

# DAY 6 DSA PRACTICE SET 1

9/11/2024

## 1. Maximum Subarray Sum – Kadane's Algorithm:

Code:

```
1  import java.util.*;
2
3  public class KadaneAlgorithm {
4      public static int maxSubArray(int[] nums) {
5          int maxSoFar = nums[0];
6          int maxEndingHere = nums[0];
7          for (int i = 1; i < nums.length; i++) {
8              maxEndingHere = Math.max(nums[i], maxEndingHere + nums[i]);
9              maxSoFar = Math.max(maxSoFar, maxEndingHere);
10         }
11         return maxSoFar;
12     }
13
14     public static void main(String[] args) {
15         Scanner scanner = new Scanner(System.in);
16
17         System.out.print("Enter the size of the array: ");
18         int size = scanner.nextInt();
19
20         int[] nums = new int[size];
21
22         System.out.println("Enter the elements of the array:");
23         for (int i = 0; i < size; i++) {
24             nums[i] = scanner.nextInt();
25         }
26
27         int maxSum = maxSubArray(nums);
28         System.out.println("Maximum Subarray Sum: " + maxSum);
29
30         scanner.close();
31     }
32 }
33
34
```

## Output:

```
Enter the size of the array: 5
Enter the elements of the array:
2 -3 5 -7 4
Maximum Subarray Sum: 5
PS F:\java> |
```

## Time Complexity:

$O(N)$

## 2. Maximum Product Subarray: Code:

```
1  import java.util.*;
2
3  public class MaximumProduct {
4
5
6      public static int maxProduct(int[] nums) {
7          if (nums.length == 0) return 0;
8
9          int maxSoFar = nums[0];
10         int maxEndingHere = nums[0];
11         int minEndingHere = nums[0];
12
13         for (int i = 1; i < nums.length; i++) {
14             if (nums[i] < 0) {
15
16                 int temp = maxEndingHere;
17                 maxEndingHere = minEndingHere;
18                 minEndingHere = temp;
19             }
20
21             maxEndingHere = Math.max(nums[i], maxEndingHere * nums[i]);
22             minEndingHere = Math.min(nums[i], minEndingHere * nums[i]);
23
24             maxSoFar = Math.max(maxSoFar, maxEndingHere);
25         }
26
27         return maxSoFar;
28     }
29
30     public static int[] readInputArray() {
31
32         try (Scanner scanner = new Scanner(System.in)) {
33             System.out.print("Enter the size of the array: ");
34             int size = scanner.nextInt();
35
36
37             int[] nums = new int[size];
38             System.out.println("Enter the elements of the array:");
39             for (int i = 0; i < size; i++) {
40                 nums[i] = scanner.nextInt();
41             }
42
43             return nums;
44         }
45     }
46
47
48     public static void main(String[] args) {
49         int[] nums = readInputArray();
50         int maxProduct = maxProduct(nums);
51         System.out.println("Maximum Subarray Product: " + maxProduct);
52     }
53
54 }
```

## Output:

```
Enter the size of the array: 4
Enter the elements of the array:
-2
3
4
-5
Maximum Subarray Product: 120
```

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

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

## Code:

```
1 public class RotatedSortedArray {
2
3     public static int searchInRotatedArray(int[] arr, int key) {
4         int low = 0;
5         int high = arr.length - 1;
6
7         while (low <= high) {
8             int mid = low + (high - low) / 2;
9
10            if (arr[mid] == key) {
11                return mid;
12            }
13
14            if (arr[low] <= arr[mid]) {
15
16                if (arr[low] <= key && key < arr[mid]) {
17                    high = mid - 1;
18                } else {
19                    low = mid + 1;
20                }
21            } else {
22
23                if (arr[mid] < key && key <= arr[high]) {
24                    low = mid + 1;
25                } else {
26                    high = mid - 1;
27                }
28            }
29        }
30
31        return -1;
32    }
33
34    public static void main(String[] args) {
35        java.util.Scanner scanner = new java.util.Scanner(System.in);
36
37        System.out.print("Enter the number of elements in the array: ");
38        int n = scanner.nextInt();
39
40        int[] arr = new int[n];
41        System.out.println("Enter the elements of the array (sorted and rotated):");
42        for (int i = 0; i < n; i++) {
43            arr[i] = scanner.nextInt();
44        }
45
46        System.out.print("Enter the key to search for: ");
47        int key = scanner.nextInt();
48        int result = searchInRotatedArray(arr, key);
49
50        if (result != -1) {
51            System.out.println("Index of key: " + result);
52        } else {
53            System.out.println("Key not found.");
54        }
55
56        scanner.close();
57    }
58 }
```

## Output:

```
Enter the number of elements in the array: 7
Enter the elements of the array (sorted and rotated):
4 5 6 7 0 1 2
Enter the key to search for: 0
Index of key: 4
```

**Time Complexity:**

**$O(\log n)$**

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

## Code:

```
1  import java.util.Scanner;
2
3  class ContainerWithMostWater {
4      static int maxArea(int[] arr) {
5          int n = arr.length;
6          int left = 0;
7          int right = n - 1;
8          int area = 0;
9
10         while (left < right) {
11             area = Math.max(area, Math.min(arr[left], arr[right]) * (right - left));
12
13             if (arr[left] < arr[right])
14                 left += 1;
15             else
16                 right -= 1;
17         }
18         return area;
19     }
20
21     public static void main(String[] args) {
22         Scanner scanner = new Scanner(System.in);
23
24         System.out.print("Enter the number of elements in the array: ");
25         int n = scanner.nextInt();
26
27         int[] arr = new int[n];
28
29         System.out.println("Enter the elements of the array:");
30         for (int i = 0; i < n; i++) {
31             arr[i] = scanner.nextInt();
32         }
33
34         System.out.println("The maximum area is: " + maxArea(arr));
35
36         scanner.close();
37     }
38 }
39
```

## Output:

```
Enter the number of elements in the array: 4
Enter the elements of the array:
1 4 5 3
The maximum area is: 6
```

## Time Complexity:

**O(n)**

### 5. Find the Factorial of a large number:

**Code:**

```
1 import java.math.BigInteger;
2 import java.util.Scanner;
3
4 public class Factorial {
5     public static BigInteger factorial(int n) {
6         BigInteger result = BigInteger.ONE;
7         for (int i = 2; i <= n; i++) {
8             result = result.multiply(BigInteger.valueOf(i));
9         }
10        return result;
11    }
12
13    public static void main(String[] args) {
14        Scanner scanner = new Scanner(System.in);
15        System.out.println("Enter a number: ");
16        int n = scanner.nextInt();
17
18        BigInteger result = factorial(n);
19        System.out.println("The factorial of " + n + " is: " + result);
20
21        scanner.close();
22
23    }
24 }
25
```

### Output:

```
Enter a number:  
100  
The factorial of 100 is: 933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000  
000000000000000
```

### Time Complexity:

**O(n)**

## 6. Trapping Rainwater Problem:

Code:

```
1  import java.util.Scanner;
2
3  class TrappingRainwater {
4
5      static int findWater(int[] arr) {
6          int n = arr.length;
7
8          int[] left = new int[n];
9          int[] right = new int[n];
10         int res = 0;
11
12         left[0] = arr[0];
13         for (int i = 1; i < n; i++)
14             left[i] = Math.max(left[i - 1], arr[i]);
15
16         right[n - 1] = arr[n - 1];
17         for (int i = n - 2; i >= 0; i--)
18             right[i] = Math.max(right[i + 1], arr[i]);
19
20         for (int i = 1; i < n - 1; i++) {
21             int minOf2 = Math.min(left[i], right[i]);
22             if (minOf2 > arr[i]) {
23                 res += minOf2 - arr[i];
24             }
25         }
26
27         return res;
28     }
29
30     public static void main(String[] args) {
31         Scanner scanner = new Scanner(System.in);
32
33         System.out.print("Enter the number of elements: ");
34         int n = scanner.nextInt();
35
36         int[] arr = new int[n];
37         System.out.println("Enter the elements of the array: ");
38         for (int i = 0; i < n; i++) {
39             arr[i] = scanner.nextInt();
40         }
41
42         int result = findWater(arr);
43         System.out.println("The amount of trapped rainwater is: " + result);
44         scanner.close();
45     }
46 }
47
```

Output:



```
Enter the number of elements: 4
Enter the elements of the array:
2 0 3 0
The amount of trapped rainwater is: 2
```

**Time Complexity:**

**$O(n)$**

## 7. Chocolate Distribution Problem: Code:

```
1  import java.util.Arrays;
2  import java.util.Scanner;
3
4  class Chocولاتdistribution {
5
6      static int findMinDiff(int[] arr, int m) {
7          int n = arr.length;
8
9          Arrays.sort(arr);
10
11          int minDiff = Integer.MAX_VALUE;
12
13          for (int i = 0; i + m - 1 < n; i++) {
14              int diff = arr[i + m - 1] - arr[i];
15              if (diff < minDiff)
16                  minDiff = diff;
17          }
18          return minDiff;
19      }
20
21      public static void main(String[] args) {
22          Scanner scanner = new Scanner(System.in);
23
24          System.out.print("Enter the number of elements in the array: ");
25          int n = scanner.nextInt();
26
27          int[] arr = new int[n];
28          System.out.println("Enter the elements of the array:");
29          for (int i = 0; i < n; i++) {
30              arr[i] = scanner.nextInt();
31          }
32
33          System.out.print("Enter the value of m: ");
34          int m = scanner.nextInt();
35
36          if (m > n) {
37              System.out.println("m cannot be greater than the number of elements in the array.");
38          } else {
39              System.out.println("The minimum difference is: " + findMinDiff(arr, m));
40          }
41
42          scanner.close();
43      }
44  }
45
```

**Output:**

```
Enter the number of elements in the array: 7
Enter the elements of the array:
7 3 12 4 8 7 56
Enter the value of m: 3
The minimum difference is: 1
```

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

## 8. Merge Overlapping Intervals:

**Code:**

```
1  import java.util.ArrayList;
2  import java.util.Arrays;
3  import java.util.List;
4  import java.util.Scanner;
5
6  class Mergeoverlapping {
7
8      static List<int[]> mergeOverlap(int[][] arr) {
9          Arrays.sort(arr, (a, b) -> Integer.compare(a[0], b[0]));
10
11          List<int[]> res = new ArrayList<>();
12          res.add(new int[]{arr[0][0], arr[0][1]});
13
14          for (int i = 1; i < arr.length; i++) {
15              int[] last = res.get(res.size() - 1);
16              int[] curr = arr[i];
17
18              if (curr[0] <= last[1])
19                  last[1] = Math.max(last[1], curr[1]);
20              else
21                  res.add(new int[]{curr[0], curr[1]});
22          }
23
24          return res;
25      }
26
27      public static void main(String[] args) {
28          Scanner scanner = new Scanner(System.in);
29
30          System.out.print("Enter the number of intervals: ");
31          int n = scanner.nextInt();
32          int[][] arr = new int[n][2]; // Create an array to hold the intervals
33
34          System.out.println("Enter the intervals (start and end):");
35          for (int i = 0; i < n; i++) {
36              System.out.print("Interval " + (i + 1) + ": ");
37              arr[i][0] = scanner.nextInt(); // Start of the interval
38              arr[i][1] = scanner.nextInt(); // End of the interval
39          }
40
41          List<int[]> res = mergeOverlap(arr);
42
43          System.out.println("Merged intervals:");
44          for (int[] interval : res)
45              System.out.println(interval[0] + " " + interval[1]);
46
47          scanner.close();
48      }
49  }
50
```

## Output:

```
Enter the number of intervals: 4
Enter the intervals (start and end):
Interval 1: 1 3 4 2
Interval 2: Interval 3: 2 5 6 9
Interval 4: Merged intervals:
5
9
```

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

## 9. A Boolean Matrix Question: Code:

```
1  import java.util.Scanner;
2
3  class BooleanMatrix {
4      public static void modifyMatrix(int mat[][], int R, int C) {
5          int row[] = new int[R];
6          int col[] = new int[C];
7          int i, j;
8
9          for (i = 0; i < R; i++)
10             row[i] = 0;
11
12          for (i = 0; i < C; i++)
13             col[i] = 0;
14
15          for (i = 0; i < R; i++) {
16              for (j = 0; j < C; j++) {
17                  if (mat[i][j] == 1) {
18                      row[i] = 1;
19                      col[j] = 1;
20                  }
21              }
22          }
23
24          for (i = 0; i < R; i++)
25              for (j = 0; j < C; j++)
26                  if (row[i] == 1 || col[j] == 1)
27                      mat[i][j] = 1;
28      }
29
30      public static void printMatrix(int mat[][], int R, int C) {
31          int i, j;
32          for (i = 0; i < R; i++) {
33              for (j = 0; j < C; j++)
34                  System.out.print(mat[i][j] + " ");
35              System.out.println();
36          }
37      }
38
39      public static void main(String[] args) {
40
41          Scanner scanner = new Scanner(System.in);
42
43          System.out.print("Enter the number of rows (R): ");
44          int R = scanner.nextInt();
45          System.out.print("Enter the number of columns (C): ");
46          int C = scanner.nextInt();
47
48          int mat[][] = new int[R][C];
49
50          System.out.println("Enter the elements of the matrix (0 or 1):");
51          for (int i = 0; i < R; i++) {
52              for (int j = 0; j < C; j++) {
53                  mat[i][j] = scanner.nextInt();
54              }
55          }
56
57          System.out.println("Matrix Initially:");
58          printMatrix(mat, R, C);
59          modifyMatrix(mat, R, C);
60          System.out.println("Matrix after modification:");
61          printMatrix(mat, R, C);
62
63          scanner.close();
64      }
65  }
66
67
68
```

### Output:

```
Enter the number of rows (R): 3
Enter the number of columns (C): 3
Enter the elements of the matrix (0 or 1):
```

```
Matrix Initially:
1 1 0
0 1 1
1 1 1
Matrix after modification:
1 1 1
1 1 1
1 1 1
```

**Time complexity:**

**$O(m*n)$**

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

Code:

```
1  import java.util.Scanner;
2
3  public class Matrixspiral {
4
5      public static void spiralPrint(int m, int n, int[][] a) {
6
7          int top = 0, bottom = m - 1, left = 0, right = n - 1;
8
9          while (top <= bottom && left <= right) {
10
11              for (int i = left; i <= right; ++i) {
12                  System.out.print(a[top][i] + " ");
13              }
14              top++;
15
16              for (int i = top; i <= bottom; ++i) {
17                  System.out.print(a[i][right] + " ");
18              }
19              right--;
20
21              if (top <= bottom) {
22                  for (int i = right; i >= left; --i) {
23                      System.out.print(a[bottom][i] + " ");
24                  }
25                  bottom--;
26              }
27
28              if (left <= right) {
29                  for (int i = bottom; i >= top; --i) {
30                      System.out.print(a[i][left] + " ");
31                  }
32                  left++;
33              }
34          }
35      }
36
37      public static void main(String[] args) {
38          Scanner scanner = new Scanner(System.in);
39
40          System.out.print("Enter the number of rows: ");
41          int m = scanner.nextInt();
42          System.out.print("Enter the number of columns: ");
43          int n = scanner.nextInt();
44
45          int[][] matrix = new int[m][n];
46
47
48          System.out.println("Enter the elements of the matrix:");
49          for (int i = 0; i < m; i++) {
50              for (int j = 0; j < n; j++) {
51                  matrix[i][j] = scanner.nextInt();
52              }
53          }
54
55          System.out.println("Spiral order of the matrix:");
56          spiralPrint(m, n, matrix);
57
58          scanner.close();
59      }
60  }
```

Output:

```

Enter the number of rows: 3
Enter the number of columns: 3
Enter the elements of the matrix:
1
0
0
0
1
0
0
0
1
Spiral order of the matrix:
1 0 0 0 1 0 0 0 1

```

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

11. Check if given Parentheses expression is balanced or not:  
**Code:**

```

1  import java.util.Scanner;
2
3  public class Parenthesis {
4
5      public static String isBalanced(String str){
6          int balance = 0;
7          for (char ch : str.toCharArray()) {
8              if (ch == '(') {
9                  balance++;
10             } else if (ch == ')') {
11                 balance--;
12             }
13
14             if (balance < 0) {
15                 return "Not Balanced";
16             }
17         }
18
19         return (balance == 0) ? "Balanced" : "Not Balanced";
20     }
21
22     public static void main(String[] args) {
23         Scanner scanner = new Scanner(System.in);
24
25         System.out.print("Enter a parentheses expression: ");
26         String str = scanner.nextLine();
27
28         String result = isBalanced(str);
29         System.out.println(result);
30
31         scanner.close();
32     }
33 }

```

**Output:**

```
Enter a parentheses expression: (()())()  
Not Balanced
```

**Time complexity:**

**$O(n^2)$**

**12. Check if two Strings are Anagrams of each other:**

**Code:**

```
1  import java.util.HashMap;  
2  import java.util.Scanner;  
3  
4  class Anagram {  
5  
6      static boolean areAnagrams(String s1, String s2) {  
7  
8          HashMap<Character, Integer> charCount = new HashMap<>();  
9  
10  
11         for (char ch : s1.toCharArray())  
12             charCount.put(ch, charCount.getOrDefault(ch, 0) + 1);  
13  
14  
15         for (char ch : s2.toCharArray())  
16             charCount.put(ch, charCount.getOrDefault(ch, 0) - 1);  
17  
18         for (var pair : charCount.entrySet()) {  
19             if (pair.getValue() != 0) {  
20                 return false;  
21             }  
22         }  
23  
24  
25         return true;  
26     }  
27  
28     public static void main(String[] args) {  
29         Scanner scanner = new Scanner(System.in);  
30  
31  
32         System.out.print("Enter the first string: ");  
33         String s1 = scanner.nextLine();  
34  
35         System.out.print("Enter the second string: ");  
36         String s2 = scanner.nextLine();  
37  
38  
39         boolean result = areAnagrams(s1, s2);  
40         System.out.println(result ? "The strings are anagrams." : "The strings are not anagrams.");  
41  
42         scanner.close();  
43     }  
44 }  
45 }
```

**Output:**

```
Enter the first string: allergy
Enter the second string: allergic
The strings are not anagrams.
```

**Time complexity:**

**O(n)**

### 13. Longest Palindromic Substring: Code:

```
1  import java.util.Scanner;
2
3  public class Longestpallindrome {
4
5      public static String longestPalindrome(String str) {
6          int n = str.length();
7          boolean[][] dp = new boolean[n][n];
8          int start = 0, maxLength = 1;
9
10         for (int i = 0; i < n; i++) {
11             dp[i][i] = true;
12         }
13
14         for (int i = 0; i < n - 1; i++) {
15             if (str.charAt(i) == str.charAt(i + 1)) {
16                 dp[i][i + 1] = true;
17                 start = i;
18                 maxLength = 2;
19             }
20         }
21
22         for (int length = 3; length <= n; length++) {
23             for (int i = 0; i < n - length + 1; i++) {
24                 int j = i + length - 1;
25
26                 if (str.charAt(i) == str.charAt(j) && dp[i + 1][j - 1]) {
27                     dp[i][j] = true;
28                     start = i;
29                     maxLength = length;
30                 }
31             }
32         }
33
34         return str.substring(start, start + maxLength);
35     }
36
37     public static void main(String[] args) {
38         Scanner scanner = new Scanner(System.in);
39
40         System.out.print("Enter a string: ");
41         String input = scanner.nextLine();
42
43         String result = longestPalindrome(input);
44         System.out.println("Longest Palindromic Substring: " + result);
45
46         scanner.close();
47     }
48 }
49
50
```

**Output:**



```
Enter a string: abc
Longest Palindromic Substring: a
```

**Time complexity:**

**$O(n^2)$**

#### **14. Longest Common Prefix using Sorting:**

**Code:**

```
1  import java.util.Scanner;
2
3  public class Commonprefix {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6
7
8          System.out.print("Enter the number of strings: ");
9          int n = scanner.nextInt();
10         scanner.nextLine();
11
12         String[] arr = new String[n];
13         System.out.println("Enter the strings:");
14         for (int i = 0; i < n; i++) {
15             arr[i] = scanner.nextLine();
16         }
17
18         String result = longestCommonPrefix(arr);
19         if (result.isEmpty()) {
20             System.out.println("-1");
21         } else {
22             System.out.println("Longest common prefix: " + result);
23         }
24
25         scanner.close();
26     }
27
28     public static String longestCommonPrefix(String[] arr) {
29         if (arr == null || arr.length == 0) {
30             return "";
31         }
32
33         String prefix = arr[0];
34         for (int i = 1; i < arr.length; i++) {
35             while (arr[i].indexOf(prefix) != 0) {
36                 prefix = prefix.substring(0, prefix.length() - 1);
37                 if (prefix.isEmpty()) {
38                     return "";
39                 }
40             }
41         }
42         return prefix;
43     }
44 }
45
46
```

**Output:**

```
Enter the number of strings: 5
Enter the strings:
hello
string

class
prefix
1
```

## 15. Delete middle element of a stack: Code:

```
1 import java.util.Scanner;
2 import java.util.Stack;
3
4 public class DelateMiddle {
5
6     public static void deleteMiddle(Stack<Integer> stack, int currentIndex, int size) {
7         if (stack.isEmpty() || currentIndex == size / 2) {
8             stack.pop();
9             return;
10        }
11
12        int topElement = stack.pop();
13
14        deleteMiddle(stack, currentIndex + 1, size);
15
16        stack.push(topElement);
17    }
18
19    public static void deleteMiddleElement(Stack<Integer> stack) {
20        int size = stack.size();
21        deleteMiddle(stack, 0, size);
22    }
23
24    public static void main(String[] args) {
25        Scanner scanner = new Scanner(System.in);
26        Stack<Integer> stack = new Stack<>();
27
28        System.out.println("Enter the number of elements in the stack:");
29        int n = scanner.nextInt();
30        System.out.println("Enter the elements of the stack:");
31        for (int i = 0; i < n; i++) {
32            stack.push(scanner.nextInt());
33        }
34
35        System.out.println("Original Stack: " + stack);
36
37        deleteMiddleElement(stack);
38
39        System.out.println("Stack after deleting middle element: " + stack);
40
41        scanner.close();
42    }
43 }
44
```

## Output:

```
Enter the number of elements in the stack:
5
Enter the elements of the stack:
1 2 3 4 5
Original Stack: [1, 2, 3, 4, 5]
Stack after deleting middle element: [1, 2, 4, 5]
```

**16. Next Greater Element (NGE) for every element in given Array:**  
**Code:**

```
1  import java.util.Scanner;
2  import java.util.Stack;
3
4  public class NextGreaterElements {
5
6      public static void main(String[] args) {
7          Scanner scanner = new Scanner(System.in);
8
9          System.out.print("Enter the size of the array: ");
10         int n = scanner.nextInt();
11
12         int[] arr = new int[n];
13
14         System.out.println("Enter the elements of the array:");
15         for (int i = 0; i < n; i++) {
16             arr[i] = scanner.nextInt();
17         }
18
19         int[] nge = new int[n];
20
21         Stack<Integer> stack = new Stack<>();
22
23         for (int i = n - 1; i >= 0; i--) {
24             while (!stack.isEmpty() && stack.peek() <= arr[i]) {
25                 stack.pop();
26             }
27
28             if (stack.isEmpty()) {
29                 nge[i] = -1;
30             } else {
31                 nge[i] = stack.peek();
32             }
33
34             stack.push(arr[i]);
35         }
36
37         System.out.println("Next Greater Element for each element:");
38         for (int i = 0; i < n; i++) {
39             System.out.print(nge[i] + " ");
40         }
41         scanner.close();
42     }
43 }
44 }
```

**Output:**

```
Enter the size of the array: 4
Enter the elements of the array:
1 2 3 4
Next Greater Element for each element:
2 3 4 -1
```

## 17. 9. Print Right View of a Binary Tree: Code:

```
1  import java.util.LinkedList;
2  import java.util.Queue;
3
4  class Node {
5      int data;
6      Node left, right;
7
8      public Node(int item) {
9          data = item;
10         left = right = null;
11     }
12 }
13
14 class BinaryTree {
15     Node root;
16
17     // Function to print the right view of the binary tree
18     void rightView() {
19         if (root == null) {
20             return;
21         }
22
23         Queue<Node> queue = new LinkedList<>();
24         queue.add(root);
25
26         while (!queue.isEmpty()) {
27             int size = queue.size();
28             Node rightmostNode = null;
29
30             for (int i = 0; i < size; i++) {
31                 Node currentNode = queue.poll();
32                 rightmostNode = currentNode; // Update the rightmost node
33
34                 // Add left and right children to the queue
35                 if (currentNode.left != null) {
36                     queue.add(currentNode.left);
37                 }
38                 if (currentNode.right != null) {
39                     queue.add(currentNode.right);
40                 }
41             }
42             // Print the rightmost node of the current level
43             System.out.print(rightmostNode.data + " ");
44         }
45     }
46
47     public static void main(String[] args) {
48         BinaryTree tree = new BinaryTree();
49         tree.root = new Node(1);
50         tree.root.left = new Node(2);
51         tree.root.right = new Node(3);
52         tree.root.left.right = new Node(4);
53         tree.root.right.right = new Node(5);
54         tree.root.right.right.right = new Node(6);
55
56         System.out.println("Right view of the binary tree:");
57         tree.rightView();
58     }
59 }
```

Output:

```
PS F:\java> & 'C:\Program Files\Java\jdk-  
\\workspaceStorage\41ec9565a616c87806451eb6  
Maximum depth of the binary tree: 3  
PS F:\java>
```

## 18. Maximum Depth or Height of Binary Tree: Code:

```
1 class TreeNode {  
2     int value;  
3     TreeNode left;  
4     TreeNode right;  
5  
6     TreeNode(int value) {  
7         this.value = value;  
8         this.left = null;  
9         this.right = null;  
10    }  
11 }  
12  
13 public class BinaryTree {  
14     public int maxDepth(TreeNode root) {  
15         if (root == null) {  
16             return 0;  
17         } else {  
18  
19             int leftDepth = maxDepth(root.left);  
20             int rightDepth = maxDepth(root.right);  
21  
22  
23             return Math.max(leftDepth, rightDepth) + 1;  
24         }  
25     }  
26  
27     public static void main(String[] args) {  
28  
29         TreeNode root = new TreeNode(1);  
30         root.left = new TreeNode(2);  
31         root.right = new TreeNode(3);  
32         root.left.left = new TreeNode(4);  
33         root.left.right = new TreeNode(5);  
34  
35         BinaryTree tree = new BinaryTree();  
36         System.out.println("Maximum depth of the binary tree: " + tree.maxDepth(root));  
37     }  
38 }
```

## Output:

```
Maximum Depth or Height of Binary Tree: 4
```