CATHERINE JESSIE J
22CS035 - CSE

1) Maximum Subarray Sum – Kadane"s Algorithm:

```
class Maximum_subarray {
    public static void main(String[] args) {
        int[] arr1 = {2, 3, -8, 7, -1, 2, 3};
        int[] arr2 = {-2, -4};
        int[] arr3 = {5, 4, 1, 7, 8};

        max_subarray(arr1);
        max_subarray(arr2);
        max_subarray(arr3);
    }

    public static void max_subarray(int[] arr){
        int maxsum = Integer.MIN_VALUE;
        int currsum = 0;

        for(int i=0; i<arr.length; i++){
            currsum += arr[i];
            if(maxsum<currsum){
                maxsum = currsum;
            }
            if(currsum<0){
                currsum = 0;
            }
        }
        System.out.println("MAXIMUM SUBARRAY SUM: "+maxsum);

    }
}
```

OUTPUT:

```
java -cp /tmp/j0KCBPproN/Maximum_subarray
MAXIMUM SUBARRAY SUM: 11
MAXIMUM SUBARRAY SUM: -2
MAXIMUM SUBARRAY SUM: 25
```

Time complexity : O(n)

2) Maximum Product Subarray

```
class MaximumProductsubarray {
    public static void main(String[] args) {
        int[] arr  = {-2, 6, -3, -10, 0, 2};
```

```
        max_subarray(arr);
    }

    public static void max_subarray(int[] arr){
        int maxprod = Integer.MIN_VALUE;
        int preprod = 1;
        int suffprod = 1;

        for(int i=0; i<arr.length; i++){
            preprod *= arr[i];
            suffprod *= arr[arr.length-i-1];
            maxprod = Math.max(maxprod,Math.max(preprod,suffprod));
            if(preprod==0){
                preprod = 1;
            }
            if(suffprod==0){
                suffprod = 1;
            }
        }
        System.out.println("MAXIMUM SUBARRAY PRODUCT: "+maxprod);

    }
}
```

OUTPUT:

```
java -cp /tmp/MkHqT8XNsT/MaximumProductsubarray
MAXIMUM SUBARRAY SUM: 180

=== Code Execution Successful ===
```

Time Complexity : O(n)

3) Search in a sorted and rotated Array

```
class SearchSortedArray {
    public static void main(String[] args) {
        int[] arr1  = {4, 5, 6, 7, 0, 1, 2};
        int key1 = 0;

        int[] arr2  = {4, 5, 6, 7, 0, 1, 2};
        int key2 = 3;

        int[] arr3  = {50, 10, 20, 30, 40};
        int key3 = 10;

        ans(arr1,key1);
```

```
        ans(arr2,key2);
        ans(arr3,key3);
    }

    public static void ans(int[] arr, int key){
        int low = 0;
        int high = arr.length-1;

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

            if(arr[mid]==key){
                System.out.println(mid);
                return;
            }

            if(arr[low]<=arr[mid]){
                if(arr[low]<=key && key<arr[mid]){
                    high = mid-1;
                }else{
                    low = mid+1;
                }
            }else{
                if(arr[mid]<=key && key<arr[high]){
                    low = mid+1;
                }else{
                    high = mid-1;
                }

            }
        }
        System.out.println(-1);


    }
}
```

OUTPUT:

```
java -cp /tmp/ebr7ghRKDT/SearchSortedArray
4
-1
1

=== Code Execution Successful ===
```

TimeComplexity : O(n)

4) Container with Most Water

```
class ContainerWithWater {
    public static void main(String[] args) {
        int[] arr1  = {1,5,4,3};
        int[] arr2  = {3,1,2,4,5};

        ans(arr1);
        ans(arr2);
    }

    public static void ans(int[] arr){
        int low = 0;
        int high = arr.length-1;
        int maxarea = 0;

        while(low<high){
            int h = Math.min(arr[low],arr[high]);
            int w = high-low;
            maxarea = Math.max(maxarea,h*w);

            if (arr[low]<arr[high]){
                low++;
            }else{
                high--;
            }
        }
        System.out.println(maxarea);


    }
}
```

OUTPUT:
java -cp /tmp/xQqXNVktiF/SearchSortedArray
6
12

=== Code Execution Successful ===

TimeComplexity : O(n)

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

```
import java.math.BigInteger;
class FactorialCalculator {
    public static void main(String[] args) {
        int n1 = 100;
        int n2 = 50;

        System.out.println(ans(n1));
        System.out.println(ans(n2));
    }

    public static BigInteger ans(int n){
        if (n == 0 || n == 1) {
            return BigInteger.ONE;
        }
        return BigInteger.valueOf(n).multiply(ans(n-1));

    }
}
```

OUTPUT:

java -cp /tmp/e5jpKaQpZh/FactorialCalculator
93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000
30414093201713378043612608166064768844377641568960512000000000000

=== Code Execution Successful ===

TimeComplexity : O(n)

6) **Trapping water**

```
class TrappingWater {
    public static void main(String[] args) {
        int[] arr1  = {3, 0, 1, 0, 4, 0, 2};
        int[] arr2  = {3, 0, 2, 0, 4};
        int[] arr3  = {1, 2, 3, 4};
        int[] arr4  = {10, 9, 0, 5};

        ans(arr1);
        ans(arr2);
        ans(arr3);
        ans(arr4);
```

```
    }

    public static void ans(int[] arr){
        int[] lm = new int[arr.length];
        int[] rm = new int[arr.length];
        lm[0] = arr[0];
        rm[arr.length-1] = arr[arr.length-1];
        for (int i=1; i<arr.length; i++){
            lm[i] = Math.max(lm[i-1],arr[i]);
        }
        for (int i=arr.length-2; i>=0; i--){
            rm[i] = Math.max(rm[i+1],arr[i]);
        }

        int c = 0;
        for (int i=0; i<arr.length; i++){
            c+=Math.min(lm[i],rm[i])-arr[i];
        }
        System.out.println(c);
    }
}
```

OUTPUT:

```
java -cp /tmp/em1sKbepoZ/TrappingWater
10
7
0
5

=== Code Execution Successful ===
```

TimeComplexity : O(n)

7) Chocolate Distribution

```
import java.util.Arrays;

public class ChocolateDistribution {

    public static void main(String[] args) {
        int[] arr1 = {7, 3, 2, 4, 9, 12, 56};
        int m1 = 3;
        ans(arr1, m1);

        int[] arr2 = {7, 3, 2, 4, 9, 12, 56};
        int m2 = 5;
        ans(arr2, m2);
```

```
    }

    public static void ans(int[] arr, int m) {
        if (m == 0 || arr.length == 0) {
            System.out.println(0);
        }
        int n = arr.length;
        if (n < m) {
            System.out.println(-1);
        }
        Arrays.sort(arr);

        int minDiff = Integer.MAX_VALUE;

        for (int i = 0; i <= n - m; i++) {
            int diff = arr[i + m - 1] - arr[i];
            minDiff = Math.min(minDiff, diff);
        }

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

OUTPUT:

java -cp /tmp/Ngu93cQfES/ChocolateDistribution

2

7

=== Code Execution Successful ===

TimeComplexity : O(n)

8) Merge Overlapping intervals

```
import java.util.*;

class MergeIntervals {
    public static void main(String[] args) {
        ans(new int[][]{{1, 3}, {2, 4}, {6, 8}, {9, 10}});
        ans(new int[][]{{7, 8}, {1, 5}, {2, 4}, {4, 6}});
    }

    public static void ans(int[][] intervals) {
        Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));

        List<int[]> result = new ArrayList<>();
```

```
        for (int[] interval : intervals) {
            if (result.isEmpty() || result.get(result.size() - 1)[1] < interval[0]) {
                result.add(interval);
            } else {
                result.get(result.size() - 1)[1] = Math.max(result.get(result.size() - 1)[1], interval[1]);
            }
        }

        for (int[] interval : result) {
            System.out.print("[" + interval[0] + ", " + interval[1] + "] ");
        }
        System.out.println();
    }
}
```

OUTPUT:
java -cp /tmp/U8vh5S2D52/MergeIntervals
[1, 4] [6, 8] [9, 10]
[1, 6] [7, 8]

=== Code Execution Successful ===

TimeComplexity : O(n)

9) Boolean Matrix

```
class BooleanMatrix {
    public static void main(String[] args) {
        ans(new int[][]{
            {1, 0},
            {0, 0}
        });

        ans(new int[][]{
            {0, 0, 0},
            {0, 0, 1}
        });

        ans(new int[][]{
            {1, 0, 0, 1},
            {0, 0, 1, 0},
            {0, 0, 0, 0}
        });
    }

    public static void ans(int[][] mat) {
        int rows = mat.length;
```

```
        int cols = mat[0].length;

        boolean[] rowFlag = new boolean[rows];
        boolean[] colFlag = new boolean[cols];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (mat[i][j] == 1) {
                    rowFlag[i] = true;
                    colFlag[j] = true;
                }
            }
        }

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (rowFlag[i] || colFlag[j]) {
                    mat[i][j] = 1;
                }
            }
        }

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print(mat[i][j] + " ");
            }
            System.out.println();
        }
        System.out.println();
    }
}
```

OUTPUT:

```
java -cp /tmp/Qvcz8f0xJI/BooleanMatrix
1 1
1 0

0 0 1
1 1 1

1 1 1 1
1 1 1 1
1 0 1 1


=== Code Execution Successful ===
```

TimeComplexity : O(m*n)

10) Print a given matrix in spiral

```
class SpiralMatrix {
    public static void main(String[] args) {
        int[][] matrix1 = {
            {1, 2, 3, 4},
            {5, 6, 7, 8},
            {9, 10, 11, 12},
            {13, 14, 15, 16}
        };

        int[][] matrix2 = {
            {1, 2, 3, 4, 5, 6},
            {7, 8, 9, 10, 11, 12},
            {13, 14, 15, 16, 17, 18}
        };

        ans(matrix1);
        ans(matrix2);
    }

    public static void ans(int[][] matrix) {
        int top = 0, left = 0;
        int bottom = matrix.length - 1, right = matrix[0].length - 1;

        while (top <= bottom && left <= right) {
            for (int i = left; i <= right; i++) {
                System.out.print(matrix[top][i] + " ");
            }
            top++;

            for (int i = top; i <= bottom; i++) {
                System.out.print(matrix[i][right] + " ");
            }
            right--;

            if (top <= bottom) {
                for (int i = right; i >= left; i--) {
                    System.out.print(matrix[bottom][i] + " ");
                }
                bottom--;
            }

            if (left <= right) {
                for (int i = bottom; i >= top; i--) {
                    System.out.print(matrix[i][left] + " ");
```

```
            }
            left++;
        }
    }
    System.out.println();
  }
}
```

OUTPUT:
java -cp /tmp/YF78tRfvUJ/SpiralMatrix
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11

=== Code Execution Successful ===

TimeComplexity : O(m*n)

11) Check if the given Parentheses expression is balanced or not

```
import java.util.Stack;
class CheckParantheses {
   public static void main(String[] args) {
      String str1 = "((()))()()";
      String str2 = "())((()))";

      ans(str1);
      ans(str2);
   }

   public static void ans(String str){
      Stack<Character> s = new Stack<>();

      for (char i : str.toCharArray()){
         if (i == '('){
            s.push(i);
         }else{
            if (s.size() == 0) {
               System.out.println("Not Balanced");
               return;
            }
            s.pop();
         }
      }
      if(s.size() == 0){
         System.out.println("Balanced");
      }else{
         System.out.println("Not Balanced");
```

```
        }

    }
}
```

OUTPUT:
java -cp /tmp/psMx8pjiEL/CheckParantheses
Balanced
Not Balanced

=== Code Execution Successful ===

TimeComplexity : O(n)

12) Check if two strings are Anagram

```
import java.util.HashMap;

class Anagram {
    public static void main(String[] args) {
        ans("geeks", "kseeg");
        ans("allergy", "allergic");
        ans("g", "g");
    }

    public static void ans(String s1, String s2) {
        if (s1.length() != s2.length()) {
            System.out.println("false");
            return;
        }

        HashMap<Character, Integer> h1 = new HashMap<>();

        for (char c : s1.toCharArray()) {
            h1.put(c, h1.getOrDefault(c, 0) + 1);
        }

        HashMap<Character, Integer> h2 = new HashMap<>();

        for (char c : s2.toCharArray()) {
            h2.put(c, h2.getOrDefault(c, 0) + 1);
        }

        if(h1.equals(h2)){
            System.out.println("true");
```

```
        }else{
            System.out.println("false");
            return;
        }
    }
}
```

OUTPUT:
java -cp /tmp/SOkUsEdCc1/Anagram
true
false
true

=== Code Execution Successful ===

TimeComplexity : O(n)

13) Longest palindromic substring

```
public class LongestPalindromic {

    public static void main(String[] args) {
        ans("forgeeksskeegfor");
        ans("Geeks");
        ans("abc");
        ans("");
    }

    public static void ans(String str) {
        if (str == null || str.length() == 0) {
            System.out.println("");
            return;
        }

        String result = "";

        for (int i = 0; i < str.length(); i++) {
            int l = i, r = i;
            while (l >= 0 && r < str.length() && str.charAt(l) == str.charAt(r)) {
                l--;
                r++;
            }
            String s1 = str.substring(l + 1, r);

            l = i;
```

```
            r = i + 1;
            while (l >= 0 && r < str.length() && str.charAt(l) == str.charAt(r)) {
                l--;
                r++;
            }
            String s2 = str.substring(l + 1, r);

            if (s1.length() > result.length()) {
                result = s1;
            }
            if (s2.length() > result.length()) {
                result = s2;
            }
        }

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

OUTPUT:
java -cp /tmp/8ym8VnfLFi/LongestPalindromic
geeksskeeg
ee
a

=== Code Execution Successful ===

TimeComplexity : O(n*n)

14) Longest common prefix using sorting

```
import java.util.Arrays;

public class LongestCommonPrefix {

    public static void main(String[] args) {
        ans(new String[]{"geeksforgeeks", "geeks", "geek", "geezer"});
        ans(new String[]{"hello", "world"});
    }

    public static void ans(String[] arr) {
        Arrays.sort(arr);

        String a = arr[0];
        String b = arr[arr.length - 1];
        StringBuilder prefix = new StringBuilder();
```

```
        int i = 0;
        while (i < a.length() && i < b.length() && a.charAt(i) == b.charAt(i)) {
            prefix.append(a.charAt(i));
            i++;
        }

        if (prefix.length() == 0) {
            System.out.println("-1");
        } else {
            System.out.println(prefix.toString());
        }
    }
}
```

OUTPUT:
java -cp /tmp/74Ptlsmsvg/LongestCommonPrefix
gee
-1

=== Code Execution Successful ===

TimeComplexity : O(nlogn*k)

15) Delete middle element of stack

```
import java.util.Stack;

class DeleteMiddleStack {

    public static void main(String[] args) {
        Stack<Integer> stack1 = new Stack<>();
        stack1.push(1);
        stack1.push(2);
        stack1.push(3);
        stack1.push(4);
        stack1.push(5);

        Stack<Integer> stack2 = new Stack<>();
        stack2.push(1);
        stack2.push(2);
        stack2.push(3);
        stack2.push(4);
        stack2.push(5);
        stack2.push(6);

        deleteMiddle(stack1);
```

```
        deleteMiddle(stack2);
    }

    public static void deleteMiddle(Stack<Integer> stack) {
        int size = stack.size();
        if (size == 0) {
            return;
        }

        int mid = size / 2;
        Stack<Integer> temp = new Stack<>();

        for (int i = 0; i < size; i++) {
            if (i == mid) {
                stack.pop();
            } else {
                temp.push(stack.pop());
            }
        }

        while (!temp.isEmpty()) {
            stack.push(temp.pop());
        }

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

OUTPUT:

```
java -cp /tmp/kdabBfB3cZ/DeleteMiddleStack
[1, 2, 4, 5]
[1, 2, 4, 5, 6]

=== Code Execution Successful ===
```

TimeComplexity : O(n)

16) Next greater element in a array of elements

```
class NextGreaterElement {
    public static void main(String[] args) {
        ans(new int[]{4, 5, 2, 25});
        System.out.println();
        ans(new int[]{13, 7, 6, 12});
    }

    public static void ans(int[] arr) {
```

```
        int[] n = new int[arr.length];
        n[arr.length - 1] = -1;

        for (int i = arr.length - 2; i >= 0; i--) {
            if (arr[i] < arr[i + 1]) {
                n[i] = arr[i + 1];
            } else {
                if(arr[i]<n[i+1]){
                    n[i] = n[i + 1];
                }else{
                    n[i] = -1;
                }

            }
        }

        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i] + " --> " + n[i]);
        }
    }
}
```

OUTPUT:

```
java -cp /tmp/MZWOd4Owsu/NextGreaterElement
4 --> 5
5 --> 25
2 --> 25
25 --> -1

13 --> -1
7 --> 12
6 --> 12
12 --> -1

=== Code Execution Successful ===
```

TimeComplexity : O(n)

17) Print right view of binary tree

```
import java.util.*;

class RightViewTree {
    static class TreeNode {
        int value;
        TreeNode left, right;
        TreeNode(int x) { value = x; left = right = null; }
```

```
    }

    public static void main(String[] args) {
        TreeNode root = new TreeNode(1);
        root.left = new TreeNode(2);
        root.right = new TreeNode(3);
        root.right.left = new TreeNode(4);
        root.right.right = new TreeNode(5);

        ans(root);
    }
    public static void ans(TreeNode root) {
        if (root == null) return;
        Queue<TreeNode> queue = new LinkedList<>();
        queue.add(root);

        while (!queue.isEmpty()) {
            int levelSize = queue.size();
            for (int i = 1; i <= levelSize; i++) {
                TreeNode current = queue.poll();
                if (i == levelSize) System.out.print(current.value + " ");
                if (current.left != null) queue.add(current.left);
                if (current.right != null) queue.add(current.right);
            }
        }
    }
}
```

OUTPUT:
java -cp /tmp/y3zkJUJMdH/ChocolateDistribution
1 3 5
=== Code Execution Successful ===

TimeComplexity : O(n)

18) Height of the binary tree

```
import java.util.LinkedList;
import java.util.Queue;

class BinaryTreeHeight {
    static class Node {
        int value;
        Node left, right;
        Node(int value) { this.value = value; }
    }
```

```
    public static void main(String[] args) {
        Node root = new Node(1);
        root.left = new Node(2);
        root.right = new Node(3);
        root.left.left = new Node(4);
        root.right.left = new Node(5);
        root.right.left.left = new Node(6);
        root.right.left.right = new Node(7);
        ans(root);
    }

    public static void ans(Node root) {
        if (root == null) {
            System.out.println("Height: 0");
            return;
        }
        Queue<Node> queue = new LinkedList<>();
        queue.add(root);
        int height = 0;

        while (!queue.isEmpty()) {
            int levelSize = queue.size();
            height++;
            for (int i = 0; i < levelSize; i++) {
                Node node = queue.poll();
                if (node.left != null) queue.add(node.left);
                if (node.right != null) queue.add(node.right);
            }
        }
        System.out.println("Height: " + height);
    }
}
```

OUTPUT:

java -cp /tmp/YADjqNJI5a/BinaryTreeHeight
Height: 4

=== Code Execution Successful ===

TimeComplexity : O(n)