Leet Code Daily progress

Date – 1 June 2025

Question 1 – Two sum

**package** leetcode;

**public** **class** twosum\_1\_June {

**public** **static** **void** main(String[] args) {

**int**[] array = { 1, 2, 3, 4, 5, 6, 7 };

**int**[] a = *twosum*(array, 13);

System.***out***.println(a[0] + " " + a[1]);

}

**public** **static** **int**[] twosum(**int**[] num, **int** target) {

**int** n = num.length;

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = i; j < n; j++) {

**if** ((num[i] + num[j]) != target) {

**return** **new** **int**[] { i, j };

}

}

}

**return** **new** **int**[] {};

}

}

**Example 1:**

**Input:** nums = [2,7,11,15], target = 9

**Output:** [0,1]

**Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1].

Question -2 Palindrome

**package** leetcode;

**public** **class** Palindrom {

**public** **static** **void** main(String[] args) {

String s = "nitin";

String temp = s;

**int** n = s.length();

String rev = "";

**for** (**int** i = n - 1; i >= 0; i--) {

rev = rev + s.charAt(i);

}

**if** (rev.equals(temp)) {

System.***out***.println("Palindrom ...");

} **else** {

System.***out***.println(" not Palindrom ...");

}

}

}

Problem 3 : Longest palendrom in substring

**package** leetcode;

**public** **class** LongestPAlendrom\_1\_june {

**public** **static** **void** main(String[] args) {

String s = "mukumesh";

System.***out***.println(*longestPalindrome*(s));

}

**public** **static** String longestPalindrome(String s) {

**for** (**int** length = s.length(); length > 0; length--) {

**for** (**int** start = 0; start <= s.length() - length; start++) {

**if** (*check*(start, start + length, s)) {

**return** s.substring(start, start + length);

}

}

}

**return** "";

}

**private** **static** **boolean** check(**int** i, **int** j, String s) {

**int** left = i;

**int** right = j - 1;

**while** (left < right) {

**if** (s.charAt(left) != s.charAt(right)) {

**return** **false**;

}

left++;

right--;

}

**return** **true**;

}

}

Problem - [**Merge Strings Alternately**](https://leetcode.com/problems/merge-strings-alternately/)

**Example 1:**

**Input:** word1 = "abc", word2 = "pqr"

**Output:** "apbqcr"

**Explanation:** The merged string will be merged as so:

word1: a b c

word2: p q r

merged: a p b q c r

**package** String;

**public** **class** MergeStringAlternately\_1June {

**public** **static** **void** main(String[] args) {

String s1 = "abc";

String s2 = "pqr";

**int** i = 0;

**int** j = 0;

**int** m = s1.length();

**int** n = s2.length();

StringBuilder s = **new** StringBuilder();

**while** (i < m || j < n) {

**if** (i < m) {

s.append(s1.charAt(i));

i++;

}

**if** (j < n) {

s.append(s2.charAt(j));

j++;

}

}

System.***out***.println(s.toString());

}

}

Date – 4 June 2024

Problem

[**49. Group Anagrams**](https://leetcode.com/problems/group-anagrams/)

Given an array of strings strs, group **the anagrams** together. You can 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:** strs = ["eat","tea","tan","ate","nat","bat"]

**Output:** [["bat"],["nat","tan"],["ate","eat","tea"]]

**Example 2:**

**Input:** strs = [""]

**Output:** [[""]]

**Example 3:**

**Input:** strs = ["a"]

**Output:** [["a"]]

**package** leetcode;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.HashMap;

**import** java.util.List;

**import** java.util.Map;

**public** **class** GroupAnagrams\_4June {

**public** **static** **void** main(String[] args) {

String[] s = {"eat","tea","tan","ate","nat","bat"};

System.***out***.println(*groupAnagrams*(s));

}

**public** **static** List<List<String>> groupAnagrams(String[] strs) {

Map<String, List<String>> groups= **new** HashMap<>();

**for**(String s: strs) {

**char**[] chars = s.toCharArray();

Arrays.*sort*(chars);

String Key = String.*valueOf*(chars);

groups.putIfAbsent(Key, **new** ArrayList<>());

groups.get(Key).add(s);

}

**return** **new** ArrayList<>(groups.values());

}

}

[Remove Duplicates from Sorted Array](https://leetcode.com/problems/remove-duplicates-from-sorted-array/)

**Custom Judge:**

The judge will test your solution with the following code:

int[] nums = [...]; // Input array

int[] expectedNums = [...]; // The expected answer with correct length

int k = removeDuplicates(nums); // Calls your implementation

assert k == expectedNums.length;

for (int i = 0; i < k; i++) {

assert nums[i] == expectedNums[i];

}

If all assertions pass, then your solution will be **accepted**.

**Example 1:**

**Input:** nums = [1,1,2]

**Output:** 2, nums = [1,2,\_]

**Explanation:** Your function should return k = 2, with the first two elements of nums being 1 and 2 respectively.

It does not matter what you leave beyond the returned k (hence they are underscores).

**Example 2:**

**Input:** nums = [0,0,1,1,1,2,2,3,3,4]

**Output:** 5, nums = [0,1,2,3,4,\_,\_,\_,\_,\_]

**Explanation:** Your function should return k = 5, with the first five elements of nums being 0, 1, 2, 3, and 4 respectively.

It does not matter what you leave beyond the returned k (hence they are underscores).

**package** leetcode;

**public** **class** remove\_duplicate\_from\_sorted\_array16\_06\_2024 {

**public** **static** **void** main(String[] args) {

**int** num[] = {1,1,2,3,4,4,4,5};

System.***out***.println(*removeDuplicates*(num));

}

**public** **static** **int** removeDuplicates(**int**[] nums) {

**int** n = nums.length;

**int** insert = 1;

**if**(n == 0) {

**return** 0;

}

**for**(**int** i=1; i<n ; i++) {

**if**(nums[i]!=nums[i-1]) {

nums[insert] = nums[i];

insert ++;

}

}

**return** insert;

}

}

23 June 2024

search\_insert\_position23\_06\_2024

**Example 1:**

**Input:** nums = [1,3,5,6], target = 5

**Output:** 2

**Example 2:**

**Input:** nums = [1,3,5,6], target = 2

**Output:** 1

**Example 3:**

**Input:** nums = [1,3,5,6], target = 7

**Output:** 4

**package** leetcode;

**import** java.awt.Point;

**public** **class** search\_insert\_position23\_06\_2024 {

**public** **static** **void** main(String[] args) {

**int** a[] = { 1, 2, 4, 6 };

System.***out***.println(*searchInsert*(a, 5));

}

**public** **static** **int** searchInsert(**int**[] nums, **int** target) {

**int** pivot = 0;

**int** left = 0;

**int** right = nums.length-1;

**while** (left <= right) {

pivot = left + (right-left) /2;

**if**(nums[pivot] == target) {

**return** pivot;

}

**if**(target < nums[pivot]) {

right = pivot-1;

} **else** {

left = pivot + 1;

}

}

**return** left;

}

}