DAY 6 DSA PRACTICE SET 1

9/11/2024

1. **Maximum Subarray Sum – Kadane‟s Algorithm:**

   Code:

```java
import java.util.*;

public class KadaneAlgorithm {
    public static int maxSubArray(int[] nums) {
        int maxSoFar = nums[0];
        int maxEndingHere = nums[0];
        for (int i = 1; i < nums.length; i++) {
            maxEndingHere = Math.max(nums[i], maxEndingHere + nums[i]);
            maxSoFar = Math.max(maxSoFar, maxEndingHere);
        }
        return maxSoFar;
    }

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

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

        int[] nums = new int[size];

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

        int maxSum = maxSubArray(nums);
        System.out.println("Maximum Subarray Sum: " + maxSum);

        scanner.close();
    }
}
```

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>
```

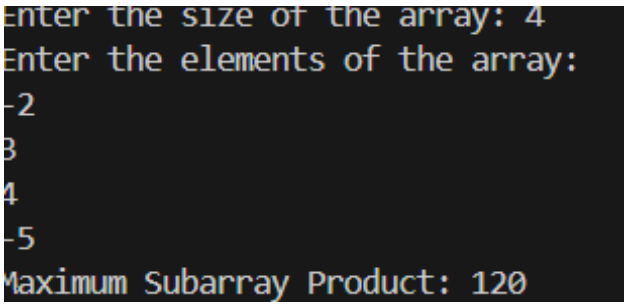
Time Complexity:

O(N)

2. **Maximum Product Subarray:**
   Code:

```java
import java.util.*;

public class MaximumProduct {

    public static int maxProduct(int[] nums) {
        if (nums.length == 0) return 0;

        int maxSoFar = nums[0];
        int maxEndingHere = nums[0];
        int minEndingHere = nums[0];

        for (int i = 1; i < nums.length; i++) {
            if (nums[i] < 0) {

                int temp = maxEndingHere;
                maxEndingHere = minEndingHere;
                minEndingHere = temp;
            }

            maxEndingHere = Math.max(nums[i], maxEndingHere * nums[i]);
            minEndingHere = Math.min(nums[i], minEndingHere * nums[i]);

            maxSoFar = Math.max(maxSoFar, maxEndingHere);
        }

        return maxSoFar;
    }


    public static int[] readInputArray() {

        try (Scanner scanner = new Scanner(System.in)) {
            System.out.print("Enter the size of the array: ");
            int size = scanner.nextInt();



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

            return nums;
        }
    }

    public static void main(String[] args) {
        int[] nums = readInputArray();
        int maxProduct = maxProduct(nums);
        System.out.println("Maximum Subarray Product: " + maxProduct);
    }
}
```

Output:



Time Complexity:
O(N)

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

Code:

```java
public class RotatedSortedArray {

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

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

            if (arr[mid] == key) {
                return mid;
            }

            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;
                }
            }
        }

        return -1;
    }

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

        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();

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

        System.out.print("Enter the key to search for: ");
        int key = scanner.nextInt();
        int result = searchInRotatedArray(arr, key);

        if (result != -1) {
            System.out.println("Index of key: " + result);
        } else {
            System.out.println("Key not found.");
        }

        scanner.close();
    }
}
```

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:

```java
import java.util.Scanner;

class ContainerWithMostWater {
    static int maxArea(int[] arr) {
        int n = arr.length;
        int left = 0;
        int right = n - 1;
        int area = 0;

        while (left < right) {
            area = Math.max(area, Math.min(arr[left], arr[right]) * (right - left));

            if (arr[left] < arr[right])
                left += 1;
            else
                right -= 1;
        }
        return area;
    }

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

        System.out.print("Enter the number of elements in 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();
        }

        System.out.println("The maximum area is: " + maxArea(arr));

        scanner.close();
    }
}

```

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:

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

public class Factorial {
    public static BigInteger factorial(int n) {
        BigInteger result = BigInteger.ONE;
        for (int i = 2; i <= n; i++) {
            result = result.multiply(BigInteger.valueOf(i));
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a number: ");
        int n = scanner.nextInt();

        BigInteger result = factorial(n);
        System.out.println("The factorial of " + n + " is: " + result);

        scanner.close();

    }
}
```

Output:

```
Enter a number:
100
The factorial of 100 is: 93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000
```

Time Complexity:
O(n)

6. Trapping Rainwater Problem:

Code:

```java
import java.util.Scanner;

class TrappingRainwater {

    static int findWater(int[] arr) {
        int n = arr.length;

        int[] left = new int[n];
        int[] right = new int[n];
        int res = 0;

        left[0] = arr[0];
        for (int i = 1; i < n; i++)
            left[i] = Math.max(left[i - 1], arr[i]);

        right[n - 1] = arr[n - 1];
        for (int i = n - 2; i >= 0; i--)
            right[i] = Math.max(right[i + 1], arr[i]);

        for (int i = 1; i < n - 1; i++) {
            int minOf2 = Math.min(left[i], right[i]);
            if (minOf2 > arr[i]) {
                res += minOf2 - arr[i];
            }
        }

        return res;
    }

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

        System.out.print("Enter the number of elements: ");
        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();
        }

        int result = findWater(arr);
        System.out.println("The amount of trapped rainwater is: " + result);
        scanner.close();
    }
}

```

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:

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

class Chocolatedistribution {

    static int findMinDiff(int[] arr, int m) {
        int n = arr.length;

        Arrays.sort(arr);

        int minDiff = Integer.MAX_VALUE;

        for (int i = 0; i + m - 1 < n; i++) {
            int diff = arr[i + m - 1] - arr[i];
            if (diff < minDiff)
                minDiff = diff;
        }
        return minDiff;
    }

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

        System.out.print("Enter the number of elements in 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();
        }

        System.out.print("Enter the value of m: ");
        int m = scanner.nextInt();

        if (m > n) {
            System.out.println("m cannot be greater than the number of elements in the array.");
        } else {
            System.out.println("The minimum difference is: " + findMinDiff(arr, m));
        }

        scanner.close();
    }
}
```

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(nlog n)

8. Merge Overlapping Intervals:

Code:

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

class Mergeoverlapping {

    static List<int[]> mergeOverlap(int[][] arr) {
        Arrays.sort(arr, (a, b) -> Integer.compare(a[0], b[0]));

        List<int[]> res = new ArrayList<>();
        res.add(new int[]{arr[0][0], arr[0][1]});

        for (int i = 1; i < arr.length; i++) {
            int[] last = res.get(res.size() - 1);
            int[] curr = arr[i];

            if (curr[0] <= last[1])
                last[1] = Math.max(last[1], curr[1]);
            else
                res.add(new int[]{curr[0], curr[1]});
        }

        return res;
    }

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

        System.out.print("Enter the number of intervals: ");
        int n = scanner.nextInt();
        int[][] arr = new int[n][2]; // Create an array to hold the intervals

        System.out.println("Enter the intervals (start and end):");
        for (int i = 0; i < n; i++) {
            System.out.print("Interval " + (i + 1) + ": ");
            arr[i][0] = scanner.nextInt(); // Start of the interval
            arr[i][1] = scanner.nextInt(); // End of the interval
        }

        List<int[]> res = mergeOverlap(arr);

        System.out.println("Merged intervals:");
        for (int[] interval : res)
            System.out.println(interval[0] + " " + interval[1]);

        scanner.close();
    }
}
```

Output:

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

Time Complexity:
O(nlogn)

9. **A Boolean Matrix Question:**
Code:

```java
import java.util.Scanner;

class BooleanMatrix {
    public static void modifyMatrix(int mat[][], int R, int C) {
        int row[] = new int[R];
        int col[] = new int[C];
        int i, j;

        for (i = 0; i < R; i++)
            row[i] = 0;

        for (i = 0; i < C; i++)
            col[i] = 0;

        for (i = 0; i < R; i++) {
            for (j = 0; j < C; j++) {
                if (mat[i][j] == 1) {
                    row[i] = 1;
                    col[j] = 1;
                }
            }
        }

        for (i = 0; i < R; i++)
            for (j = 0; j < C; j++)
                if (row[i] == 1 || col[j] == 1)
                    mat[i][j] = 1;
    }


    public static void printMatrix(int mat[][], int R, int C) {
        int i, j;
        for (i = 0; i < R; i++) {
            for (j = 0; j < C; j++)
                System.out.print(mat[i][j] + " ");
            System.out.println();
        }
    }


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

        System.out.print("Enter the number of rows (R): ");
        int R = scanner.nextInt();
        System.out.print("Enter the number of columns (C): ");
        int C = scanner.nextInt();

        int mat[][] = new int[R][C];

        System.out.println("Enter the elements of the matrix (0 or 1):");
        for (int i = 0; i < R; i++) {
            for (int j = 0; j < C; j++) {
                mat[i][j] = scanner.nextInt();
            }
        }

        System.out.println("Matrix Initially:");
        printMatrix(mat, R, C);
        modifyMatrix(mat, R, C);
        System.out.println("Matrix after modification:");
        printMatrix(mat, R, C);

        scanner.close();
    }
}
```

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
```

Time complexity:
O(m*n)

10. Print a given matrix in spiral form:

Code:

```java
import java.util.Scanner;

public class Matrixspiral {

    public static void spiralPrint(int m, int n, int[][] a) {

        int top = 0, bottom = m - 1, left = 0, right = n - 1;

        while (top <= bottom && left <= right) {

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

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

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

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

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

        System.out.print("Enter the number of rows: ");
        int m = scanner.nextInt();
        System.out.print("Enter the number of columns: ");
        int n = scanner.nextInt();

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


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

        System.out.println("Spiral order of the matrix:");
        spiralPrint(m, n, matrix);

        scanner.close();
    }
}
```

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:

```java
import java.util.Scanner;

public class Parenthesis {

    public static String isBalanced(String str){
        int balance = 0;
        for (char ch : str.toCharArray()) {
            if (ch == '(') {
                balance++;
            } else if (ch == ')') {
                balance--;
            }

            if (balance < 0) {
                return "Not Balanced";
            }
        }

        return (balance == 0) ? "Balanced" : "Not Balanced";
    }

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

        System.out.print("Enter a parentheses expression: ");
        String str = scanner.nextLine();

        String result = isBalanced(str);
        System.out.println(result);

        scanner.close();
    }
}
```

Output:

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

Time complexity:

O(n^2)

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

Code:

```java
import java.util.HashMap;
import java.util.Scanner;

class Anagram {

    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.print("Enter the first string: ");
        String s1 = scanner.nextLine();

        System.out.print("Enter the second string: ");
        String s2 = scanner.nextLine();


        boolean result = areAnagrams(s1, s2);
        System.out.println(result ? "The strings are anagrams." : "The strings are not anagrams.");

        scanner.close();
    }
}
```

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:

```java
import java.util.Scanner;

public class Longestpallindrome {

    public static String longestPalindrome(String str) {
        int n = str.length();
        boolean[][] dp = new boolean[n][n];
        int start = 0, maxLength = 1;

        for (int i = 0; i < n; i++) {
            dp[i][i] = true;
        }

        for (int i = 0; i < n - 1; i++) {
            if (str.charAt(i) == str.charAt(i + 1)) {
                dp[i][i + 1] = true;
                start = i;
                maxLength = 2;
            }
        }

        for (int length = 3; length <= n; length++) {
            for (int i = 0; i < n - length + 1; i++) {
                int j = i + length - 1;

                if (str.charAt(i) == str.charAt(j) && dp[i + 1][j - 1]) {
                    dp[i][j] = true;
                    start = i;
                    maxLength = length;
                }
            }
        }

        return str.substring(start, start + maxLength);
    }

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

        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        String result = longestPalindrome(input);
        System.out.println("Longest Palindromic Substring: " + result);

        scanner.close();
    }
}
```

Output:

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

Time complexity:

O(n^2)

14. Longest Common Prefix using Sorting:

Code:

```java
import java.util.Scanner;

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


        System.out.print("Enter the number of strings: ");
        int n = scanner.nextInt();
        scanner.nextLine();

        String[] arr = new String[n];
        System.out.println("Enter the strings:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextLine();
        }

        String result = longestCommonPrefix(arr);
        if (result.isEmpty()) {
            System.out.println("-1");
        } else {
            System.out.println("Longest common prefix: " + result);
        }

        scanner.close();
    }

    public static String longestCommonPrefix(String[] arr) {
        if (arr == null || arr.length == 0) {
            return "";
        }

        String prefix = arr[0];
        for (int i = 1; i < arr.length; i++) {
            while (arr[i].indexOf(prefix) != 0) {
                prefix = prefix.substring(0, prefix.length() - 1);
                if (prefix.isEmpty()) {
                    return "";
                }
            }
        }
        return prefix;
    }
}
```

Output:

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

class
prefix
-1
```

15. Delete middle element of a stack:

Code:

```java
import java.util.Scanner;
import java.util.Stack;

public class DelateMiddle {

    public static void deleteMiddle(Stack<Integer> stack, int currentIndex, int size) {
        if (stack.isEmpty() || currentIndex == size / 2) {
            stack.pop();
            return;
        }

        int topElement = stack.pop();

        deleteMiddle(stack, currentIndex + 1, size);

        stack.push(topElement);
    }

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

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Stack<Integer> stack = new Stack<>();

        System.out.println("Enter the number of elements in the stack:");
        int n = scanner.nextInt();
        System.out.println("Enter the elements of the stack:");
        for (int i = 0; i < n; i++) {
            stack.push(scanner.nextInt());
        }

        System.out.println("Original Stack: " + stack);

        deleteMiddleElement(stack);

        System.out.println("Stack after deleting middle element: " + stack);

        scanner.close();
    }
}

```

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:

```java
import java.util.Scanner;
import java.util.Stack;

public class NextGreaterelements {

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

        System.out.print("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();
        }

        int[] nge = new int[n];

        Stack<Integer> stack = new Stack<>();

        for (int i = n - 1; i >= 0; i--) {
            while (!stack.isEmpty() && stack.peek() <= arr[i]) {
                stack.pop();
            }


            if (stack.isEmpty()) {
                nge[i] = -1;
            } else {
                nge[i] = stack.peek();
            }

            stack.push(arr[i]);
        }

        System.out.println("Next Greater Element for each element:");
        for (int i = 0; i < n; i++) {
            System.out.print(nge[i] + " ");
        }
        scanner.close();
    }
}
```

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:

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

class Node {
    int data;
    Node left, right;

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

class BinaryTree {
    Node root;

    // Function to print the right view of the binary tree
    void rightView() {
        if (root == null) {
            return;
        }

        Queue<Node> queue = new LinkedList<>();
        queue.add(root);

        while (!queue.isEmpty()) {
            int size = queue.size();
            Node rightmostNode = null;

            for (int i = 0; i < size; i++) {
                Node currentNode = queue.poll();
                rightmostNode = currentNode; // Update the rightmost node

                // Add left and right children to the queue
                if (currentNode.left != null) {
                    queue.add(currentNode.left);
                }
                if (currentNode.right != null) {
                    queue.add(currentNode.right);
                }
            }
            // Print the rightmost node of the current level
            System.out.print(rightmostNode.data + " ");
        }
    }

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

        System.out.println("Right view of the binary tree:");
        tree.rightView();
    }
}
```

Output:

```
PS F:\java>  & 'C:\Program Files\Java\jdk-
\workspaceStorage\41ec9565a616c87806451eb0
Maximum depth of the binary tree: 3
PS F:\java>
```

18. Maximum Depth or Height of Binary Tree:

Code:

```java
class TreeNode {
    int value;
    TreeNode left;
    TreeNode right;

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

public class BinaryTree {
    public int maxDepth(TreeNode root) {
        if (root == null) {
            return 0;
        } else {

            int leftDepth = maxDepth(root.left);
            int rightDepth = maxDepth(root.right);


            return Math.max(leftDepth, rightDepth) + 1;
        }
    }

    public static void main(String[] args) {

        TreeNode root = new TreeNode(1);
        root.left = new TreeNode(2);
        root.right = new TreeNode(3);
        root.left.left = new TreeNode(4);
        root.left.right = new TreeNode(5);

        BinaryTree tree = new BinaryTree();
        System.out.println("Maximum depth of the binary tree: " + tree.maxDepth(root));
    }
}
```

Output:

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