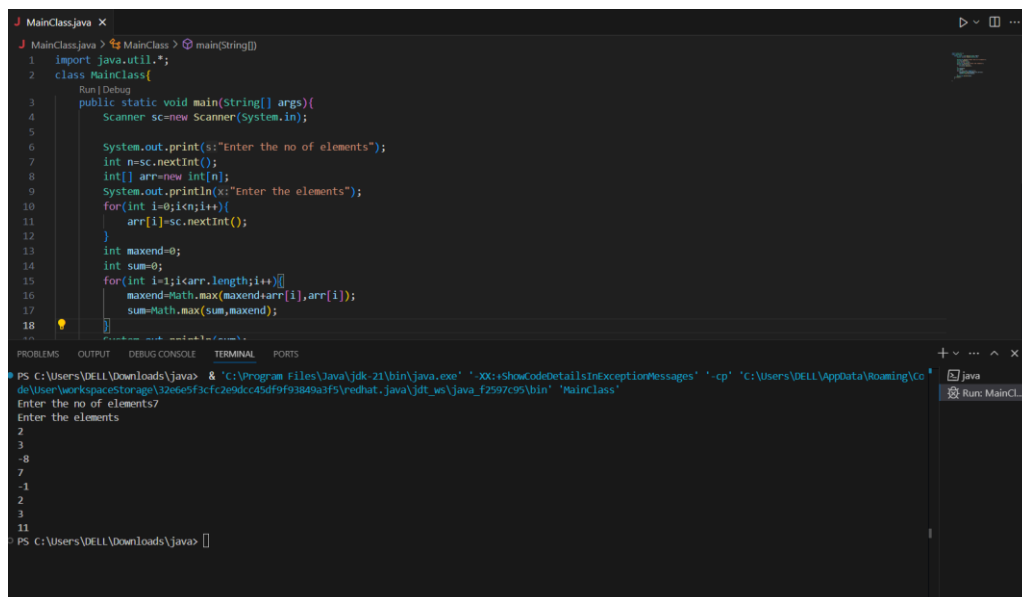**Date**: 09/11/2024

DSA Practice Problems

1. Maximum Subarray Sum – Kadan's Algorithm

```java
import java.util.Scanner;

public class MaximumSubarray {
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the no of elements");
        int n=sc.nextInt();
        int[] arr=new int[n];
        System.out.println("Enter the elements");
        for(int i=0;i<n;i++){
            arr[i]=sc.nextInt();
        }
        int sum = arr[0];
        int maxend = arr[0];
        for (int i = 1; i < arr.length; i++) {
            maxend = Math.max(maxend + arr[i], arr[i]);
            sum = Math.max(sum, maxend);
        }
        sc.close();
    }
}
```
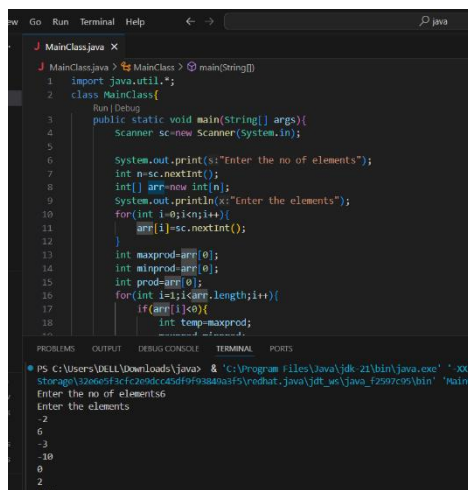

Time Complexity: O(n)

2. Maximum Product Subarray

```java
import java.util.Scanner;

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

        System.out.print("Enter the no of elements");
        int n=sc.nextInt();
        int[] arr=new int[n];
        System.out.println("Enter the elements");
        for(int i=0;i<n;i++){
            arr[i]=sc.nextInt();
        }
        int maxprod = arr[0];
        int minprod = arr[0];
        int prod = arr[0];
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] < 0) {
                int temp = maxprod;
                maxprod = minprod;
                minprod = temp;
            }
            maxprod = Math.max(arr[i], maxprod * arr[i]);
            minprod = Math.min(arr[i], minprod * arr[i]);
            prod = Math.max(prod, maxprod);
        }
        System.out.println(prod);
        sc.close();
    }
}
```

Time Complexity: O(n)

3. Search in a sorted and rotated Array

```java
import java.util.Scanner;

public class sortedarray {
   public static void main(String[] args){
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter the no of elements");
      int n=sc.nextInt();
      int[] arr=new int[n];
      System.out.println("Enter the elements");
      for(int i=0;i<n;i++){
         arr[i]=sc.nextInt();
      }
      int key=sc.nextInt();
      int ind=-1;
      int left = 0, right = arr.length - 1;
      while (left <= right) {
         int mid = left + (right - left) / 2;
         if (arr[mid] == key) {
            ind=mid;
            break;
         }

         if (arr[left] <= arr[mid]) {
            if (arr[left] <= key && key < arr[mid]) {
               right = mid - 1;
            } else {
               left = mid + 1;
            }
         }
         else {
            if (arr[mid] < key && key <= arr[right]) {
               left = mid + 1;
            } else {
               right = mid - 1;
            }
         }
      }
      System.out.println(ind);
      sc.close();
   }
}
```

Time Complexity: O(log n)

4. Container with Most Water

```java
import java.util.*;

class Containerwithmostwater{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the no of elements");
        int n=sc.nextInt();
        int[] arr=new int[n];
        System.out.println("Enter the elements");
        for(int i=0;i<n;i++){
            arr[i]=sc.nextInt();
        }
        int left=0;
        int maxarea=1;
        int right=arr.length-1;
        while(left<right){
            int base=right-left;
            int height=Math.min(arr[left],arr[right]);
            int area=base*height;
            maxarea=Math.max(area,maxarea);
            if(arr[left]<arr[right]){
                left++;
            }
            else{
                right--;
```

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

}
```


Time Complexity: O(n)
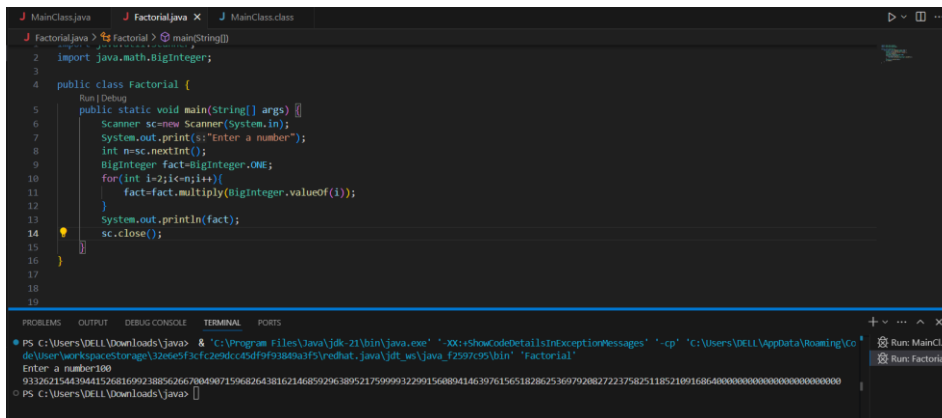
5. Find the Factorial of a large number

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

public class Factorial {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter a number");
        int n=sc.nextInt();
        BigInteger fact=BigInteger.ONE;
        for(int i=2;i<=n;i++){
            fact=fact.multiply(BigInteger.valueOf(i));
        }
        System.out.println(fact);
        sc.close();
    }
```

```
}
```


Time Complexity: O(n)

6. Trapping Rainwater Problem

```java
import java.util.*;

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

      System.out.print("Enter the no of elements");
      int n=sc.nextInt();
      int[] arr=new int[n];
      System.out.println("Enter the elements");
      for(int i=0;i<n;i++){
         arr[i]=sc.nextInt();
      }
      int result=0;
      for(int i=0;i<arr.length;i++){
         int left=arr[i];
         for(int j=0;j<i;j++){
            left=Math.max(left,arr[j]);
         }
         int right=arr[i];
         for(int j=i+1;j<arr.length;j++){
            right=Math.max(right,arr[j]);
         }
         result += Math.min(left, right) - arr[i];
      }
      System.out.println(result);
      sc.close();
   }

}
```

```java
J TrappingRainWater.java > 🦹 TrappingRainWater > 🔷 main(String[])
1   import java.util.*;
2   class TrappingRainWater{
        Run | Debug
3       public static void main(String[] args){
4           Scanner sc=new Scanner(System.in);
5
6           System.out.print(s:"Enter the no of elements");
7           int n=sc.nextInt();
8           int[] arr=new int[n];
9           System.out.println(x:"Enter the elements");
10          for(int i=0;i<n;i++){
11              arr[i]=sc.nextInt();
12          }
13          int result=0;
14          for(int i=0;i<arr.length;i++){
15              int left=arr[i];
16              for(int j=0;j<i;j++){
17                  left=Math.max(left,arr[j]);
18              }
                int right arr[i].
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

ogram Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionM
5f3cfc2e9dcc45df9f93849a3f5\redhat.java\jdt_ws\java_f2597c95\bin' 'Trap
Enter the no of elements7
Enter the elements
3
0
1
0
4
0
2
10
```

Time Complexity: O(n^2)

7. Chocolate Distribution Problem

```java
import java.util.*;

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

        System.out.print("Enter the no of elements");
        int num=sc.nextInt();
        int[] arr=new int[num];
        System.out.println("Enter the elements");
        for(int i=0;i<num;i++){
            arr[i]=sc.nextInt();
        }
        System.out.println("Enter the value of m");
        int m=sc.nextInt();
        Arrays.sort(arr);
```

```java
        int mindiff=Integer.MAX_VALUE;
        int n=arr.length;
        for(int i=0;i+m-1<n;i++){
            int diff=arr[i+m-1]-arr[i];
        if(diff<mindiff){
            mindiff=diff;
        }

        }
    System.out.println(mindiff);
    sc.close();
    }

}
```

```
J ChocolateDistribution.java > ⅔ ChocolateDistributionClass > ⊘ main(String[])
 1    import java.util.*;
 2    class ChocolateDistributionClass{
          Run | Debug
 3        public static void main(String[] args){
 4            Scanner sc=new Scanner(System.in);
 5
 6            System.out.print(s:"Enter the no of elements");
 7            int num=sc.nextInt();
 8            int[] arr=new int[num];
 9            System.out.println(x:"Enter the elements");
10            for(int i=0;i<num;i++){
11                arr[i]=sc.nextInt();
12            }
13            System.out.println(x:"Enter the value of m");
14            int m=sc.nextInt();
15            Arrays.sort(arr);
16            int mindiff=Integer.MAX_VALUE;
17            int n=arr.length;
18            for(int i=0;i+m-1<n;i++){
                   int diff onn[irm 1] onn[il.

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

● ' 'C:\Users\DELL\AppData\Roaming\Code\User\workspaceStorage\32e6e5f3cf
  tionClass'
  Enter the no of elements7
  Enter the elements
  7
  3
  2
  4
  9
  12
  56
  Enter the value of m
  3
  2
```

Time Complexity: O(n log n)

8. Merge Overlapping Intervals
   import java.util.*;

   public class MergeOverlappingIntervals {

```java
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.err.println("Enter the value of n");
    int n = sc.nextInt();
    int[][] arr = new int[n][2];
    System.err.println("Enter the elements");
    for (int i = 0; i < n; i++) {
        arr[i][0] = sc.nextInt();
        arr[i][1] = sc.nextInt();
    }
    Arrays.sort(arr, (a, b) -> a[0] - b[0]);
    List<List<Integer>> res = new ArrayList<>();
    for (int i = 0; i < n; i++) {
        if (res.isEmpty() || arr[i][0] > res.get(res.size() - 1).get(1)) {
            res.add(Arrays.asList(arr[i][0], arr[i][1]));
        } else {
            res.get(res.size() - 1).set(1, Math.max(res.get(res.size() - 1).get(1),
arr[i][1]));
        }
    }

    for (List<Integer> interval : res) {
        System.out.print("[" + interval.get(0) + ", " + interval.get(1) + "] ");
    }
    System.out.println();

    sc.close();
  }
}
```

Time Complexity: O(n log n)

9. Boolean Matrix

```java
import java.util.Scanner;

class BooleanMatrix {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of rows: ");
        int m = sc.nextInt();
        System.out.print("Enter the number of columns: ");
        int n = sc.nextInt();
        int mat[][] = new int[m][n];
        System.out.println("Enter the elements:");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                mat[i][j] = sc.nextInt();
            }
        }
        int row[] = new int[m];
        int col[] = new int[n];
        for (int i = 0; i < m; i++) row[i] = 0;
        for (int i = 0; i < n; i++) col[i] = 0;
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (mat[i][j] == 1) {
                    row[i] = 1;
                    col[j] = 1;
                }
            }
        }
```

```java
        for (int i = 0; i < m; i++) {

            for (int j = 0; j < n; j++) {

                if (row[i] == 1 || col[j] == 1) {

                    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();

    }

}
```

Time Complexity: O(m*n)

10. Print a given matrix in spiral form

```java
import java.util.Scanner;

public class SpiralMatrix {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of rows: ");
        int m = sc.nextInt();
        System.out.print("Enter the number of columns: ");
        int n = sc.nextInt();
        int[][] mat = new int[m][n];
        System.out.println("Enter the elements");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                mat[i][j] = sc.nextInt();
            }
        }
        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(mat[top][i] + " ");
            }
            top++;
            for (int i = top; i <= bottom; ++i) {
                System.out.print(mat[i][right] + " ");
            }
            right--;
            if (top <= bottom) {
                for (int i = right; i >= left; --i) {
                    System.out.print(mat[bottom][i] + " ");
                }
                bottom--;
            }
            if (left <= right) {
                for (int i = bottom; i >= top; --i) {
                    System.out.print(mat[i][left] + " ");
                }
                left++;
            }
        }
        sc.close();
    }
```

```
        }
```


Time Complexity: O(m*n)

11. Check if given Parentheses expression is balanced or not

import java.util.*;

public class Parantheses {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string");
        String s = sc.nextLine();
        Stack<Character> stack1 = new Stack<>();
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) == '(' || s.charAt(i) == '{' || s.charAt(i) == '[') {
                stack1.push(s.charAt(i));
            } else {
                if (!stack1.empty() &&
                    ((stack1.peek() == '(' && s.charAt(i) == ')') ||
                     (stack1.peek() == '{' && s.charAt(i) == '}') ||
                     (stack1.peek() == '[' && s.charAt(i) == ']'))) {
                    stack1.pop();
                } else {
                    System.out.println("false");
```

```java
                sc.close();
                return;
            }
        }
    }

    if (stack1.empty()) {
        System.out.println("Balanced");
    } else {
        System.out.println("Not Balanced");
    }
    sc.close();
    }
}
```
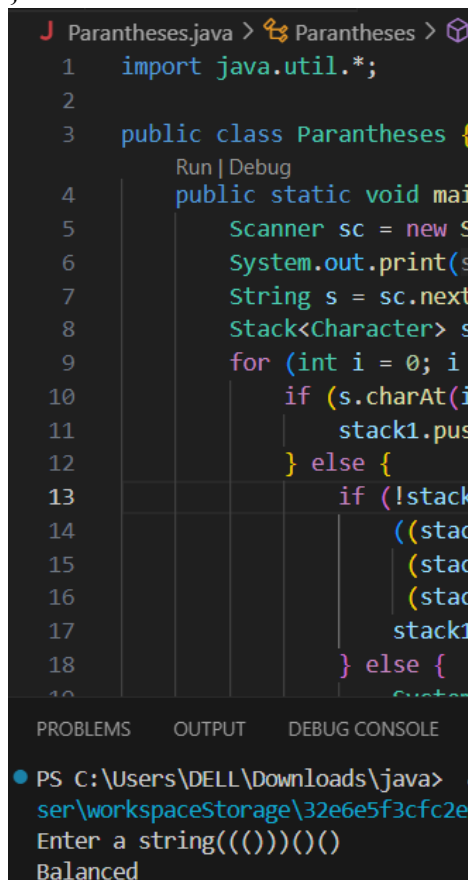


```
J Parantheses.java > ⁒ Parantheses > ◈
  1     import java.util.*;
  2
  3     public class Parantheses {
            Run | Debug
  4         public static void mai
  5             Scanner sc = new S
  6             System.out.print(s
  7             String s = sc.next
  8             Stack<Character> s
  9             for (int i = 0; i
 10                 if (s.charAt(i
 11                     stack1.pus
 12                 } else {
 13                     if (!stack
 14                         ((stac
 15                         (stac
 16                         (stac
 17                         stack1
 18                 } else {

PROBLEMS    OUTPUT    DEBUG CONSOLE
● PS C:\Users\DELL\Downloads\java>  &
ser\workspaceStorage\32e6e5f3cfc2e9
Enter a string((()))()()
Balanced
```

Time Complexity: O(n)

12. Check if two Strings are Anagrams of each other
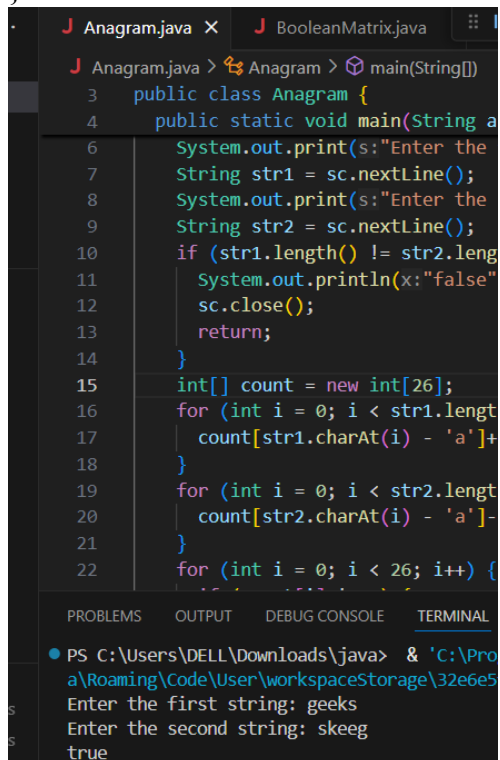
```java
import java.util.Scanner;

public class Anagram {
 public static void main(String args[]) {
   Scanner sc = new Scanner(System.in);
```

```java
System.out.print("Enter the first string: ");
String str1 = sc.nextLine();
System.out.print("Enter the second string: ");
String str2 = sc.nextLine();
if (str1.length() != str2.length()) {
  System.out.println("false");
  sc.close();
  return;
}
int[] count = new int[26];
for (int i = 0; i < str1.length(); i++) {
  count[str1.charAt(i) - 'a']++;
}
for (int i = 0; i < str2.length(); i++) {
  count[str2.charAt(i) - 'a']--;
}
for (int i = 0; i < 26; i++) {
  if (count[i] != 0) {
    System.out.println("false");
    sc.close();
    return;
  }
}
System.out.println("true");
sc.close();
}
}
```

Time Complexity: O(n)

13. Longest Palindromic Substring

```java
import java.util.*;
public class PalindromicSubstring {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.nextLine();
        int n = s.length();
        boolean[][] dp = new boolean[n][n];
        int maxlen = 1;
        int start = 0;
        for (int i = 0; i < n; ++i)
            dp[i][i] = true;
        for (int i = 0; i < n - 1; ++i) {
            if (s.charAt(i) == s.charAt(i + 1)) {
                dp[i][i + 1] = true;
                start = i;
                maxlen = 2;
            }
        }
        for (int k = 3; k <= n; ++k) {
            for (int i = 0; i < n - k + 1; ++i) {
                int j = i + k - 1;
                if (dp[i + 1][j - 1] && s.charAt(i) == s.charAt(j)) {
                    dp[i][j] = true;
                    if (k > maxlen) {
                        start = i;
                        maxlen = k;
                    }
                }
            }
        }
        System.out.println(s.substring(start, start + maxlen));
        sc.close();
    }
}
```

Time Complexity: O(n^2)

14. Longest Common Prefix using Sorting

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

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

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

        if (arr == null || arr.length == 0) {
            System.out.println("-1");
            sc.close();
            return;
        }
        Arrays.sort(arr);
```
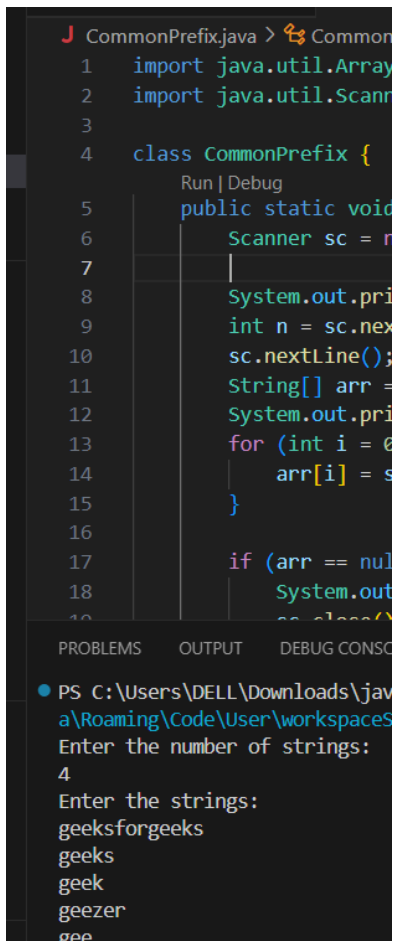
```java
            String start = arr[0];
            String end = arr[arr.length - 1];
            int minlen = Math.min(start.length(), end.length());
            int i = 0;
            while (i < minlen && start.charAt(i) == end.charAt(i)) {
                i++;
            }
            if (i == 0) {
                System.out.println("-1");
                sc.close();
            } else {
                System.out.println(start.substring(0, i));
                sc.close();
            }

        }
}
```

```
J CommonPrefix.java > ⅗ Common
  1      import java.util.Array
  2      import java.util.Scann
  3
  4      class CommonPrefix {
           Run | Debug
  5         public static void
  6            Scanner sc = r
  7            |
  8            System.out.pri
  9            int n = sc.nex
 10            sc.nextLine();
 11            String[] arr =
 12            System.out.pri
 13            for (int i = 0
 14               arr[i] = s
 15            }
 16
 17            if (arr == nul
 18               System.out
```

```
PROBLEMS    OUTPUT    DEBUG CONSO

PS C:\Users\DELL\Downloads\jav
a\Roaming\Code\User\workspaceS
Enter the number of strings:
4
Enter the strings:
geeksforgeeks
geeks
geek
geezer
gee
```

Time Complexity: O(n log n + m)
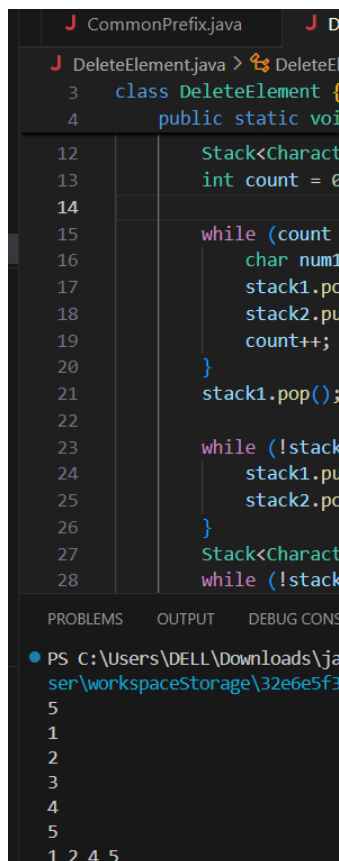
15. Delete middle element of a stack

```java
import java.util.*;

class DeleteElement {
  public static void main(String args[]) {
    Scanner scanner = new Scanner(System.in);
    int n = scanner.nextInt();
    Stack<Character> stack1 = new Stack<>();
    for (int i = 0; i < n; i++) {
      stack1.push(scanner.next().charAt(0));
    }
    int size = stack1.size();
    Stack<Character> stack2 = new Stack<>();
    int count = 0;

    while (count < size / 2) {
      char num1 = stack1.peek();
      stack1.pop();
      stack2.push(num1);
      count++;
    }
    stack1.pop();

    while (!stack2.isEmpty()) {
      stack1.push(stack2.peek());
      stack2.pop();
    }
    Stack<Character> finalStack = new Stack<>();
    while (!stack1.isEmpty()) {
      finalStack.push(stack1.peek());
      stack1.pop();
    }
    while (!finalStack.isEmpty()) {
      char num2 = finalStack.peek();
      finalStack.pop();
      System.out.print(num2 + " ");
      scanner.close();
    }
  }
}
```

Time Complexity: O(n)

16. Next Greater Element

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

public class NextGreaterElement {

    public static void NextGreaterElements(int[] arr) {
        int n = arr.length;
        int[] sol = new int[n];
        Stack<Integer> stack1 = new Stack<>();

        for (int i = 0; i < n; i++) {
            sol[i] = -1;
        }

        for (int i = n - 1; i >= 0; i--) {
            while (!stack1.isEmpty() && stack1.peek() <= arr[i]) {
                stack1.pop();
            }

            if (!stack1.isEmpty()) {
```

```java
            sol[i] = stack1.peek();
        }

        stack1.push(arr[i]);
    }

    for (int i = 0; i < n; i++) {
        System.out.println(arr[i] + " --> " + (sol[i] != -1 ? sol[i] : "-1"));
    }
}

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();
    }
    NextGreaterElements(arr);
    sc.close();
}
}
```


Time Complexity: O(n)

17. Right View Of Binary Tree

```java
import java.util.*;

class Node {
    int data;
    Node left, right;

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

public class RightViewBinaryTree {

    public static ArrayList<Integer> getRightView(Node rootNode) {
        ArrayList<Integer> ViewList = new ArrayList<>();
        if (rootNode == null) {
            return ViewList;
        }

        Queue<Node> queue1 = new LinkedList<>();
        queue1.add(rootNode);

        while (!queue1.isEmpty()) {
            int currentLevelSize = queue1.size();
            for (int i = 0; i < currentLevelSize; i++) {
                Node currentNode = queue1.poll();
                if (i == currentLevelSize - 1) {
                    ViewList.add(currentNode.data);
                }
                if (currentNode.left != null) {
                    queue1.add(currentNode.left);
                }
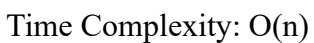                if (currentNode.right != null) {
                    queue1.add(currentNode.right);
                }
            }
        }

        return ViewList;
    }

    public static void main(String[] args) {
        Node tree1Root = new Node(1);
```

```
        tree1Root.left = new Node(2);
        tree1Root.right = new Node(3);
        tree1Root.right.left = new Node(4);
        tree1Root.right.right = new Node(5);

        ArrayList<Integer> tree1RightView = getRightView(tree1Root);
        System.out.println(tree1RightView);

        Node tree2Root = new Node(1);
        tree2Root.left = new Node(2);
        tree2Root.right = new Node(3);
        tree2Root.left.left = new Node(4);
        tree2Root.left.left.right = new Node(5);

        ArrayList<Integer> tree2RightView = getRightView(tree2Root);
        System.out.println(tree2RightView);
    }
}
```

```
xtGreaterElement.java

  J  RightViewBinaryTree.ja
    1    import java.u
    2
    3    class Node {
    4        int data;
    5        Node left
    6
    7        Node(int
    8            this.
    9            left
   10        }
   11    }
   12
   13    public class
   14
   15        public st
   16            Array
   17            if (r
   18                r
   19            }

PROBLEMS    OUTPUT

● PS C:\Users\DELL\Dow
  a\Roaming\Code\User\
  [1, 3, 5]
  [1, 3, 4, 5]
```

Time Complexity: O(n)

18. Maximum Depth or Height of Binary Tree

```java
    class Node {
    int data;
    Node left, right;

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


public class FindMaxDepth {
    public static int MaxDepth(Node rootNode) {
        if (rootNode == null) {
            return 0;
        }
        int leftHeight = MaxDepth(rootNode.left);
        int rightHeight = MaxDepth(rootNode.right);
        return Math.max(leftHeight, rightHeight) + 1;
    }

    public static void main(String[] args) {
        Node root1 = new Node(12);
        root1.left = new Node(8);
        root1.right = new Node(18);
        root1.left.left = new Node(5);
        root1.left.right = new Node(11);

        System.out.println(MaxDepth(root1));
```

```java
        Node root2 = new Node(1);

        root2.left = new Node(2);

        root2.right = new Node(3);

        root2.left.left = new Node(4);

        root2.right.left = new Node(5);

        root2.right.left.left = new Node(6);

        root2.right.left.right = new Node(7);


        System.out.println(MaxDepth(root2));

    }

}
```


Time Complexity: O(n)