# DSA PRACTICE - 3

Name: Tharani Kumar K

DEPT : IT

Date : 12/11/2024

1. **Anagram Strings**

Given two strings **s1** and **s2** consisting of lowercase characters. The task is to check whether two given strings are an anagram 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. For example, act and tac are an anagram of each other. Strings **s1** and **s2** can only contain lowercase alphabets.
Note: You can assume both the strings s1 & s2 are **non-empty**.

**Examples :**

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

**Output:** true

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

**Code:**

```
import java.util.*;
class Anagram{

   public static boolean helper(String s1, String s2) {


      HashMap<Character,Integer> hp=new HashMap<>();
      for(char i:s1.toCharArray()){
         hp.put(i,hp.getOrDefault(i,0)+1);
      }
      for(char i:s2.toCharArray()){
         if(!hp.containsKey(i)) return false;
         else{
            hp.put(i,hp.get(i)-1);
```

```
            }
            if(hp.get(i)==0) hp.remove(i);
        }
        return hp.isEmpty();
    }
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        String s1 = sc.next();
        String s2 = sc.next();
        System.out.println(helper(s1,s2));
    }
}
```

**Output**:

```
C:\Users\thara\.jdks\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
geeks
ekegs
true

Process finished with exit code 0
```

Time Complexity:O(n+m)

Space Complexity:O(k)  (Note: k denotes no of unique characters)

**2.Row With Max 1's**

You need to find and return the index of the first row that has the most number of 1s. If no such row exists, return -1.
Note: 0-based indexing is followed.

Examples:

Input: arr[][] = [[0, 1, 1, 1],
[0, 0, 1, 1],
[1, 1, 1, 1],
[0, 0, 0, 0]]

Output: 2

Explanation: Row 2 contains 4 1's.

**Code:**

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

```
class rowMaxones {
   public static int helper(int[][] arr) {
      int row = arr.length;
      int col = arr[0].length;
      int ind = -1;
      int l = 0;
      int r = col - 1;

      while (l < row && r >= 0) {
         if (arr[l][r] == 0) {
            l++;
         } else {
            ind = l;
            r--;
         }
      }
      return ind;
   }

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

      System.out.print("Enter the number of rows: ");
      int rows = sc.nextInt();
      System.out.print("Enter the number of columns: ");
      int cols = sc.nextInt();

      int[][] arr = new int[rows][cols];
      System.out.println("Enter the matrix elements (0 or 1):");
      for (int i = 0; i < rows; i++) {
         for (int j = 0; j < cols; j++) {
            arr[i][j] = sc.nextInt();
         }
      }

      System.out.println("The row with the maximum number of 1s is: " +
helper(arr));

      sc.close();
```

```
    }
}
```

**Output**:

```
C:\Users\thara\.jdks\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
Enter the number of rows: 2
Enter the number of columns: 2
Enter the matrix elements (0 or 1):
0 1
1 1
The row with the maximum number of 1s is: 1
```

3. **Longest consequtive subsequence**

Given an array **arr** of non-negative integers. Find the **length** of the longest sub-sequence such that elements in the subsequence are consecutive integers, the **consecutive numbers** can be in **any order.**

**Examples:**

**Input:** arr[] = [2, 6, 1, 9, 4, 5, 3]

**Output:** 6

**Explanation:** The consecutive numbers here are 1, 2, 3, 4, 5, 6. These 6 numbers form the longest consecutive subsquence.

**Code:**

```
import java.util.*;
class Longestsubsequence{

    public static int helper(int[] arr) {
        // code here
        int[] sub=new int[100000];
        int max=Integer.MIN_VALUE;
        int c=0;
        for(int i:arr){
            sub[i+1]++;
        }
        for(int i:sub){
            if(i!=0){
```

```
            c++;
            max=Math.max(max,c);
          }
          else c=0;
        }
        return max;
    }
    public static void main(String[] ar){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of Array");
        int n = sc.nextInt();
        System.out.println("Enter the elements of Array");
        int[] arr= new int[n];
        for(int i=0;i<n;i++) {
            arr[i] = sc.nextInt();
        }
        System.out.println(helper(arr));
    }

}
```

**Output**:

```
C:\Users\thara\.jdks\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
Enter the size of Array
6
Enter the elements of Array
2 6 1 9 4 5
3
```

**Time Complexity:O(n)**

**Space Complexity:O(1)**

**4 ) Longest palindrome in a string**

**CODE :**

```
import java.util.*;
public class Main {
   public static void main(String... argv) {
      Scanner scan = new Scanner(System.in);
      System.out.println("Enter the String :");
      String s = scan.next();
      int n = s.length();
      int start = 0;
      int max = 1;
      for(int i=0;i<n;i++){
         int len1 = helper(s,i,i);
         int len2 = helper(s,i,i+1);
         int len = Math.max(len1,len2);
         if(max < len){
            max = len;
            start = i - (len - 1)/2;
         }
      }
      System.out.println ("Result is :" + s.substring(start,start + max));
   }
   public static int helper(String s,int l,int r{
      while(l>=0 && r<s.length() && s.charAt(l) == s.charAt(r)){
         l--;
         r++;
      }
      return r - l - 1;

   }
}
```

OUTPUT:

```
C:\Users\thara\.jdks\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
Enter the String :
aaabbbaa
Result is :aabbbaa
```

**5. Rat in a maze problem**

**CODE :**

```
import java.util.*;
public class Main {
   public static void main(String... argv) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter the Number of rows :");
        int n = scan.nextInt();
        System.out.println("Enter the Number of Columns :");
        int m1 = scan.nextInt();
        int[][] m = new int[n][m1];
        System.out.println("Enter the Elements in the matrix 0 or 1 :");
        for(int i=0;i<n;i++){
          for(int j=0;j<m1;j++){
                m[i][j] = scan.nextInt();
          }
        }
        ArrayList<String> result = new ArrayList<>();
      boolean[][] board = new boolean[n][n];
      if(m[0][0]==0){
        result.add("-1");
        System.out.println(result);
      }else{
           helper(0,0,m,board,n,"",result);
        System.out.println(result);
        }

   }
```

```
    public static void helper(int row,int col,int[][] m,boolean[][] board,int
n,String s,ArrayList<String> result){
        if(row<0 || row>=n || col<0 || col>=n || board[row][col] ||
m[row][col]!=1) return;
        if(row==n-1 && col==n-1 && m[row][col]==1){
            result.add(s);
            return;
        }
        board[row][col] = true;
        helper(row+1,col,m,board,n,s+"D",result);
        helper(row-1,col,m,board,n,s+"U",result);
        helper(row,col+1,m,board,n,s+"R",result);
        helper(row,col-1,m,board,n,s+"L",result);
        board[row][col] = false;
    }
}
```

**Output:**

```
C:\Users\thara\.jdks\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
Enter the Number of rows :
3
Enter the Number of Columns :
3
Enter the Elements in the matrix 0 or 1 :
1 0 0
1 1 0
0 1 1
[DRDR]
```