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

Output:

```
C:\Users\pugaz\OneDrive\Desktop\jlpt>javac Main.java

C:\Users\pugaz\OneDrive\Desktop\jlpt>java Main.java
Enter the size of the array: 9
Enter the elements of the array: -2 1 -3 4 -1 2 1 -5 4
Maximum Subarray Sum: 6

C:\Users\pugaz\OneDrive\Desktop\jlpt>
```

Complexity: O(n)

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

Code:

```
import java.util.Scanner;
public class Product {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int[] nums = new int[n];
      for (int i = 0; i < n; i++) {
         nums[i] = sc.nextInt();
      }
      if (nums.length == 1) {
         System.out.println(nums[0]);
         return;
      }
      int max = Integer.MIN_VALUE;
      for (int i = 0; i < nums.length; i++) {
         int product = 1;
         for (int j = i; j < nums.length; j++) {
            product *= nums[j];
            if (product > max) {
               max = product;
            }
         }
      }
      System.out.println(max);
      sc.close();
   }
}
```

Output:

```
C:\Users\pugaz\OneDrive\Desktop\jlpt>javac Product.java

C:\Users\pugaz\OneDrive\Desktop\jlpt>java Product.java
6
-2 6 -3 -10 0 2
180

C:\Users\pugaz\OneDrive\Desktop\jlpt>
```

Complexity: O(n^2)

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.Scanner;
public class Array {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int[] nums = new int[n];
      for (int i = 0; i < n; i++) {
         nums[i] = sc.nextInt();
      }

      int target = sc.nextInt();

      int low = 0, high = nums.length - 1;
      int result = -1;

      while (low <= high) {
         int mid = (low + high) / 2;

         if (nums[mid] == target) {
            result = mid;
            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;
            }
         }
      }

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

Output:

```
C:\Windows\System32\cmd.e

Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pugaz\OneDrive\Desktop\jlpt>javac Array.java

C:\Users\pugaz\OneDrive\Desktop\jlpt>java Array.java
7
4 5 6 7 0 1 2
0
4

C:\Users\pugaz\OneDrive\Desktop\jlpt>
```

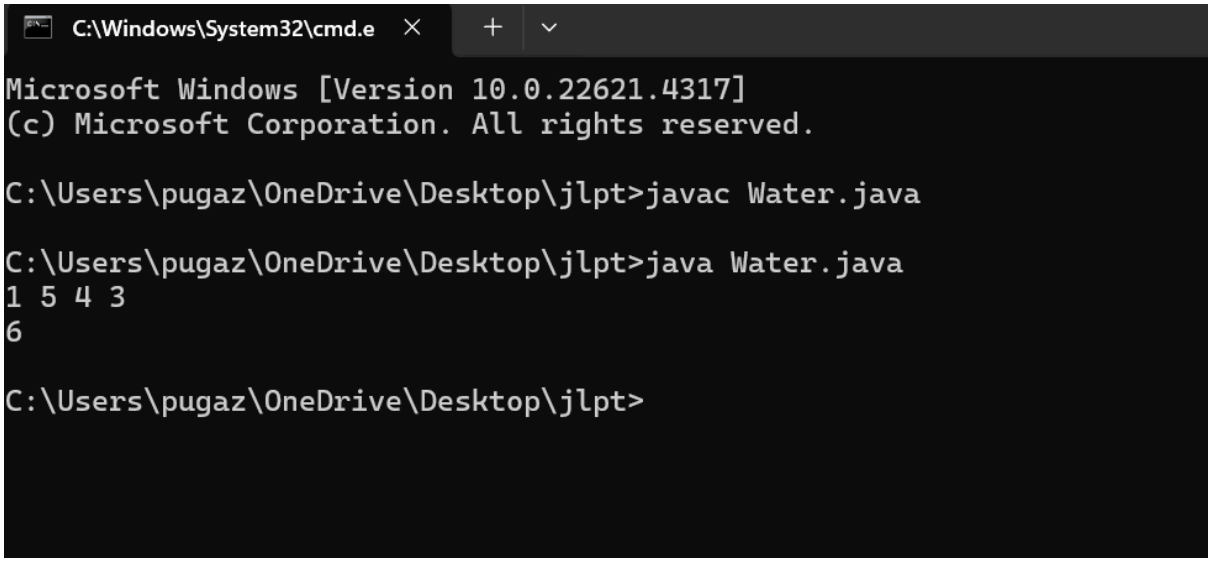
Complexity: O(logn)

4. Container with Most Water

Code:

```
import java.util.Scanner;
import java.util.List;
import java.util.ArrayList;
public class Water {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String[] input = sc.nextLine().split(" ");
      List<Integer> height = new ArrayList<>();
      for (String s : input) {
         height.add(Integer.parseInt(s));
      }
      int left = 0;
      int right = height.size() - 1;
      int maxArea = 0;
      while (left < right) {
         int currentArea = Math.min(height.get(left), height.get(right)) * (right - left);
         maxArea = Math.max(maxArea, currentArea);
         if (height.get(left) < height.get(right)) {
            left++;
         } else {
            right--;
         }
      }
      System.out.println(maxArea);
      sc.close();
   }
}
```

Output:



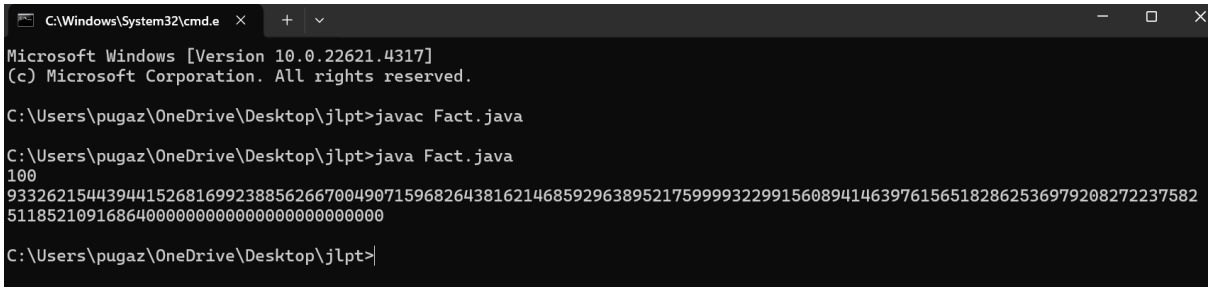
Complexity: O(n)

5. Find the Factorial of a large number

Code:

```
import java.util.Scanner;
import java.math.BigInteger;
public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int a = sc.nextInt();
      BigInteger  b= BigInteger.ONE;
      for(int i=1;i<=a ;i++){
         b=b.multiply(BigInteger.valueOf(i));
      }
      System.out.println(b);
   }
}
```

Output:



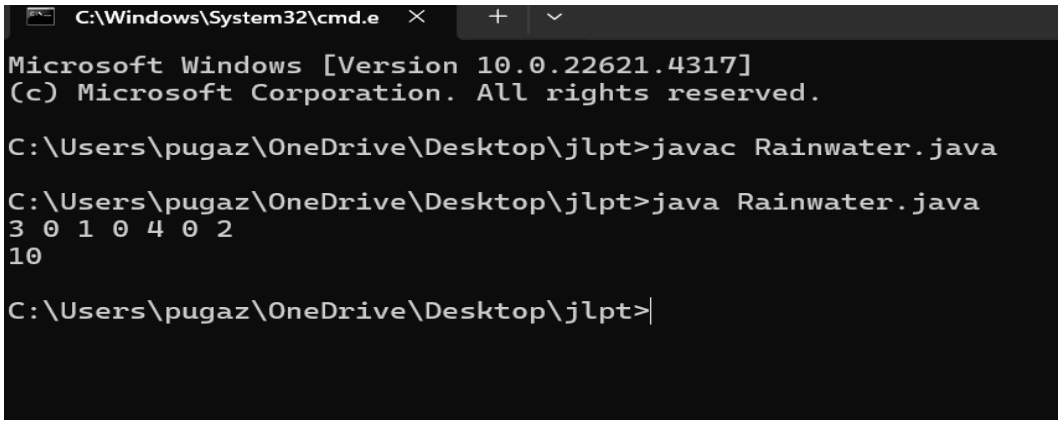
Complexity: O(n)

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.Scanner;
public class Rainwater {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String[] input = sc.nextLine().split(" ");
      int n = input.length;
      int[] height = new int[n];
      for (int i = 0; i < n; i++) {
         height[i] = Integer.parseInt(input[i]);
      }
      if (n == 0) {
         System.out.println(0);
         sc.close();
         return;
      }
      int[] leftMax = new int[n];
      int[] rightMax = new int[n];
      leftMax[0] = height[0];
      for (int i = 1; i < n; i++) {
         leftMax[i] = Math.max(leftMax[i - 1], height[i]);
      }
      rightMax[n - 1] = height[n - 1];
      for (int i = n - 2; i >= 0; i--) {
         rightMax[i] = Math.max(rightMax[i + 1], height[i]);
      }
      int waterTrapped = 0;
      for (int i = 0; i < n; i++) {
         waterTrapped += Math.min(leftMax[i], rightMax[i]) - height[i];
      }
      System.out.println(waterTrapped);
      sc.close();
   }
}
```

Output:



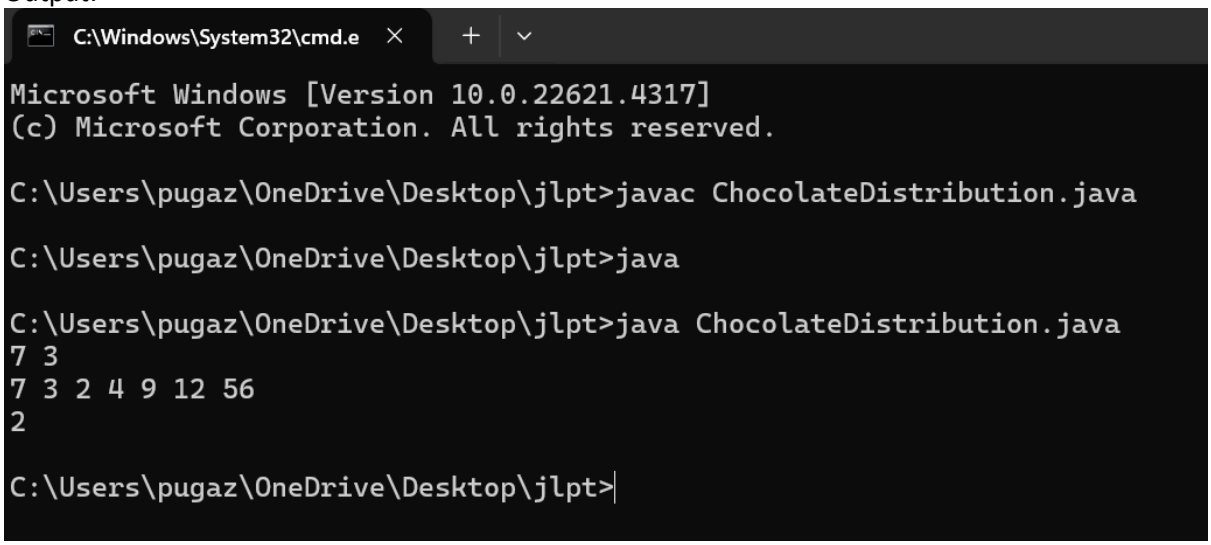
Complexity: O(n)

7. Chocolate Distribution Problem

Code:

```
import java.util.Arrays;
import java.util.Scanner;
public class ChocolateDistribution {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int m = sc.nextInt();
      if (n < m) {
         System.out.println("Not enough packets to distribute to all students.");
         return;
      }
      int[] arr = new int[n];
      for (int i = 0; i < n; i++) {
         arr[i] = sc.nextInt();
      }
      Arrays.sort(arr);
      int minDifference = Integer.MAX_VALUE;
      for (int i = 0; i + m - 1 < n; i++) {
         int currentDifference = arr[i + m - 1] - arr[i];
         minDifference = Math.min(minDifference, currentDifference);
      }
      System.out.println(minDifference);
      sc.close();
   }
}
```

Output:



Complexity: O(n logn)

8. Merge Overlapping Intervals

Code:

```
import java.util.*;
public class MergeIntervals {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[][] intervals = new int[n][2];
        for (int i = 0; i < n; i++) {
            intervals[i][0] = sc.nextInt();
            intervals[i][1] = sc.nextInt();
        }
        int[][] result = merge(intervals);
        for (int[] interval : result) {
            System.out.println(interval[0] + " " + interval[1]);
        }
        sc.close();
    }
    public static int[][] merge(int[][] arr) {
        Arrays.sort(arr, (a, b) -> Integer.compare(a[0], b[0]));
        List<int[]> merged = new ArrayList<>();
        int[] prev = arr[0];
        for (int i = 1; i < arr.length; i++) {
            int[] current = arr[i];
            if (current[0] <= prev[1]) {
                prev[1] = Math.max(prev[1], current[1]);
            } else {
                merged.add(prev);
                prev = current;
            }
        }
        merged.add(prev);
        return merged.toArray(new int[merged.size()][]);
    }
}
```

Output:

```
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pugaz\OneDrive\Desktop\jlpt>javac MergeIntervals.java

C:\Users\pugaz\OneDrive\Desktop\jlpt>java MergeIntervals.java
4
1 3
2 4
6 8
9 10
1 4
6 8
9 10

C:\Users\pugaz\OneDrive\Desktop\jlpt>
```

Complecity: O(n log n)

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 BooleanMatrix {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int m = sc.nextInt();
      int n = sc.nextInt();
      int[][] mat = new int[m][n];
      for (int i = 0; i < m; i++) {
         for (int j = 0; j < n; j++) {
            mat[i][j] = sc.nextInt();
         }
      }
      boolean[] rowFlags = new boolean[m];
      boolean[] colFlags = new boolean[n];
      for (int i = 0; i < m; i++) {
         for (int j = 0; j < n; j++) {
            if (mat[i][j] == 1) {
               rowFlags[i] = true;
               colFlags[j] = true;
            }
         }
      }
      for (int i = 0; i < m; i++) {
         for (int j = 0; j < n; j++) {
            if (rowFlags[i] || colFlags[j]) {
               mat[i][j] = 1;
            }
         }
      }
      for (int i = 0; i < m; i++) {
         for (int j = 0; j < n; j++) {
            System.out.print(mat[i][j] + " ");
         }
         System.out.println();
      }
      sc.close();
   }
}
```

Output:

```
C:\Windows\System32\cmd.e

Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pugaz\OneDrive\Desktop\jlpt>javac BooleanMatrix.java

C:\Users\pugaz\OneDrive\Desktop\jlpt>java BooleanMatrix.java
3 4
1 0 0 1
0 0 1 0
0 0 0 0
1 1 1 1
1 1 1 1
1 0 1 1

C:\Users\pugaz\OneDrive\Desktop\jlpt>
```

Complexity: O(m*n)

10. Print a given matrix in spiral form

Code:

```
import java.util.*;
public class Solution {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int m = sc.nextInt();
      int n = sc.nextInt();
      int[][] mat = new int[m][n];
      for (int i = 0; i < m; i++) {
         for (int j = 0; j < n; j++) {
            mat[i][j] = sc.nextInt();
         }
      }
      System.out.println(spiralOrder(mat));
      sc.close();
   }
   public static List<Integer> spiralOrder(int[][] mat) {
      int r = mat.length, c = mat[0].length, row = 0, col = -1, dir = 1;
      List<Integer> res = new ArrayList<>();
      while (r > 0 && c > 0) {
         for (int i = 0; i < c; i++) {
            col += dir;
            res.add(mat[row][col]);
         }
         r--;
         for (int i = 0; i < r; i++) {
            row += dir;
            res.add(mat[row][col]);
         }
         c--;
         dir *= -1;
      }
      return res;
   }
}
```

Output:

```
C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pugaz\OneDrive\Desktop\jlpt>javac Solution.java

C:\Users\pugaz\OneDrive\Desktop\jlpt>java Solution.java
4 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]

C:\Users\pugaz\OneDrive\Desktop\jlpt>
```

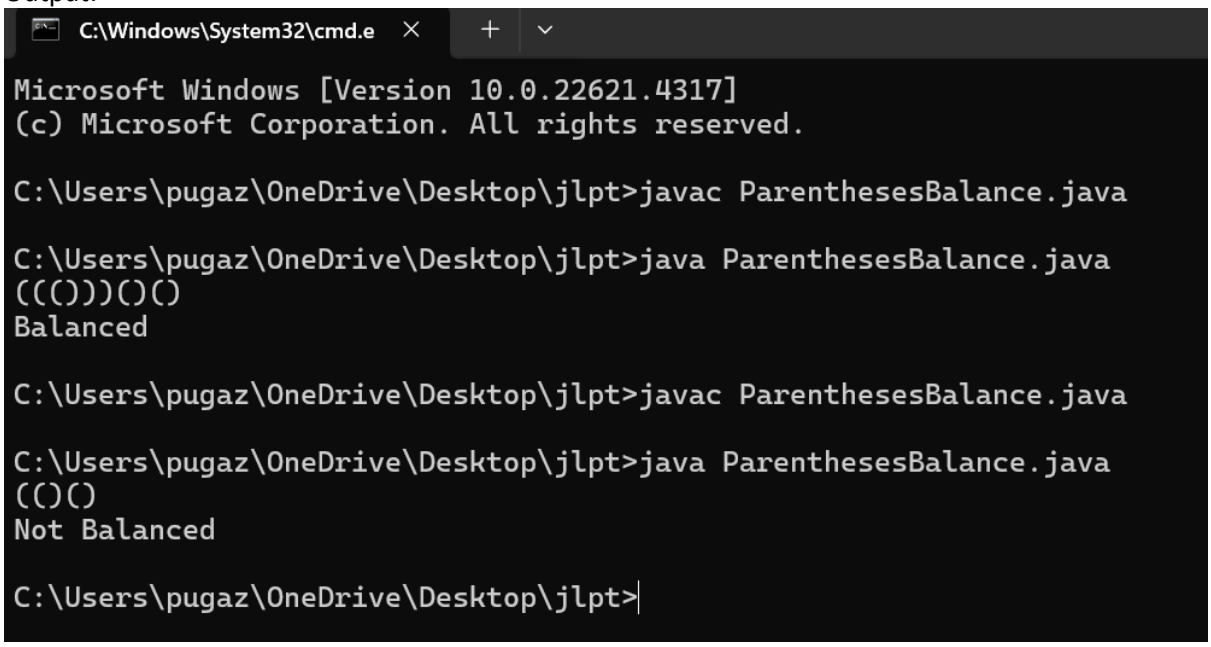
Complexity: O(m*n)

13. Check if given Parentheses expression is balanced or not

Code:

```
import java.util.Scanner;
public class ParenthesesBalance {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String str = sc.nextLine();
      int balance = 0;
      for (char c : str.toCharArray()) {
         if (c == '(') {
            balance++;
         } else if (c == ')') {
            balance--;
         }
         if (balance < 0) {
            System.out.println("Not Balanced");
            return;
         }
      }
      if (balance == 0) {
         System.out.println("Balanced");
      } else {
         System.out.println("Not Balanced");
      }
      sc.close();
   }
}
```

Output:



Complexity: O(n)

14. Check if two Strings are Anagrams of each other

Code:

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

class Anagram {
   public static boolean isAnagram(String x, String y) {
      char[] p = x.toCharArray();
      char[] q = y.toCharArray();

      Arrays.sort(p);
      Arrays.sort(q);

      return Arrays.equals(p, q);
   }

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String str1 = sc.nextLine();
      String str2 = sc.nextLine();

      boolean result = isAnagram(str1, str2);

      System.out.println(result);

      sc.close();
   }
}
```

Output:

```
C:\Users\pugaz\OneDrive\Desktop\jlpt>javac Anagram.java

C:\Users\pugaz\OneDrive\Desktop\jlpt>java Anagram.java
geeks
kseeg
true

C:\Users\pugaz\OneDrive\Desktop\jlpt>
```

Complexity: O(n log n)

15. Longest Palindromic Substring

Code:

```
import java.util.Scanner;
public class palindrome {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter the string:");
      String str = scanner.nextLine();
      if (str.length() <= 1) {
         System.out.println(str);
         return;
      }
      int maxLength = 1;
      String maxSubstring = str.substring(0, 1);
      for (int i = 0; i < str.length(); i++) {
         for (int j = i + maxLength; j <= str.length(); j++) {
            if (j - i > maxLength && isPalindrome(str.substring(i, j))) {
               maxLength = j - i;
               maxSubstring = str.substring(i, j);
            }
         }
      }
      System.out.println("Longest Palindromic Substring: " + maxSubstring);
      scanner.close();
   }
   private static boolean isPalindrome(String str) {
      int leftIndex = 0;
      int rightIndex = str.length() - 1;
      while (leftIndex < rightIndex) {
         if (str.charAt(leftIndex) != str.charAt(rightIndex)) {
            return false;
         }
         leftIndex++;
         rightIndex--;
      }
      return true;
   }
}
```

Output:

```
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pugaz\OneDrive\Desktop\jlpt>javac palindrome.java

C:\Users\pugaz\OneDrive\Desktop\jlpt>java palindrome.java
Enter the string:
forgeeksskeegfor
Longest Palindromic Substring: geeksskeeg

C:\Users\pugaz\OneDrive\Desktop\jlpt>
```

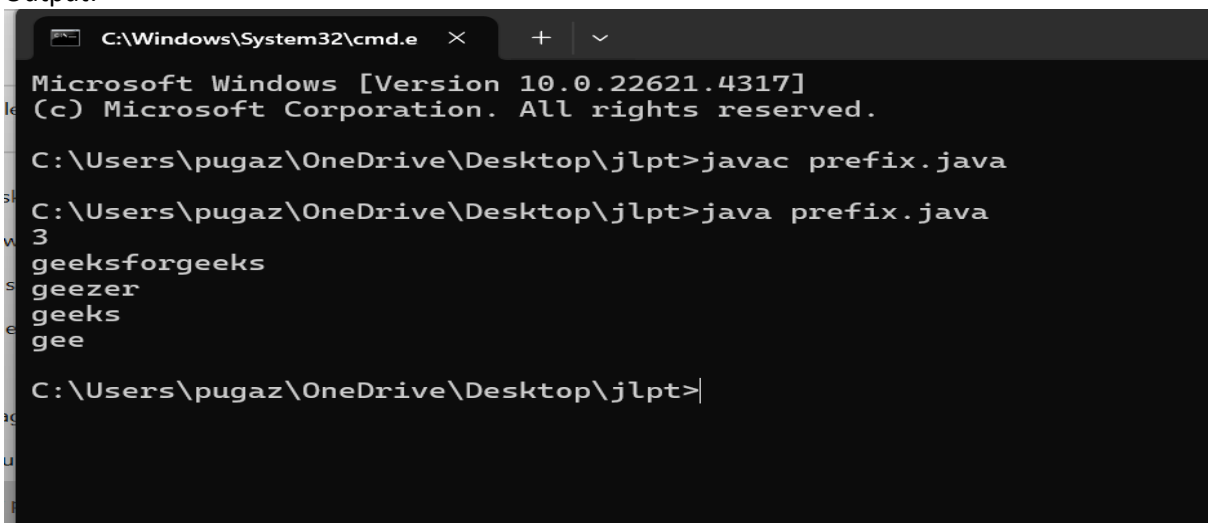
Complexity: O(n^2)

16. Longest Common Prefix using Sorting

Code:

```
import java.util.Scanner;
import java.util.Arrays;
class Prefix {
   public String longestCommonPrefix(String[] arr) {
      StringBuilder result = new StringBuilder();
      Arrays.sort(arr);
      String first = arr[0];
      String last = arr[arr.length - 1];
      for (int i = 0; i < Math.min(first.length(), last.length()); i++) {
         if (first.charAt(i) != last.charAt(i)) {
            return result.toString();
         }
         result.append(first.charAt(i));
      }
      return result.toString();
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      sc.nextLine();
      String[] arr = new String[n];
      for (int i = 0; i < n; i++) {
         arr[i] = sc.nextLine();
      }
      Prefix prefix = new Prefix();
      String result = prefix.longestCommonPrefix(arr);
      System.out.println(result);
      sc.close();
   }
}
```

Output:



Complexity: O(n log n * m)

17. Delete middle element of a stack

Code:

```
import java.util.Stack;
import java.util.Scanner;
public class Middle {
   public static void deleteMiddle(Stack<Integer> stack, int size, int current) {
      if (current == size / 2) {
         stack.pop();
         return;
      }
      int top = stack.pop();
      deleteMiddle(stack, size, current + 1);
      stack.push(top);
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      Stack<Integer> stack = new Stack<>();
      for (int i = 0; i < n; i++) {
         stack.push(sc.nextInt());
      }
      int size = stack.size();
      deleteMiddle(stack, size, 0);
      System.out.println(stack);
      sc.close();
   }
}
```

Output:

```
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pugaz\OneDrive\Desktop\jlpt>javac Middle.java

C:\Users\pugaz\OneDrive\Desktop\jlpt>java Middle.java
5
1 2 3 4 5
[1, 2, 4, 5]

C:\Users\pugaz\OneDrive\Desktop\jlpt>
```

Complexity: O(n)

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

Code:

```
import java.util.Scanner;
import java.util.Stack;
public class Next {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int[] arr = new int[n];

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

Output:

```
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pugaz\OneDrive\Desktop\jlpt>javac next.java

C:\Users\pugaz\OneDrive\Desktop\jlpt>java next.java
4
13 7 6 12
-1 12 12 -1
C:\Users\pugaz\OneDrive\Desktop\jlpt>
```

Complexity: O(n)

19. Print Right View of a Binary Tree

Code:

```
import java.util.*;
public class Tree {
    public List<Integer> rightSideView(TreeNode root) {
        List<Integer> result = new ArrayList<>();
        exploreRightSide(root, result, 0);
        return result;
    }
    private void exploreRightSide(TreeNode currentNode, List<Integer> result, int depth) {
        if (currentNode == null) {
            return;
        }
        if (depth == result.size()) {
            result.add(currentNode.val);
        }
        exploreRightSide(currentNode.right, result, depth + 1);
        exploreRightSide(currentNode.left, result, depth + 1);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int numberOfNodes = sc.nextInt();
        TreeNode root = createTree(sc, numberOfNodes);
        Tree tree = new Tree();
        List<Integer> rightSide = tree.rightSideView(root);
        System.out.println(rightSide);
    }
    private static TreeNode createTree(Scanner sc, int numberOfNodes) {
        if (numberOfNodes == 0) {
            return null;
        }
        int rootValue = sc.nextInt();
        TreeNode root = new TreeNode(rootValue);
        Queue<TreeNode> queue = new LinkedList<>();
        queue.add(root);
        for (int i = 1; i < numberOfNodes; i++) {
            TreeNode currentNode = queue.poll();
            int leftValue = sc.nextInt();
            if (leftValue != -1) {
                currentNode.left = new TreeNode(leftValue);
                queue.add(currentNode.left);
            }
            int rightValue = sc.nextInt();
            if (rightValue != -1) {
                currentNode.right = new TreeNode(rightValue);
                queue.add(currentNode.right);
            }
        }
        return root;
```

```
    }
}
class TreeNode {
    int val;
    TreeNode left, right;
    TreeNode(int x) { val = x; }
}
```

Output:

```
C:\Users\pugaz\OneDrive\Desktop\jlpt>javac Tree.java

C:\Users\pugaz\OneDrive\Desktop\jlpt>java Tree.java
5
1
2
3
-1
-1
4
5
-1
-1
[1, 3, 5]

C:\Users\pugaz\OneDrive\Desktop\jlpt>
```

Complexity: O(n)

20. Maximum Depth or Height of Binary Tree

Code:

```
import java.util.Scanner;
class Depth {
   public int maxDepth(TreeNode root) {
      if (root == null) {
         return 0;
      }
      int a = maxDepth(root.left);
      int b = maxDepth(root.right);
      return Math.max(a, b) + 1;
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the number of nodes: ");
      int n = sc.nextInt();
      TreeNode root = null;
      Depth depth = new Depth();
      System.out.println("Enter the values for the nodes: ");
      for (int i = 0; i < n; i++) {
         int value = sc.nextInt();
         root = insertNode(root, value);
      }
      System.out.println("Maximum depth of the tree: " + depth.maxDepth(root));
      sc.close();
   }
   public static TreeNode insertNode(TreeNode root, int value) {
      if (root == null) {
         return new TreeNode(value);
      }
      if (value < root.val) {
         root.left = insertNode(root.left, value);
      } else {
         root.right = insertNode(root.right, value);
      }
      return root;
   }
}
class TreeNode {
   int val;
   TreeNode left;
   TreeNode right;
   TreeNode(int val) {
      this.val = val;
   }
}
```

Output:

```
C:\Windows\System32\cmd.e

Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pugaz\OneDrive\Desktop\jlpt>javac Depth.java

C:\Users\pugaz\OneDrive\Desktop\jlpt>java Depth.java
Enter the number of nodes: 5
Enter the values for the nodes:
12 8 5 11 18
Maximum depth of the tree: 3

C:\Users\pugaz\OneDrive\Desktop\jlpt>
```

Complexity: O(n)