CODING PRACTICE PROBLEMS

(Sowmya A) **Date: 12/11/2024**

1. Anagrams

```
import java.util.HashMap;

class anagrams {

    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) {
        String s1 = "geeks";
        String s2 = "kseeg";
        System.out.println(areAnagrams(s1, s2) ? "true" : "false");
    }
}
```

O/P:

```
D:\Java learn>javac anagrams.java

D:\Java learn>java anagrams
true
```

Time Complexity: O(m*log(m) + n*log(n))

2. Row with max 1’s

```
import java.util.*;

class rowmax {

static int R = 4 ;
static int C = 4 ;

static int rowWithMax1s(int mat[][], int R, int C)
{

      boolean flag = true;

    int max_row_index = 0, max_ones = 0;;

    for(int i = 0 ; i < R ; i++){

            int count1 = 0 ;
            for(int j = 0 ; j < C ; j++){
                if(mat[i][j] == 1){
                    count1++;
                    flag = false;
                }
            }
            if(count1>max_ones){
                max_ones = count1;
                max_row_index = i;
            }

        }

      if(flag){
            return -1;
        }

    return max_row_index;
}

    public static void main(String[] args) {

    int mat[][] = { {0, 0, 0, 1},
                    {0, 1, 1, 1},
                    {1, 1, 1, 1},
                    {0, 0, 0, 0}};

    System.out.print("Index of row with maximum 1s is " + rowWithMax1s(mat,R,C));
    }
}
```

O/P:

```
D:\Java learn>javac rowwithmax1.java

D:\Java learn>java rowmax
Index of row with maximum 1s is 2
```

Time Complexity: O(M*N)

3. Longest consecutive subsequence

```
import java.io.*;
import java.util.*;

class longconsequtive{

    static int findLongestConseqSubseq(int arr[], int n)
    {

        // Sort the array
        Arrays.sort(arr);

        int ans = 0, count = 0;

        ArrayList<Integer> v = new ArrayList<Integer>();
        v.add(10);

        for (int i = 1; i < n; i++) {
            if (arr[i] != arr[i - 1])
                v.add(arr[i]);
        }

        for (int i = 0; i < v.size(); i++) {

            // Check if the current element is
            // equal to previous element +1
            if (i > 0 && v.get(i) == v.get(i - 1) + 1)
                count++;
            else
                count = 1;

            // Update the maximum
            ans = Math.max(ans, count);
        }
        return ans;
    }

    public static void main(String[] args)
    {
        int arr[] = { 1, 9, 3, 10, 4, 20, 2 };
        int n = arr.length;
```

```
        System.out.println(
            "Length of the Longest "
            + "contiguous subsequence is "
            + findLongestConseqSubseq(arr, n));
    }
}
```

O/P:

```
D:\Java learn>javac longconsequtive.java

D:\Java learn>java longconsequtive
Length of the Longest contiguous subsequence is 3
```

Time complexity: O(Nlog(N))

4. Longest palindrome in a string

```java
public class longpalindrome {

    static boolean checkPal(String s, int low, int high) {
        while (low < high) {
            if (s.charAt(low) != s.charAt(high))
                return false;
            low++;
            high--;
        }
        return true;
    }

    static String longestPalSubstr(String s) {
        int n = s.length();

        int maxLen = 1, start = 0;

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

                if (checkPal(s, i, j) && (j - i + 1) > maxLen) {
                    start = i;
                    maxLen = j - i + 1;
                }
            }
        }

        return s.substring(start, start + maxLen);
    }

    public static void main(String[] args) {
        String s = "forgeeksskeegfor";
        System.out.println(longestPalSubstr(s));
    }
}
```

O/P:

```
D:\Java learn>javac longpalindrome.java

D:\Java learn>java longpalindrome
geeksskeeg
```

Time complexity: O(N3)

5. Rat in a maze problem

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

class mazePaths {

    static String direction = "DLRU";

    static int[] dr = { 1, 0, 0, -1 };
    static int[] dc = { 0, -1, 1, 0 };

    static boolean isValid(int row, int col, int n,
                           int[][] maze)
    {
        return row >= 0 && col >= 0 && row < n && col < n
            && maze[row][col] == 1;
    }

    static void findPath(int row, int col, int[][] maze,
                         int n, ArrayList<String> ans,
                         StringBuilder currentPath)
    {

        if (row == n - 1 && col == n - 1) {
            ans.add(currentPath.toString());
            return;
        }

        maze[row][col] = 0;

        for (int i = 0; i < 4; i++) {

            int nextrow = row + dr[i];
            int nextcol = col + dc[i];

            if (isValid(nextrow, nextcol, n, maze)) {
                currentPath.append(direction.charAt(i));

                findPath(nextrow, nextcol, maze, n, ans,
                         currentPath);
```

```
                currentPath.deleteCharAt(
                    currentPath.length() - 1);
            }
        }
        maze[row][col] = 1;
    }

    public static void main(String[] args)
    {
        int[][] maze = { { 1, 0, 0, 0 },
                         { 1, 1, 0, 1 },
                         { 1, 1, 0, 0 },
                         { 0, 1, 1, 1 } };

        int n = maze.length;

        ArrayList<String> result = new ArrayList<>();
        StringBuilder currentPath = new StringBuilder();

        if (maze[0][0] != 0 && maze[n - 1][n - 1] != 0) {
            findPath(0, 0, maze, n, result, currentPath);
        }

        if (result.size() == 0)
            System.out.println(-1);
        else
            for (String path : result)
                System.out.print(path + " ");
        System.out.println();
    }
}
```

O/P:

```
D:\Java learn>javac mazepaths.java

D:\Java learn>java mazePaths
DDRDRR DRDDRR
```

Time Complexity: O(3^(m*n))