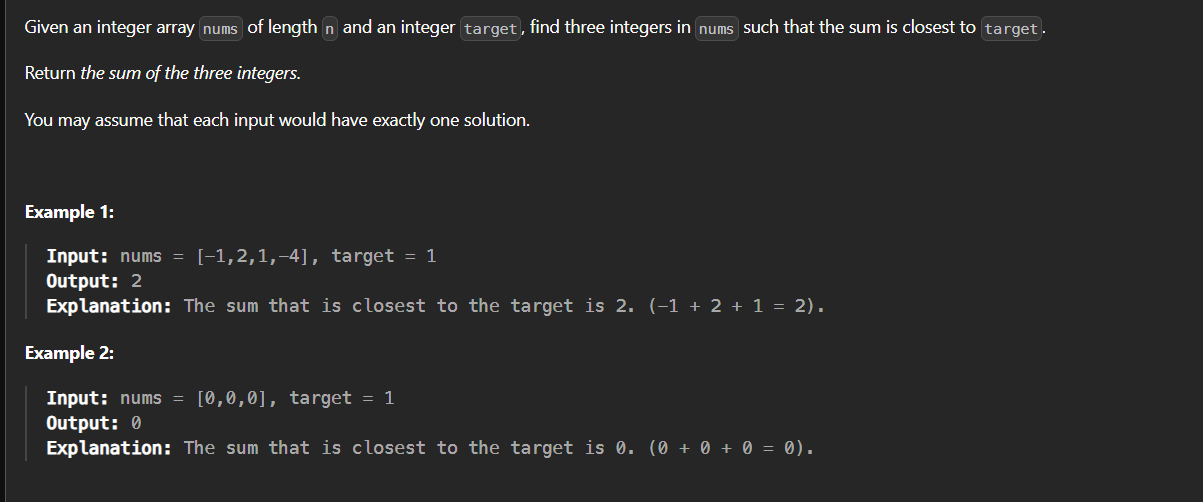
**DSA Practice Test – 8** 20th Nov 2024

**1. 3 Sum Closest:**



**Code:**

import java.util.Arrays;

public class Closest3Sum {

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

int c=Integer.MAX\_VALUE;

int l,r;

Arrays.sort(nums);

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

l=i+1;

r=nums.length-1;

while(l<r){

int s=nums[i]+nums[l]+nums[r];

if(s>target)

r--;

else

l++;

if(Math.abs(s-target)<Math.abs(c-target)){

c=s;

}

}

}

return c;

}

public static void main(String ar[]){

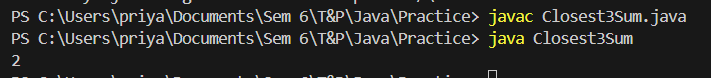
int[] nums={-1,2,1,-4};

System.out.println(threeSumClosest(nums, 2));

}

}

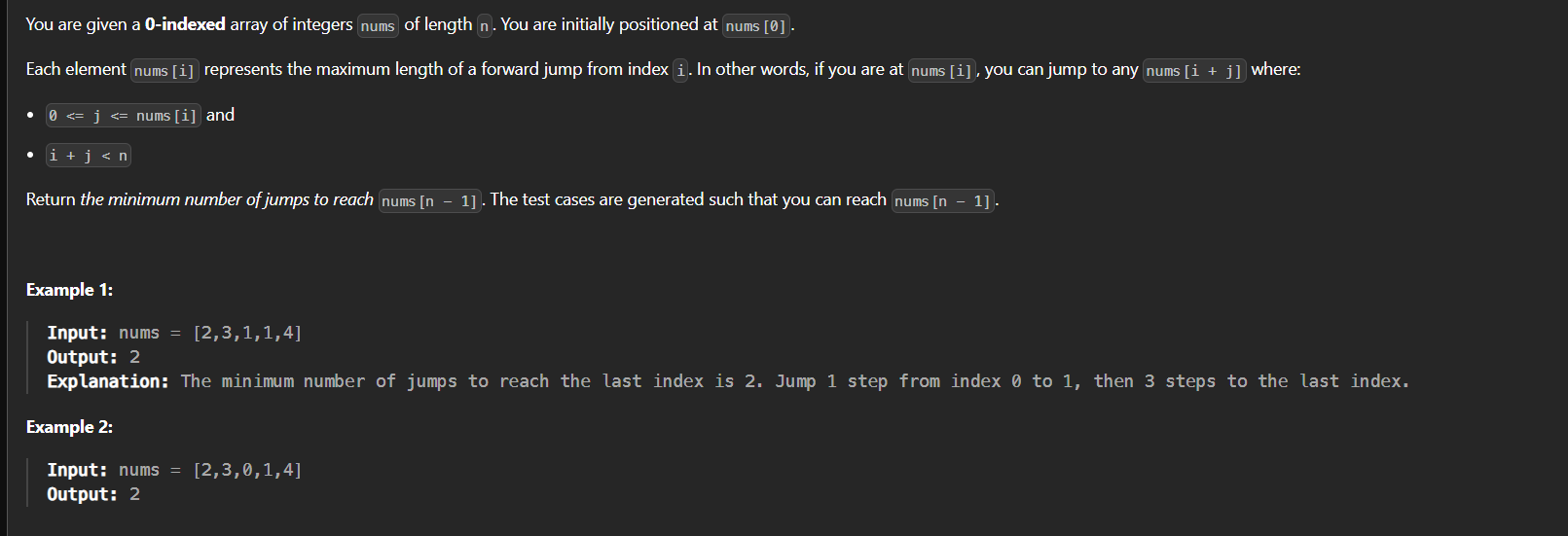
**Output**:



**Time Complexity:** O (n2)

**Space Complexity:** O (1)

**2. Jump Game II**



**Code:**

class JumpGame {

    static int jump(int[] nums) {

        int t = 0, c = 0, far = 0;

        if (nums.length == 1) return 0;

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

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

            if (i == c) {

                if (c < nums.length - 1) {

                    t++;

                    c = far;

                }

            }

        }

        return t;

    }

    public static void main(String ar[]){

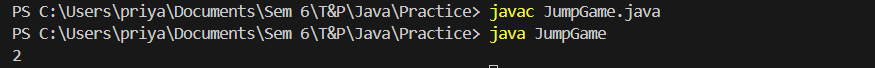
        int[] nums={2,3,1,1,4};

        System.out.print(jump(nums));

    }

}

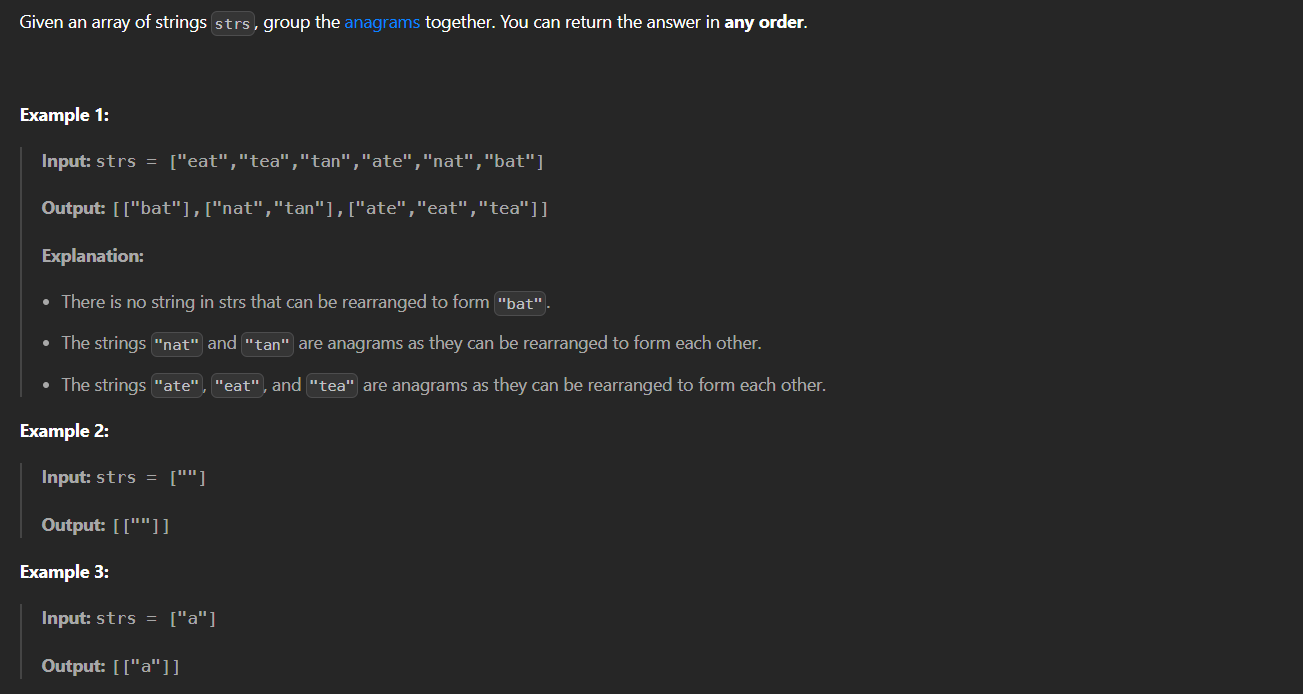
**Output:**



**Time Complexity:** O (n)

**Space Complexity:** O (1)

**3. Group Anagrams**



**Code:**

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

class GroupAnagrams {

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

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

for(String i:strs){

char[] c=i.toCharArray();

Arrays.sort(c);

String s=new String(c);

if(!map.containsKey(s)){

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

}

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

}

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

}

public static void main(String ar[]){

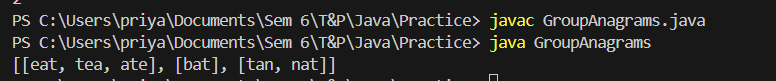
String[] strs={"eat","tea","tan","ate","nat","bat"};

System.out.print(groupAnagrams(strs));

}

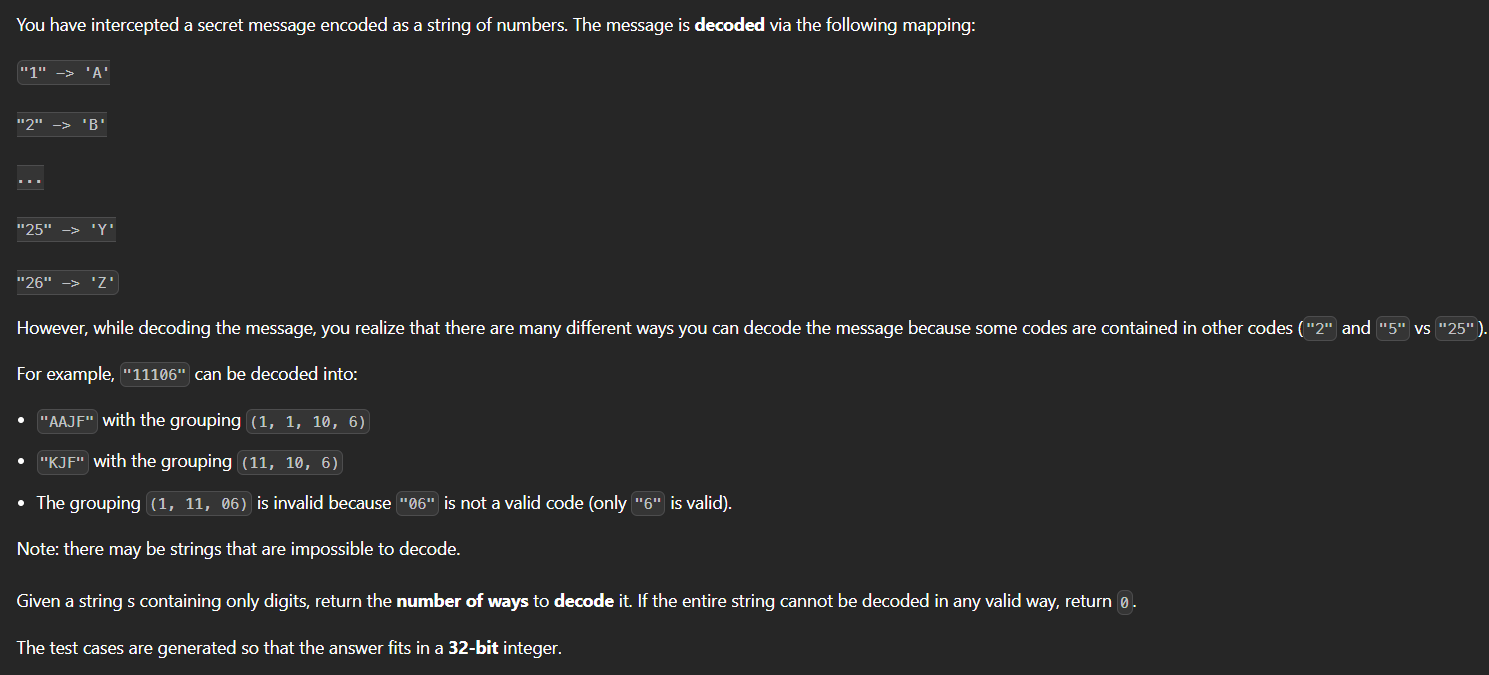
}

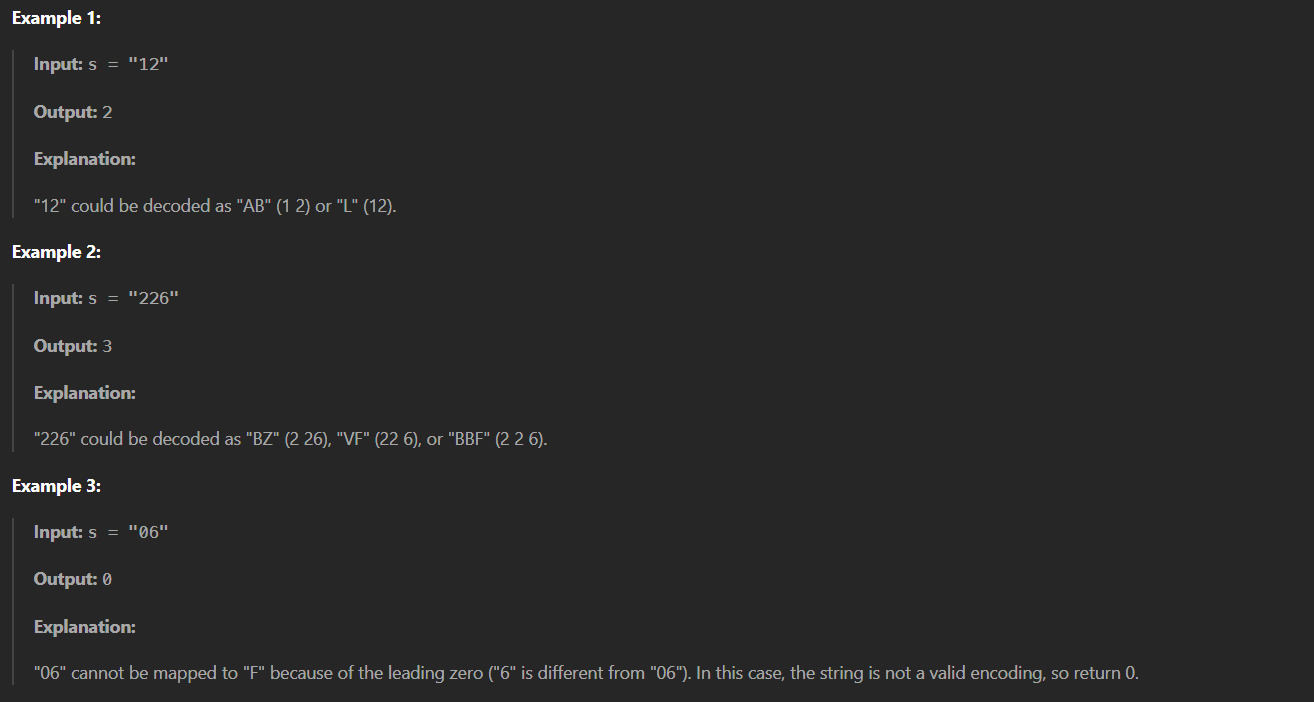
**Output:**



**Time Complexity:** O (n log n)  
**Space Complexity:** O (1)

**4. Decode Ways**





**Code:**

public class DecodeWays {

public static int numDecodings(String s) {

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

return 0;

}

int n = s.length();

int prev2 = 1;

int prev1 = 1;

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

int current = 0;

int oneDigit = s.charAt(i) - '0';

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

if (oneDigit != 0) {

current += prev1;

}

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

current += prev2;

}

prev2 = prev1;

prev1 = current;

}

return prev1;

}

public static void main(String ar[]){

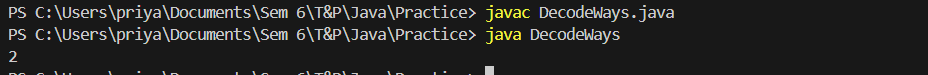
String s="12";

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

}

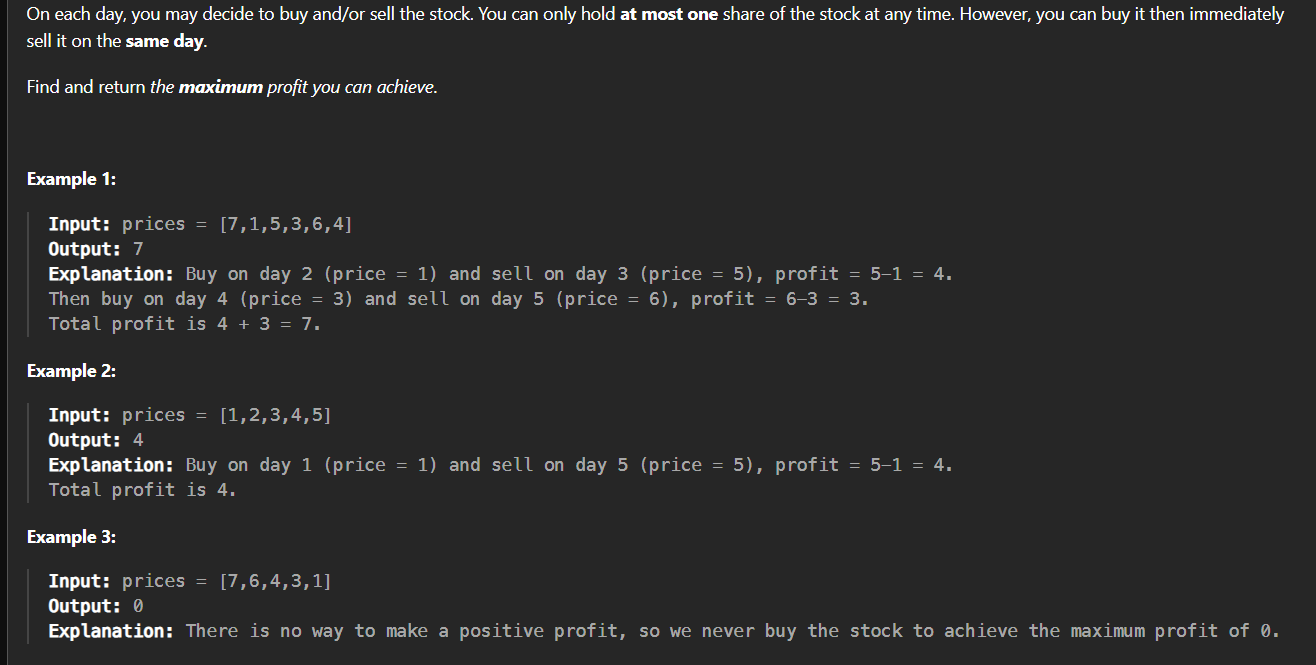
}

**Output:**



**Time Complexity:** O (n)  
**Space Complexity:** O (1)

**5. Best Time to Buy and Sell Stock II**



**Code:**class BuyandSell2 {

    public static int maxProfit(int[] prices) {

        int b=prices[0];

        int tp=0;

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

            if(i!=0){

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

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

                    b=prices[i];

                }

            }

        }

        return tp;

    }

    public static void main (String ar[]){

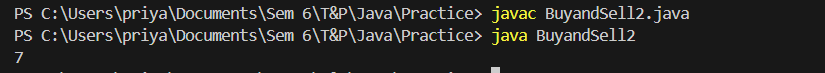
        int[] p={7,1,5,3,6,4};

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

    }

}

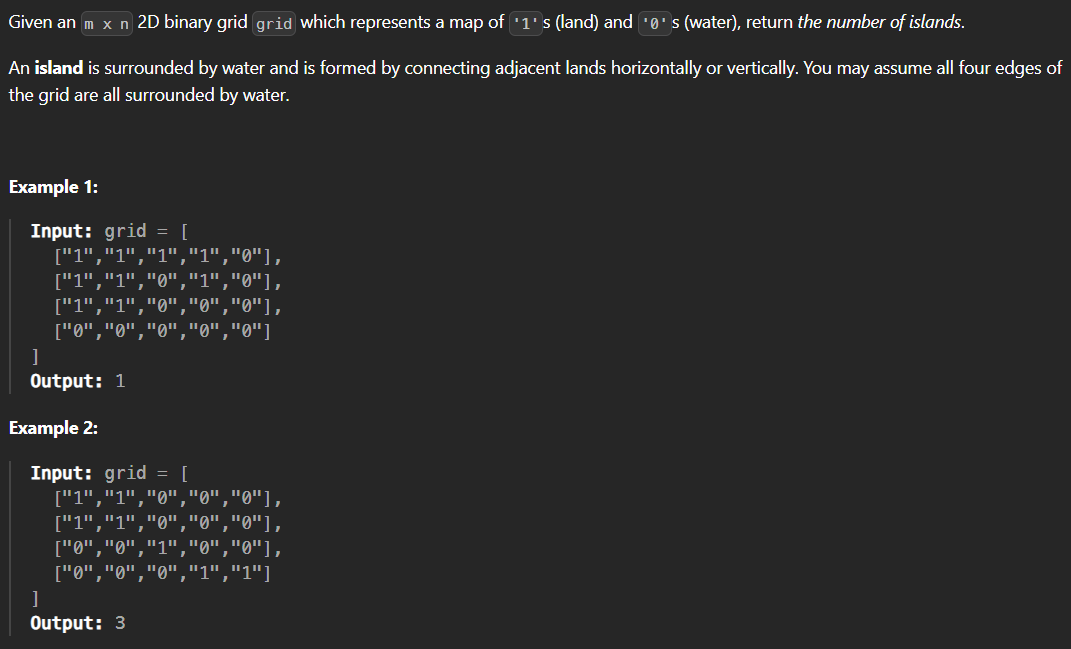
**Output:**

****

**Time Complexity:** O (n)

**Space Complexity:** O (1)

**6. Number Of Islands**



**Code:**

import java.util.LinkedList;

import java.util.Queue;

class Island {

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

int islands = 0;

int rows = grid.length;

int cols = grid[0].length;

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

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

if (grid[r][c] == '1') {

islands++;

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

}

}

}

return islands;

}

private static void bfs(char[][] grid, int r, int c, int rows, int cols) {

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

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

grid[r][c] = '0';

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

while (!queue.isEmpty()) {

int[] point = queue.poll();

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

for (int[] direction : directions) {

int nr = row + direction[0];

int nc = col + direction[1];

if (nr >= 0 && nr < rows && nc >= 0 && nc < cols && grid[nr][nc] == '1') {

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

grid[nr][nc] = '0';

}

}

}

}

public static void main(String priya[]){

char[][] grid={

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

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

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

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

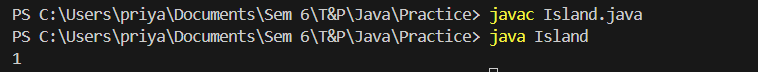
};

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

}

}

**Output:**

****

**Time Complexity:** O (r\*c)

**Space Complexity:** O (min(r,c))

**7. Quick Sort**

**Code:**

class QuickSort {

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

int pivot = arr[high];

int i = low - 1;

for (int j = low; j <= high - 1; j++) {

if (arr[j] < pivot) {

i++;

swap(arr, i, j);

}

}

swap(arr, i + 1, high);

return i + 1;

}

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

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

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

if (low < high) {

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

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

public static void main(String[] args) {

int[] arr = {10, 7, 8, 9, 1, 5};

int n = arr.length;

quickSort(arr, 0, n - 1);

for (int val : arr) {

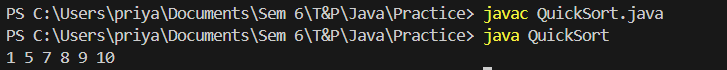
System.out.print(val + " ");

}

}

}

**Output:**



**Time Complexity:** O (n)

**Space Complexity:** O (1)

**8. Merge Sort**

**Code:**

// Java program for Merge Sort

class MergeSort {

static void merge(int arr[], int l, int m, int r)

{

int n1 = m - l + 1;

int n2 = r - m;

int L[] = new int[n1];

int R[] = new int[n2];

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

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

for (int j = 0; j < n2; ++j)

R[j] = arr[m + 1 + j];

int i = 0, j = 0;

int k = l;

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++;

}

}

static void sort(int arr[], int l, int r)

{

if (l < r) {

int m = l + (r - l) / 2;

sort(arr, l, m);

sort(arr, m + 1, r);

merge(arr, l, m, r);

}

}

static void printArray(int arr[])

{

int n = arr.length;

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

System.out.print(arr[i] + " ");

System.out.println();

}

public static void main(String args[])

{

int arr[] = { 12, 11, 13, 5, 6, 7 };

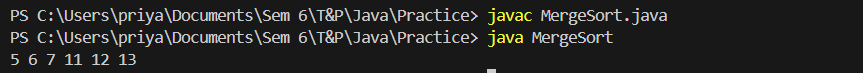
sort(arr, 0, arr.length - 1);

printArray(arr);

}

}

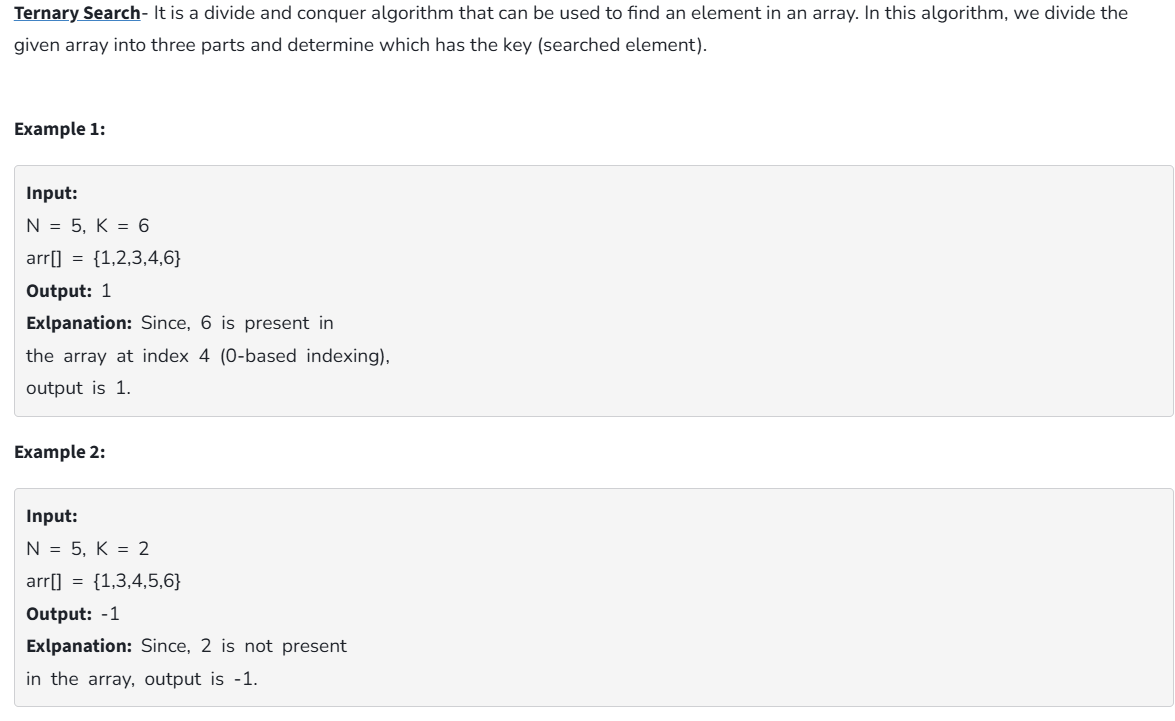
**Output:**



**Time Complexity:** O (n log n)

**Space Complexity:** O (n)

**9. Ternary Search**



**Code:**

class TernarySearch {

static int ternarySearch(int l, int r, int key, int ar[])

{

if (r >= l) {

int mid1 = l + (r - l) / 3;

int mid2 = r - (r - l) / 3;

if (ar[mid1] == key) {

return mid1;

}

if (ar[mid2] == key) {

return mid2;

}

if (key < ar[mid1]) {

return ternarySearch(l, mid1 - 1, key, ar);

}

else if (key > ar[mid2]) {

return ternarySearch(mid2 + 1, r, key, ar);

}

else {

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

}

}

return -1;

}

public static void main(String args[])

{

int l, r, p, key;

int ar[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

l = 0;

r = ar.length-1;

key = 5;

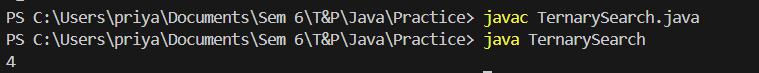
p = ternarySearch(l, r, key, ar);

System.out.println(p);

}

}

**Output:**

****

**Time Complexity:** O(2 \* log3n)

**Space Complexity:** O(log3n)

**10. Interpolation Search**

Given a sorted array of n uniformly distributed values arr[], write a function to search for a particular element x in the array.

**Code:**

class InterpolationSearch {

public static int interpolationSearch(int arr[], int lo,

int hi, int x)

{

int pos;

if (lo <= hi && x >= arr[lo] && x <= arr[hi]) {

pos = lo

+ (((hi - lo) / (arr[hi] - arr[lo]))

\* (x - arr[lo]));

if (arr[pos] == x)

return pos;

if (arr[pos] < x)

return interpolationSearch(arr, pos + 1, hi,

x);

if (arr[pos] > x)

return interpolationSearch(arr, lo, pos - 1,

x);

}

return -1;

}

public static void main(String[] args)

{

int arr[] = { 10, 12, 13, 16, 18, 19, 20, 21,

22, 23, 24, 33, 35, 42, 47 };

int n = arr.length;

int x = 18;

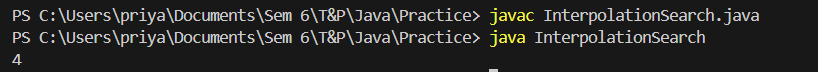
int index = interpolationSearch(arr, 0, n - 1, x);

System.out.println(index);

}

}

**Output:**

****

**Time Complexity:** O(log2(log2 n))

**Space Complexity:** O (1)