

DSA Programming Practice 1 – 09/11/24 – KAVYA U – CSBS(III yr) – 2026 BATCH – 22CB024

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.

Input: `arr[] = {2, 3, -8, 7, -1, 2, 3}`

Output: 11

Explanation: The subarray {7, -1, 2, 3} has the largest sum 11.

Input: `arr[] = {-2, -4}`

Output: -2

Explanation: The subarray {-2} has the largest sum -2.

Code and Output:

```
1  import java.util.*;
2  |
Codeium: Refactor | Explain
3  class maxsubarr {
Codeium: Refactor | Explain | Generate Javadoc | X
4      public static int maxisubarr(int[] arr, int n){
5          int maxi = Integer.MIN_VALUE;
6          int sum=0;
7          for(int i=0; i<n; i++){
8              sum=Math.max(arr[i],sum+arr[i]);
9              maxi=Math.max(maxi,sum);
10         }
11         return maxi;
12     }
Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
13     public static void main(String[] args) {
14         Scanner s = new Scanner(System.in);
15         System.out.println(x:"Enter the size of the array: ");
16         int n = s.nextInt();
17         int[] arr = new int[n];
18         System.out.println(x:"Enter the elements of the array: ");
19         for(int i=0; i<n; i++){
20             arr[i] = s.nextInt();
21         }
22         int res = maxisubarr(arr, n);
23         System.out.println("The maximum subarray sum is: " + res);
24     }
25 }
```

OUTPUT PROBLEMS 1 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

```
7
Enter the elements of the array:
2 3 -8 7 -1 2 3
The maximum subarray sum is: 11
```

Time Complexity : $O(N)$

2. Maximum Product Subarray :

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

Input: arr[] = {-2, 6, -3, -10, 0, 2}

Output: 180

Explanation: The subarray with maximum product is {6, -3, -10} with product = $6 * (-3) * (-10) = 180$

Input: arr[] = {-1, -3, -10, 0, 60}

Output: 60

Code and Output:

```
2  class maxprodarr {
    Codeium: Refactor | Explain | Generate Javadoc | X
3      public static int maxiprodarr(int[] arr, int n){
4          int p1 = arr[0], p2 = arr[0], res = arr[0];
5          for(int i=1; i<arr.length; i++) {
6              int temp = Math.max(arr[i], Math.max(p1*arr[i], p2*arr[i]));
7              p2 = Math.min(arr[i], Math.min(p1*arr[i], p2*arr[i]));
8              p1 = temp;
9              res = Math.max(res, p1);
10         }
11         return res;
12     }
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
13     public static void main(String[] args) {
14         Scanner s = new Scanner(System.in);
15         System.out.println(x:"Enter the size of the array: ");
16         int n = s.nextInt();
17         int[] arr = new int[n];
18         System.out.println(x:"Enter the elements of the array: ");
19         for(int i=0; i<n; i++){
20             arr[i] = s.nextInt();
21         }
22         int res = maxiprodarr(arr, n);
23         System.out.println("The maximum subarray sum is: " + res);
24     }
25 }
```

OUTPUT PROBLEMS 2 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

Enter the size of the array:

6

Enter the elements of the array:

-2 6 -3 -10 0 2

▢ The maximum subarray sum is: 180

Time Complexity : $O(N)$

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.

Input : `arr[] = {4, 5, 6, 7, 0, 1, 2}`, `key = 0`

Output : 4

Input : `arr[] = { 4, 5, 6, 7, 0, 1, 2 }`, `key = 3`

Output : -1

Input : `arr[] = {50, 10, 20, 30, 40}`, `key = 10`

Output : 1

Code and Output:

```
import java.util.*;
Codeium: Refactor | Explain
class searchinrotarr {
    Codeium: Refactor | Explain | Generate Javadoc | X
    public static int search(int[] arr, int n, int target){
        int l=0;
        int h=n-1;
        int m=0;
        while(l<=h){
            m=(l+h)/2;
            if(arr[m]==target){
                return m;
            }
            if(arr[l]<=arr[m]){
                if(target>=arr[l] && target<=arr[m]){
                    h=m-1;
                }
                else{
                    l=m+1;
                }
            }else{
                if(target>=arr[m] && target<=arr[h]){
                    l=m+1;
                }else{
                    h=m-1;
                }
            }
        }
        }return -1;
    }
```

```

28     public static void main(String[] args) {
29         Scanner s = new Scanner(System.in);
30         System.out.println(x:"Enter the size of the array: ");
31         int n = s.nextInt();
32         int[] arr = new int[n];
33         System.out.println(x:"Enter the elements of the array: ");
34         for(int i=0; i<n; i++){
35             arr[i] = s.nextInt();
36         }
37         System.out.println(x:"Enter the target to search: ");
38         int target = s.nextInt();
39         int res = search(arr, n, target);
40         System.out.println("The element is found at index: " + res);
41     }
42 }
43

```

OUTPUT PROBLEMS 3 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

```

Enter the size of the array:
7
Enter the elements of the array:
4 5 6 7 0 1 2
Enter the target to search:
0
The element is found at index: 4

```

Time Complexity: $O(N)$

4. [Container with most water:](#)

Input: arr = [1, 5, 4, 3]

Output: 6

Explanation:

5 and 3 are distance 2 apart. So the size of the base = 2.

Height of container = $\min(5, 3) = 3$. So total area = $3 * 2 = 6$

Input: arr = [3, 1, 2, 4, 5]

Output: 12

Explanation:

5 and 3 are distance 4 apart. So the size of the base = 4.

Height of container = $\min(5, 3) = 3$. So total area = $4 * 3 = 12$

Code and Output:

Code and Output:

Code and Output:

```
Codeium: Refactor | Explain | Generate Javadoc | X
class trappingrainwater {
    static int trap(int[] height) {
        int n = height.length;
        int l = 0, r = n - 1;
        int res = 0;
        int maxl = 0, maxr = 0;
        while (l <= r) {
            if (height[l] <= height[r]) {
                if (height[l] >= maxl) {
                    maxl = height[l];
                } else {
                    res += maxl - height[l];
                }
                l++;
            } else {
                if (height[r] >= maxr) {
                    maxr = height[r];
                } else {
                    res += maxr - height[r];
                }
                r--;
            }
        }
        return res;
    }
}
```

```
Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
27 public static void main(String args[]) {
28     Scanner s = new Scanner(System.in);
29     System.out.println(x:"Enter the size of the array: ");
30     int n = s.nextInt();
31     int[] arr = new int[n];
32     System.out.println(x:"Enter the elements of the array: ");
33     for(int i=0; i<n; i++){
34         arr[i] = s.nextInt();
35     }
36     System.out.println("The max capacity is " + trap(arr));
37 }
38 }
39
```

OUTPUT PROBLEMS 5 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

```
Enter the size of the array:
7
Enter the elements of the array:
3 0 1 0 4 0 2
The max capacity is 10
```

Time Complexity: $O(N)$

7. Chocolate Distribution:

Given an array `arr[]` of n integers where `arr[i]` represents the number of chocolates in i th packet. Each packet can have a variable number of chocolates. There are m students, the task is to distribute chocolate packets such that:

Each student gets exactly one packet.

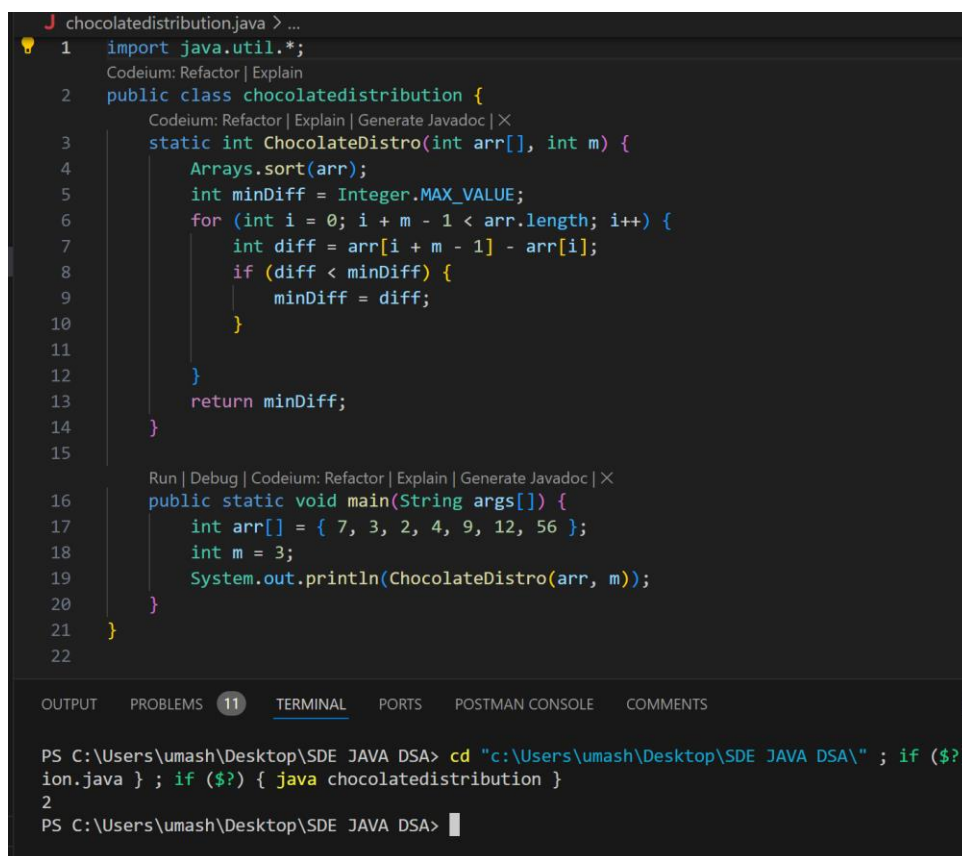
The difference between the maximum and minimum number of chocolates in the packets given to the students is minimized.

Input: `arr[] = {7, 3, 2, 4, 9, 12, 56}`, $m = 3$

Output: 2

Explanation: If we distribute chocolate packets {3, 2, 4}, we will get the minimum difference, that is 2.

Code and Output:



```
J chocolatedistribution.java > ...
1  import java.util.*;
   Codeium: Refactor | Explain
2  public class chocolatedistribution {
   Codeium: Refactor | Explain | Generate Javadoc | X
3      static int ChocolateDistro(int arr[], int m) {
4          Arrays.sort(arr);
5          int minDiff = Integer.MAX_VALUE;
6          for (int i = 0; i + m - 1 < arr.length; i++) {
7              int diff = arr[i + m - 1] - arr[i];
8              if (diff < minDiff) {
9                  minDiff = diff;
10             }
11         }
12         return minDiff;
13     }
14 }
15
   Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
16 public static void main(String args[]) {
17     int arr[] = { 7, 3, 2, 4, 9, 12, 56 };
18     int m = 3;
19     System.out.println(ChocolateDistro(arr, m));
20 }
21 }
22

OUTPUT  PROBLEMS 11  TERMINAL  PORTS  POSTMAN CONSOLE  COMMENTS 2
PS C:\Users\umash\Desktop\SDE JAVA DSA> cd "c:\Users\umash\Desktop\SDE JAVA DSA\" ; if ($?)
ion.java } ; if ($?) { java chocolatedistribution }
2
PS C:\Users\umash\Desktop\SDE JAVA DSA> 
```

Time Complexity: $O(N)$

8. [Merge Overlapping Intervals:](#)

Given an array of time intervals where `arr[i] = [starti, endi]`, the task is to merge all the overlapping intervals into one and output the result which should have only mutually exclusive intervals.

Input: `arr[] = [[1, 3], [2, 4], [6, 8], [9, 10]]`

Output: `[[1, 4], [6, 8], [9, 10]]`

Explanation: In the given intervals, we have only two overlapping intervals [1, 3] and [2, 4]. Therefore, we will merge these two and return [[1, 4]], [6, 8], [9, 10]].

Input: arr[] = [[7, 8], [1, 5], [2, 4], [4, 6]]

Output: [[1, 6], [7, 8]]

Explanation: We will merge the overlapping intervals [[1, 5], [2, 4], [4, 6]] into a single interval [1, 6].

Code and Output:

```
public class mergeintervals {
    Codeium: Refactor | Explain | Generate Javadoc | X
    public static List<List<Integer>> mergeOverlappingIntervals(int[][] arr) {
        int n = arr.length;
        Arrays.sort(arr, new Comparator<int[]>() {
            Codeium: Refactor | Explain | Generate Javadoc | X
            public int compare(int[] a, int[] b) {
                return a[0] - b[0];
            }
        });
        List<List<Integer>> ans = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            if (ans.isEmpty() || arr[i][0] > ans.get(ans.size() - 1).get(index:1)) {
                ans.add(Arrays.asList(arr[i][0], arr[i][1]));
            }
            else {
                ans.get(ans.size() - 1).set(index:1, Math.max(ans.get(ans.size() - 1).get(index:1), arr[i][1]));
            }
        }
        return ans;
    }
}

26 public static void main(String[] args) {
27     Scanner sc = new Scanner(System.in);
28     int n = sc.nextInt();
29     int m = sc.nextInt();
30     int[][] arr = new int[n][m];
31     for (int i = 0; i < n; i++) {
32         for (int j = 0; j < m; j++) {
33             arr[i][j] = sc.nextInt();
34         }
35     }
36     List<List<Integer>> ans = mergeOverlappingIntervals(arr);
37     System.out.print(s:"The merged intervals are: \n");
38     for (List<Integer> it : ans) {
39         System.out.print "[" + it.get(index:0) + ", " + it.get(index:1) + " ";
40     }
41     System.out.println();
42 }
43 }
44
45
```

OUTPUT PROBLEMS 6 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

```
4
2
1 3
8 9
2 6
15 18
The merged intervals are:
[1, 6] [8, 9] [15, 18]
```

Time Complexity : $O(N \cdot \log N) + O(N)$

9. [A Boolean Matrix](#) :

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.

Input: {{1, 0},

{0, 0}}

Output: {{1, 1}

{1, 0}}

Input: {{0, 0, 0},

{0, 0, 1}}

Output: {{0, 0, 1},

{1, 1, 1}}

Input: {{1, 0, 0, 1},

{0, 0, 1, 0},

{0, 0, 0, 0}}

Output: {{1, 1, 1, 1},

{1, 1, 1, 1},

{1, 0, 1, 1}}

Code and Output:

```
class booleanmatrix {
    public static void modifyMatrix(int mat[][]) {
        boolean row_state = false;
        boolean col_state = false;
        for (int i = 0; i < mat.length; i++) {
            for (int j = 0; j < mat[0].length; j++) {
                if (i == 0 && mat[i][j] == 1)
                    row_state = true;
                if (j == 0 && mat[i][j] == 1)
                    col_state = true;
                if (mat[i][j] == 1) {
                    mat[0][j] = 1;
                    mat[i][0] = 1;
                }
            }
        }
        for (int i = 1; i < mat.length; i++)
            for (int j = 1; j < mat[0].length; j++)
                if (mat[0][j] == 1 || mat[i][0] == 1)
                    mat[i][j] = 1;
        if (row_state == true)
            for (int i = 0; i < mat[0].length; i++)
                mat[0][i] = 1;
        if (col_state == true)
            for (int i = 0; i < mat.length; i++)
                mat[i][0] = 1;
    }

    public static void printMatrix(int mat[][]) {
        for (int i = 0; i < mat.length; i++) {
            for (int j = 0; j < mat[0].length; j++)
                System.out.print(mat[i][j] + " ");
            System.out.println("\n");
        }
    }
}
```

```
Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
37 public static void main(String args[]) {
38     int mat[][] = { { 1, 0, 0, 1 },
39                     { 0, 0, 1, 0 },
40                     { 0, 0, 0, 0 } };
41
42     modifyMatrix(mat);
43     printMatrix(mat);
44 }
45 }
46
```

OUTPUT PROBLEMS 11 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

```
PS C:\Users\umash\Desktop\SDE JAVA DSA> cd "c:\Users\umash\Desktop\SDE JAVA DSA\"
} ; if ($?) { java booleanmatrix }
1 1 1 1
1 1 1 1
1 0 1 1
PS C:\Users\umash\Desktop\SDE JAVA DSA>
```

Time Complexity: $O(M*N)$

10. Spiral Matrix:

Print a given matrix in spiral form

Given an $m \times n$ matrix, the task is to print all elements of the matrix in spiral form.

Input: matrix = {{1, 2, 3, 4},

{5, 6, 7, 8},

{9, 10, 11, 12},

{13, 14, 15, 16 }}

Output: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

Input: matrix = { {1, 2, 3, 4, 5, 6},

{7, 8, 9, 10, 11, 12},

{13, 14, 15, 16, 17, 18}}

Output: 1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11

Explanation: The output is matrix in spiral format.

Code and Output:

Codeium: Refactor | Explain

```
public class spiralmatrix {
```

Codeium: Refactor | Explain | Generate Javadoc | X

```
public static List<Integer> printSpiral(int[][] mat) {
    List<Integer> ans = new ArrayList<>();
    int n = mat.length;
    int m = mat[0].length;
    int top = 0, left = 0, bottom = n - 1, right = m - 1;
    while (top <= bottom && left <= right) {
        for (int i = left; i <= right; i++)
            ans.add(mat[top][i]);
        top++;
        for (int i = top; i <= bottom; i++)
            ans.add(mat[i][right]);
        right--;
        if (top <= bottom) {
            for (int i = right; i >= left; i--)
                ans.add(mat[bottom][i]);
            bottom--;
        }
        if (left <= right) {
            for (int i = bottom; i >= top; i--)
                ans.add(mat[i][left]);
            left++;
        }
    }
    return ans;
}
```

Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X

```
31 public static void main(String[] args) {
32     int[][] mat = {{1, 2, 3, 4},
33                   {5, 6, 7, 8},
34                   {9, 10, 11, 12},
35                   {13, 14, 15, 16}};
36     List<Integer> ans = printSpiral(mat);
37     for(int i = 0; i < ans.size(); i++){
38         System.out.print(ans.get(i) + " ");
39     }
40     System.out.println();
41 }
42 }
```

OUTPUT PROBLEMS 6 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

PS C:\Users\umash\Desktop\SDE JAVA DSA>

Time Complexity: $O(M*N)$

11. Parenthesis Checker:

Check if given Parentheses expression is balanced or not

Given a string str of length N, consisting of „(, „, and „),„ only, the task is to check whether it is balanced or not.

Input: str = “((()))()()”

Output: Balanced

Input: str = “())((())”

Output: Not Balanced

Code and Output:

```
2  class parenthesischeck {
Codeium: Refactor | Explain | Generate Javadoc | X
3  public static boolean isValid(String s) {
4      Stack<Character> st = new Stack<Character>();
5      for (char it : s.toCharArray()) {
6          if (it == '(' || it == '[' || it == '{')
7              st.push(it);
8          else {
9              if(st.isEmpty()) return false;
10             char ch = st.pop();
11             if((it == ')' && ch == '(') || (it == ']' && ch == '[')
12                || (it == '}' && ch == '{')) continue;
13             else return false;
14         }
15     }
16     return st.isEmpty();
17 }
Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
18 public static void main (String[] args) {
19     Scanner sc = new Scanner(System.in);
20     System.out.println(x:"Enter string:");
21     String s = sc.nextLine();
22     if(isValid(s)==true)
23         System.out.println(x:"Balanced");
24     else
25         System.out.println(x:"Not Balanced");
26 }
27 }
```

OUTPUT PROBLEMS 7 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

Enter string:
()[{}]()
Balanced

Time Complexity: O(N)

12. Anagrams:

Given two strings s1 and s2 consisting of lowercase characters, the task is to check whether the two given strings are anagrams of each other or not. An anagram of a string is another string that contains the same characters, only the order of characters can be different.

Input: s1 = "geeks" s2 = "kseeg"

Output: true

Explanation: Both the string have same characters with same frequency. So, they are anagrams.

Input: s1 = "allergy" s2 = "allergic"

Output: false

Explanation: Characters in both the strings are not same. s1 has extra character „y“ and s2 has extra characters „i“ and „c“, so they are not anagrams.

Input: s1 = "g", s2 = "g"

Output: true

Explanation: Characters in both the strings are same, so they are anagrams.

Code and Output:

```
2 public class anagram{
3     public static String SortString(String str)
4     {
5         char c[] = str.toCharArray();
6         Arrays.sort(c);
7         return new String(c);
8     }
9     public static boolean checkAnagrams(String str1, String str2)
10    {
11        if (str1.length() != str2.length())
12            return false;
13        str1 = SortString(str1);
14        str2 = SortString(str2);
15        for (int i = 0; i < str1.length(); i++)
16        {
17            if (str1.charAt(i) != str2.charAt(i))
18                return false;
19        }
20        return true;
21    }
22    public static void main(String args[])
23    {
24        String Str1 = "geeks";
25        String Str2 = "kseeg";
26        System.out.println(checkAnagrams(Str1, Str2));
27    }
28 }
```

Codeium: Refactor | Explain | Generate Javadoc | X

Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X

OUTPUT PROBLEMS 7 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

true

PS C:\Users\umash\Desktop\SDE JAVA DSA>

Time Complexity: $O(N \log N)$

13. Longest Palindromic Substring:

Given a string str, the task is to find the longest substring which is a palindrome. If there are multiple answers, then return the first appearing substring.

Input: str = "forgeeksskeegfor"

Output: "geeksskeeg"

Explanation: There are several possible palindromic substrings like "kssk", "ss", "eeksske" etc. But the substring "geeksskeeg" is the longest among all.

Input: str = "Geeks"

Output: "ee"

Input: str = "abc"

Output: "a"

Input: str = ""

Output: ""

Code and Output:

```
1  import java.util.*;
   Codeium: Refactor | Explain
2  public class longestpalindrome {
   Codeium: Refactor | Explain | Generate Javadoc | X
3      public static String lpalindrome(String s) {
4          if (s.length() <= 1) {
5              return s;
6          }
7          int maxLen = 1;
8          int start = 0;
9          int end = 0;
10         boolean[][] dp = new boolean[s.length()][s.length()];
11         for (int i = 0; i < s.length(); ++i) {
12             dp[i][i] = true;
13             for (int j = 0; j < i; ++j) {
14                 if (s.charAt(j) == s.charAt(i) && (i - j <= 2 || dp[j + 1][i - 1])) {
15                     dp[j][i] = true;
16                     if (i - j + 1 > maxLen) {
17                         maxLen = i - j + 1;
18                         start = j;
19                         end = i;}}}}
20         return s.substring(start, end + 1);
21     }
   Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
22     public static void main(String[] args) {
23         String s="forgeeksskeegfor";
24         String res=lpalindrome(s);
25         System.out.println("Longest palindromic substring is:"+res);
26     }
27 }
```

OUTPUT PROBLEMS 8 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

Longest palindromic substring is:geeksskeeg
PS C:\Users\umash\Desktop\SDE JAVA DSA>

Time Complexity : $O(N^2)$

14. Longest Common Prefix using Sorting :

Given an array of strings arr[]. The task is to return the longest common prefix among each and every strings present in the array. If there's no prefix common in all the strings, return "-1".

Input: arr[] = ["geeksforgeeks", "geeks", "geek", "geezer"]

Output: gee

Explanation: "gee" is the longest common prefix in all the given strings.

Input: arr[] = ["hello", "world"]

Output: -1

Explanation: There's no common prefix in the given strings.

Code and Output:

```
1  import java.util.*;
2
3  Codeium: Refactor | Explain
4  class commonprefix {
5      Codeium: Refactor | Explain | Generate Javadoc | X
6      public static String longestCommonPrefix(String[] v) {
7          StringBuilder ans = new StringBuilder();
8          Arrays.sort(v);
9          String first = v[0];
10         String last = v[v.length - 1];
11
12         for (int i = 0; i < Math.min(first.length(), last.length()); i++) {
13             if (first.charAt(i) != last.charAt(i)) {
14                 return ans.toString();
15             }
16             ans.append(first.charAt(i));
17         }
18         return ans.toString();
19     }
20
21     Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
22     public static void main(String[] args) {
23         Scanner sc = new Scanner(System.in);
24         System.out.println(x:"Enter the number of strings:");
25         int n = sc.nextInt();
26         sc.nextLine();
27         String[] strings = new String[n];
28         System.out.println(x:"Enter the strings:");
29         for (int i = 0; i < n; i++) {
30             strings[i] = sc.nextLine();
31         }
32         String res = longestCommonPrefix(strings);
33         System.out.println("Longest common prefix is: " + res);
34     }
35 }
```

```
Enter the number of strings:
4
Enter the strings:
geeksforgeeks
geeks
geek
geezer
Longest common prefix is: gee
```

Time Complexity : $O(N \log N)$

15. Delete middle element of a stack :

Given a stack with push(), pop(), and empty() operations, The task is to delete the middle element of it without using any additional data structure.

Input : Stack[] = [1, 2, 3, 4, 5]

Output : Stack[] = [1, 2, 4, 5]

Input : Stack[] = [1, 2, 3, 4, 5, 6]

Output : Stack[] = [1, 2, 4, 5, 6]

Code and Output:

```
1  import java.util.*;
2
3  Codeium: Refactor | Explain
4  public class deletemiddleofstack {
5      Codeium: Refactor | Explain | Generate Javadoc | X
6      public static void deletemiddle(Stack<Integer> stack) {
7          int mid = stack.size() / 2;
8          deletemiddlehelper(stack, mid);
9      }
10     Codeium: Refactor | Explain | Generate Javadoc | X
11     private static void deletemiddlehelper(Stack<Integer> stack, int mid) {
12         if (mid == 0) {
13             stack.pop();
14             return;
15         }
16         int top = stack.pop();
17         deletemiddlehelper(stack, mid - 1);
18         stack.push(top);
19     }
20
21     Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
22     public static void main(String[] args) {
23         Scanner sc = new Scanner(System.in);
24         Stack<Integer> stack = new Stack<>();
25         System.out.println(x:"Enter number of elements in stack:");
26         int n = sc.nextInt();
27         System.out.println(x:"Enter stack elements:");
28         for (int i = 0; i < n; i++) {
29             stack.push(sc.nextInt());
30         }
31         System.out.println("Original Stack: " + stack);
32         deletemiddle(stack);
33         System.out.println("Stack after deleting middle element: " + stack);
34     }
35 }
```

Enter number of elements in stack:

5

Enter stack elements:

1

2

3

4

5

Original Stack: [1, 2, 3, 4, 5]

Stack after deleting middle element: [1, 2, 4, 5]

PS C:\Users\umash\Desktop\SDE JAVA DSA>

Time Complexity : $O(N)$

16. Next Greater Element:

Given an array, print the Next Greater Element (NGE) for every element.

Note: The Next greater Element for an element x is the first greater element on the right side of x in the array. Elements for which no greater element exist, consider the next greater element as -1.

Input: arr[] = [4 , 5 , 2 , 25]

Output: 4 → 5

5 → 25

2 → 25

25 → -1

Explanation: Except 25 every element has an element greater than them present on the right side

Input: arr[] = [13 , 7, 6 , 12]

Output: 13 → -1

7 → 12

6 → 12

12 → -1

Explanation: 13 and 12 don't have any element greater than them present on the right side

Code and Output:

```
1 import java.util.*;
2 public class nextgreaterelement {
3     public static void nextLargerElement(int[] arr) {
4         int n = arr.length;
5         int[] result = new int[n];
6         Arrays.fill(result, -1);
7         Stack<Integer> stack = new Stack<>();
8         for (int i = 0; i < n; i++) {
9             while (!stack.isEmpty() && arr[stack.peek()] < arr[i]) {
10                 result[stack.pop()] = arr[i];
11             }
12             stack.push(i);
13         }
14         for (int i = 0; i < n; i++) {
15             System.out.println(arr[i] + " --> " + result[i]);
16         }
17     }
18     public static void main(String[] args) {
19         int[] arr = {4, 5, 2, 25};
20         System.out.println(x:"Output:");
21         nextLargerElement(arr);
22     }
23 }
24
```

Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X

OUTPUT PROBLEMS 10 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

```
4 --> 5
5 --> 25
2 --> 25
25 --> -1
```

PS C:\Users\umash\Desktop\SDE JAVA DSA>

Time Complexity : $O(N)$

17. Print Right View of a Binary Tree:

Given a Binary Tree, the task is to print the Right view of it. The right view of a Binary Tree is a set of rightmost nodes for every level.

Code and output:

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

    Node(int x) {
        data = x;
        left = right = null;
    }
}
Codeium: Refactor | Explain
class rightviewBT {
    Codeium: Refactor | Explain | Generate Javadoc | X
    static void RecursiveRightView(Node root, int level,
        int[] maxLevel, ArrayList<Integer> result) {
        if (root == null)
            return;
        if (level > maxLevel[0]) {
            result.add(root.data);
            maxLevel[0] = level;
        }
        RecursiveRightView(root.right, level + 1,
            maxLevel, result);
        RecursiveRightView(root.left, level + 1,
            maxLevel, result);
    }
    Codeium: Refactor | Explain | Generate Javadoc | X
    static ArrayList<Integer> rightView(Node root) {
        ArrayList<Integer> result = new ArrayList<>();
        int[] maxLevel = new int[] { -1 };
        RecursiveRightView(root, level:0, maxLevel, result);
        return result;
    }
}
Codeium: Refactor | Explain | Generate Javadoc | X
33 static void printArray(ArrayList<Integer> arr) {
34     for (int val : arr) {
35         System.out.print(val + " ");
36     }
37     System.out.println();
38 }
Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
39 public static void main(String[] args) {
40     Node root = new Node(x:1);
41     root.left = new Node(x:2);
42     root.right = new Node(x:3);
43     root.right.left = new Node(x:4);
44     root.right.right = new Node(x:5);
45     ArrayList<Integer> result = rightView(root);
46     printArray(result);
47 }
48 }
49
OUTPUT PROBLEMS 12 TERMINAL PORTS POSTMAN CONSOLE COMMENTS
1 3 5
PS C:\Users\umash\Desktop\SDE JAVA DSA> 
```

Time Complexity: $O(N)$

18. Maximum Depth or Height of Binary Tree :

Given a binary tree, the task is to find the maximum depth or height of the tree. The height of the tree is the number of vertices in the tree from the root to the deepest node.

Code and Output:

```
Codeium: Refactor | Explain
4  class Node {
5      int key;
6      Node left, right;
7      Node(int val) {
8          key = val;
9          left = null;
10         right = null;
11     }
12 }

Codeium: Refactor | Explain
13 class maxdepthBT {
14     Codeium: Refactor | Explain | Generate Javadoc | X
15     static int height(Node root) {
16         if (root == null)
17             return 0;
18         Queue<Node> q = new LinkedList<>();
19         q.add(root);
20         int h = 0;
21         while (!q.isEmpty()) {
22             int size = q.size();
23             for (int i = 0; i < size; i++) {
24                 Node temp = q.poll();
25
26                 if (temp.left != null)
27                     q.add(temp.left);
28                 if (temp.right != null)
29                     q.add(temp.right);
30             }
31             h++;
32         }
33         return h;
34     }
35 }
```

```
34 static Node buildTree() {
35     Scanner sc = new Scanner(System.in);
36     System.out.println("Enter the number of nodes in the binary tree:");
37     int n = sc.nextInt();
38
39     if (n == 0) {
40         return null;
41     }
42     System.out.println("Enter the values in level-order (space separated):");
43     int rootVal = sc.nextInt();
44     Node root = new Node(rootVal);
45     Queue<Node> q = new LinkedList<>();
46     q.add(root);
47     for (int i = 1; i < n; i++) {
48         Node temp = q.poll();
49         System.out.println("Enter left and right children of node " + temp.key + ":");
50         int leftVal = sc.nextInt();
51         int rightVal = sc.nextInt();
52         if (leftVal != -1) {
53             temp.left = new Node(leftVal);
54             q.add(temp.left);
55         }
56         if (rightVal != -1) {
57             temp.right = new Node(rightVal);
58             q.add(temp.right);
59         }
60     }
61     return root;
62 }
```

Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X

```
63 public static void main(String[] args) {
64     Node root = buildTree();
65     int treeHeight = height(root);
66     System.out.println("The height of the binary tree is: " + treeHeight);
67 }
68 }
69
```

OUTPUT PROBLEMS 12 TERMINAL PORTS POSTMAN CONSOLE COMMENTS

Enter the number of nodes in the binary tree:
5
Enter the values in level-order (space separated):
1 2
Enter left and right children of node 1:
3 4
Enter left and right children of node 2:
5 6
Enter left and right children of node 3:
7 8
Enter left and right children of node 4:
9 10
The height of the binary tree is: 4
PS C:\Users\umash\Desktop\SDE JAVA DSA> |

Time Complexity : $O(N)$