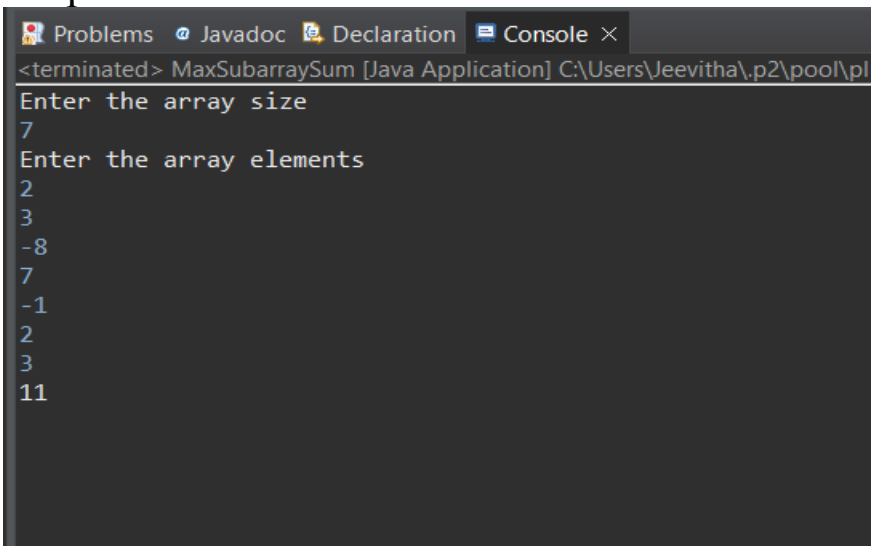
DSA Practice Questions

Name: Jeevitha R
Reg No: 22IT040

1.Maximum Subarray Sum – Kadane‟s Algorithm

```
package practiceset;
import java.util.*;
public class MaxSubarraySum {

    public static void main(String[] args) {
            Scanner sc= new Scanner(System.in);
            System.out.println("Enter the array size");
            int n= sc.nextInt();
            int[] nums= new int[n];
            System.out.println("Enter the array elements");
            for(int i=0;i<n;i++) {
                    nums[i]= sc.nextInt();
            }
            int max=Integer.MIN_VALUE;
      int sum=0;
      for(int num:nums){
         sum=Math.max(sum,0)+num;
         max=Math.max(sum,max);
      }
      System.out.println(max);
    }
}
```

Output

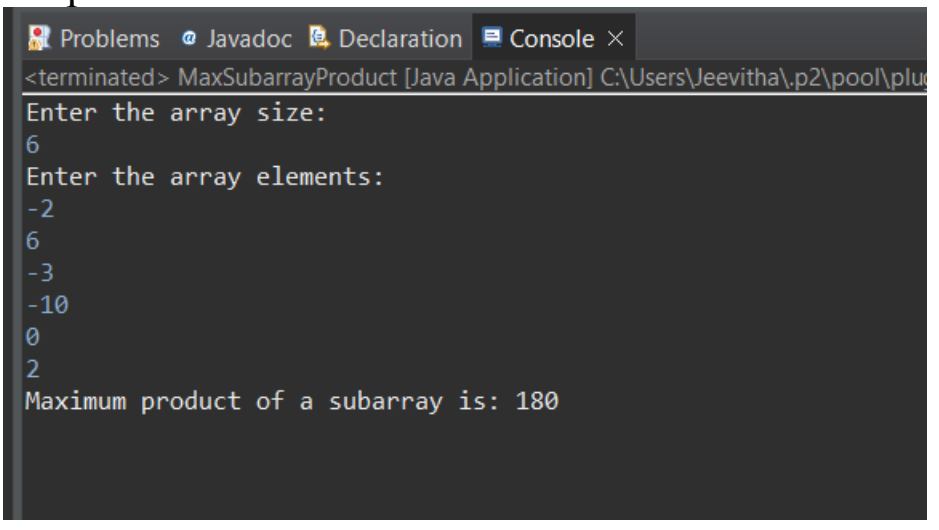


Time complexity: O(n)
Space complexity: O(1)

2. Maximum Product Subarray

```
package practiceset;
import java.util.*;
public class MaxSubarrayProduct {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the array size:");
        int n = sc.nextInt();
        int[] nums = new int[n];
        System.out.println("Enter the array elements:");
        for (int i=0;i<n;i++) {
            nums[i] = sc.nextInt();
        }
        int max = nums[0];
        int min = nums[0];
        int res = nums[0];
        for (int i=1; i<nums.length; i++) {
            if (nums[i]<0){
                int temp = max;
                max = min;
                min = temp;
            }
            max = Math.max(nums[i],max*nums[i]);
            min = Math.min(nums[i],min*nums[i]);
            res = Math.max(res, max);
        }
        System.out.println("Maximum product of a subarray is: " + res);
    }
}
```

Output



Time complexity: O(n)
Space complexity: O(1)

3. Search in a sorted and rotated Array

```
package practiceset;
```

```
import java.util.*;
public class RotatedArray {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the size of the array:");
        int n=sc.nextInt();
        int[] arr=new int[n];
        System.out.println("Enter the elements of the rotated sorted array:");
        for (int i=0; i<n; i++) {
            arr[i]=sc.nextInt();
        }
        System.out.println("Enter the target element to search:");
        int key=sc.nextInt();
        int index=search(arr, key);
        if (index!=-1) {
            System.out.println("Element found at index: "+index);
        } else {
            System.out.println("Element not found in the array.");
        }
    }

    public static int search(int[] arr, int key) {
        int left=0;
        int right=arr.length-1;
        while (left<=right) {
            int mid=left+(right-left)/2;
            if (arr[mid]==key) return mid;
            if (arr[left]<=arr[mid]) {
                if (arr[left]<=key && key<=arr[mid]) {
                    right=mid-1;
                } else {
                    left=mid+1;
                }
            } else {
                if (arr[mid]<=key && key<=arr[right]) {
                    left=mid+1;
                } else {
                    right=mid-1;
                }
            }
        }
        return -1;
    }
}
```

Output

```
Problems  Javadoc  Declaration  Console ×
<terminated> RotatedArray [Java Application] C:\Users\Jeevitha\.p2\pool\plugins\
Enter the size of the array:
7
Enter the elements of the rotated sorted array:
4
5
6
7
0
1
2
Enter the target element to search:
0
Element found at index: 4
```

Time complexity: O(log n)
Space complexity: O(1)

4.Container with Most Water

```
package practiceset;
import java.util.Scanner;
public class Container {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the number of lines:");
        int n=sc.nextInt();
        int[] h=new int[n];
        System.out.println("Enter the heights of each line:");
        for (int i=0; i<n; i++) {
            h[i]=sc.nextInt();
        }
        int l=0, r=h.length-1, max=0;
        while (l<r) {
            int area=Math.min(h[l],h[r])*(r-l);
            max=Math.max(max,area);
            if (h[l]<h[r]) {
                l++;
            } else {
                r--;
            }
        }
        System.out.println("The maximum area that can be contained is: "+max);
    }
}
```

```
Problems  Javadoc  Declaration  Console ×
<terminated> Container [Java Application] C:\Users\Jeevitha\.p2\pool\plugins\org.
Enter the number of lines:
4
Enter the heights of each line:
1
5
4
3
The maximum area that can be contained is: 6
```

Time complexity: O(n)
Space complexity:O(1)

5.Find the Factorial of a large number

```
package practiceset;
import java.util.*;
import java.math.BigInteger;
public class Factorial {
	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter a large number:");
		int n=sc.nextInt();
		BigInteger fact=BigInteger.ONE;
		for (int i=1; i<=n; i++) {
		   fact=fact.multiply(BigInteger.valueOf(i));
		}
		System.out.println("Factorial of "+n+" is:\n"+fact);
	}
}
```

Output

```
Problems  Javadoc  Declaration  Console ×
<terminated> Factorial [Java Application] C:\Users\Jeevitha\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.1.v20241024-1700\jre\bin\javaw.exe  (10 Nov 2024, 1:08:30 am – 1:08:43 am) [pid: 1144]
Enter a large number:
100
Factorial of 100 is:
93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000
```

Time complexity: O(n)
Space complexity: O(nlogn)

6. Trapping Rain Water

```
package practiceset;
import java.util.*;
public class TrappingWater {

        public static void main(String[] args) {
                Scanner sc=new Scanner(System.in);
            System.out.print("Enter the number of elements:");
            int n=sc.nextInt();
            int[] h=new int[n];
            System.out.println("Enter the elements of the array:");
            for(int i=0;i<n;i++){
                h[i]=sc.nextInt();
            }
            int l=0,lMax=h[0],sum=0;
            int r=h.length-1,rMax=h[r];
            while(l<r){
                if(lMax<=rMax){
                    sum+=(lMax-h[l]);
                    l++;
                    lMax=Math.max(lMax,h[l]);
                }else{
                    sum+=(rMax-h[r]);
                    r--;
                    rMax=Math.max(rMax,h[r]);
                }
            }
            System.out.println("Total trapped water:"+sum);
        }

}
```

Output

```
Problems  Javadoc  Declaration  Console ×
<terminated> TrappingWater [Java Application] C:\Users\Jeevitha\.p2\pool\pl
Enter the number of elements:7
Enter the elements of the array:
3 0 1 0 4 0 2
Total trapped water:10
```

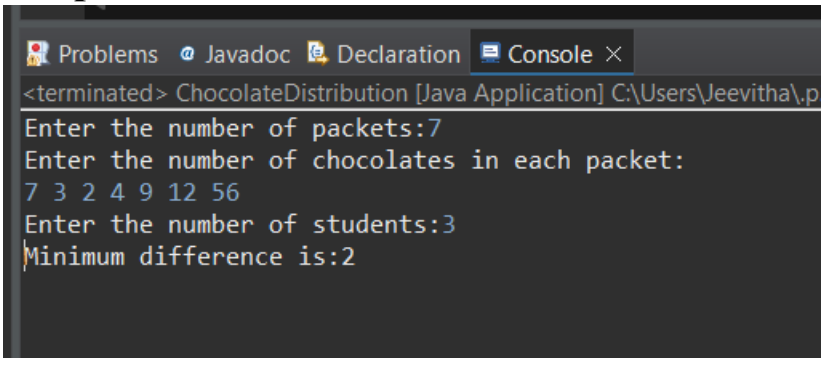
Time complexity: O(n)
Space complexity: O(1)

7. Chocolate Distribution Problem

```
package practiceset;
import java.util.*;
public class ChocolateDistribution {
```

```
        public static void main(String[] args) {
            Scanner sc=new Scanner(System.in);
            System.out.print("Enter the number of packets:");
            int n=sc.nextInt();
            ArrayList<Integer> a=new ArrayList<>();
            System.out.println("Enter the number of chocolates in each packet:");
            for(int i=0;i<n;i++){
               a.add(sc.nextInt());
            }
            System.out.print("Enter the number of students:");
            int m=sc.nextInt();
            if(m>n){
                    System.out.println("Distribution not possible, more students than packets.");
            return;
            }
            Collections.sort(a);
            int diff=Integer.MAX_VALUE;
            for(int i=0;i+m-1<n;i++){
            int min=a.get(i);
            int max=a.get(i+m-1);
            int currDiff=max-min;
            if(currDiff<diff){
            diff=currDiff;
                    }
            }
            System.out.println("Minimum difference is:"+diff);
        }
}
```

Output



Time complexity: O(nlogn)
Space complexity: O(n)

8. Merge Overlapping Intervals

```
package practiceset;
import java.util.*;
public class MergeOverlapIntervals {
    public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter the number of intervals: ");
    int n=sc.nextInt();
    int[][] in=new int[n][2];
```

```
        System.out.println("Enter the intervals:");
        for(int i=0;i<n;i++){
            in[i][0]=sc.nextInt();
            in[i][1]=sc.nextInt();
        }
        Arrays.sort(in,Comparator.comparingInt(a->a[0]));
        Stack<int[]> stack=new Stack<>();
        stack.push(new int[]{in[0][0],in[0][1]});
        for(int i=1;i<in.length;i++){
            int[] top=stack.peek();
            if(in[i][0]<=top[1]){
                int[] mergedInterval=new int[]{top[0],Math.max(in[i][1],top[1])};
                stack.pop();
                stack.push(mergedInterval);
            }else{
                stack.push(new int[]{in[i][0],in[i][1]});
            }
        }
        List<int[]> mergedIntervals=new ArrayList<>(stack);
        System.out.println("Merged intervals:");
        for(int[] interval:mergedIntervals){
            System.out.println("["+interval[0]+", "+interval[1]+"]");
        }
            }
}
```

Output:

```
Problems  Javadoc  Declaration  Console ×
<terminated> MergeOverlapIntervals [Java Application] C:\Users
Enter the number of intervals: 4
Enter the intervals:
1 3
2 4
6 8
9 10
Merged intervals:
[1, 4]
[6, 8]
[9, 10]
```

Time complexity: O(nlogn)
Space complexity: O(n)

9.A Boolean Matrix Question

```
package practiceset;
import java.util.*;
public class BooleanMatrix {
        public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the number of rows: ");
        int M=sc.nextInt();
        System.out.print("Enter the number of columns: ");
        int N=sc.nextInt();
        int[][] mat=new int[M][N];
```

```
        System.out.println("Enter the matrix values (0 or 1):");
        for(int i=0;i<M;i++){
            for(int j=0;j<N;j++){
                mat[i][j]=sc.nextInt();
            }
        }
        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;
                }
            }
        }
        System.out.println("Modified matrix:");
        for(int i=0;i<M;i++){
            for(int j=0;j<N;j++){
                System.out.print(mat[i][j]+" ");
            }
            System.out.println();
                    }
          }
}
```

Output

```
Problems  Javadoc  Declaration  Console ×
<terminated> BooleanMatrix [Java Application] C:\Users\Jeevitha\.p2\pool\plugins\org
Enter the number of rows: 2
Enter the number of columns: 2
Enter the matrix values (0 or 1):
1 0
0 1
Modified matrix:
1 1
1 1
```

Time complexity: O(M X N)
Space complexity: O(M+N)

10.Print a given matrix in spiral form

```
package practiceset;
import java.util.*;
```

```
public class SpiralMatrix {
    public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter the number of rows: ");
    int m=sc.nextInt();
    System.out.print("Enter the number of columns: ");
    int n=sc.nextInt();
    int[][] matrix=new int[m][n];
    System.out.println("Enter the matrix values:");
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            matrix[i][j]=sc.nextInt();
        }
    }
    List<Integer> result=new ArrayList<>();
    int row=0,col=-1,direction=1;
    while(m>0&&n>0){
        for(int i=0;i<n;i++){
            col+=direction;
            result.add(matrix[row][col]);
        }
        m--;
        for(int i=0;i<m;i++){
            row+=direction;
            result.add(matrix[row][col]);
        }
        n--;
        direction*=-1;
    }
    System.out.println("Spiral order of the matrix:");
    for(int num:result){
        System.out.print(num+" ");
    }
        }
}
```

Output:

```
Problems  Javadoc  Declaration  Console ×
<terminated> SpiralMatrix [Java Application] C:\Users\Jeevitha\.p2\pool\pl
Enter the number of rows: 4
Enter the number of columns: 4
Enter the matrix values:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Spiral order of the matrix:
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
```

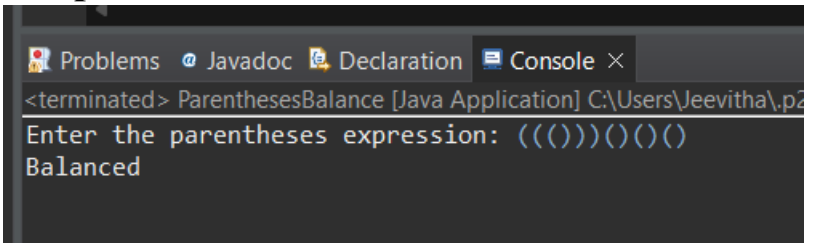
Time complexity: O(mxn)
Space complexity: O(mxn)

13.Check if given Parentheses expression is balanced or not

package practiceset;

```
import java.util.*;
public class ParenthesesBalance {
	public static void main(String[] args) {
	Scanner sc=new Scanner(System.in);
	System.out.print("Enter the parentheses expression: ");
	String str=sc.nextLine();
	Stack<Character> stk=new Stack<Character>();
	boolean isBalanced=true;
	for(char ch:str.toCharArray()){
	   if(ch=='(')stk.push(')');
	   else if(stk.isEmpty()||stk.pop()!=ch){
	      isBalanced=false;
	      break;
	   }
	}
	if(stk.isEmpty()&&isBalanced)System.out.println("Balanced");
	else System.out.println("Not Balanced");
		}
}
```

Output:



Time complexity: O(N)
Space complexity: O(N)

14.Check if two Strings are Anagrams of each other

```
package practiceset;
import java.util.*;
public class AnagramCheck {
	public static void main(String[] args) {
	Scanner sc=new Scanner(System.in);
	System.out.print("Enter the first string: ");
	String s1=sc.nextLine();
	System.out.print("Enter the second string: ");
	String s2=sc.nextLine();
	if(s1.length()!=s2.length())System.out.println("false");
	else{
	   char[] sChars=s1.toCharArray();
	   char[] tChars=s2.toCharArray();
	   Arrays.sort(sChars);
	   Arrays.sort(tChars);
	   if(Arrays.equals(sChars,tChars))System.out.println("true");
	   else System.out.println("false");
	}
		}
```

}

Output:

```
Problems  Javadoc  Declaration  Console ×
<terminated> AnagramCheck [Java Application] C:\Users\Jeevitha\
Enter the first string: geeks
Enter the second string: kseeg
true
```

Time complexity: O(N\log N).
Space complexity: O(N)

15.Longest Palindromic Substring

```
package practiceset;
import java.util.*;
public class LongestPalindromicSubstring {
	public static void main(String[] args) {
	Scanner sc=new Scanner(System.in);
	System.out.print("Enter the string: ");
	String s=sc.nextLine();
	int n=s.length();
	if(n==0){System.out.println("");return;}
	boolean[][] dp=new boolean[n][n];
	String res="";
	for(int len=1;len<=n;len++){
	   for(int i=0;i<=n-len;i++){
	      int j=i+len-1;
	      if(s.charAt(i)==s.charAt(j)){
	         if(len==1||len==2){dp[i][j]=true;}
	         else{dp[i][j]=dp[i+1][j-1];}
	      }
	      if(dp[i][j]&&len>res.length()){res=s.substring(i,j+1);}
	   }
	}
	System.out.println(res);
	    }
}
```

Output:

```
Problems  Javadoc  Declaration  Console ×
<terminated> LongestPalindromicSubstring [Java Application] C
Enter the string: “forgeeksskeegfor”
geeksskeeg
```

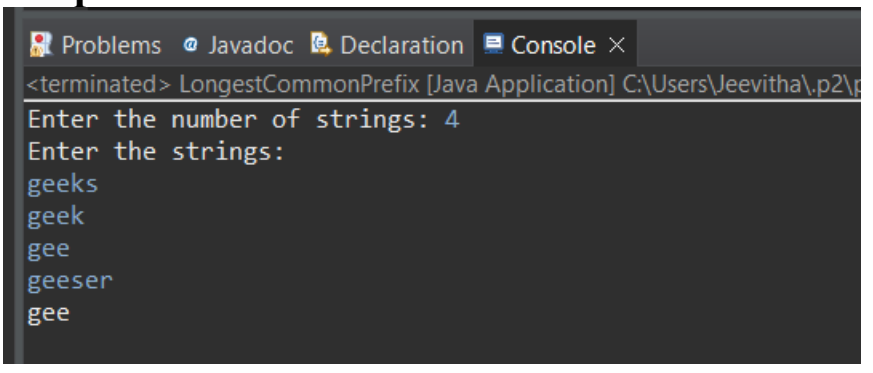
Time complexity: O(N^2)
Space complexity: O(N^2)

16.Longest Common Prefix using Sorting

```
package practiceset;
import java.util.*;
public class LongestCommonPrefix {
```

```
    public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter the number of strings: ");
    int n=sc.nextInt();
    sc.nextLine();
    String[] arr=new String[n];
    System.out.println("Enter the strings:");
    for(int i=0;i<n;i++){
        arr[i]=sc.nextLine();
    }
    Arrays.sort(arr);
    String first=arr[0];
    String last=arr[arr.length-1];
    StringBuilder ans=new StringBuilder();
    for(int i=0;i<Math.min(first.length(),last.length());i++){
        if(first.charAt(i)!=last.charAt(i)){
            System.out.println("-1");
            sc.close();
            return;
    }
        ans.append(first.charAt(i));
    }
    System.out.println(ans.toString().isEmpty()?"-1":ans.toString());
        }
}
```

Output:



Time complexity: O(NlogN+M)
Space complexity: O(N)

17. Delete middle element of a stack

```
package practiceset;
import java.util.*;
public class DeleteMiddleElement {
    public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter the number of elements in the stack: ");
    int n=sc.nextInt();
    Stack<Integer> stack=new Stack<>();
    System.out.println("Enter the elements of the stack:");
    for(int i=0;i<n;i++){
        stack.push(sc.nextInt());
    }
    int middleIndex=n/2;
```

```
        Stack<Integer> tempStack=new Stack<>();
        for(int i=0;i<=middleIndex;i++){
            tempStack.push(stack.pop());
        }
        tempStack.pop();
        while(!stack.isEmpty()){
            tempStack.push(stack.pop());
        }
        while(!tempStack.isEmpty()){
            System.out.print(tempStack.pop()+" ");
        }
            }
}
```

Output:

```
Problems  Javadoc  Declaration  Console ×
<terminated> DeleteMiddleElement [Java Application] C:\Users\Jeevith
Enter the number of elements in the stack: 5
Enter the elements of the stack:
1 2 3 4 5
1 2 4 5
```

Time complexity: O(n)
Space complexity: O(n)

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

```
package practiceset;
import java.util.*;
public class NextGreaterElement {
    public static void printNextGreaterElements(int[] arr) {
        Stack<Integer> stack=new Stack<>();
        HashMap<Integer, Integer> ngeMap=new HashMap<>();

        for (int i=arr.length-1; i>=0; i--) {
            int current=arr[i];

            while (!stack.isEmpty() && stack.peek()<=current) {
                stack.pop();
            }

            int nextGreater=stack.isEmpty()?-1:stack.peek();
            ngeMap.put(current, nextGreater);

            stack.push(current);
        }
        for (int num:arr) {
            System.out.println(num+" -> "+ngeMap.get(num));
        }
    }

    public static void main(String[] args) {
```

```
        int[] arr1={4, 5, 2, 25};
        System.out.println("Array: "+java.util.Arrays.toString(arr1));
        printNextGreaterElements(arr1);

        int[] arr2={13, 7, 6, 12};
        System.out.println("\nArray: "+java.util.Arrays.toString(arr2));
        printNextGreaterElements(arr2);
            }

}
```

Output

```
Problems  Javadoc  Declaration  Console ×
<terminated> NextGreaterElement [Java Application] C:\Users\Jeevitha\.p2\
Array: [4, 5, 2, 25]
4 -> 5
5 -> 25
2 -> 25
25 -> -1

Array: [13, 7, 6, 12]
13 -> -1
7 -> 12
6 -> 12
12 -> -1
```

Time complexity: O(n)
Space complexity: O(n)

19. Print Right View of a Binary Tree

```
package practiceset;
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 BinaryTreeRightView {
    public List<Integer> rightSideView(TreeNode root) {
        List<Integer> result=new ArrayList<>();
        if (root==null) {
            return result;
        }

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

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

            for (int i=0; i<levelSize; i++) {
                TreeNode curr=queue.poll();
                rightNode=curr;

                if (curr.left!=null) {
                    queue.add(curr.left);
                }
                if (curr.right!=null) {
                    queue.add(curr.right);
                }
            }

            result.add(rightNode.val);
        }

        return result;
    }

    public static void main(String[] args) {
        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);
        root.right.right=new TreeNode(6);
        root.left.left.right=new TreeNode(7);

        BinaryTreeRightView  solution=new BinaryTreeRightView ();
        List<Integer> rightView=solution.rightSideView(root);

        System.out.println("Right View of the Binary Tree: "+rightView);
    }
}
```

Output:

```
Problems  Javadoc  Declaration  Console ×
<terminated> BinaryTreeRightView [Java Application] C:\Users\Jeevitha\.p2\
Right View of the Binary Tree: [1, 3, 6, 7]
```

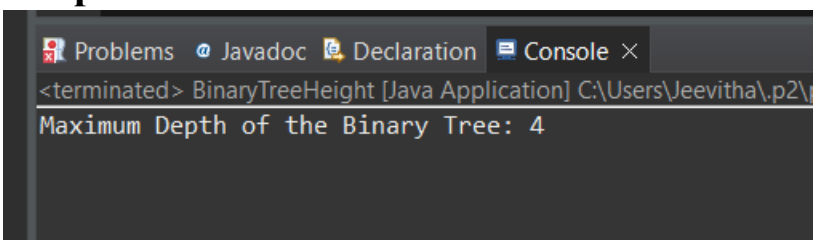
Time complexity: O(n)
Space complexity: O(n)

20.Maximum Depth or Height of Binary Tree

```
package practiceset;

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 BinaryTreeHeight {
    public int maxDepth(TreeNode root) {
        if (root==null) {
            return 0;
        }

        int leftHeight=maxDepth(root.left);
        int rightHeight=maxDepth(root.right);

        return Math.max(leftHeight,rightHeight)+1;
    }

    public static void main(String[] args) {
        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);
        root.right.right=new TreeNode(6);
```

```
        root.left.left.right=new TreeNode(7);

        BinaryTreeHeight tree=new BinaryTreeHeight();
        int height=tree.maxDepth(root);

        System.out.println("Maximum Depth of the Binary Tree: "+height);
    }
}
```

Output



Time complexity: O(n)
Space complexity: O(n)