DSA Coding Practice Problems

09-11-2024

1) Maximum Subarray Sum – Kadane‟s Algorithm:

Given an array arr[], the task is to find the subarray that has the maximum sum and return its sum.

Input: arr[] = {2, 3, -8, 7, -1, 2, 3}

Output: 11

Explanation: The subarray {7, -1, 2, 3} has the largest sum 11.

Input: arr[] = {-2, -4}

Output: –2

Explanation: The subarray {-2} has the largest sum -2.

Program :

```java
import java.util.Scanner;
import java.util.ArrayList;

public class Main {
    public static int maxSubArraySum(ArrayList<Integer> arr) {
        int maxCurrent = arr.get(0);
```

```java
        int maxGlobal = arr.get(0);

        for (int i = 1; i < arr.size(); i++) {
            maxCurrent = Math.max(arr.get(i), maxCurrent + arr.get(i));
            maxGlobal = Math.max(maxGlobal, maxCurrent);
        }

        return maxGlobal;
    }

    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();

        ArrayList<Integer> arr = new ArrayList<>(n);
        System.out.println("Enter the elements of the array: ");
        for (int i = 0; i < n; i++) {
            arr.add(scanner.nextInt());
        }

        int maxSum = maxSubArraySum(arr);
        System.out.println("Maximum Subarray Sum: " + maxSum);

        scanner.close();
    }
}
```

Output :

```
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$ javac SubArraySum.java
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$ java SubArraySum
Enter the number of elements in the array: 7
Enter the elements of the array:
2
3
-8
7
-1
2
3
Maximum Subarray Sum: 11
```

The time complexity is O(n) and the space complexity is O(1).

2) Maximum Product Subarray

Given an integer array, the task is to find the maximum product of any subarray.

Input: arr[] = {-2, 6, -3, -10, 0, 2}

Output: 180

Explanation: The subarray with maximum product is {6, -3, -10} with product = 6 * (-3) * (-10) = 180

Input: arr[] = {-1, -3, -10, 0, 60}

Output: 60

Explanation: The subarray with maximum product is {60}.

Program :

```java
import java.util.Scanner;
import java.util.ArrayList;

public class MaxProductSubarray {
    public static int MaxProductSubarray(ArrayList<Integer> arr) {
        int maxP = arr.get(0), minP = arr.get(0), res = arr.get(0);

        for (int i = 1; i < arr.size(); i++) {
            int current = arr.get(i);

            if (current < 0) {
                int temp = maxP;
                maxP = minP;
                minP = temp;
            }

            maxP = Math.max(current, maxP * current);
            minP = Math.min(current, minP * current);

            res = Math.max(res, maxP);
        }

        return res;
    }
}
```

```java
    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();

        ArrayList<Integer> arr = new ArrayList<>(n);
        System.out.println("Enter the array elements: ");
        for (int i = 0; i < n; i++) {
            arr.add(scanner.nextInt());
        }

        int maxProduct = MaxProductSubarray(arr);
        System.out.println("Maximum Product Subarray: " + maxProduct);

        scanner.close();
    }
}
```

Output :

```
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$ javac MaxProductSubarray.java
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$ java MaxProductSubarray
Enter the number of elements in the array: 6
Enter the array elements:
-2 6 -3 -10 0 2
Maximum Product Subarray: 180
abishek@abishek HP 250 G8 Notebook PC:~/Downloads/SDE Hws/Problems$
```

The time complexity is O(n) and the space O(1).

3) Search in a sorted and rotated Array

Given a sorted and rotated array arr[] of n distinct elements, the task is to find the index of given

key in the array. If the key is not present in the array, return -1.

Input : arr[] = {4, 5, 6, 7, 0, 1, 2}, key = 0

Output : 4

Input : arr[] = { 4, 5, 6, 7, 0, 1, 2 }, key = 3

Output : -1

Input : arr[] = {50, 10, 20, 30, 40}, key = 10

Output : 1

Program :

import java.util.Scanner;

public class SearchInRotatedArray {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();

```java
        int[] arr = new int[n];
        System.out.println("Enter the elements for the array: ");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.print("Enter any target element to find: ");
        int target = sc.nextInt();

        int result = searchInRotatedArray(arr, target);
        if (result != -1) {
            System.out.println("The Target element " + target + " is found at index " + result);
        } else {
            System.out.println("-1\nThe Target element " + target + " is not present in the given array...");
        }

        sc.close();
    }

    public static int searchInRotatedArray(int[] arr, int target) {
        int left = 0, right = arr.length - 1;

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

            if (arr[mid] == target) {
```

```
            return mid;
        }

        if (arr[left] <= arr[mid]) {

            if (arr[left] <= target && target < arr[mid]) {
                right = mid - 1;
            } else {
                left = mid + 1;
            }
        }

        else {

            if (arr[mid] < target && target <= arr[right]) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
    }

    return -1;
  }
}
```

Output :

```
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
java SearchInRotatedArray
Enter the size of the array: 7
Enter the elements for the array:
4 5 6 7 0 1 2
Enter any target element to find: 0
The Target element 0 is found at index 4
```

The time complexity is O(log n) and the space complexity is O(1).

4)Container with most water

Given an array of non-negative integers `arr` where each element arr[i] represents the height of a vertical line at coordinate (i, arr[i]):

Objective: Find two lines that, together with the x-axis, form a container that can hold the maximum amount of water.

Output: Return an integer representing the maximum area of water that can be contained.

Rules and Assumptions:

1. You cannot tilt the container (it remains vertical).

2. The container's area is calculated by the distance between the two lines (right - left) multiplied by the height of the shorter line (min(arr[left], arr[right])).

Examples:

Example 1:

Input: arr = [1, 5, 4, 3]

Output: 6

Explanation:

Choose the lines at indices 1 and 3.

Distance between the lines = 3 - 1 = 2.

Height of the container = min(5, 3) = 3.

Area = 3 * 2 = 6.

Example 2:

Input: arr = [3, 1, 2, 4, 5]`

Output: 12

Explanation:

Choose the lines at indices 0 and 4.

Distance between the lines = 4 - 0 = 4.

Height of the container = min(3, 5) = 3.

Area = 3 * 4 = 12.

Program :

```java
import java.util.ArrayList;
import java.util.Scanner;

public class MaxArea {
    public static int MaxArea(ArrayList<Integer> arr) {
```

```java
        int left = 0, right = arr.size() - 1;
        int maxArea = 0;

        while (left < right) {

            int height = Math.min(arr.get(left), arr.get(right));
            int width = right - left;
            int area = height * width;


            maxArea = Math.max(maxArea, area);


            if (arr.get(left) < arr.get(right)) {
                left++;
            } else {
                right--;
            }
        }

    return maxArea;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter the number of elements in the height array: ");
    int n = scanner.nextInt();
```

```java
        ArrayList<Integer> arr = new ArrayList<>(n);
        System.out.println("Enter the height elements: ");
        for (int i = 0; i < n; i++) {
            arr.add(scanner.nextInt());
        }

        int maxWater = MaxArea(arr);
        System.out.println("Maximum Area of Water that can be contained: " +
maxWater);

        scanner.close();
    }
}
```

Output :

The time complexity is O(n) and the space complexity is O(1).

5)Find the Factorial of a large number

Input: 100

Output:

93326215443944152681699238856266700490715968264381
62146859296389521759999322999

15608941463976156518286253697920827223758251185210
9168640000000000000000000000000

Input: 50

Output:
30414093201713378043612608166064768844377641568960
512000000000000

Program :

import java.math.BigInteger;
import java.util.Scanner;

public class FactorialCalculator {

   public static BigInteger factorial(int n) {
     BigInteger result = BigInteger.ONE;
     for (int i = 1; i <= n; i++) {

```java
            result = result.multiply(BigInteger.valueOf(i));
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number to calculate its factorial: ");
        int number = scanner.nextInt();

        BigInteger factorialResult = factorial(number);
        System.out.println("Factorial of " + number + " is:");
        System.out.println(factorialResult);

        scanner.close();
    }
}
```

Output :

```
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
javac FactorialCalculator.java
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
java FactorialCalculator
Enter a number to calculate its factorial: 100
Factorial of 100 is:
93326215443944152681699238856266700490715968264381621468592963895217
59999322991560894146397615651828625369792082722375825118521091686400
0000000000000000000000000
```

The time complexity is O(n) and the space complexity is O(1).

6)Trapping Rainwater Problem

Given an array of n non-negative integers arr[] representing an elevation map where the width of each bar is 1, compute how much water it can trap after rain.

Input: arr[] = {3, 0, 1, 0, 4, 0, 2}

Output: 10

Explanation: The expected rainwater to be trapped is shown in the above image.

Input: arr[] = {3, 0, 2, 0, 4}

Output: 7

Explanation: We trap 0 + 3 + 1 + 3 + 0 = 7 units.

Input: arr[] = {1, 2, 3, 4}

Output: 0

Explanation : We cannot trap water as there is no height bound on both sides

Input: arr[] = {10, 9, 0, 5}

Output: 5

Explanation : We trap 0 + 0 + 5 + 0 = 5

Program :

```java
public class TrappingRainwater {

    public static int trapWater(int[] arr) {
        int n = arr.length;
        if (n <= 2) return 0;

        int[] leftMax = new int[n];
        int[] rightMax = new int[n];

        leftMax[0] = arr[0];
        for (int i = 1; i < n; i++) {
            leftMax[i] = Math.max(leftMax[i - 1], arr[i]);
        }

        rightMax[n - 1] = arr[n - 1];
        for (int i = n - 2; i >= 0; i--) {
            rightMax[i] = Math.max(rightMax[i + 1], arr[i]);
        }

        int trappedWater = 0;
        for (int i = 0; i < n; i++) {
            trappedWater += Math.min(leftMax[i], rightMax[i]) - arr[i];
        }

        return trappedWater;
```

```java
    }

    public static void main(String[] args) {
        int[] arr1 = {3, 0, 1, 0, 4, 0, 2};
        int[] arr2 = {3, 0, 2, 0, 4};
        int[] arr3 = {1, 2, 3, 4};
        int[] arr4 = {10, 9, 0, 5};

        System.out.println("Trapped water for arr1: " + trapWater(arr1));
        System.out.println("Trapped water for arr2: " + trapWater(arr2));
        System.out.println("Trapped water for arr3: " + trapWater(arr3));
        System.out.println("Trapped water for arr4: " + trapWater(arr4));
    }
}
```

Output :

```
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
javac TrappingRainwater.java
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
java TrappingRainwater
Trapped water for arr1: 10
Trapped water for arr2: 7
Trapped water for arr3: 0
Trapped water for arr4: 5
```

The time complexity is O(n) and the space complexity is O(n).

7) Chocolate Distribution Problem

Given an array arr[] of n integers where arr[i] represents the number of chocolates in ith packet.

Each packet can have a variable number of chocolates. There are m students, the task is to distribute chocolate packets such that:

Each student gets exactly one packet.

The difference between the maximum and minimum number of chocolates in the packets given to the students is minimized.

Input: arr[] = {7, 3, 2, 4, 9, 12, 56}, m = 3

Output: 2

Explanation: If we distribute chocolate packets {3, 2, 4}, we will get the minimum difference, that is 2.

Input: arr[] = {7, 3, 2, 4, 9, 12, 56}, m = 5

Output: 7

Explanation: If we distribute chocolate packets {3, 2, 4, 9, 7}, we will get the minimum difference, that is 9 – 2 = 7.

Program :

```java
import java.util.Arrays;
import java.util.Scanner;

public class ChocolateDistribution {

    public static int findMinDifference(int[] packets, int n, int m) {

        if (m == 0 || n == 0) return 0;

        Arrays.sort(packets);

        if (n < m) return -1;

        int minDiff = Integer.MAX_VALUE;

        for (int i = 0; i + m - 1 < n; i++) {
            int diff = packets[i + m - 1] - packets[i];
            minDiff = Math.min(minDiff, diff);
        }

        return minDiff;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of packets: ");
        int n = sc.nextInt();
```

```java
        int[] packets = new int[n];

        System.out.println("Enter the number of chocolates in each packet:");
        for (int i = 0; i < n; i++) {
            packets[i] = sc.nextInt();
        }

        System.out.print("Enter the number of students: ");
        int m = sc.nextInt();

        int result = findMinDifference(packets, n, m);
        if (result == -1) {
            System.out.println("The number of students cannot be more than the
number of packets.");
        } else {
            System.out.println("The minimum difference is " + result);
        }

        sc.close();
    }
}
```

Output :

```
^Cabishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems
java ChocolateDistribution
Enter the number of packets: 7
Enter the number of chocolates in each packet:
7 3 2 9 4 12 56
Enter the number of students: 3
The minimum difference is 2
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
```

The time complexity is O(n log n) and the space complexity is O(1)

8)Merge Overlapping Intervals

Given an array of time intervals where arr[i] = [starti, endi], the task is to merge all the overlapping intervals into one and output the result which should have only mutually exclusive

intervals.

Input: arr[] = [[1, 3], [2, 4], [6, 8], [9, 10]]

Output: [[1, 4], [6, 8], [9, 10]]

Explanation: In the given intervals, we have only two overlapping intervals [1, 3] and [2, 4].

Therefore, we will merge these two and return [[1, 4}], [6, 8], [9, 10]].Input: arr[] = [[7, 8], [1, 5], [2, 4], [4, 6]]

Output: [[1, 6], [7, 8]]

Explanation: We will merge the overlapping intervals [[1, 5], [2, 4], [4, 6]] into a single interval

[1, 6].

Program :

```java
import java.util.*;

public class MergeIntervals {

    public static List<int[]> mergeIntervals(int[][] intervals) {
        if (intervals == null || intervals.length == 0) {
            return new ArrayList<>();
        }

        Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));

        List<int[]> merged = new ArrayList<>();
        merged.add(intervals[0]);

        for (int i = 1; i < intervals.length; i++) {
            int[] current = intervals[i];
            int[] lastMerged = merged.get(merged.size() - 1);

            if (current[0] <= lastMerged[1]) {
                lastMerged[1] = Math.max(lastMerged[1], current[1]);
            } else {
                merged.add(current);
            }
        }

        return merged;
    }
```

```java
    public static void main(String[] args) {
        int[][] intervals1 = {{1, 3}, {2, 4}, {6, 8}, {9, 10}};
        int[][] intervals2 = {{7, 8}, {1, 5}, {2, 4}, {4, 6}};

        System.out.println("Merged Intervals 1: " +
formatIntervals(mergeIntervals(intervals1)));
        System.out.println("Merged Intervals 2: " +
formatIntervals(mergeIntervals(intervals2)));
    }


    private static String formatIntervals(List<int[]> intervals) {
        StringBuilder sb = new StringBuilder();
        sb.append("[");
        for (int[] interval : intervals) {
            sb.append("[").append(interval[0]).append(",
").append(interval[1]).append("], ");
        }
        if (sb.length() > 1) {
            sb.setLength(sb.length() - 2);  // Remove the last comma and space
        }
        sb.append("]");
        return sb.toString();
    }
}
```

Output :

```
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
javac MergeIntervals.java
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
java MergeIntervals
Merged Intervals 1: [[1, 4], [6, 8], [9, 10]]
Merged Intervals 2: [[1, 6], [7, 8]]
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
```

The time complexity is O(n log n) and the space complexity is O(n).

9)A Boolean Matrix Question

Given a boolean matrix mat[M][N] of size M X N, modify it such that if a matrix cell mat[i][j] is 1 (or true) then make all the cells of ith row and jth column as 1.

Input: {{1, 0},{0, 0}}

Output: {{1, 1}{1, 0}}

Input: {{0, 0, 0},{0, 0, 1}}

Output: {{0, 0, 1},{1, 1, 1}}

Input: {{1, 0, 0, 1},{0, 0, 1, 0},{0, 0, 0, 0}}

Output: {{1, 1, 1, 1},{1, 1, 1, 1},{1, 0, 1, 1}}

Program :

```java
public class BooleanMatrix {

    public static void modifyMatrix(int[][] mat) {
        int m = mat.length;
        int n = mat[0].length;
        boolean firstRow = false, firstCol = false;
        for (int j = 0; j < n; j++) {
            if (mat[0][j] == 1) {
                firstRow = true;
                break;
            }
```

```
    }
    for (int i = 0; i < m; i++) {
        if (mat[i][0] == 1) {
            firstCol = true;
            break;
        }
    }
    for (int i = 1; i < m; i++) {
        for (int j = 1; j < n; j++) {
            if (mat[i][j] == 1) {
                mat[i][0] = 1;  // Mark the row
                mat[0][j] = 1;  // Mark the column
            }
        }
    }
    for (int i = 1; i < m; i++) {
        for (int j = 1; j < n; j++) {
            if (mat[i][0] == 1 || mat[0][j] == 1) {
                mat[i][j] = 1;
            }
        }
    }
    if (firstRow) {
        for (int j = 0; j < n; j++) {
            mat[0][j] = 1;
        }
    }
    if (firstCol) {
        for (int i = 0; i < m; i++) {
```

```java
            mat[i][0] = 1;
        }
    }
}

public static void printMatrix(int[][] mat) {
    for (int i = 0; i < mat.length; i++) {
        for (int j = 0; j < mat[i].length; j++) {
            System.out.print(mat[i][j] + " ");
        }
        System.out.println();
    }
}

public static void main(String[] args) {
    int[][] mat1 = {{1, 0}, {0, 0}};
    int[][] mat2 = {{0, 0, 0}, {0, 0, 1}};
    int[][] mat3 = {{1, 0, 0, 1}, {0, 0, 1, 0}, {0, 0, 0, 0}};

    System.out.println("Modified Matrix 1:");
    modifyMatrix(mat1);
    printMatrix(mat1);

    System.out.println("\nModified Matrix 2:");
    modifyMatrix(mat2);
    printMatrix(mat2);

    System.out.println("\nModified Matrix 3:");
    modifyMatrix(mat3);
```

```
        printMatrix(mat3);

    }

}
```

Output :

```
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
javac BooleanMatrix.java
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
java BooleanMatrix
Modified Matrix 1:
1 1
1 0

Modified Matrix 2:
0 0 1
1 1 1

Modified Matrix 3:
1 1 1 1
1 1 1 1
1 0 1 1
```

The time complexity is O(m*n) and the space complexity is O(1)

10. Print a given matrix in spiral form

Given an m x n matrix, the task is to print all elements of the matrix in spiral form.

Input: matrix = {{1, 2, 3, 4},{5, 6, 7, 8},{9, 10, 11, 12},{13, 14, 15, 16 }}

Output: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

Input: matrix = { {1, 2, 3, 4, 5, 6},{7, 8, 9, 10, 11, 12},{13, 14, 15, 16, 17, 18}}

Output: 1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11

Explanation: The output is matrix in spiral format.

Program :

```
public class SpiralMatrix {
    public static void printSpiral(int[][] matrix) {
        if (matrix == null || matrix.length == 0 || matrix[0].length == 0) {
            return;
        }

        int top = 0, bottom = matrix.length - 1;
        int left = 0, right = matrix[0].length - 1;

        while (top <= bottom && left <= right) {
            for (int i = left; i <= right; i++) {
```

```java
            System.out.print(matrix[top][i] + " ");
        }
        top++;

        for (int i = top; i <= bottom; i++) {
            System.out.print(matrix[i][right] + " ");
        }
        right--;

        if (top <= bottom) {
            for (int i = right; i >= left; i--) {
                System.out.print(matrix[bottom][i] + " ");
            }
            bottom--;
        }

        if (left <= right) {
            for (int i = bottom; i >= top; i--) {
                System.out.print(matrix[i][left] + " ");
            }
            left++;
        }
    }
}

public static void main(String[] args) {
    int[][] matrix1 = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
```

```java
        {9, 10, 11, 12},
        {13, 14, 15, 16}
    };

    int[][] matrix2 = {
        {1, 2, 3, 4, 5, 6},
        {7, 8, 9, 10, 11, 12},
        {13, 14, 15, 16, 17, 18}
    };

    System.out.println("Spiral Order of Matrix 1:");
    printSpiral(matrix1);

    System.out.println("\nSpiral Order of Matrix 2:");
    printSpiral(matrix2);
  }
}
```

Output :

```
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
javac SpiralMatrix.java
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
java SpiralMatrix
Spiral Order of Matrix 1:
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
Spiral Order of Matrix 2:
1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11 abishek@abishek-HP-250-
```

The time complexity is O(m*n) and the space complexity is O(1)

13. Check if given Parentheses expression is balanced or not

Given a string str of length N, consisting of „(„ and „)„ only, the task is to check whether it is balanced or not.

Input: str = "((()))()()"

Output: Balanced

Input: str = "())((())"

Output: Not Balanced

Program :
```
import java.util.Stack;

public class ParenthesisBalance {

  public static String checkBalance(String str) {
    Stack<Character> stack = new Stack<>();

    for (int i = 0; i < str.length(); i++) {
      char ch = str.charAt(i);

      if (ch == '(') {
        stack.push(ch);
      }
```

```java
            else if (ch == ')') {
                if (stack.isEmpty()) {
                    return "Not Balanced";
                }
                stack.pop();
            }
        }

        return stack.isEmpty() ? "Balanced" : "Not Balanced";
    }

    public static void main(String[] args) {
        String str1 = "((()))()()";
        String str2 = "())((())";

        System.out.println("For str1: " + checkBalance(str1));
        System.out.println("For str2: " + checkBalance(str2));
    }
}
```

Output :

```
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
javac ParenthesisBalance.java
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
java ParenthesisBalance
For str1: Balanced
For str2: Not Balanced
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
```

The space complexity is O(n) and the time complexity is O(n).

14) Check if two Strings are Anagrams of each other

Given two strings s1 and s2 consisting of lowercase characters, the task is to check whether the two given strings are anagrams of each other or not. An anagram of a string is another string that contains the same characters, only the order of characters can be different.

Input: s1 = "geeks" s2 = "kseeg"

Output: true

Explanation: Both the string have same characters with same frequency. So, they are anagrams.

Input: s1 = "allergy" s2 = "allergic"

Output: false

Explanation: Characters in both the strings are not same. s1 has extra character „y" and s2 has

extra characters „i" and „c", so they are not anagrams.

Input: s1 = "g", s2 = "g"

Output: true

Explanation: Characters in both the strings are same, so they are anagrams.

Program :

```java
import java.util.Arrays;

public class AnagramChecker {

    public static boolean areAnagrams(String s1, String s2) {

        if (s1.length() != s2.length()) {
            return false;
        }

        char[] arr1 = s1.toCharArray();
        char[] arr2 = s2.toCharArray();


        Arrays.sort(arr1);
        Arrays.sort(arr2);

        return Arrays.equals(arr1, arr2);
    }

    public static void main(String[] args) {

        String s1 = "geeks";
        String s2 = "kseeg";
```
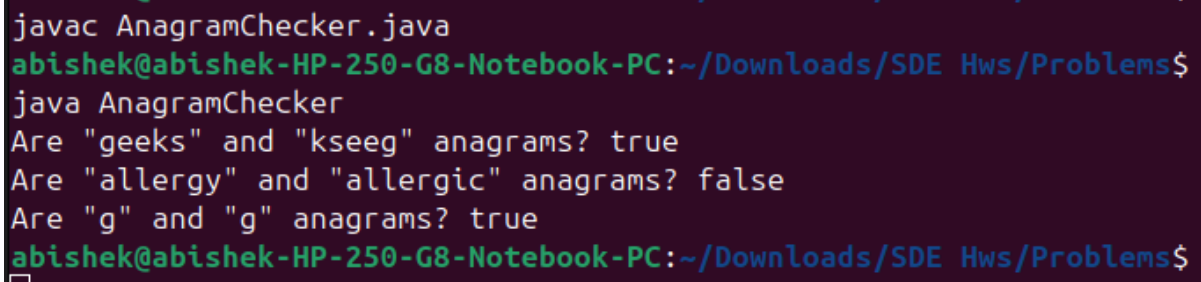
```java
        System.out.println("Are \"" + s1 + "\" and \"" + s2 + "\" anagrams? "
+ areAnagrams(s1, s2));


        String s1_2 = "allergy";
        String s2_2 = "allergic";
        System.out.println("Are \"" + s1_2 + "\" and \"" + s2_2 + "\"
anagrams? " + areAnagrams(s1_2, s2_2));


        String s1_3 = "g";
        String s2_3 = "g";
        System.out.println("Are \"" + s1_3 + "\" and \"" + s2_3 + "\"
anagrams? " + areAnagrams(s1_3, s2_3));
    }
}
```

Output :

```
javac AnagramChecker.java
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
java AnagramChecker
Are "geeks" and "kseeg" anagrams? true
Are "allergy" and "allergic" anagrams? false
Are "g" and "g" anagrams? true
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
```

The time complexity is O(n log n) and space complexity is O(n).

15. Longest Palindromic Substring

Given a string str, the task is to find the longest substring which is a palindrome. If there are multiple answers, then return the first appearing substring.

Input: str = "forgeeksskeegfor"

Output: "geeksskeeg"

Explanation: There are several possible palindromic substrings like "kssk", "ss", "eeksskee" etc.

But the substring "geeksskeeg" is the longest among all.

Input: str = "Geeks"

Output: "ee"

Input: str = "abc"

Output: "a"

Input: str = ""

Program :

```
public class LongestPalindromicSubstring {

    public static String longestPalindrome(String str) {
```

```java
        if (str == null || str.length() == 0) {
            return "";
        }

        int start = 0, end = 0;

        for (int i = 0; i < str.length(); i++) {
            int len1 = expandAroundCenter(str, i, i);
            int len2 = expandAroundCenter(str, i, i + 1);

            int len = Math.max(len1, len2);

            if (len > end - start) {
                start = i - (len - 1) / 2;
                end = i + len / 2;
            }
        }

        return str.substring(start, end + 1);
    }

    private static int expandAroundCenter(String str, int left, int right) {
        while (left >= 0 && right < str.length() && str.charAt(left) ==
str.charAt(right)) {
            left--;
            right++;
        }
        return right - left - 1;
    }
```
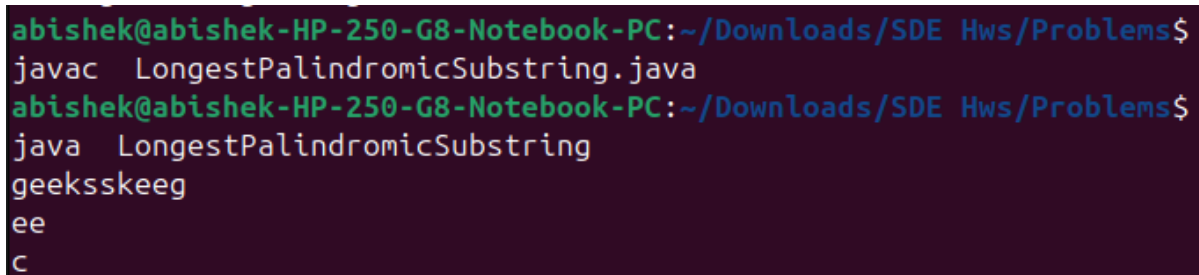
```java
    public static void main(String[] args) {

        System.out.println(longestPalindrome("forgeeksskeegfor"));

        System.out.println(longestPalindrome("Geeks"));

        System.out.println(longestPalindrome("abc"));

        System.out.println(longestPalindrome(""));}
}
```

Output :

```
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
javac  LongestPalindromicSubstring.java
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
java  LongestPalindromicSubstring
geeksskeeg
ee
c
```

The time complexity is O(n^2) and space complexity is O(1).

16. Longest Common Prefix using Sorting

Given an array of strings arr[]. The task is to return the longest common prefix among each and every strings present in the array. If there"s no prefix common in all the strings, return "-1".

Input: arr[] = ["geeksforgeeks", "geeks", "geek", "geezer"]

Output: gee

Explanation: "gee" is the longest common prefix in all the given strings.Input: arr[] = ["hello", "world"]

Output: -1

Explanation: There"s no common prefix in the given strings.

Program :
```
import java.util.Arrays;
public class LongestCommonPrefix {
    public static String longestCommonPrefix(String[] arr) {
        if (arr.length == 0) {
            return "-1";
        }
        Arrays.sort(arr);
        String first = arr[0];
        String last = arr[arr.length - 1];
        int minLength = Math.min(first.length(), last.length());
        int i = 0;
```

```java
        while (i < minLength && first.charAt(i) == last.charAt(i)) {
            i++;
        }
        if (i == 0) {
            return "-1";
        }
        return first.substring(0, i);
    }
    public static void main(String[] args) {
        String[] arr1 = {"geeksforgeeks", "geeks", "geek", "geezer"};
        System.out.println("Longest common prefix: " +
longestCommonPrefix(arr1));
        String[] arr2 = {"hello", "world"};
        System.out.println("Longest common prefix: " +
longestCommonPrefix(arr2));
    }
}
```

Output :

```
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
javac LongestCommonPrefix.java
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
java LongestCommonPrefix
Longest common prefix: gee
Longest common prefix: -1
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
```

The time complexity is O(nlogn+m) and space complexity is O(1)

17. Delete middle element of a stack

Given a stack with push(), pop(), and empty() operations, The task is to delete the middle element of it without using any additional data structure.

Input : Stack[] = [1, 2, 3, 4, 5]

Output : Stack[] = [1, 2, 4, 5]

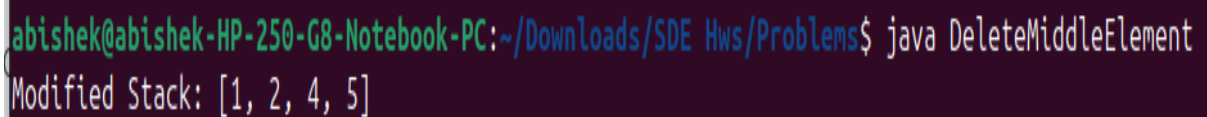Input : Stack[] = [1, 2, 3, 4, 5, 6]

Program :

```java
import java.util.Stack;
public class DeleteMiddleElement {
    public static void deleteMiddleElement(Stack<Integer> stack, int count, int size) {
        if (count == size / 2) {
            stack.pop();
            return;
        }
        int temp = stack.pop();
        deleteMiddleElement(stack, count + 1, size);
        stack.push(temp);
    }
    public static void main(String[] args) {
        Stack<Integer> stack = new Stack<>();
```

```
        stack.push(1);

        stack.push(2);

        stack.push(3);

        stack.push(4);

        stack.push(5);

        int size = stack.size();

        deleteMiddleElement(stack, 0, size);

        System.out.println("Modified Stack: " + stack);

    }

}
```

Output :

The time complexity is O(n) and the space complexity is O(n).

18. Next Greater Element (NGE) for every element in given Array

Given an array, print the Next Greater Element (NGE) for every element.

Note: The Next greater Element for an element x is the first greater element on the right side of x in the array. Elements for which no greater element exist, consider the next greater element as -1.

Input: arr[] = [4 , 5 , 2 , 25]

Output: 4 –> 5

5 –> 25

2 –> 25

25 –> -1

Explanation: Except 25 every element has an element greater than them present on the right side

Input: arr[] = [13 , 7, 6 , 12]

Output: 13 –> -1

7–> 12

6–> 12

12–> -1

Explanation: 13 and 12 don"t have any element greater than them present on the right side

Program :

```java
import java.util.Stack;

public class NextGreaterElement {
    public static void nextGreaterElement(int[] arr) {
        int n = arr.length;
        int[] result = new int[n];
        Stack<Integer> stack = new Stack<>();

        for (int i = n - 1; i >= 0; i--) {
            while (!stack.isEmpty() && stack.peek() <= arr[i]) {
                stack.pop();
            }

            if (!stack.isEmpty()) {
                result[i] = stack.peek();
            } else {
                result[i] = -1;
            }
```

```java
            stack.push(arr[i]);
        }
        for (int i = 0; i < n; i++) {
            System.out.println(arr[i] + " --> " + result[i]);
        }
    }

    public static void main(String[] args) {
        int[] arr1 = {4, 5, 2, 25};
        System.out.println("Next Greater Element for arr1:");
        nextGreaterElement(arr1);

        int[] arr2 = {13, 7, 6, 12};
        System.out.println("Next Greater Element for arr2:");
        nextGreaterElement(arr2);
    }
}
```

Output :

```
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$ javac NextGreaterElement.java
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$ java NextGreaterElement
Next Greater Element for arr1:
4 --> 5
5 --> 25
2 --> 25
25 --> -1
Next Greater Element for arr2:
13 --> -1
7 --> 12
6 --> 12
12 --> -1
```
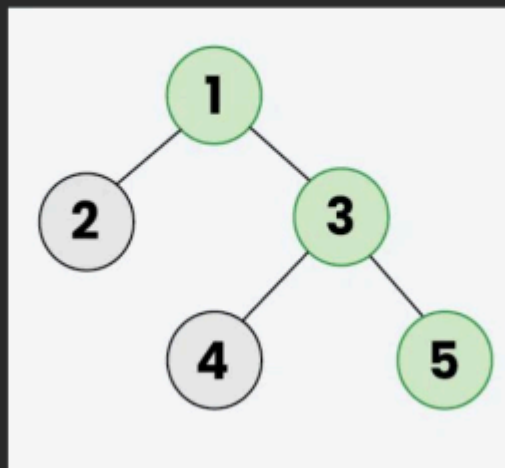
**The time complexity is O(n) and the space complexity is O(n).**

19)Print Right View of a Binary Tree

Given a Binary Tree, the task is to print the Right view of it. The right view of a Binary Tree is a set of rightmost nodes for every level.

Example 1: The **Green** colored nodes (1, 3, 5) represents the Right view in the below Binary tree.

Example 2: The **Green** colored nodes (1, 3, 4, 5) represents the Right view in the below Binary tree.
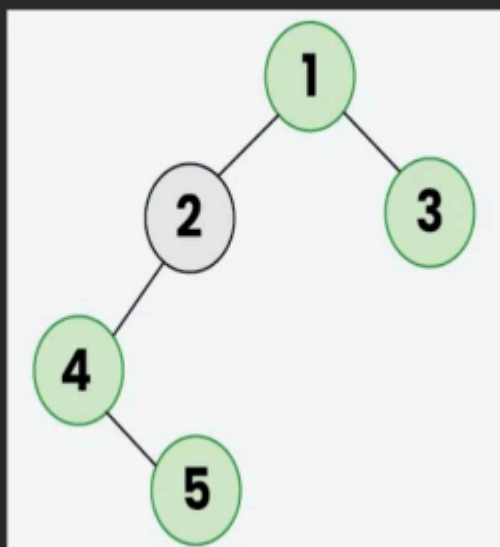
Program :

```java
import java.util.*;
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
    TreeNode() {}
    TreeNode(int val) { this.val = val; }
    TreeNode(int val, TreeNode left, TreeNode right) {
        this.val = val;
        this.left = left;
        this.right = right;
    }
}
public class BinaryTreeRightSideView {
    public List<Integer> rightSideView(TreeNode root) {
        List<Integer> rightView = new ArrayList<>();

        if (root == null) {
            return rightView;
        }

        Queue<TreeNode> queue = new LinkedList<>();
        queue.offer(root);

        while (!queue.isEmpty()) {
            int levelSize = queue.size();

            for (int i = 0; i < levelSize; i++) {
```

```java
            TreeNode currentNode = queue.poll();

            if (i == levelSize - 1) {
                rightView.add(currentNode.val);
            }

            if (currentNode.left != null) {
                queue.offer(currentNode.left);
            }

            if (currentNode.right != null) {
                queue.offer(currentNode.right);
            }
        }
    }

    return rightView;
}

public static void main(String[] args) {
    BinaryTreeRightSideView treeView = new BinaryTreeRightSideView();

    TreeNode root = new TreeNode(1);
    root.left = new TreeNode(2);
    root.right = new TreeNode(3);
    root.left.right = new TreeNode(5);
    root.right.right = new TreeNode(4);

    System.out.println(treeView.rightSideView(root)); // Output: [1, 3, 4]
```
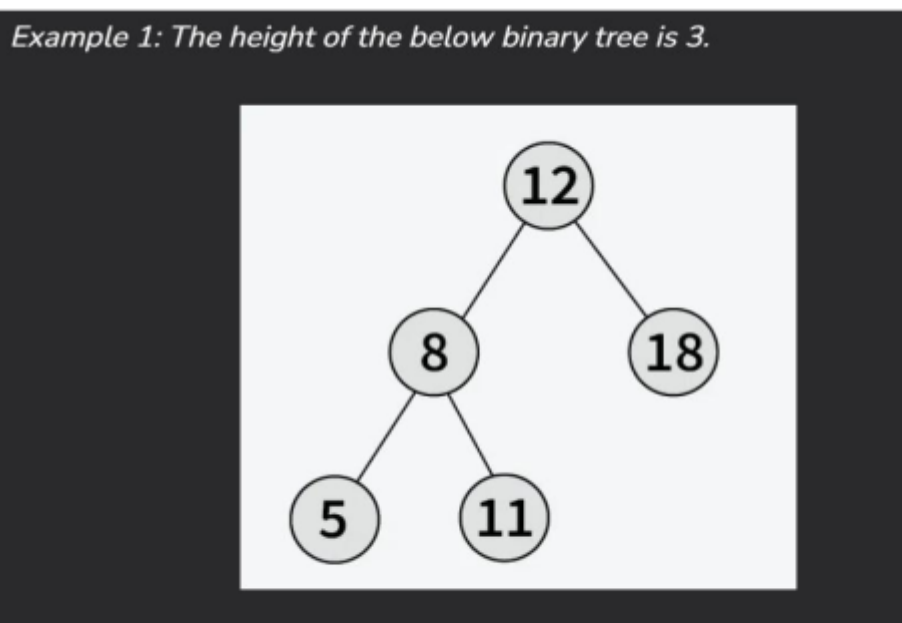
```
        }
}
```

Output :

The time complexity is O(n) and space complexity is O(n).

20. Maximum Depth or Height of Binary Tree

Given a binary tree, the task is to find the maximum depth or height of the tree. The height of the tree is the number of vertices in the tree from the root to the deepest node.

Example 1: The height of the below binary tree is 3.

Example 2: The height of the below binary tree is 4
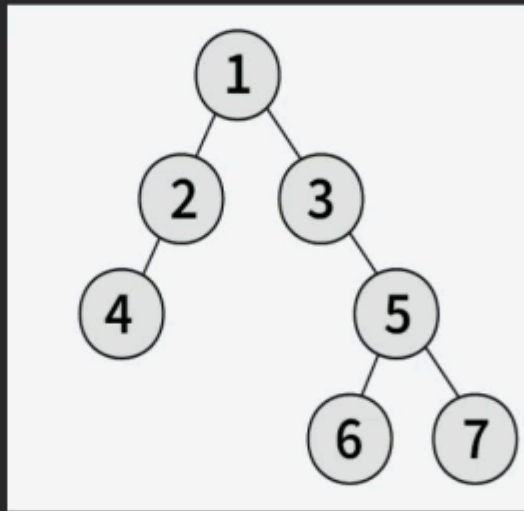
Program :

```
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
    TreeNode(int val) { this.val = val; }
    TreeNode(int val, TreeNode left, TreeNode right) {
        this.val = val;
        this.left = left;
        this.right = right;
    }
}
public class BinaryTreeMaxDepth {
    public int maxDepth(TreeNode root) {
        if (root == null) {
            return 0;
        }
```

```java
        int leftDepth = maxDepth(root.left);
        int rightDepth = maxDepth(root.right);
        return Math.max(leftDepth, rightDepth) + 1;
    }
    public static void main(String[] args) {
        BinaryTreeMaxDepth treeDepth = new BinaryTreeMaxDepth();

        TreeNode root = new TreeNode(1);
        root.left = new TreeNode(2);
        root.right = new TreeNode(3);
        root.left.left = new TreeNode(4);
        root.left.right = new TreeNode(5);

        System.out.println("Maximum Depth of the Tree: " +
treeDepth.maxDepth(root));
    }
}
```

Output :

```
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$ javac BinaryTreeMaxDepth.java
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$ java BinaryTreeMaxDepth
Maximum Depth of the Tree: 3
abishek@abishek-HP-250-G8-Notebook-PC:~/Downloads/SDE Hws/Problems$
```

The time complexity is O(n) and the space complexity O(log n).