4 matches.<br>We made 5 calls to master.guess, and one of them was the secret, so we pass the test case.<br>**Example 2:**<br>**Input:** secret = "hamada", words = ["hamada","khaled"], allowedGuesses = 10<br>**Output:** You guessed the secret word correctly.<br>**Explanation:** Since there are two words, you can guess both.<br><br>**Constraints:**<br>• 1 <= words.length <= 100<br>• words[i].length == 6<br>• words[i] consist of lowercase English letters.<br>• All the strings of wordlist are **unique**.<br>• secret exists in words.<br>• 10 <= allowedGuesses <= 30 | |
| **10** | **Alien dictionary**<br><br>**Problem statement:** You have been given a sorted (lexical order) dictionary of an alien language.<br><br>Write a function that returns the order of characters as a string in the alien language. This dictionary will be given to you as an array of strings called ***'dictionary'***, of size ***'N'***.<br><br>**Example :**<br>If the dictionary consists of the following words:-<br>["caa", "aaa", "aab"], and 'K' is 3. | Hard |

Then, the order of the alphabet is -

['c', 'a', 'b']

**Note:**

If the language consists of four letters, the four letters should be the starting four letters of the English language.

However, their order might differ in the alien language.

**Sample Input 1 :**

3 1

a aa aaa

**Sample Output 1 :**

true

**Explanation For Sample Output 1 :**

The words are 'a', 'aa', and 'aaa'. Since the unique character here is 'a', the array to be returned will just be ['a'].

The 'true' being printed signifies that the output returned by the function is valid.

**Sample Input 2 :**

3 3

caa aaa aab

**Sample Output 2 :**

true

**Constraints :**

1 ≤ N ≤ 300

1 ≤ K ≤ 26

1 ≤ Length of words ≤ 50