

## DSA PRACTICE PROBLEMS – SET 1

Madhulekha R – AIML - [09/11/2024]

### 1. **Question:** 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

#### **Code:**

```
import java.util.*;
public class Main{
    public static void main(String[] args){
        int[] arr={2, 3, -8, 7, -1, 2, 3};
        int[] arr1={-2, -4};
        int ans = maxisum(arr);
        System.out.println(ans);
        int ans1 = maxisum(arr1);
        System.out.println(ans1);
    }

    static int maxisum(int[] arr){
        int cursum=arr[0];
        int maxsum=arr[0];

        for (int i=1; i<arr.length; i++){
            cursum=Math.max(cursum,0);
            cursum+=arr[i];
            maxsum=Math.max(maxsum, cursum);
        }
        return maxsum;
    }
}
```

**Output:**

```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
11
-2
```

**Time Complexity:**  $O(n)$ **2. Question:** 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  $\{60\}$ .

**Code:**

```
import java.util.*;

public class Main{
    public static void main(String[] args){
        int[] arr ={-2, 6, -3, -10, 0, 2};
        System.out.println(maxipro(arr));
        int[] arr1 ={-1, -3, -10, 0, 60};
        System.out.println(maxipro(arr1));
    }

    static int maxipro(int[] arr){
        int maxpro=arr[0];
        int minpro=arr[0];
        int res=arr[0];

        for (int i=1; i<arr.length; i++){
            if (arr[i]<0){
                int temp = maxpro;
                maxpro=minpro;
                minpro=temp;
            }
            maxpro=Math.max(maxpro*arr[i], minpro*arr[i]);
            minpro=Math.min(maxpro*arr[i], minpro*arr[i]);
            res=Math.max(res, Math.max(maxpro, minpro));
        }
        return res;
    }
}
```

```

    }

    maxpro=Math.max(arr[i],maxpro*arr[i]);
    minpro=Math.min(arr[i],minpro*arr[i]);
    res=Math.max(res,maxpro);
}
return res;
}
}

```

**Output:**

```

Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
180
60

```

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

### 3. **Question:** 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

**Code:**

```

import java.util.*;

public class Main{
    public static void main(String[] args){
        int[] arr = {4,5,6,7,0,1,2};
        int target = 0;
        System.out.println(rotatesearch(arr, target));
        int[] arr1 = { 4, 5, 6, 7, 0, 1, 2};
        int target1 = 3;
    }
}

```

```

        System.out.println(rotatesearch(arr1, target1));
        int[] arr2 = {50, 10, 20, 30, 40};
        int target2 = 10;
        System.out.println(rotatesearch(arr2, target2));
    }

    static int rotatesearch(int[] arr, int target){
        int start=0;
        int end=arr.length-1;

        while (start<=end){
            int mid = start+(end-start)/2;

            if (target==arr[mid]){
                return mid;
            }

            if (arr[start]<=arr[mid]){
                if (target>=arr[start] && target<arr[mid]){
                    end=mid-1;
                }
                else{
                    start=mid+1;
                }
            }
            else{
                if (target>=arr[mid] && target<arr[end]){
                    start=mid+1;
                }
                else{
                    end=mid-1;
                }
            }
        }
        return -1;
    }
}

```

**Output:**

```
C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
4
-1
1
```

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

**4. Question:** Container with Most Water

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

Output: 6

Explanation: 5 and 3 are distance 2 apart. So the size of the base = 2.

Height of container =  $\min(5, 3) = 3$ . So total area =  $3 * 2 = 6$

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

Output: 12

Explanation: 5 and 3 are distance 4 apart. So the size of the base = 4. Height of container =  $\min(5, 3) = 3$ . So total area =  $4 * 3 = 12$

**Code:**

```
class Main{
    public static void main(String[] args){
        int[] height={ 1, 5, 4, 3};
        System.out.println(maxArea(height));
        int[] height1={3, 1, 2, 4, 5};
        System.out.println(maxArea(height1));
    }

    static int maxArea(int[] height) {
        int l=0;
        int r = height.length-1;
        int res=0;

        while (l<r){
            int maxarea = (r-l)*Math.min(height[l],height[r]);
            res = Math.max(res,maxarea);
        }
    }
}
```

```

        if (height[l]<height[r]){
            l+=1;
        }
        else{
            r-=1;
        }
    }
    return res;
}
}

```

**Output:**

```

C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
6
12

```

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

**5. Question:** Find the Factorial of a large number

Input: 100

Output: 332621544394415268169923885626670049071596826438162146859296389521759999  
322991560894146397615651828625369792082722375825118521091686400000000000000000  
0000000

Input: 50

Output: 30414093201713378043612608166064768844377641568960512000000000000

**Code:**

```

import java.util.*;
import java.math.BigInteger;

class Main{
    public static void main(String[] args){
        int n = 50;
        System.out.println(bigfactorial(n));
    }
}

```

```
static BigInteger bigfactorial(int n) {
    BigInteger result = BigInteger.ONE;

    for (int i = 2; i <= n; i++) {
        result = result.multiply(BigInteger.valueOf(i));
    }

    return result;
}
```

**Output:**

[illegible]

### Time Complexity: $O(n)$

- 6. Question:** Trapping Rainwater Problem states that 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  $0 + 0 + 5 + 0 = 5$

**Code:**

```
import java.util.*;
```

```
class Main{
    public static void main(String[] args){
        int[] arr = {3, 0, 1, 0, 4, 0, 2};
```

```

        System.out.println(trap(arr));
        int[] arr1 = {3, 0, 2, 0, 4};
        System.out.println(trap(arr1));
        int[] arr2 = {1, 2, 3, 4};
        System.out.println(trap(arr2));
    }

    static int trap(int[] height) {
        if (height == null || height.length == 0) return 0;

        int left = 0, right = height.length - 1;
        int leftmax = 0, rightmax = 0;
        int watertrapped = 0;

        while (left <= right) {
            if (height[left] <= height[right]) {
                if (height[left] >= leftmax) {
                    leftmax = height[left];
                } else {
                    watertrapped += leftmax - height[left];
                }
                left++;
            } else {
                if (height[right] >= rightmax) {
                    rightmax = height[right];
                } else {
                    watertrapped += rightmax - height[right];
                }
                right--;
            }
        }
        return watertrapped;
    }
}

```

**Output:**

```

C:\Users\HP\Documents>javac main.java

C:\Users\HP\Documents>java Main
10
7
0

```



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

**7. Question:** Chocolate Distribution Problem

Given an array `arr[]` of  $n$  integers where `arr[i]` represents the number of chocolates in  $i$ th 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  $9 - 2 = 7$

**Code:**

```
import java.util.Arrays;

class Main{
    public static void main(String[] args) {
        int[] arr1 = {7, 3, 2, 4, 9, 12, 56};
        int m1 = 3;
        System.out.println(findMinDifference(arr1, m1));
        int[] arr2 = {7, 3, 2, 4, 9, 12, 56};
        int m2 = 5;
        System.out.println(findMinDifference(arr2, m2));
    }

    static int findMinDifference(int[] arr, int m) {
        int n = arr.length;

        if (n < m) {
            return -1;
        }

        Arrays.sort(arr);

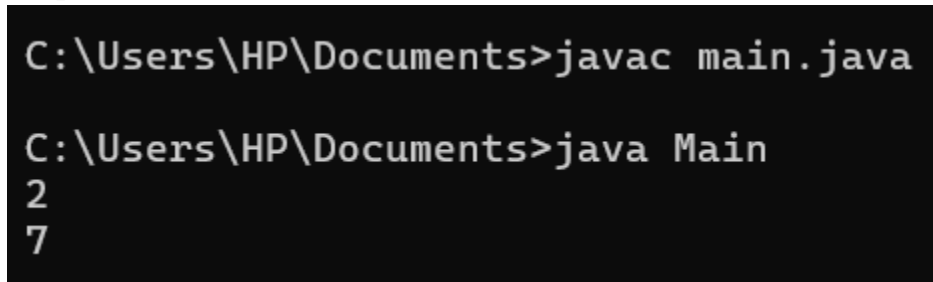
        int minDiff = Integer.MAX_VALUE;
        for (int i = 0; i <= n - m; i++) {
```

```

        int diff = arr[i + m - 1] - arr[i];
        minDiff = Math.min(minDiff, diff);
    }
    return minDiff;
}
}

```

**Output:**



```

C:\Users\HP\Documents>javac main.java

C:\Users\HP\Documents>java Main
2
7

```

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

**8. Question:** Merge Overlapping Intervals

Given an array of time intervals where  $arr[i] = [start_i, end_i]$ , 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  $[1, 6]$ .

**Code:**

```

import java.util.*;

class Solution {
    public static void main(String[] args) {
        int[][] intervals1 = {{1, 3}, {2, 4}, {6, 8}, {9, 10}};
        int[][] result1 = mergeIntervals(intervals1);
        System.out.println(Arrays.deepToString(result1));
        int[][] intervals2 = {{7, 8}, {1, 5}, {2, 4}, {4, 6}};
        int[][] result2 = mergeIntervals(intervals2);
        System.out.println(Arrays.deepToString(result2));
    }
}

```

```

    }

    static int[][] mergeIntervals(int[][] intervals) {
        if (intervals.length == 0) {
            return new int[0][0];
        }

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

        List<int[]> merged = new ArrayList<>();

        for (int[] interval : intervals) {
            if (merged.isEmpty() || merged.get(merged.size() - 1)[1] < interval[0]) {
                merged.add(interval);
            } else {
                merged.get(merged.size()-1)[1]=Math.max(merged.get(merged.size()-1)[1],interval[1]);
            }
        }
        return merged.toArray(new int[merged.size()][]);
    }
}

```

**Output:**

```

C:\Users\HP\Documents>javac main.java

C:\Users\HP\Documents>java Main
[1, 4], [6, 8], [9, 10]
[1, 6], [7, 8]

```

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

## 9. Question: A Boolean Matrix Question

Given a boolean matrix `mat[M][N]` of size  $M \times N$ , modify it such that if a matrix cell `mat[i][j]` is 1 (or true) then make all the cells of *i*th row and *j*th 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}}

**Code:**

```
import java.util.*;

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

    static void modifyMatrix(int[][] mat) {
        int M = mat.length;
        int N = mat[0].length;

        boolean[] rowFlag = new boolean[M];
        boolean[] colFlag = new boolean[N];

        for (int i = 0; i < M; i++) {
            for (int j = 0; j < N; j++) {
                if (mat[i][j] == 1) {
                    rowFlag[i] = true;
                    colFlag[j] = true;
                }
            }
        }

        for (int i = 0; i < M; i++) {
            for (int j = 0; j < N; j++) {
                if (rowFlag[i] || colFlag[j]) {
                    mat[i][j] = 1;
                }
            }
        }
    }

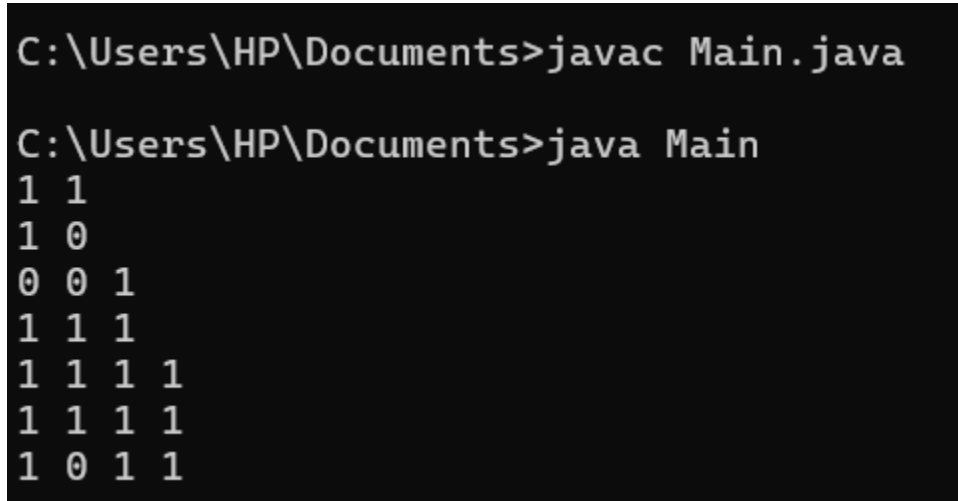
    static void printMatrix(int[][] mat) {
        for (int i = 0; i < mat.length; i++) {
```

```

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

```

**Output:**



```

C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
1 1
1 0
0 0 1
1 1 1
1 1 1 1
1 1 1 1
1 0 1 1

```

**Time Complexity:**  $O(m*n)$

#### 10. Question: Print a given matrix in spiral form

Given an  $m \times n$  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.

**Code:**

```

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

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

```

```

while (top <= bottom && left <= right) {
    for (int i = left; i <= right; i++) {
        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},
        {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("\n\nSpiral order of matrix 2:");

```

```

        printSpiral(matrix2);
    }
}

```

**Output:**

```

C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
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

```

**Time Complexity:**  $O(m*n)$

**11. Question:** 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

**Code:**

```

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

        System.out.println(isBalanced(str1));
        System.out.println(isBalanced(str2));
    }

    public static String isBalanced(String str) {
        int count = 0;
        for (int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);
            if (ch == '(') {
                count++;
            }
        }
    }
}

```

```

    } else if (ch == ')') {
        count--;
        if (count < 0) {
            return "Not Balanced";
        }
    }
}

return (count == 0) ? "Balanced" : "Not Balanced";
}
}

```

**Output:**

```

C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
Balanced
Not Balanced

```

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

## 12. Question: 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

**Code:**

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

```
public class Main{
```



```

public static void main(String[] args) {
    String s1 = "geeks";
    String s2 = "kseeg";
    System.out.println(areAnagrams(s1, s2));
    String s3 = "allergy";
    String s4 = "allergic";
    System.out.println(areAnagrams(s3, s4));
    String s5 = "g";
    String s6 = "g";
    System.out.println(areAnagrams(s5, s6));
}

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);
}
}

```

**Output:**

```

C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
Balanced
Not Balanced

C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
true
false
true

```

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

### 13. Question: 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", "eeksske" etc. But the substring "geeksskeeg" is the longest among all.

Input: str = "Geeks"

Output: "ee"

Input: str = "abc"

Output: "a"

Input: str = ""

Output: ""

#### Code:

```
public class Main{
    public static void main(String[] args) {
        String str1 = "forgeeksskeegfor";
        String str2 = "Geeks";
        String str3 = "abc";
        String str4 = "";

        System.out.println(longestPalindrome(str1)); // Output: geeksskeeg
        System.out.println(longestPalindrome(str2)); // Output: ee
        System.out.println(longestPalindrome(str3)); // Output: a
        System.out.println(longestPalindrome(str4)); // Output: ""
    }

    public static String longestPalindrome(String str) {
        if (str == null || str.length() == 0) {
            return "";
        }

        int start = 0, maxLength = 1;

        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 > maxLength) {
                maxLength = len;
            }
        }

        return str.substring(start, start + maxLength);
    }

    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;
    }
}
```

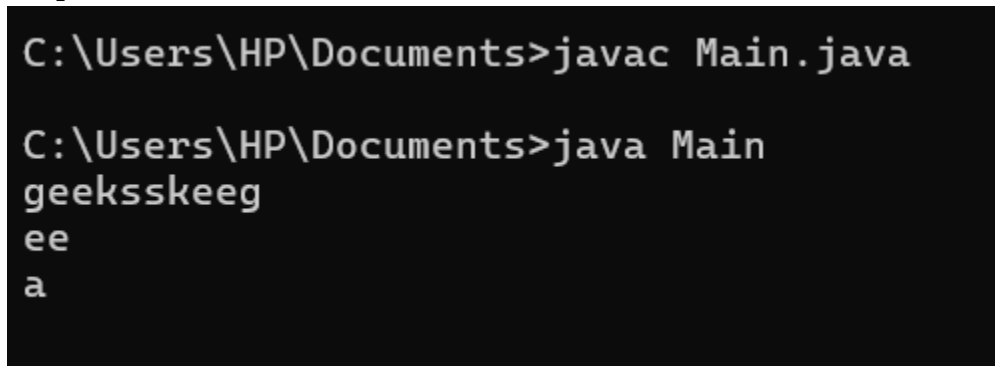
```

        start = i - (len - 1) / 2;
    }
}
return str.substring(start, start + maxLength);
}

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;
}
}

```

**Output:**



```

C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
geeksskeeg
ee
a

```

**Time Complexity:**  $O(n^2)$

**14. Question:** 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.

**Code:**

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

```
public class Main{
```

```

public static void main(String[] args) {
    String[] arr1 = {"geeksforgeeks", "geeks", "geek", "geezer"};
    String[] arr2 = {"hello", "world"};
    System.out.println(longestCommonPrefix(arr1));
    System.out.println(longestCommonPrefix(arr2));
}

public static String longestCommonPrefix(String[] arr) {
    // If the array is empty, return "-1"
    if (arr == null || arr.length == 0) {
        return "-1";
    }
    Arrays.sort(arr);

    String first = arr[0];
    String last = arr[arr.length - 1];
    int i = 0;
    while (i < first.length() && i < last.length() && first.charAt(i) == last.charAt(i)) {
        i++;
    }
    if (i == 0) {
        return "-1";
    }
    return first.substring(0, i);
}
}

```

**Output:**

```

C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
gee
-1

```

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

#### 15. Question: 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]

Output : Stack[] = [1, 2, 4, 5, 6]

**Code:**

```
import java.util.Stack;
```

```
public class Main{
    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();
        deleteMiddle(stack, 0, size);
        System.out.println("Stack after removing the middle element: " + stack);

        stack.clear();
        stack.push(1);
        stack.push(2);
        stack.push(3);
        stack.push(4);
        stack.push(5);
        stack.push(6);

        size = stack.size();
        deleteMiddle(stack, 0, size);
        System.out.println("Stack after removing the middle element: " + stack);
    }

    static void deleteMiddle(Stack<Integer> stack, int currentIndex, int size) {
        if (stack.isEmpty()) {
            return;
        }

        int top = stack.pop();

        if (currentIndex == size / 2) {
            return;
        }
    }
}
```

```

        deleteMiddle(stack, currentIndex + 1, size);
        stack.push(top);
    }
}

```

**Output:**

```

C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
Stack after removing the middle element: [1, 2, 4, 5]
Stack after removing the middle element: [1, 2, 4, 5, 6]

```

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

**16. Question:** 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

**Code:**

```

import java.util.Stack;

public class Main{
    public static void main(String[] args) {
        int[] arr1 = {4, 5, 2, 25};
        int[] arr2 = {13, 7, 6, 12};
        printNGE(arr1);
        System.out.println();
        printNGE(arr2);
    }

    public static void printNGE(int[] arr) {
        Stack<Integer> stack = new Stack<>();
        for (int i = 0; i < arr.length; i++) {
            while (!stack.isEmpty() && stack.peek() <= arr[i]) {

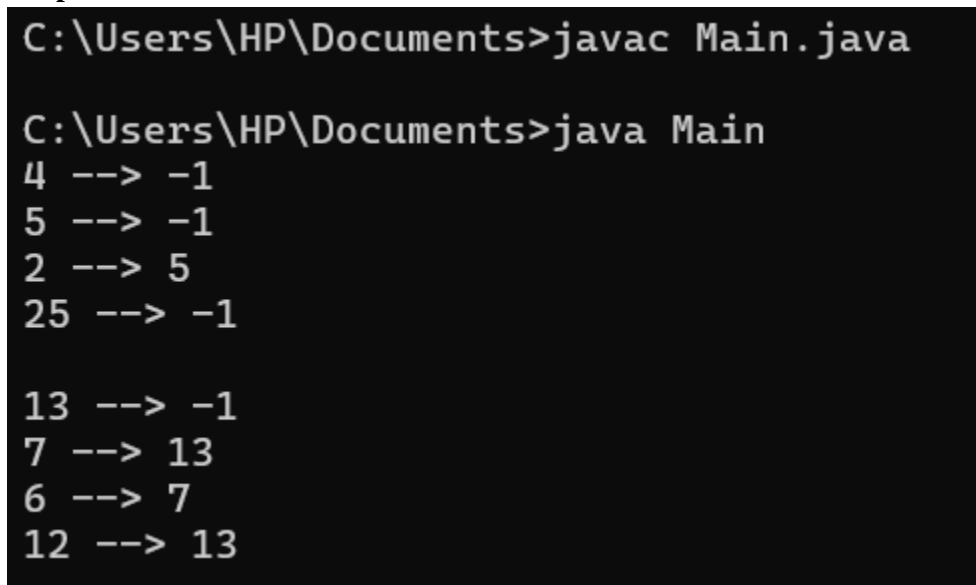
```

```

        stack.pop();
    }
    if (!stack.isEmpty()) {
        System.out.println(arr[i] + " --> " + stack.peek());
    } else {
        System.out.println(arr[i] + " --> -1");
    }
    stack.push(arr[i]);
}
}
}

```

**Output:**



```

C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
4 --> -1
5 --> -1
2 --> 5
25 --> -1

13 --> -1
7 --> 13
6 --> 7
12 --> 13

```

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

#### 17. Question: 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

**Code:**

```

import java.util.*;

public class Main{
    static class TreeNode {
        int data;
        TreeNode left, right;
    }
}

```

```

    public TreeNode(int item) {
        data = item;
        left = right = null;
    }
}

public static void main(String[] args) {
    Main tree = new Main();
    tree.root = new TreeNode(1);
    tree.root.left = new TreeNode(2);
    tree.root.right = new TreeNode(3);
    tree.root.left.left = new TreeNode(4);
    tree.root.left.right = new TreeNode(5);
    tree.root.right.right = new TreeNode(6);
    tree.root.left.left.left = new TreeNode(7);

    System.out.println("Right view of the binary tree:");
    tree.rightView();
}

```

TreeNode root;

```

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

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

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

        for (int i = 0; i < levelSize; i++) {
            currentNode = queue.poll();
            if (currentNode.left != null) {
                queue.add(currentNode.left);
            }
            if (currentNode.right != null) {
                queue.add(currentNode.right);
            }
        }
    }
}

```



```

        System.out.print(currentNode.data + " ");
    }
}
}

```

**Output:**

```

C:\Users\HP\Documents>javac Main.java

C:\Users\HP\Documents>java Main
Right view of the binary tree:
1 3 6 7

```

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

#### 18. Question: 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.

**Code:**

```

public class Main{
    static class TreeNode {
        int data;
        TreeNode left, right;

        public TreeNode(int item) {
            data = item;
            left = right = null;
        }
    }

    public static void main(String[] args) {
        Main tree = new Main();
        tree.root = new TreeNode(12);
        tree.root.left = new TreeNode(8);
        tree.root.right = new TreeNode(18);
        tree.root.left.left = new TreeNode(5);
        tree.root.left.right = new TreeNode(11);

        System.out.println("Height of the binary tree: " + tree.maxDepth(tree.root));
    }
}

```

```
TreeNode root;  
  
public int maxDepth(TreeNode node) {  
    if (node == null) {  
        return 0;  
    }  
  
    int leftHeight = maxDepth(node.left);  
    int rightHeight = maxDepth(node.right);  
    return Math.max(leftHeight, rightHeight) + 1;  
}  
}
```

**Output:**

```
C:\Users\HP\Documents>javac Main.java  
  
C:\Users\HP\Documents>java Main  
Height of the binary tree: 3
```

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