**20-11-2024**

**CODING PRACTICE PROBLEMS**

**1.3Sum Closest**

import java.util.\*;

public class problem1 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

int[] nums = new int[n];

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

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

nums[i] = sc.nextInt();

}

System.out.println("Enter the target:");

int target = sc.nextInt();

System.out.println(threeSumClosest(nums, target));

}

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

Arrays.sort(nums);

int closestSum = 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 sum = nums[i] + nums[left] + nums[right];

if (Math.abs(target - sum) < Math.abs(target - closestSum)) {

closestSum = sum;

}

if (sum < target) {

left++;

} else {

right--;

}

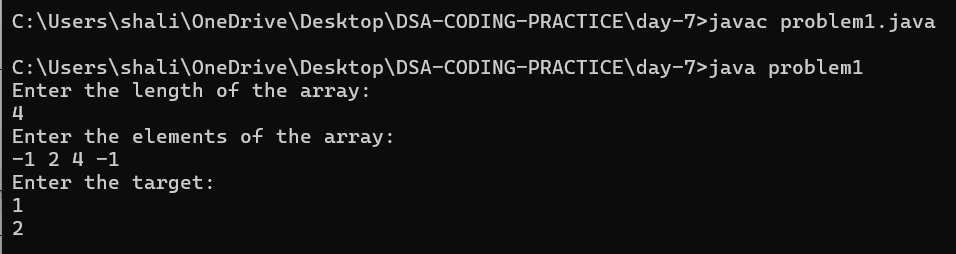
}

}

return closestSum;

}

}



**2.Jump game II**

import java.util.\*;

public class problem2 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

int[] nums = new int[n];

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

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

nums[i] = sc.nextInt();

}

System.out.println(minJumps(nums));

}

public static int minJumps(int[] nums) {

int jumps = 0, currentEnd = 0, farthest = 0;

for (int i = 0; i < nums.length - 1; i++) {

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

if (i == currentEnd) {

jumps++;

currentEnd = farthest;

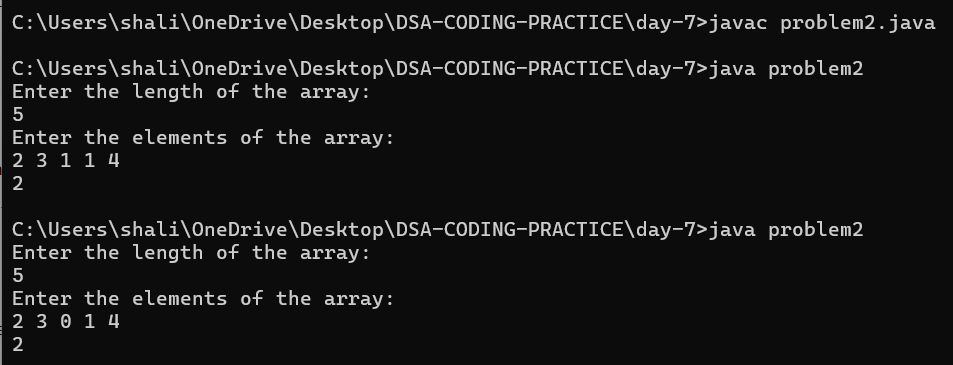
}

}

return jumps;

}

}



**3.Group Anagrams**

import java.util.\*;

public class problem3 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of strings:");

int n = sc.nextInt();

sc.nextLine();

String[] strs = new String[n];

System.out.println("Enter the strings:");

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

strs[i] = sc.nextLine();

}

List<List<String>> result = groupAnagrams(strs);

System.out.println(result);

}

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

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

for (String s : strs) {

char[] charArray = s.toCharArray();

Arrays.sort(charArray);

String sorted = new String(charArray);

map.putIfAbsent(sorted, new ArrayList<>());

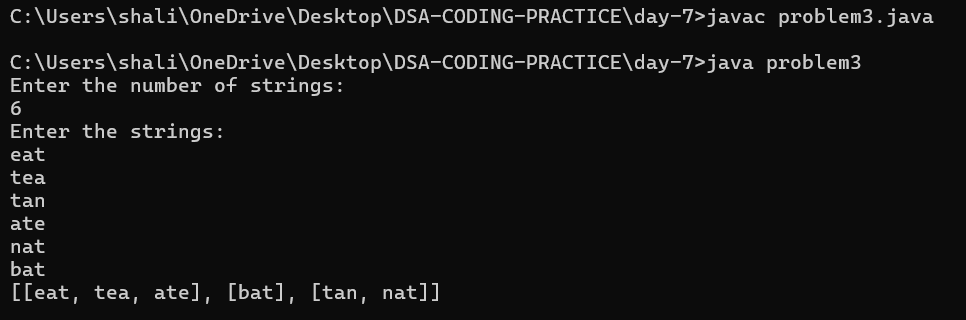
map.get(sorted).add(s);

}

return new ArrayList<>(map.values());

}

}



**4.Decode ways**

import java.util.\*;

public class problem4 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the encoded message:");

String s = sc.nextLine();

System.out.println(numDecodings(s));

}

public static int numDecodings(String s) {

if (s == null || s.length() == 0 || s.charAt(0) == '0') {

return 0;

}

int n = s.length();

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

dp[0] = 1;

dp[1] = s.charAt(0) != '0' ? 1 : 0;

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

int oneDigit = Integer.parseInt(s.substring(i - 1, i));

int twoDigits = Integer.parseInt(s.substring(i - 2, i));

if (oneDigit >= 1 && oneDigit <= 9) {

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

}

if (twoDigits >= 10 && twoDigits <= 26) {

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

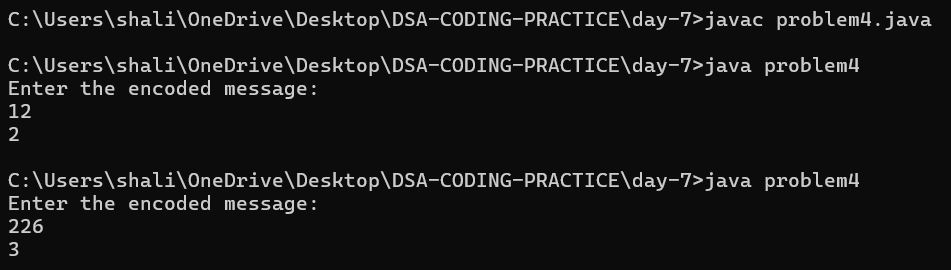
}

}

return dp[n];

}

}



**5.Best time to buy and sell tock II**

import java.util.\*;

public class problem5 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of days:");

int n = sc.nextInt();

int[] prices = new int[n];

System.out.println("Enter the stock prices:");

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

prices[i] = sc.nextInt();

}

System.out.println(maxProfit(prices));

}

public static int maxProfit(int[] prices) {

int maxProfit = 0;

for (int i = 1; i < prices.length; i++) {

if (prices[i] > prices[i - 1]) {

maxProfit += prices[i] - prices[i - 1];

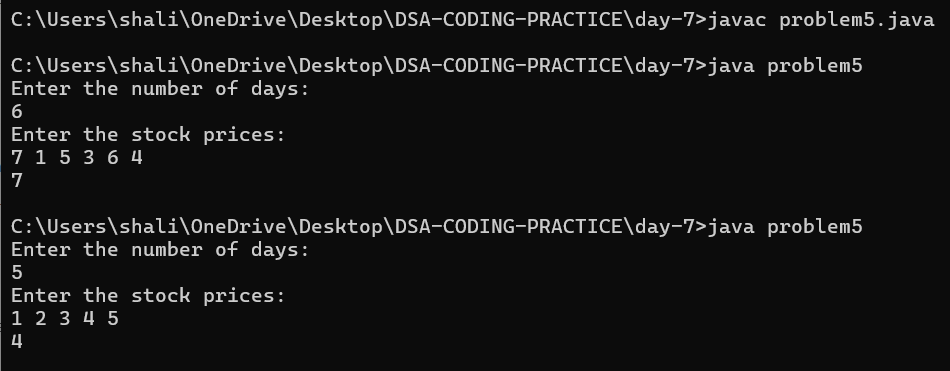
}

}

return maxProfit;

}

}



**6.Number of Islands**

import java.util.\*;

public class problem6 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of rows:");

int m = sc.nextInt();

System.out.println("Enter the number of columns:");

int n = sc.nextInt();

System.out.println("Enter the grid (row by row, space-separated):");

char[][] grid = new char[m][n];

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

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

grid[i][j] = sc.next().charAt(0);

}

}

System.out.println(numIslands(grid));

}

public static int numIslands(char[][] grid) {

int m = grid.length;

int n = grid[0].length;

int count = 0;

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

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

if (grid[i][j] == '1') {

count++;

dfs(grid, i, j);

}

}

}

return count;

}

private static void dfs(char[][] grid, int i, int j) {

int m = grid.length;

int n = grid[0].length;

if (i < 0 || j < 0 || i >= m || j >= n || grid[i][j] == '0') {

return;

}

grid[i][j] = '0';

dfs(grid, i - 1, j);

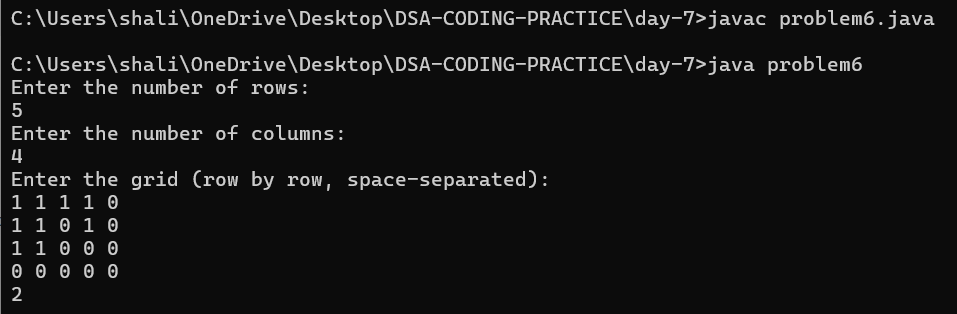
dfs(grid, i + 1, j);

dfs(grid, i, j - 1);

dfs(grid, i, j + 1);

}

}



**7.Quick Sort**

import java.util.\*;

public class problem7 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

int[] arr = new int[n];

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

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

arr[i] = sc.nextInt();

}

quickSort(arr, 0, n - 1);

System.out.println("Sorted array: " + Arrays.toString(arr));

}

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

if (low < high) {

int pivotIndex = partition(arr, low, high);

quickSort(arr, low, pivotIndex - 1);

quickSort(arr, pivotIndex + 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++;

swap(arr, i, j);

}

}

swap(arr, i + 1, high);

return i + 1;

}

private static void swap(int[] arr, int i, int j) {

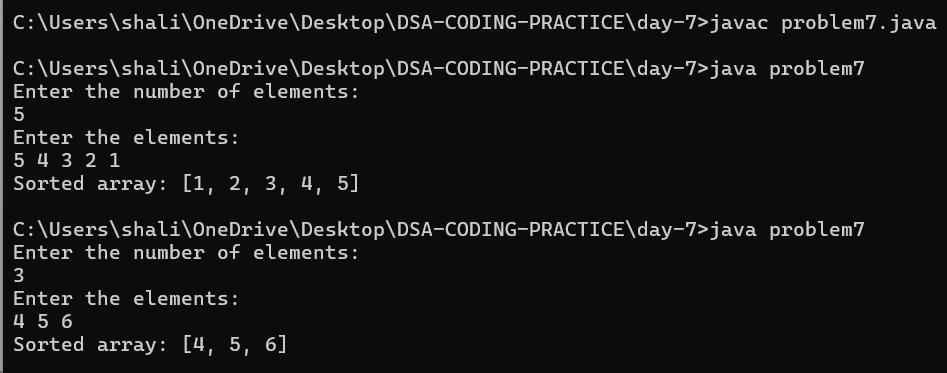
int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}



**8.Merge sort**

import java.util.\*;

public class problem8 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

int[] arr = new int[n];

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

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

arr[i] = sc.nextInt();

}

mergeSort(arr, 0, n - 1);

System.out.println("Sorted array: " + Arrays.toString(arr));

}

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[] L = new int[n1];

int[] R = new int[n2];

for (int i = 0; i < n1; i++) {

L[i] = arr[left + i];

}

for (int i = 0; i < n2; i++) {

R[i] = arr[mid + 1 + i];

}

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

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

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

} else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

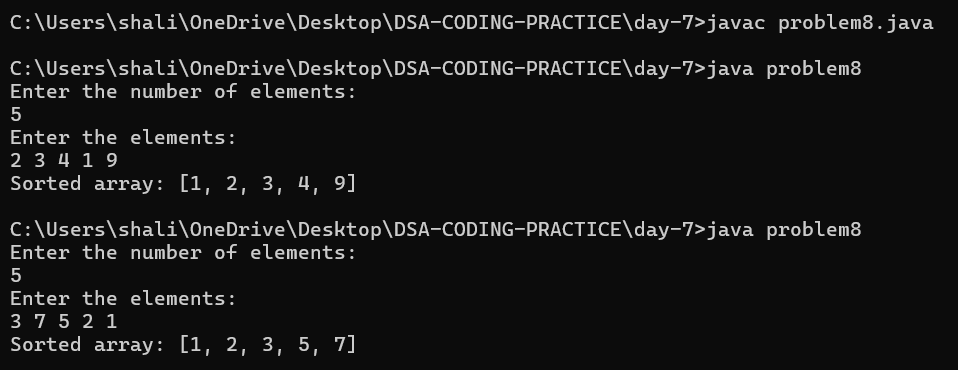
j++;

k++;

}

}

}



**9.Ternary Search**

import java.util.\*;

public class problem9 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

int[] arr = new int[n];

System.out.println("Enter the sorted elements:");

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

arr[i] = sc.nextInt();

}

System.out.println("Enter the element to search:");

int key = sc.nextInt();

int result = ternarySearch(arr, 0, n - 1, key);

System.out.println(result == -1 ? "Element not found" : "Element found at index: " + result);

}

public static int ternarySearch(int[] arr, int left, int right, int key) {

if (right >= left) {

int mid1 = left + (right - left) / 3;

int mid2 = right - (right - left) / 3;

if (arr[mid1] == key) return mid1;

if (arr[mid2] == key) return mid2;

if (key < arr[mid1]) return ternarySearch(arr, left, mid1 - 1, key);

else if (key > arr[mid2]) return ternarySearch(arr, mid2 + 1, right, key);

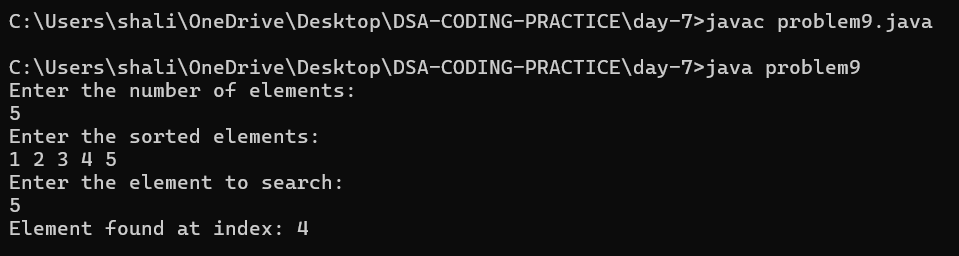
else return ternarySearch(arr, mid1 + 1, mid2 - 1, key);

}

return -1;

}

}



**10.Interpolation Search**

import java.util.\*;

public class problem10 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

int[] arr = new int[n];

System.out.println("Enter the sorted elements:");

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

arr[i] = sc.nextInt();

}

System.out.println("Enter the element to search:");

int key = sc.nextInt();

int result = interpolationSearch(arr, key);

System.out.println(result == -1 ? "Element not found" : "Element found at index: " + result);

}

public static int interpolationSearch(int[] arr, int key) {

int low = 0, high = arr.length - 1;

while (low <= high && key >= arr[low] && key <= arr[high]) {

if (low == high) {

if (arr[low] == key) return low;

return -1;

}

int pos = low + ((key - arr[low]) \* (high - low)) / (arr[high] - arr[low]);

if (arr[pos] == key) return pos;

if (arr[pos] < key) low = pos + 1;

else high = pos - 1;

}

return -1;

}

}

