DATE : (9/11/2024)

DSA Pracitice Problem

-by Thamizhvendhan V IT

1)Problem 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.

Code :

```
import java.util.*;

public class Main{
    public static void main(String... argv){
        System.out.println("Enter the Size of The Array");
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        int[] nums = new int[n];
        for(int i=0;i<n;i++){
            nums[i] = scan.nextInt();
        }
        int[] dp = new int[nums.length];
    dp[0] = nums[0];
    int ans = dp[0];
    for(int i=1;i<nums.length;i++){
            dp[i] = Math.max(nums[i],dp[i-1]+nums[i]);
            ans = Math.max(ans,dp[i]);
```

```
        }
                System.out.println(ans);
        }
}
```

Output :

```
C:\thamizh\Java Practice Test>java Main.java
Enter the Size of The Array
7
2 3 -8 7 -1 2 3
11

C:\thamizh\Java Practice Test>java Main.java
Enter the Size of The Array
2
-2 -4
-2

C:\thamizh\Java Practice Test>
```

Time Complexity : O(n);

2)Problem2.

Maximum Product Subarray

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

Code:

```
import java.util.*;

public class Main{
        public static void main(String... argv){
            System.out.println("Enter the Size of The Array");
            Scanner scan = new Scanner(System.in);
```

```
        int n = scan.nextInt();
        int[] nums = new int[n];
        for(int i=0;i<n;i++){
                nums[i] = scan.nextInt();
        }
        double ans=Integer.MIN_VALUE;
    double product=1;
    for(int i=0;i<nums.length;i++){
        product *= nums[i];
        ans = Math.max(ans,product);
        if(nums[i]==0){
                product=1;
        }
    }
    product=1;
    for(int i=nums.length-1;i>=0;i--){
        product *= nums[i];
        ans  =Math.max(ans,product);
        if(nums[i]==0){
                product=1;
        }

    }
```

```
        System.out.println((int)ans);
        }
}
```

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
Enter the Size of The Array
6
-2 6 -3 -10 0 2
180

C:\thamizh\Java Practice Test>java Main.java
Enter the Size of The Array
5
-1 -3 -10 0 60
60
```

Time Complexity : O(n);

PROBLEM 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.

CODE :

```
import java.util.*;

public class Main{
        public static void main(String... argv){
```

```
System.out.println("Enter the Size of The Array");
Scanner scan = new Scanner(System.in);
int n = scan.nextInt();
int[] nums = new int[n];
for(int i=0;i<n;i++){
        nums[i] = scan.nextInt();
}
System.out.println("Enter the Terget Element : ");
int target = scan.nextInt();
int low = 0;
int high = n-1;
boolean found = false;
while(low <= high){
        int mid = low + (high - low)/2;
        if(nums[mid]==target){
                System.out.println("Index is " + mid);
                found = true;
                break;
        }
        if(nums[low]<=nums[mid]){
                if(nums[low] <= target && target < nums[mid]){
                        high = mid - 1;
                }else{
                        low = mid + 1;
```

```
                }
            }else{
                if(nums[mid]<target && target <= nums[high]){
                    low = mid + 1;
                }else{
                    high = mid - 1;
                }
            }
        }
        if(!found)
        System.out.println("Not Found " + -1);
    }
}
```

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
Enter the Size of The Array
7
4 5 6 7 0 1 2
Enter the Terget Element :
0
Index is 4

C:\thamizh\Java Practice Test>java Main.java
Enter the Size of The Array
7
4 5 6 7 0 1 2
Enter the Terget Element :
3
Not Found -1
```

TIME COMPLEXITY : O(log n)

Problem 4 :

Container With Most Water

Given n non-negative integers $a_1, a_2, \ldots, a_n$ where each represents a point at coordinate (i, a_i). ' n ' vertical lines are drawn such that the two endpoints of line i is at (i, a_i) and $(i, 0)$. Find two lines, which together with x-axis forms a container, such that the container contains the most water.

The program should return an integer which corresponds to the maximum area of water that can be contained (maximum area instead of maximum volume sounds weird but this is the 2D plane we are working with for simplicity).

Note: You may not slant the container.

Code :

```
import java.util.*;
public class Main{
        public static void main(String... argv){
           System.out.println("Enter the Size of The Array");
           Scanner scan = new Scanner(System.in);
           int n = scan.nextInt();
           int[] nums = new int[n];
           for(int i=0;i<n;i++){
                   nums[i] = scan.nextInt();
           }
           int result = 0;
           int left = 0;
           int right = n-1;
```

```
            while(left<right){
                int currentArea = Math.min(nums[left],nums[right])*(right-
left);
                result = Math.max(result,currentArea);
                if(nums[left]>nums[right]) right--;
                else left++;
            }
            System.out.println("Maximum water :" +  result);
        }
}
```

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
Enter the Size of The Array
4
1 5 4 3
Maximum water :6

C:\thamizh\Java Practice Test>java Main.java
Enter the Size of The Array
5
3 1 2 4 5
Maximum water :12
```

TIME COMPLEXITY : O(n)

PROBLEM 5 :

Find the Factorial of a large number.

CODE :

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

```
public class Main{
	public static void main(String... argv){
		Scanner scan = new Scanner(System.in);
		System.out.println("Enter the Number to find the factorial ");
		int n = scan.nextInt();
		BigInteger result = new BigInteger("1");
		for(int i=1;i<=n;i++){
			result = result.multiply(BigInteger.valueOf(i));
		}
		System.out.println(result);
	}
}
```

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
Enter the Number to find the factorial
100
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000000

C:\thamizh\Java Practice Test>java Main.java
Enter the Number to find the factorial
50
30414093201713378043612608166064768844377641568960512000000000000
```

TIME COMPLEXITY : O(n)

PROBLEM 6 :

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.

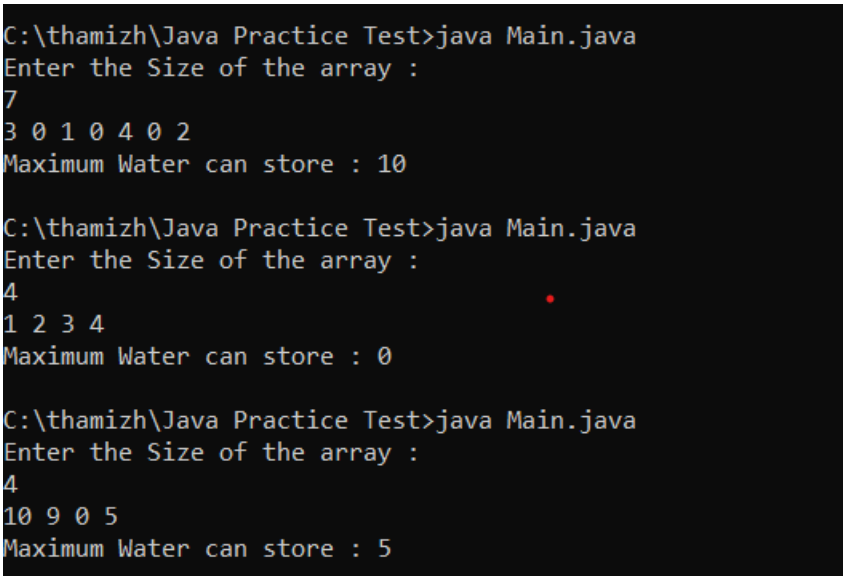
CODE :

```
import java.util.*;

public class Main{
```

```
public static void main(String... argv){
	Scanner scan = new Scanner(System.in);
	System.out.println("Enter the Size of the array : ");
	int n = scan.nextInt();
	int[] nums = new int[n];
	for(int i=0;i<n;i++){
		nums[i] = scan.nextInt();
	}
	int[] leftMax = new int[n];
	int[] rightMax = new int[n];
	for(int i=0;i<n;i++){
		if(i==0){
			leftMax[i] = nums[i];
			rightMax[n-1] = nums[n-1];
		}else{
			leftMax[i]  = Math.max(nums[i],leftMax[i-1]);
			rightMax[n-i-1] = Math.max(nums[n-i-1],rightMax[n-i]);
		}
	}
	int result = 0;
	for(int i=0;i<n;i++){
		result += Math.min(leftMax[i],rightMax[i]) - nums[i];
	}
	System.out.println("Maximum Water can store : " + result);
```

```
        }
}
```

OUTPUT :



TIME COMPLEXITY : O(n)

PROBLEM 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.

CODE :

```
import java.util.*;

public class Main{

                public static void main(String... argv){
```

```
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter the Size of the array : ");
        int n = scan.nextInt();
        int[] nums = new int[n];
        for(int i=0;i<n;i++){
            nums[i] = scan.nextInt();
        }
        System.out.println("Enter the Number of Students : ");
        int m = scan.nextInt();
        int left = 0;
        int right = 0;
        Arrays.sort(nums);
        int min = Integer.MAX_VALUE;
        while(right < n){
            if((right-left+1) == m){
                min = Math.min(nums[right] - nums[left],min);
                left++;
            }
            right++;
        }
        System.out.println("Minimum Difference is " + min);
    }
}
```

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
Enter the Size of the array :
7
7 3 2 4 9 12 56
Enter the Number of Students :
3
Minimum Difference is 2

C:\thamizh\Java Practice Test>java Main.java
Enter the Size of the array :
7
7 3 2 4 9 12 56
Enter the Number of Students :
5
Minimum Difference is 7
```

TIME COMPLEXITY : O(nlog n)

PROBLEM 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.

CODE :

```
import java.util.*;


public class Main {
    public static void main(String... argv) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter the Number of Intervals: ");
        int n = scan.nextInt();
        int[][] intervals = new int[n][2];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < 2; j++) {
```

```
            intervals[i][j] = scan.nextInt();
        }
    }
    Arrays.sort(intervals, (arr1, arr2) -> {
        if (arr1[0] == arr2[0]) {
            return arr1[1] - arr2[1];
        } else {
            return arr1[0] - arr2[0];
        }
    });
    List<int[]> result = new ArrayList<>();
    int[] current = intervals[0];
    result.add(current);
    for (int[] interval : intervals) {
        int currBegin = current[0];
        int currEnd = current[1];
        int nextBegin = interval[0];
        int nextEnd = interval[1];
        if (currEnd >= nextBegin) {
            current[1] = Math.max(currEnd, nextEnd);
        } else {
            current = interval;
            result.add(current);
        }
```

```
        }
        System.out.println("Merged Intervals: ");
        for (int[] interval : result) {
            System.out.println(Arrays.toString(interval));
        }
    }
}
```

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
Enter the Number of Intervals:
4
1 3
2 4
6 8
9 10
Merged Intervals:
[1, 4]
[6, 8]
[9, 10]

C:\thamizh\Java Practice Test>java Main.java
Enter the Number of Intervals:
4

7 8
1 5
2 4
4 6
Merged Intervals:
[1, 6]
[7, 8]
```

TIME COMPLEXITY : O(nlogn)

PROBLEM 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.

CODE :

```
import java.util.*;

public class Main {

   public static void main(String... argv) {

      Scanner scan = new Scanner(System.in);

      System.out.println("Enter the Number of Rows ");

      int n = scan.nextInt();

            System.out.println("Enter the Number of Columns ");

            int m = scan.nextInt();

      int[][] mat = new int[n][m];

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

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

                  mat[i][j] = scan.nextInt();

               }

            }

            int[][] result = new int[n][m];

            boolean[] row = new boolean[n];

            boolean[] col = new boolean[m];

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

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

                 if(mat[i][j]==1 && !row[i] && !col[j]){
```

```
                    helper(result,i,j);
                    row[i] = true;
                    col[j] = true;
                }
            }
        }
        System.out.println("result is : ");
        for(int[] arr : result){
            System.out.println(Arrays.toString(arr));
        }
    }
    public static void helper(int[][] result,int row,int col){
        for(int i=0;i<result[0].length;i++){
            result[row][i] = 1;
        }
        for(int i=0;i<result.length;i++){
            result[i][col] = 1;
        }
    }
}
```

OUTPUT:

```
C:\thamizh\Java Practice Test>java Main.java
Enter the Number of Rows
2
Enter the Number of Columns
2
1 0
0 0
result is :
[1, 1]
[1, 0]

C:\thamizh\Java Practice Test>java Main.java
Enter the Number of Rows
2
Enter the Number of Columns
3
0 0 0
0 0 1
result is :
[0, 0, 1]
[1, 1, 1]
```

TIME COMPLEXITY : O((n×m)×(m+n))

PROBLEM 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.

CODE :

```
import java.util.*;

public class Main {

   public static void main(String... argv) {

      Scanner scan = new Scanner(System.in);

      System.out.println("Enter the Number of Rows ");

      int n = scan.nextInt();

         System.out.println("Enter the Number of Columns ");

         int m = scan.nextInt();

      int[][] mat = new int[n][m];
```

```
for(int i=0;i<n;i++){
      for(int j=0;j<m;j++){
            mat[i][j] = scan.nextInt();
      }
   }
   List<Integer> result = new ArrayList<>();
   int startRow = 0;
   int endRow = n-1;
   int startCol = 0;
   int endCol = m-1;
   while(result.size() < n*m){
     for(int i=startCol;i<=endCol;i++){
            result.add(mat[startRow][i]);
     }
     if(result.size()== m*n) break;
     startRow++;
     for(int i=startRow;i<=endRow;i++){
            result.add(mat[i][endCol]);
     }
     if(result.size()== m*n) break;
     endCol--;
     for(int i=endCol;i>=startCol;i--){
            result.add(mat[endRow][i]);
   }
```

```
        if(result.size()== m*n) break;
          endRow--;
          for(int i=endRow;i>=startRow;i--){
                result.add(mat[i][startCol]);
          }
        if(result.size()== m*n) break;
          startCol++;
        }
        System.out.println(result);
  }
}
```

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
Enter the Number of Rows
3
Enter the Number of Columns
6
1 2 3 4 5 6
7 8 9 10 11 12
13 14 15 16 17 18
[1, 2, 3, 4, 5, 6, 12, 18, 17, 16, 15, 14, 13, 7, 8, 9, 10, 11]

C:\thamizh\Java Practice Test>java Main.java
Enter the Number of Rows
4
Enter the Number of Columns
4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
[1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10]
```

TIME COMPLEXITY : O(n×m)

PROBLEM 13 :

Check if given Parentheses expression is balanced or not

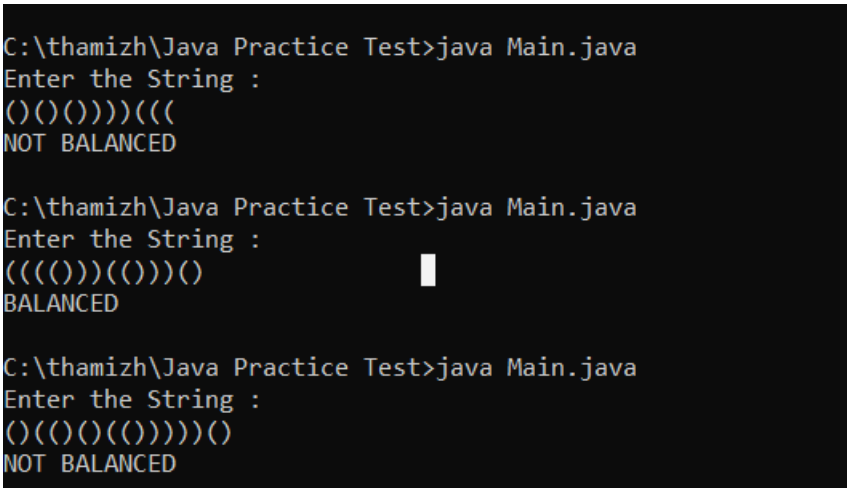
Given a string str of length N, consisting of „(„ and „)„ only, the task is to check whether it is balanced or not.

CODE :

```
import java.util.*;

public class Main {

   public static void main(String... argv) {

      Scanner scan = new Scanner(System.in);

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

         String str = scan.next();

         int open = 0;

         int close = 0;

         boolean flag = true;

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

            if(str.charAt(i)=='('){

                  open++;

            }else{

                  if(open==0){

                    flag = false;

                    break;

                  }else{

                    open--;

                  }

            }

         }
```

```
        if(!flag) System.out.println("NOT BALANCED");
        else System.out.println("BALANCED");
    }
}
```

OUTPUT :



TIME COMPLEXITY : O(n)

PROBLEM 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.

CODE:

```
import java.util.*;


public class Main {
   public static void main(String... argv) {
      Scanner scan = new Scanner(System.in);
```

```
    System.out.println("Enter the String1 : " );
      String str1 = scan.next();
      System.out.println("Enter the String2 : ");
      String str2 = scan.next();
      if(str1.length() != str2.length()){
         System.out.println("false");
         return;
      }
      int[] arr1 = new int[26];
      for(int i=0;i<str1.length();i++){
         char ch = str1.charAt(i);
         arr1[ch -'a']++;
      }
      int[] arr2 = new int[26];
      for(int i=0;i<str2.length();i++){
         char ch = str2.charAt(i);
         arr2[ch - 'a']++;
      }
      if(Arrays.equals(arr1,arr2)){
        System.out.println("true");
      }else{
        System.out.println("false");
      }
}
```

}

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
Enter the String1 :
ahhree
Enter the String2 :
eerhha
true

C:\thamizh\Java Practice Test>java Main.java
Enter the String1 :
qwertqwe
Enter the String2 :
qqwewerte
false
```

TIME COMPLEXITY : O(n)

PROBLEM 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.

CODE :

```
import java.util.*;

public class Main {

   public static void main(String... argv) {

      Scanner scan = new Scanner(System.in);

      System.out.println("Enter the String1 : " );

         String s = scan.next();

         int max = 1;

         int start = 0;

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

```
            int len1 = helper(s,i,i);
            int len2 = helper(s,i,i+1);
            int len = Math.max(len1,len2);
            if(len > max){
                  start = i - (len - 1)/2;
               max = len;
            }
         }
         System.out.println("Longest polindrome substring is " +
s.substring(start,start+max));
  }
  public static int helper(String s,int left,int right){
    while(left >= 0 && right<s.length() && s.charAt(left)==s.charAt(right)){
         left--;
         right++;
    }
    return right-left-1;
  }
}
```

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
Enter the String1 :
forgeeksskeegfor
Longest polindrome substring is geeksskeeg

C:\thamizh\Java Practice Test>java Main.java
Enter the String1 :
Geeeks
Longest polindrome substring is eee
```

TIME COMPLEXITY : O(n^2)

PROBLEM 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”.

CODE :

```
import java.util.*;


public class Main {
   public static void main(String... argv) {
      Scanner scan = new Scanner(System.in);
      System.out.println("Enter the number of String : " );
         int n = scan.nextInt();
         String[] strs = new String[n];
         for(int i=0;i<n;i++){
            strs[i] = scan.next();
         }
         StringBuilder ans = new StringBuilder();
      Arrays.sort(strs);
      String first = strs[0];
      String last = strs[strs.length-1];
      for(int i=0;i<Math.min(first.length(),last.length());i++){
         if(first.charAt(i)!=last.charAt(i)){
```

```
            break;
        }
        ans.append(first.charAt(i));
    }
        if(ans.length()==0){
            System.out.println(-1);
        }else{
          System.out.println(ans.toString());
        }
  }
}
```

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
Enter the number of String :
3
geeks  geeksforgeeks  geezer
gee

C:\thamizh\Java Practice Test>java Main.java
Enter the number of String :
2
fror trye
-1
```

TIME COMPLEXITY : O(n logn)

PROBLEM 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.

CODE :

```
import java.util.*;


public class Main {
    public static void main(String... argv) {
        Scanner scan = new Scanner(System.in);
          System.out.println("Enter the NUmber of elements in stack : ");
        int n = scan.nextInt();
          Stack<Integer> st = new Stack<>();
          for(int i=0;i<n;i++){
             int num = scan.nextInt();
             st.push(num);
          }
          for(int i=0;i<(int)(n/2);i++){
            st.pop();
          }
          System.out.println(st.peek());
   }
}
```

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
Enter the NUmber of elements in stack :
5
1 2 3 4 5
3

C:\thamizh\Java Practice Test>java Main.java
Enter the NUmber of elements in stack :
4
1 2 3 4
2

C:\thamizh\Java Practice Test>java Main.java
Enter the NUmber of elements in stack :
10
1 2 3 4 5 6 7 8 9 10
5
```

TIME COMPLEXITY : O(n)

PROBLEM 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.

CODE :

```
import java.util.*;


public class Main {
   public static void main(String... argv) {
      Scanner scan = new Scanner(System.in);
         System.out.println("Enter the Number of elements in stack : ");
      int n = scan.nextInt();
         int[] arr = new int[n];
         int[] nums = new int[n];
```

```
        for(int i=0;i<n;i++){
           arr[i] = scan.nextInt();
        }
        Stack<Integer> st = new Stack<>();
        int max = -1;
        for(int i=n-1;i>=0;i--){
           while(!st.isEmpty() && st.peek()<=arr[i]){
                st.pop();
           }
           if(st.isEmpty()){
                nums[i] = max;
           }else{
                nums[i] = st.peek();
           }
           st.push(arr[i]);
        }
        for(int i=0;i<n;i++){
           System.out.println(arr[i] + "-->" + nums[i]);
        }
  }
}
```

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
Enter the Number of elements in stack :
4
4 5 2 25
4-->5
5-->25
2-->25
25-->-1

C:\thamizh\Java Practice Test>java Main.java
Enter the Number of elements in stack :
4
13 7 6 12
13-->-1
7-->12
6-->12
12-->-1
```

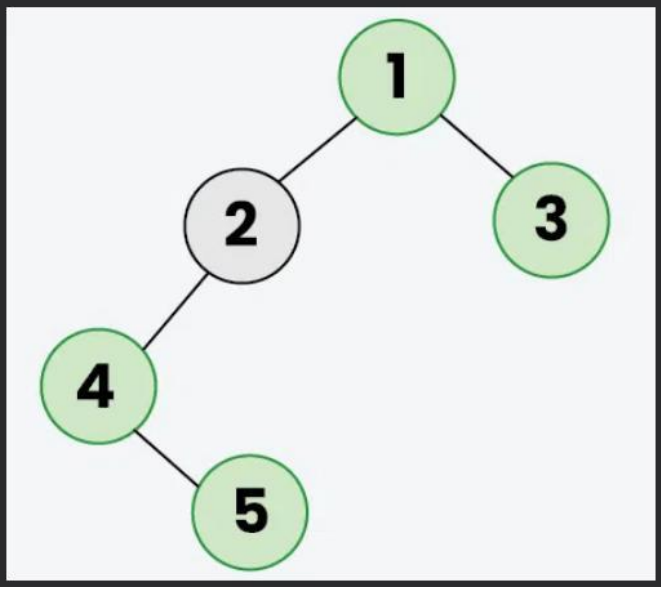
TIME COMPLEXITY : O(n)

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



CODE :

```
import java.util.*;
public class Main {
   public static void main(String... argv) {
      TreeNode root = new TreeNode(1);
      TreeNode node2 = new TreeNode(2);
      TreeNode node3 = new TreeNode(3);
      TreeNode node4 = new TreeNode(4);
      TreeNode node5 = new TreeNode(5);
      root.left = node2;
      root.right = node3;
      node2.left = node4;
      node4.right = node5;
        Queue<TreeNode> queue = new LinkedList<>();
        queue.add(root);
        List<Integer> result = new ArrayList<>();
        while(!queue.isEmpty()){
           int n = queue.size();
           for(int i=0;i<n;i++){
                TreeNode node = queue.remove();
                if(i==n-1) result.add(node.val);
                if(node.left!=null) queue.offer(node.left);
                if(node.right!=null) queue.offer(node.right);
           }
```

```
            }
            System.out.println(result);
        }
    }

    class TreeNode {
        int val;
        TreeNode left;
        TreeNode right;

        TreeNode(int val) {
            this.val = val;
            left = null;
            right = null;
        }
    }
```

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
[1, 3, 4, 5]
```

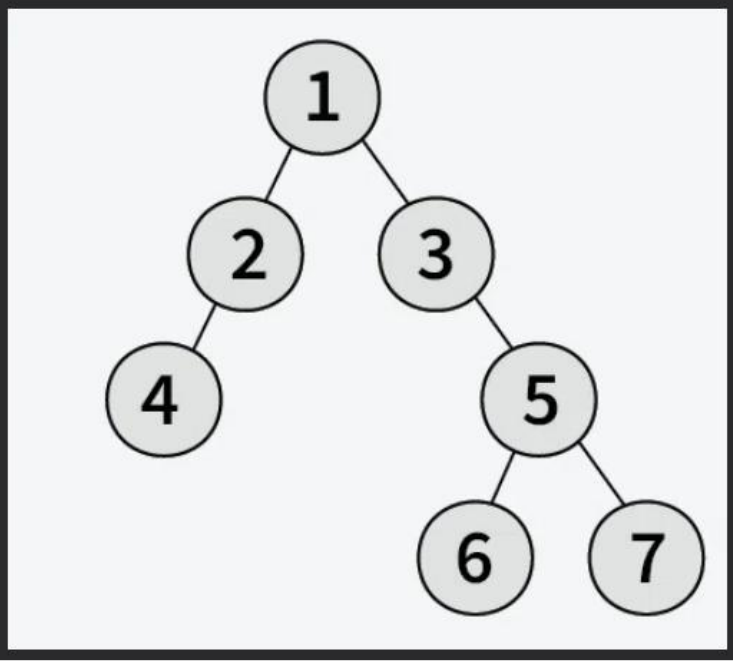
TIME COMPLEXITY : O(n)

PROBLEM 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 2: The height of the below binary tree is 4



CODE :

```
import java.util.*;
public class Main {
    public static void main(String... argv) {
        TreeNode root = new TreeNode(1);
        TreeNode node2 = new TreeNode(2);
        TreeNode node3 = new TreeNode(3);
        TreeNode node4 = new TreeNode(4);
        TreeNode node5 = new TreeNode(5);
           TreeNode node6 = new TreeNode(6);
        TreeNode node7 = new TreeNode(7);
        root.left = node2;
```

```
        root.right = node3;
        node2.left = node4;
        node3.right = node5;
            node5.left = node6;
            node5.right = node7;
            int result = helper(root);
            System.out.println(result);
    }
    public static int helper(TreeNode root){
            if(root==null) return 0;
            int left = helper(root.left);
            int right = helper(root.right);
            int max = Math.max(left,right)+1;
            return max;
    }
}

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int val) {
        this.val = val;
```

```
        left = null;
        right = null;
    }
}
```

OUTPUT :

```
C:\thamizh\Java Practice Test>java Main.java
4
```

TIME COMPLEXITY : O(n)