12/11/2024
ARIKALEESWARAN G
22AM002
CSE(AI&ML)

1.
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.

Program:

```
class Solution {
      public static boolean areAnagrams(String s1, String s2) {
        char[] s1A = s1.toCharArray();
        char[] s2A = s2.toCharArray();
        Arrays.sort(s1A);
        Arrays.sort(s2A);

    return new String(s1A).equals(new String(s2A));
    }
}
```

TC: O(nlogn)
SC: O(n)

2. Given a boolean 2D array, where each row is sorted. Find the row with the maximum number of 1s.

Program:

```
class Sol
{
    public static int maxOnes (int Mat[][], int N, int M)
    {
        int maxm = 0;
        int ind=-1;
        int count=0;
        for(int j=0;j<N;j++){
            count=0;
            for(int i=0;i<M;i++){
                if((Mat[j][i]) == 1) count++;
            }
            if(maxm < count){
                maxm = count;
                ind = j;
            }
        }
        return ind;

    }
}
```

TC: O(N×M)
SC: O(1)

3. Given an array 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.

Program:

```
class Solution {
    public int findLongestConseqSubseq(int[] arr) {
        Arrays.sort(arr);
        int count=1;
        int maxm =0;
        for(int i=0;i<arr.length -1;i++){
            if((arr[i + 1] == arr[i])){
                continue;
            }else if(arr[i + 1] - arr[i] == 1){
                count++;
            }
            else{
                maxm = Math.max(maxm,count);
                count=1;
            }
        }
         maxm = Math.max(maxm, count);
        return maxm;
    }
```

TC: O(nlogn)
SC: O(n)