**DSA PROBLEMS**

1. 3sum closest

import java.util.Arrays;

public class Main {

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

Arrays.sort(nums);

int closestSum = nums[0] + nums[1] + nums[2];

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

int left = i + 1;

int right = nums.length -1;

while (left < right) {

int currentSum = nums[i] + nums[left] + nums[right];

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

closestSum = currentSum;

}

if (currentSum < target) {

left++;

} else if (currentSum > target) {

right--;

} else {

return currentSum;

}

}

}

return closestSum;

}

public static void main(String[] args) {

Main solution = new Main();

int[] nums1 = {-1, 2, 1, -4};

int target1 = 1;

System.out.println("Closest Sum: " + solution.threeSumClosest(nums1, target1));

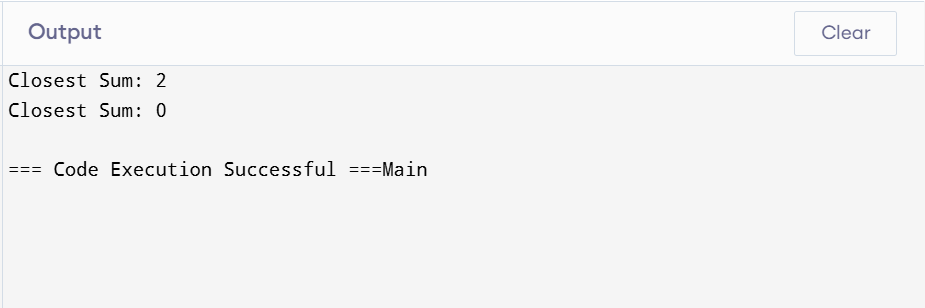
int[] nums2 = {0, 0, 0};

int target2 = 1;

System.out.println("Closest Sum: " + solution.threeSumClosest(nums2, target2));

}

}



1. Jump game II

public class Main {

public static void main(String[] args) {

int[] nums1 = {2, 3, 1, 1, 4};

int[] nums2 = {2, 3, 0, 1, 4};

System.out.println(minJumps(nums1));

System.out.println(minJumps(nums2));

}

public static int minJumps(int[] nums) {

int jumps = 0;

int currentEnd = 0;

int farthest = 0;

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

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

if (i == currentEnd) {

jumps++;

currentEnd = farthest;

if (currentEnd >= nums.length - 1) {

break;

}

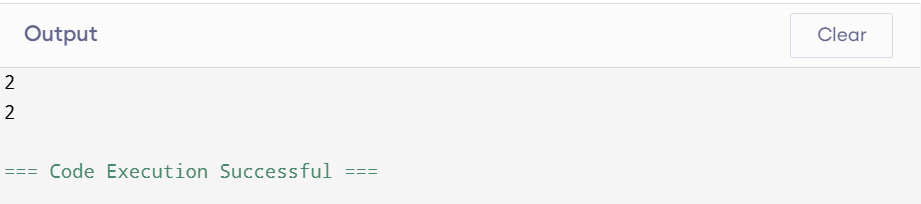
}

}

return jumps;

}

}



3 . group anagrams

import java.util.\*;

public class Main {

public static void main(String[] args) {

String[] strs1 = {"eat", "tea", "tan", "ate", "nat", "bat"};

String[] strs2 = {""};

String[] strs3 = {"a"};

System.out.println(groupAnagrams(strs1));

System.out.println(groupAnagrams(strs2));

System.out.println(groupAnagrams(strs3));

}

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

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

for (String str : strs) {

char[] charArray = str.toCharArray();

Arrays.sort(charArray);

String sortedStr = new String(charArray);

anagramGroups.putIfAbsent(sortedStr, new ArrayList<>());

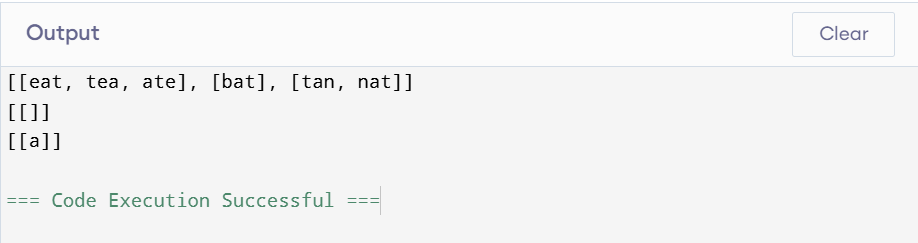
anagramGroups.get(sortedStr).add(str);

}

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

}

}



4.Decode ways

public class Main {

public static void main(String[] args) {

// Test cases

System.out.println(numDecodings("12"));

System.out.println(numDecodings("226"));

System.out.println(numDecodings("06"));

}

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' ? 0 : 1;

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

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

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

}

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

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

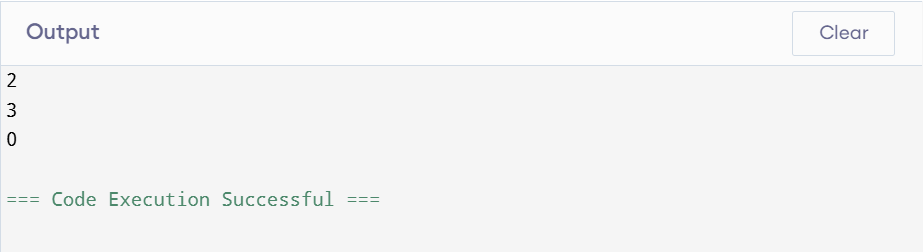
dp[i] += dp[i - 2];

}

}

return dp[n];

}

}  
  


5. Best time to buy and sell stocks II

public class Main {

public static void main(String[] args) {

// Test cases

int[] prices1 = {7, 1, 5, 3, 6, 4};

int[] prices2 = {1, 2, 3, 4, 5};

int[] prices3 = {7, 6, 4, 3, 1};

System.out.println(maxProfit(prices1)); // Output: 7

System.out.println(maxProfit(prices2)); // Output: 4

System.out.println(maxProfit(prices3)); // Output: 0

}

public static int maxProfit(int[] prices) {

int maxProfit = 0;

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

// Add profit whenever the current price is higher than the previous price

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

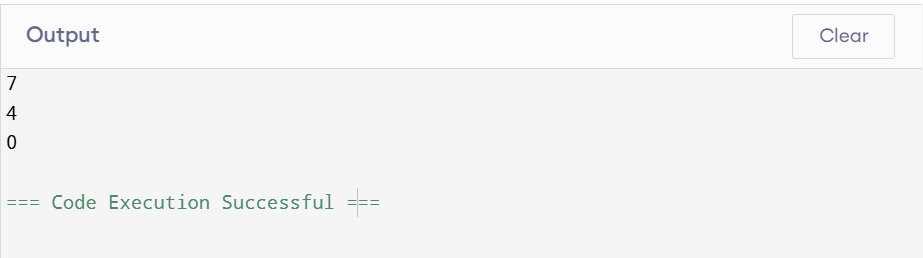
maxProfit += prices[i] - prices[i - 1];

}

}

return maxProfit;

}

}  
  


6. Number of islands

public class Main {

public static void main(String[] args) {

char[][] grid1 = {

{'1', '1', '1', '1', '0'},

{'1', '1', '0', '1', '0'},

{'1', '1', '0', '0', '0'},

{'0', '0', '0', '0', '0'}

};

char[][] grid2 = {

{'1', '1', '0', '0', '0'},

{'1', '1', '0', '0', '0'},

{'0', '0', '1', '0', '0'},

{'0', '0', '0', '1', '1'}

};

System.out.println(numIslands(grid1));

System.out.println(numIslands(grid2));

}

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

if (grid == null || grid.length == 0) {

return 0;

}

int numIslands = 0;

int rows = grid.length;

int cols = grid[0].length;

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

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

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

numIslands++;

dfs(grid, i, j);

}

}

}

return numIslands;

}

private static void dfs(char[][] grid, int row, int col) {

if (row < 0 || col < 0 || row >= grid.length || col >= grid[0].length || grid[row][col] == '0') {

return;

}

grid[row][col] = '0';

dfs(grid, row - 1, col);

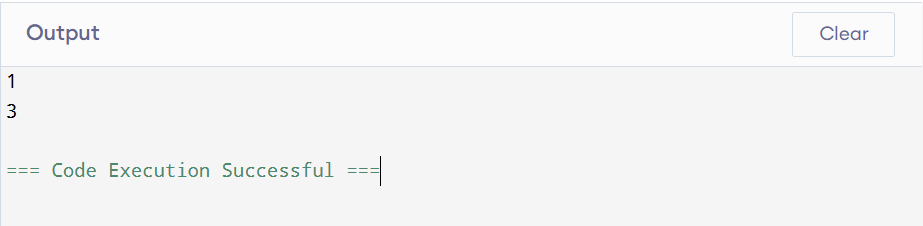
dfs(grid, row + 1, col);

dfs(grid, row, col - 1);

dfs(grid, row, col + 1);

}

}



7. Quick sort

import java.util.Scanner;

public class Main {

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

if (low < high) {

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

quickSort(array, low, pivotIndex - 1);

quickSort(array, pivotIndex + 1, high);

}

}

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

int pivot = array[high];

int i = low - 1;

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

if (array[j] <= pivot) {

i++;

int temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

int temp = array[i + 1];

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

array[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: ");

int n = scanner.nextInt();

int[] array = new int[n];

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

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

array[i] = scanner.nextInt();

}

quickSort(array, 0, array.length - 1);

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

for (int num : array) {

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

}

}

}



8 . Merge sort

import java.util.Scanner;

public class Main {

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

if (left < right) {

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

mergeSort(array, left, mid);

mergeSort(array, mid + 1, right);

merge(array, left, mid, right);

}

}

private static void merge(int[] array, 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] = array[left + i];

for (int j = 0; j < n2; j++) rightArray[j] = array[mid + 1 + j];

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

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

if (leftArray[i] <= rightArray[j]) {

array[k] = leftArray[i];

i++;

} else {

array[k] = rightArray[j];

j++;

}

k++;

}

while (i < n1) {

array[k] = leftArray[i];

i++;

k++;

}

while (j < n2) {

array[k] = rightArray[j];

j++;

k++;

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

int[] array = new int[n];

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

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

array[i] = scanner.nextInt();

}

mergeSort(array, 0, array.length - 1);

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

for (int num : array) {

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

}

}

}



9.ternary search

import java.util.Scanner;

public class Main {

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

if (right >= left) {

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

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

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

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

if (key < array[mid1]) {

return ternarySearch(array, left, mid1 - 1, key);

} else if (key > array[mid2]) {

return ternarySearch(array, mid2 + 1, right, key);

} else {

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

}

}

return -1; // Key not found

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

int[] array = new int[n];

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

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

array[i] = scanner.nextInt();

}

System.out.print("Enter the key to search: ");

int key = scanner.nextInt();

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

if (result == -1) {

System.out.println("Element not found");

} else {

System.out.println("Element found at index: " + result);

}

}

}



10 . Interpolation search

import java.util.Scanner;

public class Main {

public static int interpolationSearch(int[] array, int low, int high, int key) {

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

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

if (array[pos] == key) {

return pos;

}

if (array[pos] < key) {

low = pos + 1;

} else {

high = pos - 1;

}

}

return -1;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

int[] array = new int[n];

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

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

array[i] = scanner.nextInt();

}

System.out.print("Enter the key to search: ");

int key = scanner.nextInt();

int result = interpolationSearch(array, 0, n - 1, key);

if (result == -1) {

System.out.println("Element not found");

} else {

System.out.println("Element found at index: " + result);

}

}

}

