P.Bhavya 21.11.2024  
AIDS  
[**16. 3Sum Closest**](https://leetcode.com/problems/3sum-closest/)

Solved

Medium

Topics

Companies

Given an integer array nums of length n and an integer target, find three integers in nums such that the sum is closest to target.

Return *the sum of the three integers*.

You may assume that each input would have exactly one solution.

**Example 1:**

**Input:** nums = [-1,2,1,-4], target = 1  
**Output:** 2  
**Explanation:** The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).

**Example 2:**

**Input:** nums = [0,0,0], target = 1  
**Output:** 0  
**Explanation:** The sum that is closest to the target is 0. (0 + 0 + 0 = 0).

class Solution {

public int threeSumClosest(int[] nums, int target) {

Arrays.sort(nums);

int closest\_sum = Integer.MAX\_VALUE / 2;

for (int i = 0; i < nums.length - 2; ++i) {

int left = i + 1, right = nums.length - 1;

while (left < right) {

int current\_sum = nums[i] + nums[left] + nums[right]

if (Math.abs(current\_sum - target) < Math.abs(closest\_sum - target)) {

closest\_sum = current\_sum;

}

if (current\_sum < target) {

++left;

} else if (current\_sum > target) {

--right;

} else {

return current\_sum;

}

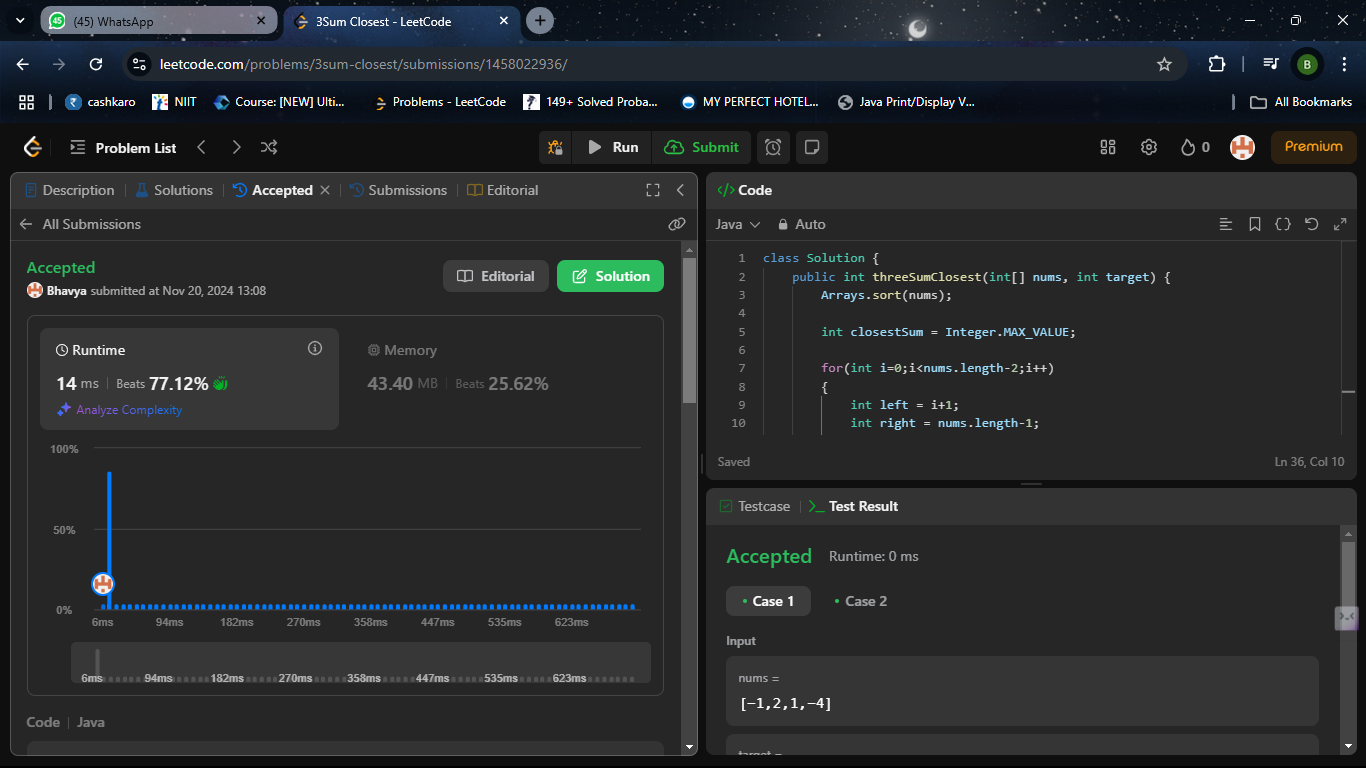
}

}

return closest\_sum;

}

}



[**45. Jump Game II**](https://leetcode.com/problems/jump-game-ii/)

Solved

Medium

Topics

Companies

You are given a **0-indexed** array of integers nums of length n. You are initially positioned at nums[0].

Each element nums[i] represents the maximum length of a forward jump from index i. In other words, if you are at nums[i], you can jump to any nums[i + j] where:

* 0 <= j <= nums[i] and
* i + j < n

Return *the minimum number of jumps to reach* nums[n - 1]. The test cases are generated such that you can reach nums[n - 1].

**Example 1:**

**Input:** nums = [2,3,1,1,4]  
**Output:** 2  
**Explanation:** The minimum number of jumps to reach the last index is 2. Jump 1 step from index 0 to 1, then 3 steps to the last index.

**Example 2:**

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

class Solution {

public int jump(int[] nums) {

int near = 0, far = 0, jumps = 0;

while (far < nums.length - 1) {

int farthest = 0;

for (int i = near; i <= far; i++) {

farthest = Math.max(farthest, i + nums[i]);

}

near = far + 1;

far = farthest;

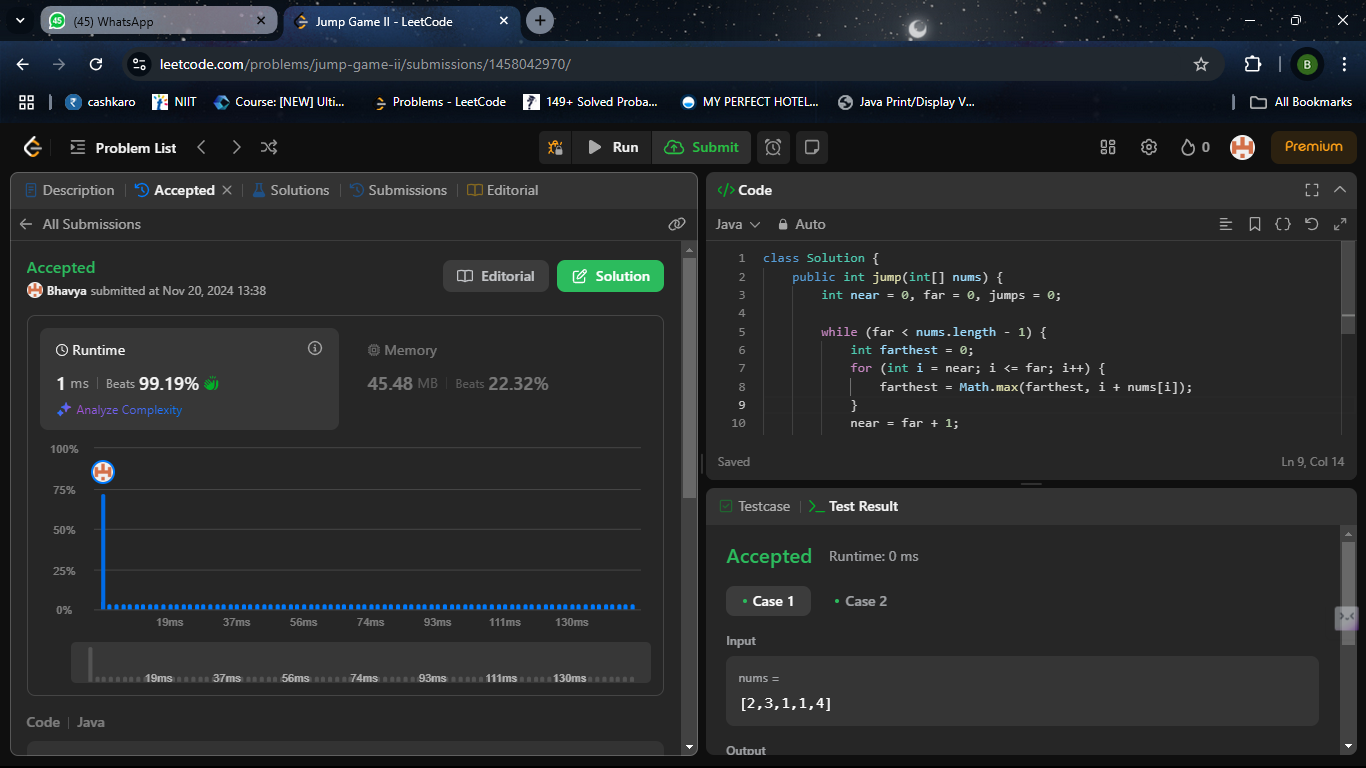
jumps++;

}

return jumps;

}

}

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

Solved

Medium

Topics

Companies

Given an array of strings strs, group the anagrams

together. You can return the answer in **any order**.

**Example 1:**

**Input:** strs = ["eat","tea","tan","ate","nat","bat"]

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

**Explanation:**

* There is no string in strs that can be rearranged to form "bat".
* The strings "nat" and "tan" are anagrams as they can be rearranged to form each other.
* The strings "ate", "eat", and "tea" are anagrams as they can be rearranged to form each other.

**Example 2:**

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

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

**Example 3:**

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

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

**Constraints:**

* 1 <= strs.length <= 104
* 0 <= strs[i].length <= 100
* strs[i] consists of lowercase English letters.{

class Solution   
public List<List<String>> groupAnagrams(String[] strs) {

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

for (String word : strs) {

char[] chars = word.toCharArray();

Arrays.sort(chars);

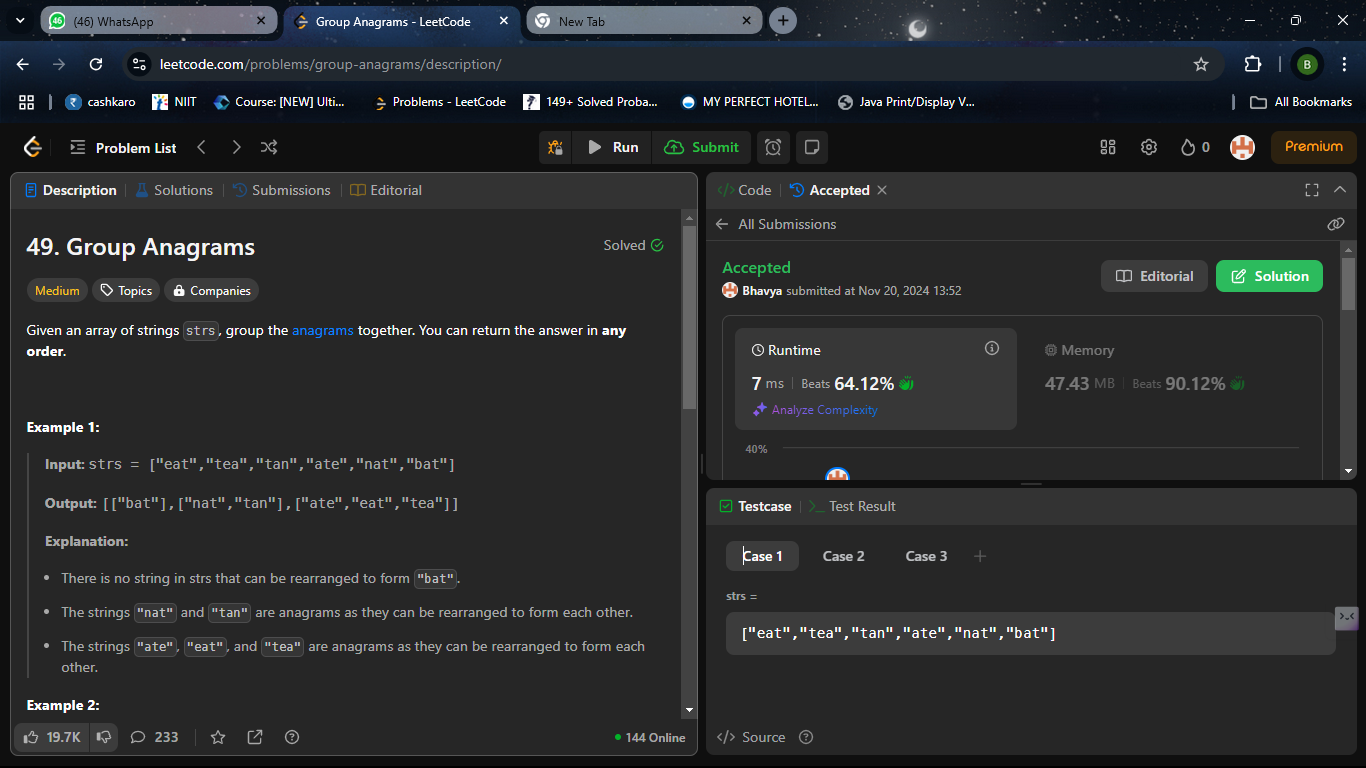
String sortedWord = new String(chars);

if (!map.containsKey(sortedWord)) {

map.put(sortedWord, new ArrayList<>());

}

map.get(sortedWord).add(word);

return new ArrayList<>(map.values());  
  
  
[**91. Decode Ways**](https://leetcode.com/problems/decode-ways/)

Solved

Medium

Topics

Companies

You have intercepted a secret message encoded as a string of numbers. The message is **decoded** via the following mapping:

"1" -> 'A'  
 "2" -> 'B'  
 ...  
 "25" -> 'Y'  
 "26" -> 'Z'

However, while decoding the message, you realize that there are many different ways you can decode the message because some codes are contained in other codes ("2" and "5" vs "25").

For example, "11106" can be decoded into:

* "AAJF" with the grouping (1, 1, 10, 6)
* "KJF" with the grouping (11, 10, 6)
* The grouping (1, 11, 06) is invalid because "06" is not a valid code (only "6" is valid).

Note: there may be strings that are impossible to decode.  
  
Given a string s containing only digits, return the **number of ways** to **decode** it. If the entire string cannot be decoded in any valid way, return 0.

The test cases are generated so that the answer fits in a **32-bit** integer.

**Example 1:**

**Input:** s = "12"

**Output:** 2

**Explanation:**

"12" could be decoded as "AB" (1 2) or "L" (12).

**Example 2:**

**Input:** s = "226"

**Output:** 3

**Explanation:**

"226" could be decoded as "BZ" (2 26), "VF" (22 6), or "BBF" (2 2 6).

**Example 3:**

**Input:** s = "06"

**Output:** 0

class Solution {

public int numDecodings(String s) {

int strLen = s.length();

int[] dp = new int[strLen + 1];

dp[0] = 1;

if (s.charAt(0) != '0') {

dp[1] = 1;

} else {

return 0;

}

for (int i = 2; i <= strLen; ++i) {

if (s.charAt(i - 1) != '0') {

dp[i] += dp[i - 1];

}

if (s.charAt(i - 2) == '1' ||

(s.charAt(i - 2) == '2' && s.charAt(i - 1) <= '6')) {

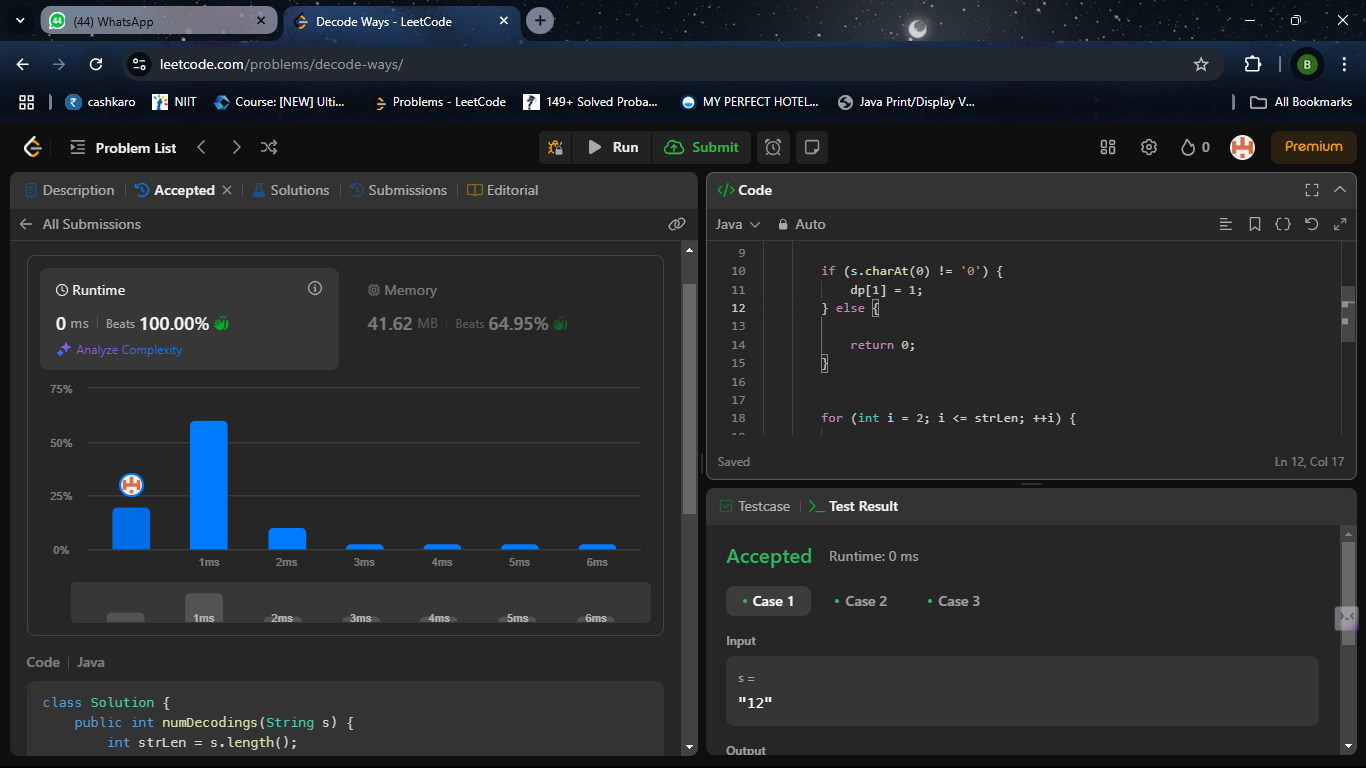
dp[i] += dp[i - 2];

}

}

return dp[strLen];

}

}  
  
  
[**122. Best Time to Buy and Sell Stock II**](https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/)

Solved

Medium

Topics

Companies

You are given an integer array prices where prices[i] is the price of a given stock on the ith day.

On each day, you may decide to buy and/or sell the stock. You can only hold **at most one** share of the stock at any time. However, you can buy it then immediately sell it on the **same day**.

Find and return *the* ***maximum*** *profit you can achieve*.

**Example 1:**

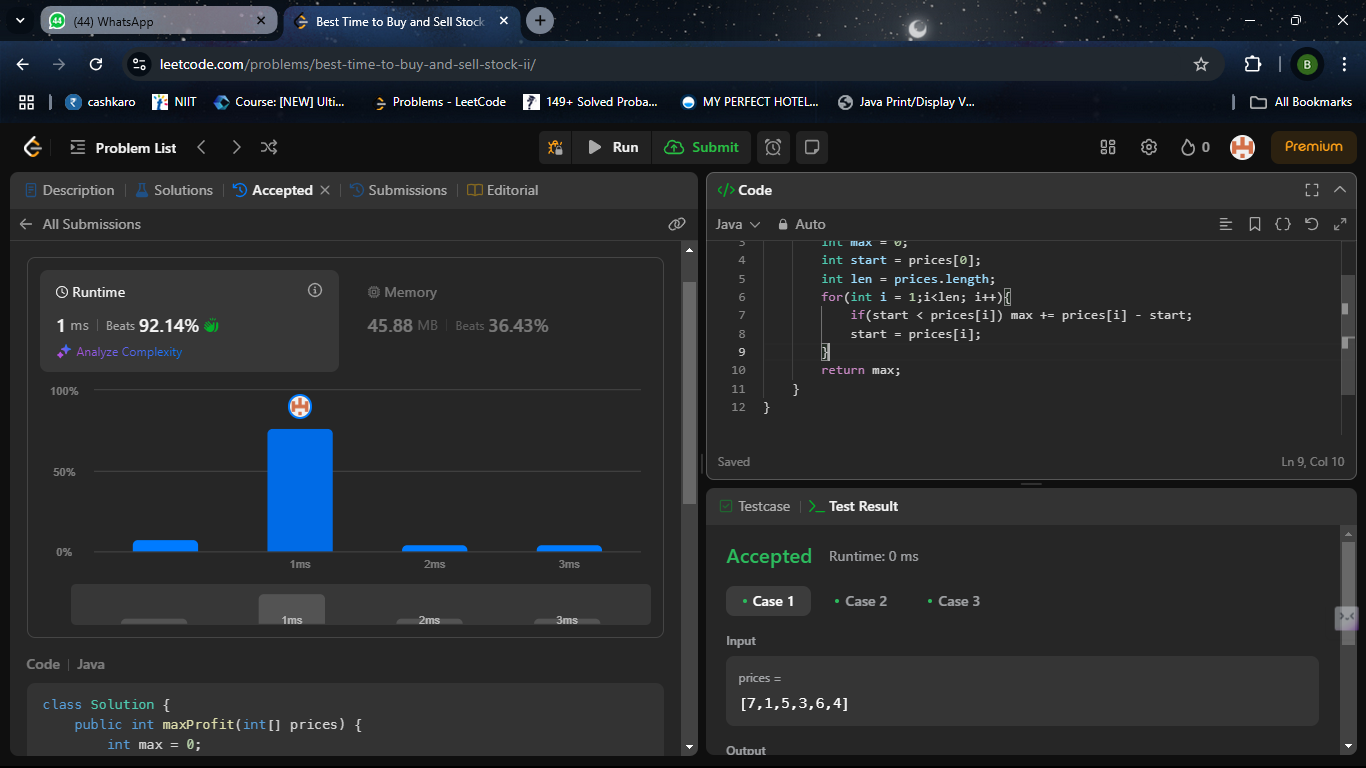
**Input:** prices = [7,1,5,3,6,4]  
**Output:** 7  
**Explanation:** Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit = 5-1 = 4.  
Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3.  
Total profit is 4 + 3 = 7.

**Example 2:**

**Input:** prices = [1,2,3,4,5]  
**Output:** 4  
**Explanation:** Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4.  
Total profit is 4.

**Example 3:**

* **Input:** prices = [7,6,4,3,1]  
  **Output:** 0  
  **Explanation:** There is no way to make a positive profit, so we never buy the stock to achieve the maximum profit of 0.  
    
    
  class Solution {
* public int maxProfit(int[] prices) {
* int max = 0;
* int start = prices[0];
* int len = prices.length;
* for(int i = 1;i<len; i++){
* if(start < prices[i]) max += prices[i] - start;
* start = prices[i];
* }
* return max;
* }
* }

  
[**200. Number of Islands**](https://leetcode.com/problems/number-of-islands/)

Medium

Given an m x n 2D binary grid grid which represents a map of '1's (land) and '0's (water), return *the number of islands*.An **island** is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.  
**Example 1:**

**Input:** grid = [  
 ["1","1","1","1","0"],  
 ["1","1","0","1","0"],  
 ["1","1","0","0","0"],  
 ["0","0","0","0","0"]  
]  
**Output:** 1

**Example 2:**

**Input:** grid = [  
 ["1","1","0","0","0"],  
 ["1","1","0","0","0"],  
 ["0","0","1","0","0"],  
 ["0","0","0","1","1"]  
]  
**Output:** 3  
  
 {

class Solution{  
public int numIslands(char[][] grid) {

int islands = 0;

int rows = grid.length;

int cols = grid[0].length;

Set<String> visited = new HashSet<>();

int[][] directions = {{1, 0}, {-1, 0}, {0, 1}, {0, -1}};

for (int r = 0; r < rows; r++) {

for (int c = 0; c < cols; c++) {

if (grid[r][c] == '1' && !visited.contains(r + "," + c)) {

islands++;

bfs(grid, r, c, visited, directions, rows, cols);

}

}

}

return islands;

}

private void bfs(char[][] grid, int r, int c, Set<String> visited, int[][] directions, int rows, int cols) {

Queue<int[]> q = new LinkedList<>();

visited.add(r + "," + c);

q.add(new int[]{r, c});

while (!q.isEmpty()) {

int[] point = q.poll();

int row = point[0], col = point[1];

for (int[] direction : directions) {

int nr = row + direction[0], nc = col + direction[1];

if (nr >= 0 && nr < rows && nc >= 0 && nc < cols && grid[nr][nc] == '1' && !visited.contains(nr + "," + nc)) {

q.add(new int[]{nr, nc});

visited.add(nr + "," + nc);

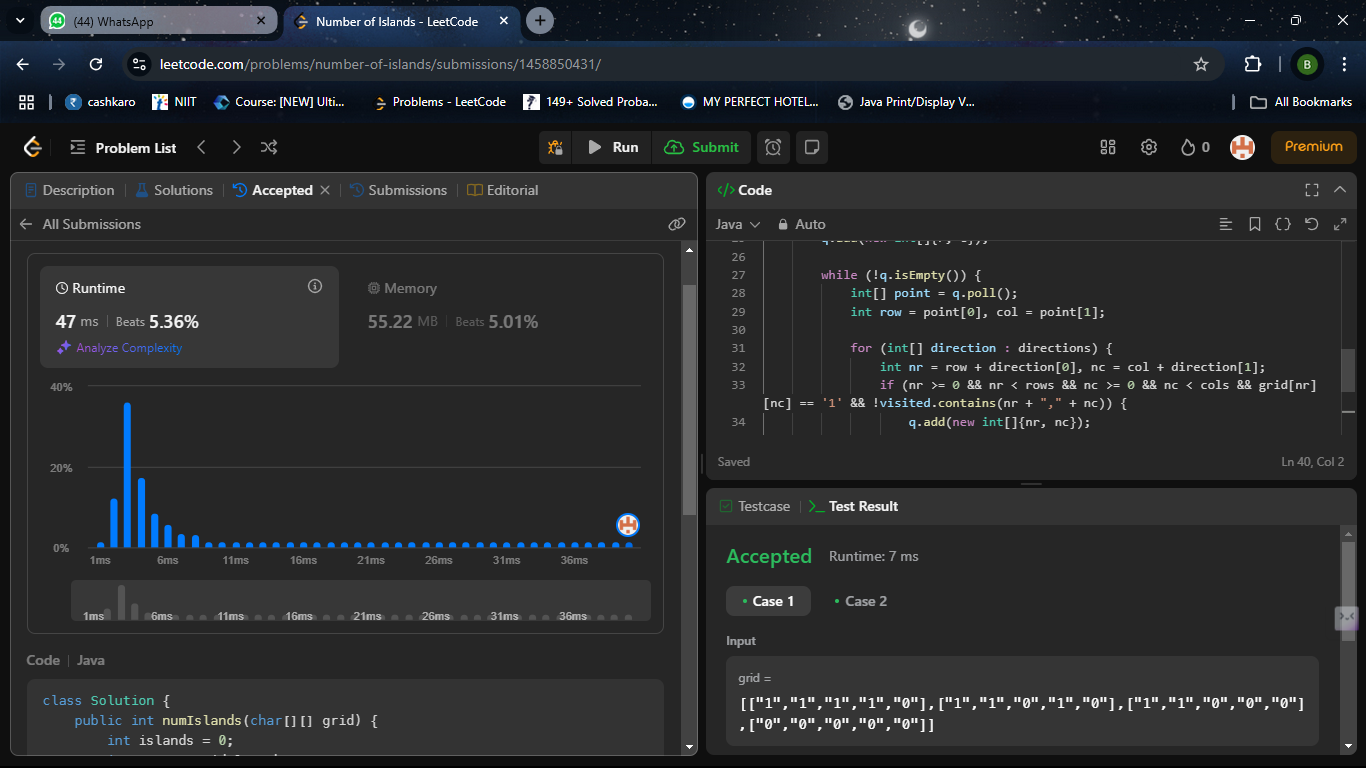
}

}

}

}

}

  
Quick sort

package maxsubarraysum;

import java.util.Scanner;

public class QuickSortExample {

public static void quickSort(int[] arr, int low, int high) {

if (low < high) {

int partitionIndex = *partition*(arr, low, high);

*quickSort*(arr, low, partitionIndex - 1);

*quickSort*(arr, partitionIndex + 1, high);

}

}

private static int partition(int[] arr, int low, int high) {

int pivot = arr[high];

int i = low - 1;

for (int j = low; j < high; j++) {

if (arr[j] <= pivot) {

i++;

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

int temp = arr[i + 1];

arr[i + 1] = arr[high];

arr[high] = temp;

return i + 1;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

System.***out***.print("Enter the number of elements in the array: ");

int n = scanner.nextInt();

int[] arr = new int[n];

System.***out***.println("Enter the elements of the array:");

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

arr[i] = scanner.nextInt();

}

*quickSort*(arr, 0, n - 1);

System.***out***.println("Sorted array:");

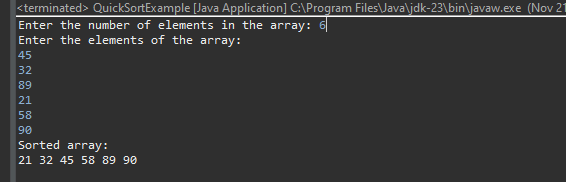
for (int num : arr) {

System.***out***.print(num + " ");

}

}

}

  
  
Merge sort

package maxsubarraysum; import java.util.Scanner;

public class MergeSortExample {

public static void mergeSort(int[] arr, int left, int right) {

if (left < right) {

int mid = left + (right - left) / 2;

*mergeSort*(arr, left, mid);

*mergeSort*(arr, mid + 1, right);

*merge*(arr, left, mid, right);

}

}

private static void merge(int[] arr, int left, int mid, int right) {

int n1 = mid - left + 1;

int n2 = right - mid;

int[] leftArray = new int[n1];

int[] rightArray = new int[n2];

for (int i = 0; i < n1; i++) leftArray[i] = arr[left + i];

for (int i = 0; i < n2; i++) rightArray[i] = arr[mid + 1 + i];

int i = 0, j = 0, k = left;

while (i < n1 && j < n2) {

if (leftArray[i] <= rightArray[j]) arr[k++] = leftArray[i++];

else arr[k++] = rightArray[j++];

}

while (i < n1) arr[k++] = leftArray[i++];

while (j < n2) arr[k++] = rightArray[j++];

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

System.***out***.print("Enter the number of elements in the array: ");

int n = scanner.nextInt();

int[] arr = new int[n];

System.***out***.println("Enter the elements of the array:");

for (int i = 0; i < n; i++) arr[i] = scanner.nextInt();

*mergeSort*(arr, 0, n - 1);

System.***out***.println("Sorted array:");

for (int num : arr) System.***out***.print(num + " ");

}

}

