Coding Practice (9/11/2024):

1. **Maximum Subarray Sum – Kadane's Algorithm**
   Solution: Time Complexity: O(n)
   Space Complexity: O(n)

```java
package problems;
import java.util.*;
public class Program1 {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the Size of the Array");
        int n = s.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the Values");
        for (int i = 0; i<n;i++) {
            arr[i]=s.nextInt();
        }
        int sum=0 , summax=Integer.MIN_VALUE;
        for (int i = 0; i < n; i++) {
            sum+=arr[i];
            summax=Math.max(summax, sum);
            if (sum<=0) {
                sum=0;
            }
        }
        System.out.println(summax);
    }
}
```

```
Enter the Size of the Array
7
Enter the Values
2
3
-8
7
-1
2
3
11
```

2. **Maximum Product Subarray**

Solution: Time Complexity: O(n)

Space Complexity: O(n)

```java
package problems;
import java.util.*;
public class Program2 {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the size");
        int n = scanner.nextInt();
        System.out.println("Enter the values");
        int[] arr = new int[n];
        for (int i = 0;i < n; i++) {
            arr[i]=scanner.nextInt();
        }
        int pre=1 , suf=1 , ans=Integer.MIN_VALUE;
        for (int i=0;i<n;i++) {
            if (pre==0) pre=1;
            if (suf==0) suf=1;
            pre*=arr[i];
            suf*=arr[n-i-1];
            ans=Math.max(ans, Math.max(pre, suf));
        }
        System.out.println(ans);
    }
}
```

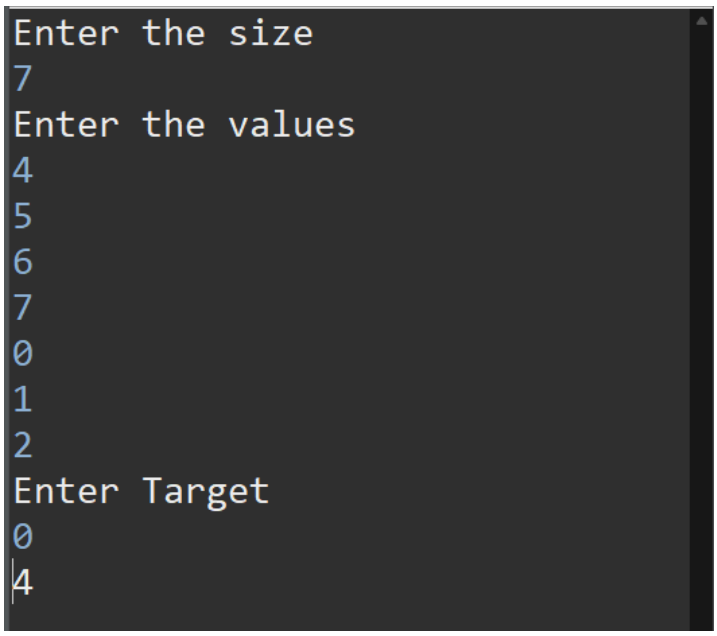
```
Enter the size
6
Enter the values
-2
6
-3
-10
0
2
180
```

3. **Search in a sorted and rotated Array**

Solution: Time Complexity: O(log n)

Space Complexity: O(1)

```java
package problems;

import java.util.Scanner;

public class Program3 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the size");
        int n = scanner.nextInt();
        System.out.println("Enter the values");
        int[] arr = new int[n];
        for (int i = 0;i < n; i++) {
            arr[i]=scanner.nextInt();
        }
        System.out.println("Enter Target");
        int tar= scanner.nextInt();
        int l=0 , r=arr.length - 1;
        boolean flag=true;
        while (l<=r) {
            int mid=(l+r)/2;
            if (arr[mid]==tar) {
                System.out.println(mid);
                flag=false;
                break;
            }
            if (arr[l]<=arr[mid]) {
                if (arr[l]<=tar && tar<=arr[mid]) {
                    r=mid-1;
                }
                else {
                    l=mid+1;
                }

            }
            else {
                if (arr[mid]<=tar && tar<=arr[r]) {
                    l=mid+1;
                }
                else {
                    r=mid-1;
                }
            }
        }
        if (flag) System.out.println(-1);
}
}
```


Enter the size
7
Enter the values
4
5
6
7
0
1
2
Enter Target
0
4

4. **Container with Most Water**

Solution: Time Complexity: O(n)

Space Complexity: O(1)

```java
package problems;

import java.util.Scanner;

public class Program4 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the size");
        int n = scanner.nextInt();
        System.out.println("Enter the values");
        int[] arr = new int[n];
        for (int i = 0;i < n; i++) {
            arr[i]=scanner.nextInt();
        }
        int res=0;
        int l=0 , r=arr.length-1;
        while (l<r) {
            res = Math.max(res,((r-l) * Math.min(arr[l], arr[r])));
            if (arr[l]<arr[r]) {
                l++;
            }
            else {
                r--;
            }
        }
        System.out.println(res);
}
}

```

```
Enter the size
4
Enter the values
1
5
4
3
6
```

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

Solution: Time Complexity: O(n)

Space Complexity: O(1)

```java
package problems;

import java.math.BigInteger;

public class Program5 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the size");
        int n = scanner.nextInt();
        BigInteger res= BigInteger.ONE;
        for (int i = 1; i <= n; i++) {
            res = res.multiply(BigInteger.valueOf(i));
        }
        System.out.println(res);
    }
}

```

```
Enter the size
100
933262154439441526816992388562667
```

6. **Trapping Rain Water**

Solution: Time Complexity: O(n)

Space Complexity: O(1)

```java
package problems;

import java.util.Scanner;

public class Program6 {
    public static int trap(int[] arr) {
        int left = 0;
        int right = arr.length - 1;
        int leftMax = arr[left];
        int rightMax = arr[right];
        int water = 0;

        while (left < right) {
            if (leftMax < rightMax) {
                left++;
                leftMax = Math.max(leftMax, arr[left]);
                water += leftMax - arr[left];
            } else {
                right--;
                rightMax = Math.max(rightMax, arr[right]);
                water += rightMax - arr[right];
            }
        }

        return water;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the size");
        int n = scanner.nextInt();
        System.out.println("Enter the values");
        int[] arr = new int[n];

        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }

        int waterTrapped = trap(arr);
        System.out.println(waterTrapped);

        scanner.close();
    }


}

```

```
Enter the size
7
Enter the values

3
0
1
0
4
0
2
10
```

7. Chocolate Distribution Problem

Solution: Time Complexity: O(n log n)

Space Complexity: O(1)

```java
package problems;

import java.util.*;


public class Program7 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the size");
        int n = scanner.nextInt();
        System.out.println("Enter the values");
        int[] arr = new int[n];

        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        System.out.println("Enter M value");
        int m = scanner.nextInt();
        Arrays.sort(arr);
        int op=Integer.MAX_VALUE;
        for (int i=0;i<n-m+1;i++) {
            int res=arr[i+m-1]-arr[i];
            op=Math.min(op,res);
        }
        System.out.println(op);
}
}
```

```
Enter the size
7
Enter the values
7
3
2
4
9
12
56
Enter M value
3
2
```

8. Merge Intervals

Solution: Time Complexity: O(n log n)

Space Complexity: O(n)

```java
package problems;

import java.util.*;

public class Program8 {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the Size of the array");
        int n = s.nextInt();

        int[][] intervals = new int[n][2];
        System.out.println("Enter the array");
        for (int i = 0; i < n; i++) {
            intervals[i][0] = s.nextInt();
            intervals[i][1] = s.nextInt();
        }

        int[][] mergedIntervals = mergeIntervals(intervals);

        System.out.println("Merged intervals:");
        for (int[] interval : mergedIntervals) {
            System.out.println(Arrays.toString(interval));
        }
    }

    public static int[][] mergeIntervals(int[][] intervals) {
        if (intervals.length <= 1) {
            return intervals;
        }

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

        ArrayList<int[]> mergedIntervals = new ArrayList<>();
        int[] currentInterval = intervals[0];
        mergedIntervals.add(currentInterval);

        for (int[] interval : intervals) {
            int currentEnd = currentInterval[1];
            int nextStart = interval[0];
            int nextEnd = interval[1];

            if (currentEnd >= nextStart) {
                currentInterval[1] = Math.max(currentEnd, nextEnd);
            } else {
                currentInterval = interval;
                mergedIntervals.add(currentInterval);
            }
        }

        return mergedIntervals.toArray(new int[mergedIntervals.size()][]);
    }
}
```

```
Enter the Size of the array

4
Enter the array
1 3
2 4
6 8
9 10
Merged intervals:
[1, 4]
[6, 8]
[9, 10]
```

9. A Boolean Matrix Question

Solution: Time Complexity: O(n * m)

Space Complexity: O(n + m)

```java
package problems;

import java.util.*;

public class Program9 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of rows:");
        int rows = scanner.nextInt();
        System.out.println("Enter the number of columns:");
        int cols = scanner.nextInt();
        int[][] matrix = new int[rows][cols];
        System.out.println("Enter the matrix values (0 or 1 only):");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = scanner.nextInt();
            }
        }
        modifyMatrix(matrix);
        System.out.println("Modified Matrix:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
        scanner.close();
    }

    public static void modifyMatrix(int[][] matrix) {
        int rows = matrix.length;
        int cols = matrix[0].length;

        boolean[] rowMark = new boolean[rows];
        boolean[] colMark = new boolean[cols];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (matrix[i][j] == 1) {
                    rowMark[i] = true;
                    colMark[j] = true;
                }
            }
        }

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (rowMark[i] || colMark[j]) {
                    matrix[i][j] = 1;
                }
            }
        }
    }
}
```

```
<terminated> Program9 [Java Application] C:\Program Files\Java\bin\javaw.exe (09-Nov-2024, 10:
Enter the number of rows:
2
Enter the number of columns:
2
Enter the matrix values (0 or 1
1 0
0 0
Modified Matrix:
1 1
1 0
```

10. Print a given matrix in spiral form

Solution: Time Complexity: O(n * m)

Space Complexity: O(n * m)

```java
package problems;
import java.util.*;
public class Program10 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of rows:");
        int rows = scanner.nextInt();
        System.out.println("Enter the number of columns:");
        int cols = scanner.nextInt();
        int[][] mat = new int[rows][cols];
        System.out.println("Enter the matrix values:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                mat[i][j] = scanner.nextInt();
            }
        }
        List<Integer> ans = printSpiral(mat);
        for (int num : ans) {
            System.out.print(num + " ");
        }
        System.out.println();
        scanner.close();
    }
    public static List<Integer> printSpiral(int[][] mat) {
        List<Integer> ans = new ArrayList<>();
        int n = mat.length;
        int m = mat[0].length;
        int top = 0, left = 0, bottom = n - 1, right = m - 1;

        while (top <= bottom && left <= right) {
            for (int i = left; i <= right; i++)
                ans.add(mat[top][i]);
            top++;

            for (int i = top; i <= bottom; i++)
                ans.add(mat[i][right]);
            right--;

            if (top <= bottom) {
                for (int i = right; i >= left; i--)
                    ans.add(mat[bottom][i]);
                bottom--;
            }

            if (left <= right) {
                for (int i = bottom; i >= top; i--)
                    ans.add(mat[i][left]);
                left++;
            }
        }
        return ans;
    }


}
```

```
<terminated> Program10 [Java Application] C:\Program Files\Java\jdk\javaw.exe (03-Nov-2024, 10:10:40 pm – 10:10:58 pm) [p
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
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
```

13. Check if given Parentheses expression is balanced or not

Solution: Time Complexity: O(n)

Space Complexity: O(1)

```java
package problems;

import java.util.Scanner;

public class Program13 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the expression:");
        String exp = scanner.nextLine();
        if (isBalanced(exp))
            System.out.println("Balanced");
        else
            System.out.println("Not Balanced");

        scanner.close();
    }
    public static boolean isBalanced(String exp) {
        boolean flag = true;
        int count = 0;

        for (int i = 0; i < exp.length(); i++) {
            if (exp.charAt(i) == '(') {
                count++;
            } else {
                count--;
            }
            if (count < 0) {
                flag = false;
                break;
            }
        }

        if (count != 0) {
            flag = false;
        }
        return flag;
    }
}
```

```
<terminated> Program13 [Java Application] C:\Program Files\Java\bin\javaw.exe (09-Nov-
Enter the expression:
((()))()()
Balanced
```

14. Check if two Strings are Anagrams of each other

Solution: Time Complexity: O(n)

Space Complexity: O(n)

```java
package problems;

import java.util.HashMap;

public class Program14 {

    static boolean areAnagrams(String s1, String s2) {
        HashMap<Character, Integer> charCount = new HashMap<>();
        for (char ch : s1.toCharArray())
            charCount.put(ch, charCount.getOrDefault(ch, 0) + 1);
        for (char ch : s2.toCharArray())
            charCount.put(ch, charCount.getOrDefault(ch, 0) - 1);
        for (var pair : charCount.entrySet()) {
            if (pair.getValue() != 0) {
                return false;
            }
        }

        return true;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the first string:");
        String s1 = scanner.nextLine();
        System.out.println("Enter the second string:");
        String s2 = scanner.nextLine();
        if(areAnagrams(s1, s2)) {
            System.out.println("True");;
        }
        else {
            System.out.println("False");
        }
        scanner.close();
    }
}
```

```
<terminated> Program14 [Java Application] C:\Program Files\Java\bin\javaw.exe  (09-Nov-
Enter the first string:
geeks
Enter the second string:
kseeg
True
```

15. Longest Palindromic Substring

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

Space Complexity: $O(n^2)$

```java
package problems;

import java.util.Scanner;

public class Program15 {

    static String longestPalSubstr(String s) {
        int n = s.length();
        boolean[][] dp = new boolean[n][n];
        int maxLen = 1;
        int start = 0;
        for (int i = 0; i < n; ++i)
            dp[i][i] = true;
        for (int i = 0; i < n - 1; ++i) {
            if (s.charAt(i) == s.charAt(i + 1)) {
                dp[i][i + 1] = true;
                start = i;
                maxLen = 2;
            }
        }

        for (int k = 3; k <= n; ++k) {
            for (int i = 0; i < n - k + 1; ++i) {
                int j = i + k - 1;
                if (dp[i + 1][j - 1] && s.charAt(i) == s.charAt(j)) {
                    dp[i][j] = true;
                    if (k > maxLen) {
                        start = i;
                        maxLen = k;
                    }
                }
            }
        }

        return s.substring(start, start + maxLen);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String s = scanner.nextLine();
        System.out.println("Longest palindrome substring is: " + longestPalSubstr(s));

        scanner.close();
    }
}

```

```
<terminated> Program15 [Java Application] C:\Program Files\Java\bin\javaw.exe  (09-Nov-2024, 10:26:07 pm – 10:26:18 pm) [pid: 22768]
Enter a string:
forgeeksskeegfor
Longest palindrome substring is: geeksskeeg
```

16. Longest Common Prefix using Sorting

Solution: Time Complexity: O(n log n + m)

Space Complexity: O(1)

```java
package problems;

import java.util.Arrays;

public class Program16 {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of strings:");
        int n = scanner.nextInt();
        String[] arr = new String[n];
        System.out.println("Enter the strings:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextLine();
        }

        System.out.println(longestCommonPrefix(arr));

        scanner.close();
    }
    static String longestCommonPrefix(String[] arr) {
        if (arr == null || 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;
        while (i < minLength && first.charAt(i) == last.charAt(i)) {
            i++;
        }

        if (i == 0)
            return "-1";

        return first.substring(0, i);
    }

}
```

```
<terminated> Program16 [Java Application] C:\Program Files\Java\bin\javaw.exe (09-Nov-202
Enter the number of strings:
2
Enter the strings:
hello world
-1
```

17. Delete middle element of a stack

Solution: Time Complexity: O(n)

Space Complexity: O(n)

```java
package problems;

import java.util.*;

public class Program17 {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Stack<Character> st = new Stack<>();
        System.out.println("Enter the stack elements:");
        String[] elements = scanner.nextLine().split(" ");
        for (String element : elements) {
            if (!element.isEmpty()) {
                st.push(element.charAt(0));
            }
        }
        Vector<Character> v = new Vector<>();
        while (!st.empty()) {
            v.add(st.pop());
        }
        int n = v.size();
        int target = n / 2;
        for (int i = 0; i < n; i++) {
            if (n % 2 == 0 && i == target || n % 2 != 0 && i == target) continue;
            st.push(v.get(i));
        }

        while (!st.empty()) {
            System.out.print(st.pop() + " ");
        }
        scanner.close();
    }
}

```

```
<terminated> Program17 [Java Application] C:\Program Files\Java\bin\javaw.exe (09-Nov-202
Enter the stack elements:
1 2 3 4 5 6
1 2 4 5 6
```

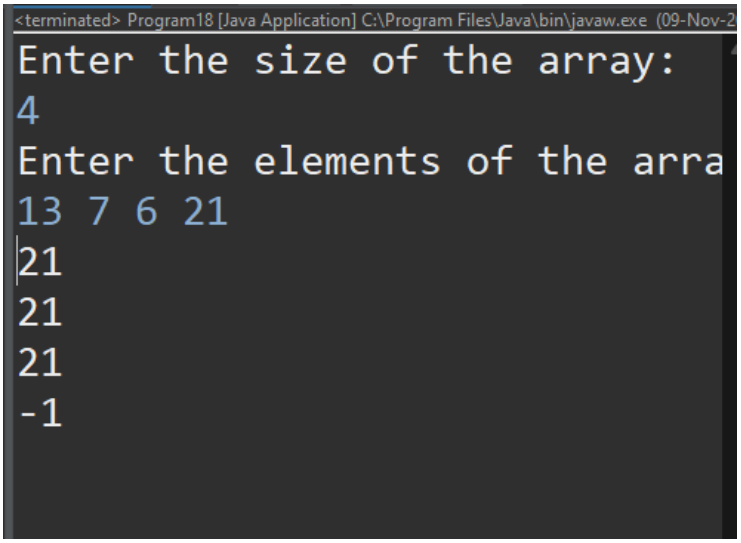
18. Next Greater Element (NGE) for every element in given Array

Solution: Time Complexity: O(n)

Space Complexity: O(n)

```java
package problems;

import java.util.Scanner;

public class Program18 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

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

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

        printNGE(arr, n);
        scanner.close();
    }
    static class Stack {
        int top;
        int[] items = new int[100];

        void push(int x) {
            if (top == 99) {
                System.out.println("Stack full");
            } else {
                items[++top] = x;
            }
        }

        int pop() {
            if (top == -1) {
                System.out.println("Underflow error");
                return -1;
            } else {
                return items[top--];
            }
        }

        boolean isEmpty() {
            return top == -1;
        }
    }

    static void printNGE(int[] arr, int n) {
        Stack s = new Stack();
        s.top = -1;

        s.push(arr[0]);

        for (int i = 1; i < n; i++) {
            int next = arr[i];

            if (!s.isEmpty()) {
```

```java
    static void printNGE(int[] arr, int n) {
        Stack s = new Stack();
        s.top = -1;

        s.push(arr[0]);

        for (int i = 1; i < n; i++) {
            int next = arr[i];

            if (!s.isEmpty()) {
                int element = s.pop();
                while (element < next) {
                    System.out.println(next);
                    if (s.isEmpty()) break;
                    element = s.pop();
                }

                if (element > next) s.push(element);
            }

            s.push(next);
        }

        while (!s.isEmpty()) {
            int element = s.pop();
            System.out.println(-1);
        }
    }


}
```

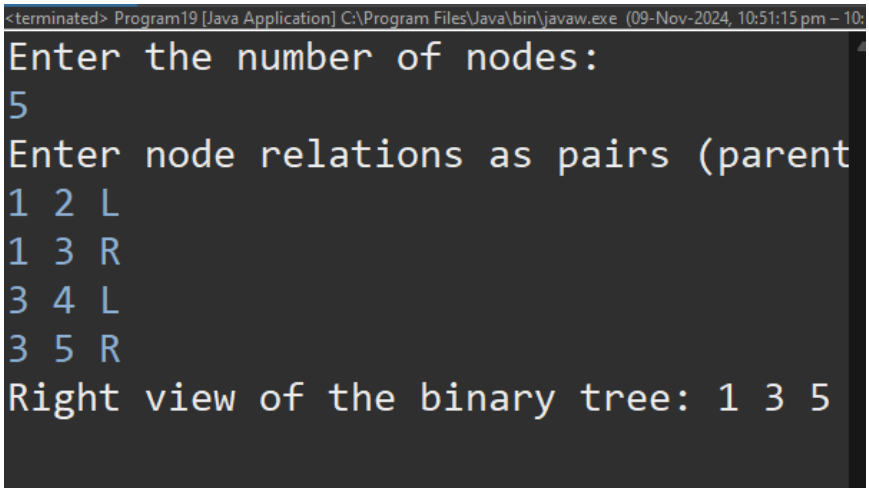

<terminated> Program18 [Java Application] C:\Program Files\Java\bin\javaw.exe (09-Nov-2
Enter the size of the array:
4
Enter the elements of the arra
13 7 6 21
21
21
21
-1

19. Print Right View of a Binary Tree

Solution: Time Complexity: O(n)

Space Complexity: O(n)

```java
package problems;

import java.util.ArrayList;

class Node {
    int data;
    Node left, right;

    Node(int x) {
        data = x;
        left = right = null;
    }
}

public class Program19 {
    public static void main(String[] args) {
        Node root = buildTree();
        ArrayList<Integer> result = rightView(root);
        System.out.print("Right view of the binary tree: ");
        printArray(result);
    }
    static void recursiveRightView(Node root, int level, int[] maxLevel, ArrayList<Integer> result) {
        if (root == null) return;

        if (level > maxLevel[0]) {
            result.add(root.data);
            maxLevel[0] = level;
        }

        recursiveRightView(root.right, level + 1, maxLevel, result);
        recursiveRightView(root.left, level + 1, maxLevel, result);
    }

    static ArrayList<Integer> rightView(Node root) {
        ArrayList<Integer> result = new ArrayList<>();
        int[] maxLevel = new int[]{-1};
        recursiveRightView(root, 0, maxLevel, result);
        return result;
    }

    static void printArray(ArrayList<Integer> arr) {
        for (int val : arr) {
            System.out.print(val + " ");
        }
        System.out.println();
    }

    static Node buildTree() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of nodes:");
        int nodeCount = scanner.nextInt();

        if (nodeCount <= 0) {
            return null;
        }

        Node[] nodes = new Node[nodeCount];
        for (int i = 0; i < nodeCount; i++) {
            nodes[i] = new Node(i + 1);
        }

        System.out.println("Enter node relations as pairs (parent, child, L/R for left/right):");
        for (int i = 0; i < nodeCount - 1; i++) {
            int parentIndex = scanner.nextInt() - 1;
            int childIndex = scanner.nextInt() - 1;
            char direction = scanner.next().charAt(0);

            if (direction == 'L' || direction == 'l') {
                nodes[parentIndex].left = nodes[childIndex];
            } else if (direction == 'R' || direction == 'r') {
                nodes[parentIndex].right = nodes[childIndex];
            }
        }
        scanner.close();
        return nodes[0];
    }


}
```


<terminated> Program19 [Java Application] C:\Program Files\Java\bin\javaw.exe (09-Nov-2024, 10:51:15 pm – 10:
Enter the number of nodes:
5
Enter node relations as pairs (parent
1 2 L
1 3 R
3 4 L
3 5 R
Right view of the binary tree: 1 3 5

20. Maximum Depth or Height of Binary Tree

Solution: Time Complexity: O(n)

Space Complexity: O(n)

```java
package problems;

import java.util.Scanner;

class Node {
    int data;
    Node left, right;

    Node(int val) {
        data = val;
        left = null;
        right = null;
    }
}

public class Program20 {

    static int maxDepth(Node node) {
        if (node == null)
            return 0;

        int lDepth = maxDepth(node.left);
        int rDepth = maxDepth(node.right);

        return Math.max(lDepth, rDepth) + 1;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Map<Integer, Node> nodes = new HashMap<>();

        System.out.println("Enter the number of nodes:");
        int n = scanner.nextInt();

        System.out.println("Enter each node's parent, child, and direction (L/R) separated by spaces:");

        Node root = null;
        for (int i = 0; i < n; i++) {
            int parentVal = scanner.nextInt();
            int childVal = scanner.nextInt();
            char direction = scanner.next().charAt(0);

            Node parentNode = nodes.computeIfAbsent(parentVal, Node::new);
            if (root == null) root = parentNode; // First node becomes root

            Node childNode = nodes.computeIfAbsent(childVal, Node::new);
            if (direction == 'L') {
                parentNode.left = childNode;
            } else if (direction == 'R') {
                parentNode.right = childNode;
            }
        }
        System.out.println("Maximum Depth of the Binary Tree: " + maxDepth(root));
        scanner.close();
    }
}
```

```
<terminated> Program20 [Java Application] C:\Program Files\Java\bin\javaw.exe  (09-Nov-2024, 10:56:23 pm – 10
Enter the number of nodes:
4
Enter each node's parent, child, and
12 8 L
12 18 R
8 5 L
8 11 R
Maximum Depth of the Binary Tree: 3
```