

## Rishi Aathithya CN, 22IT086

### DSA PRACTICE SET 3

#### 1. Anagram Problem

```
class Main {  
    public static boolean areAnagrams(String s1, String s2) {  
  
        // Your code here  
  
        int[] freq=new int[26];  
        for(char c:s1.toCharArray()){  
            freq[c-'a']++;  
        }  
        for(char c:s2.toCharArray()){  
            freq[c-'a']--;  
        }  
        for(int i=0;i<26;i++){  
            if(freq[i]!=0){  
                return false;  
            }  
        }  
        return true;  
    }  
  
    public static void main(String[] args) {  
        String s1 = "listen";  
        String s2 = "silent";  
        System.out.println(areAnagrams(s1, s2));  
  
        s1 = "hello";  
        s2 = "world";  
        System.out.println(areAnagrams(s1, s2));  
    }  
}
```

```
true  
false  
}
```

## 2. Row With Maximum Ones

```
class Solution {  
    public int[] rowAndMaximumOnes(int[][] mat) {  
        int m = mat.length;  
        int n = mat[0].length;  
        int max = 0;  
        int ind = 0;  
        for (int i = 0; i < m; i++) {  
            int crr = 0;  
            for (int j = 0; j < n; j++) {  
                if (mat[i][j] == 1) {  
                    crr++;  
                }  
            }  
            if (crr > max) {  
                ind = i;  
                max = crr;  
            }  
        }  
        return new int[] { ind, max };  
    }  
    public static void main(String[] args) {  
        Solution solution = new Solution();  
  
        int[][] mat1 = {  
            {1, 0, 1, 1},  
        }  
    }  
}
```

```

        {0, 1, 1, 0},
        {1, 1, 1, 1}
    };

    int[] result1 = solution.rowAndMaximumOnes(mat1);

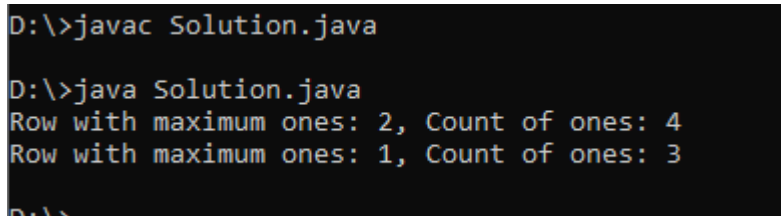
    System.out.println("Row with maximum ones: " + result1[0] + ", Count of ones: " + result1[1]);

    int[][] mat2 = {
        {0, 0, 0},
        {1, 1, 1},
        {0, 1, 0}
    };

    int[] result2 = solution.rowAndMaximumOnes(mat2);

    System.out.println("Row with maximum ones: " + result2[0] + ", Count of ones: " + result2[1]);
}
}

```



```

D:\>javac Solution.java

D:\>java Solution.java
Row with maximum ones: 2, Count of ones: 4
Row with maximum ones: 1, Count of ones: 3
D:\>

```

### 3. Longest consecutive subsequence

```

import java.util.*;

class Solution {

    public int findLongestConseqSubseq(int[] arr) {

        // code here

        Arrays.sort(arr);

        int crr=1;

        int max=1;

        int n=arr.length;

        for(int i=1;i<n;i++){

            if(arr[i]==arr[i-1]+1 ){

```

```

        crr++;
    }else if(arr[i]!=arr[i-1]){
        crr=1;
    }
    max=Math.max(max,crr);
}

return max;
}

public static void main(String[] args) {
    Solution solution = new Solution();

    int[] arr1 = {1, 9, 3, 10, 4, 20, 2};

    System.out.println("Length of longest consecutive subsequence: " +
        solution.findLongestConseqSubseq(arr1));

    int[] arr2 = {36, 41, 56, 35, 37, 34, 33, 42};

    System.out.println("Length of longest consecutive subsequence: " +
        solution.findLongestConseqSubseq(arr2));
}
}

```

```

D:\>java Solution.java
Length of longest consecutive subsequence: 4
Length of longest consecutive subsequence: 5

```

#### 4. Rat in a Maze Problem – I

```

import java.util.*;

class Solution {
    public int[] drow={-1,0,1,0};
    public int[] dcol={0,1,0,-1};
    ArrayList<String> ls=new ArrayList<>();
    public void dfs(int row,int col,int[][] mat,boolean[][] visited,StringBuilder sb){

```

```

        if(row==mat.length-1 && col==mat.length-1){
            ls.add(sb.toString());
            return;
        }
        visited[row][col]=true;
        for(int i=0;i<4;i++){
            int nr=row+drow[i];
            int nc=col+dcol[i];

            if(nr>=0 && nr<mat.length && nc>=0 && nc<mat.length && !visited[nr][nc] &&
mat[nr][nc]==1){
                if(i==0) sb.append('U');
                if(i==1) sb.append('R');
                if(i==2) sb.append('D');
                if(i==3) sb.append('L');
                dfs(nr,nc,mat,visited,sb);
                sb.deleteCharAt(sb.length()-1);
            }
        }
        visited[row][col]=false;

    }

    public ArrayList<String> findPath(int[][] mat) {
        // Your code here
        int n=mat.length;
        boolean[][] visited=new boolean[n][n];
        if(mat[0][0]==1) dfs(0,0,mat,visited,new StringBuilder());
        return ls;
    }

    public static void main(String[] args) {
        Solution solution = new Solution();

```

```
int[][] mat1 = {  
    {1, 0, 0, 0},  
    {1, 1, 0, 1},  
    {0, 1, 0, 0},  
    {1, 1, 1, 1}  
};  
ArrayList<String> paths1 = solution.findPath(mat1);  
System.out.println("Paths to reach the destination: " + paths1);  
}  
}
```

```
D:\>java Solution.java  
Paths to reach the destination: [DRDDRR]
```