DSA Practice

Date: 12/11/2024

Problems:

anagram program

row with max 1s'

Longest consequtive subsequence

longest palindrome in a string

rat in a maze problem

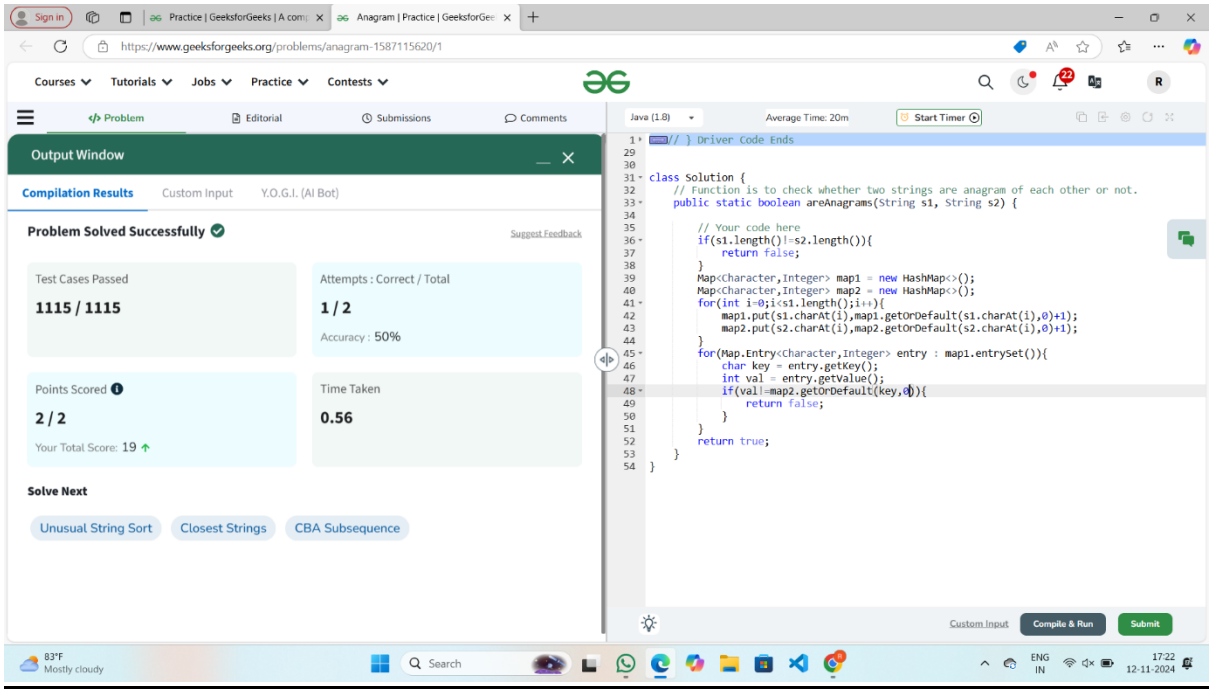
Problem 1:

```
class Solution {
   // Function is to check whether two strings are anagram of each other or not.
   public static boolean areAnagrams(String s1, String s2) {

      // Your code here
      if(s1.length()!=s2.length()){
         return false;
      }
      Map<Character,Integer> map1 = new HashMap<>();
      Map<Character,Integer> map2 = new HashMap<>();
      for(int i=0;i<s1.length();i++){
         map1.put(s1.charAt(i),map1.getOrDefault(s1.charAt(i),0)+1);
         map2.put(s2.charAt(i),map2.getOrDefault(s2.charAt(i),0)+1);
      }
      for(Map.Entry<Character,Integer> entry : map1.entrySet()){
         char key = entry.getKey();
```

```
            int val = entry.getValue();
            if(val!=map2.getOrDefault(key,0)){
                return false;
            }
        }
        return true;
    }
}
```

Output:

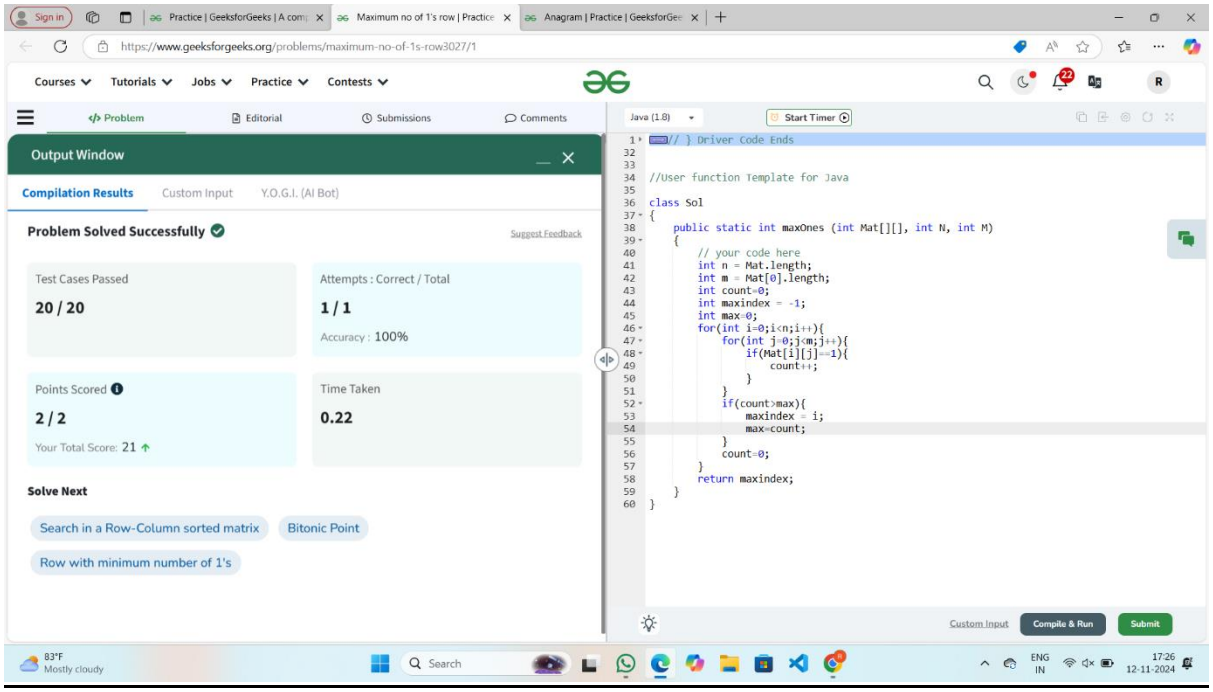


Problem 2:

```
class Sol
{
    public static int maxOnes (int Mat[][], int N, int M)
    {
        // your code here
        int n = Mat.length;
```

```
        int m = Mat[0].length;
        int count=0;
        int maxindex = -1;
        int max=0;
        for(int i=0;i<n;i++){
            for(int j=0;j<m;j++){
                if(Mat[i][j]==1){
                    count++;
                }
            }
            if(count>max){
                maxindex = i;
                max=count;
            }
            count=0;
        }
        return maxindex;
    }
}
```

Output:



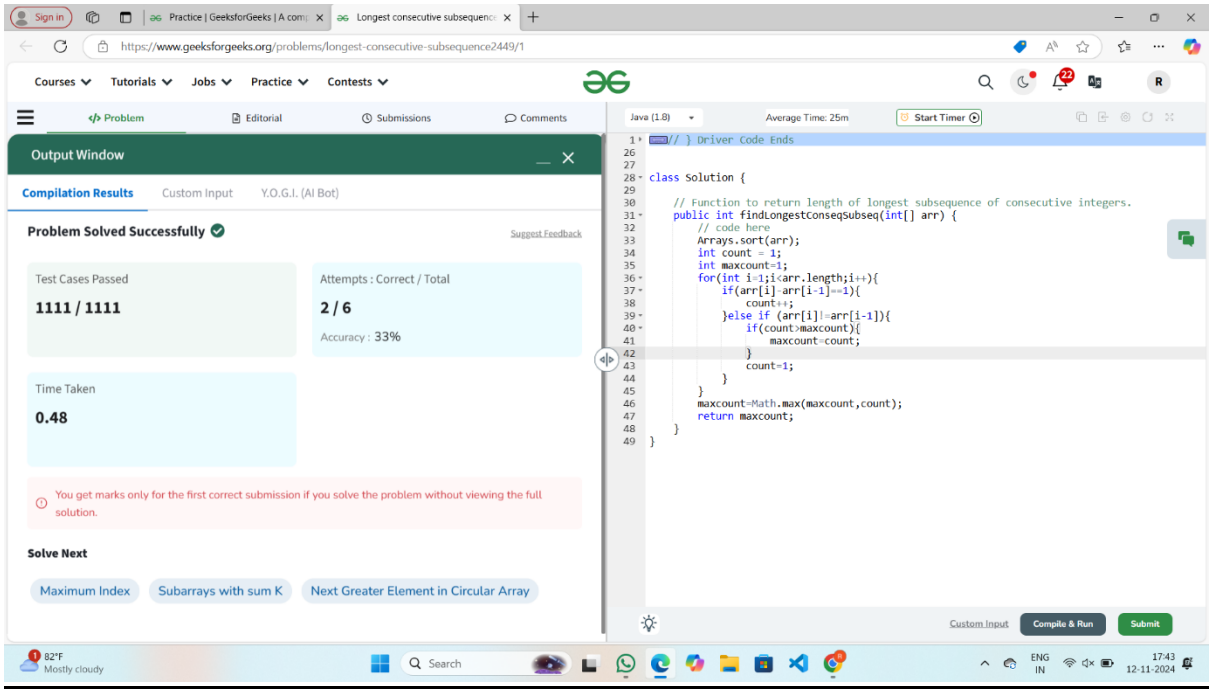
Problem 3:

```
class Solution {


    // Function to return length of longest subsequence of consecutive integers.
    public int findLongestConseqSubseq(int[] arr) {
        // code here
        Arrays.sort(arr);
        int count = 1;
        int maxcount=1;
        for(int i=1;i<arr.length;i++){
            if(arr[i]-arr[i-1]==1){
                count++;
            }else if (arr[i]!=arr[i-1]){
                if(count>maxcount){
                    maxcount=count;
                }
```

```
            count=1;
          }
        }
        maxcount=Math.max(maxcount,count);
        return maxcount;
    }
}
```

Output:

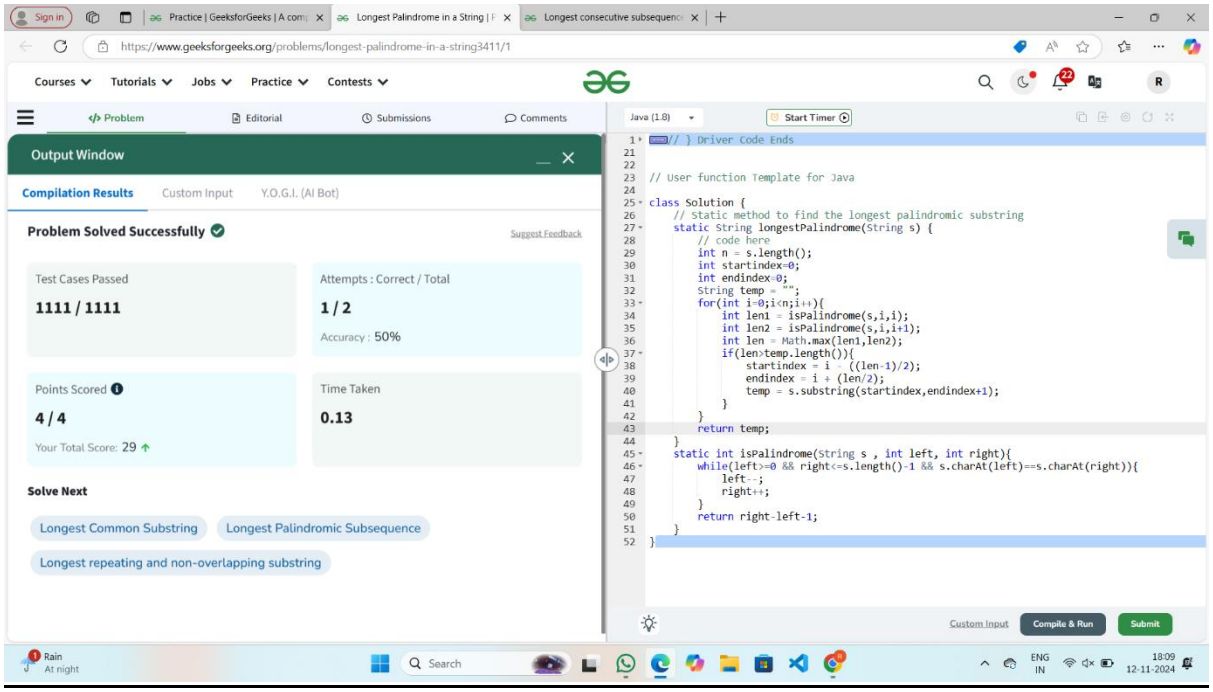


Problem 4:

```
class Solution {
    // Static method to find the longest palindromic substring
    static String longestPalindrome(String s) {
        // code here
        int n = s.length();
        int startindex=0;
        int endindex=0;
```

```
        String temp = "";
        for(int i=0;i<n;i++){
            int len1 = isPalindrome(s,i,i);
            int len2 = isPalindrome(s,i,i+1);
            int len = Math.max(len1,len2);
            if(len>temp.length()){
                startindex = i - ((len-1)/2);
                endindex = i + (len/2);
                temp = s.substring(startindex,endindex+1);
            }
        }
        return temp;
    }
    static int isPalindrome(String s , int left, int right){
        while(left>=0 && right<=s.length()-1 && s.charAt(left)==s.charAt(right)){
            left--;
            right++;
        }
        return right-left-1;
    }
}
```

Output:

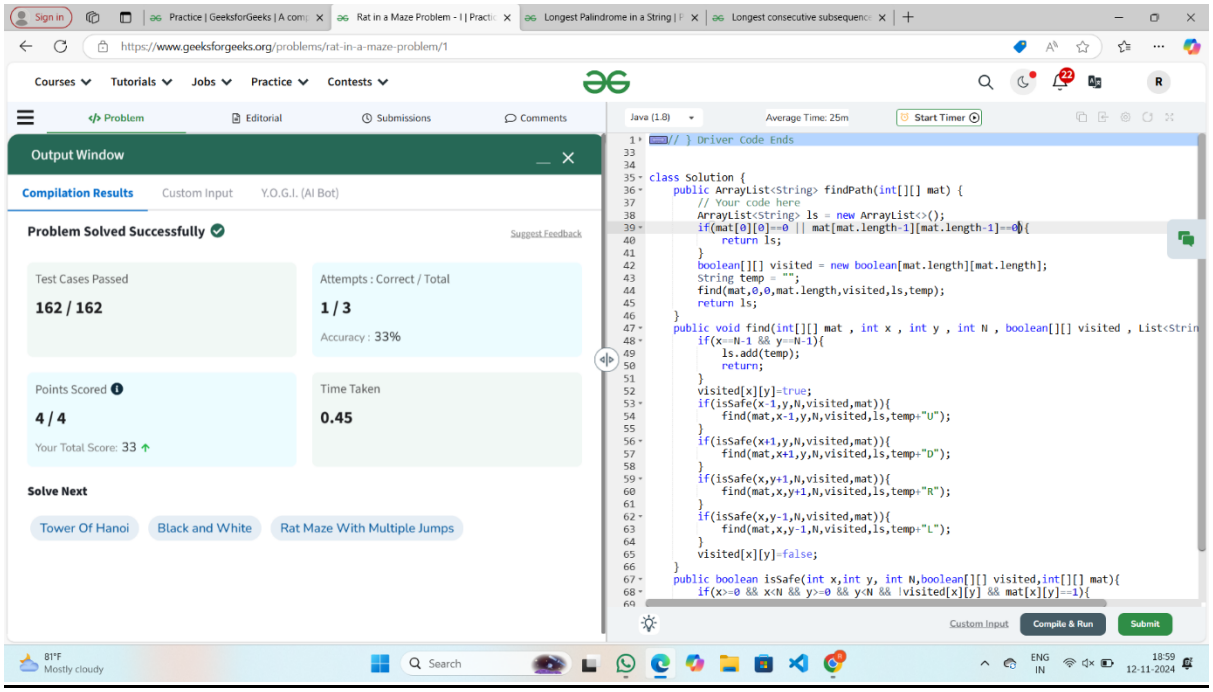


Problem 5:

```
class Solution {
    public ArrayList<String> findPath(int[][] mat) {
        // Your code here
        ArrayList<String> ls = new ArrayList<>();
        if(mat[0][0]==0 || mat[mat.length-1][mat.length-1]==0){
            return ls;
        }
        boolean[][] visited = new boolean[mat.length][mat.length];
        String temp = "";
        find(mat,0,0,mat.length,visited,ls,temp);
        return ls;
    }
    public void find(int[][] mat , int x , int y , int N , boolean[][] visited ,
List<String> ls,String temp){
        if(x==N-1 && y==N-1){
```

```
            ls.add(temp);
            return;
        }
        visited[x][y]=true;
        if(isSafe(x-1,y,N,visited,mat)){
            find(mat,x-1,y,N,visited,ls,temp+"U");
        }
        if(isSafe(x+1,y,N,visited,mat)){
            find(mat,x+1,y,N,visited,ls,temp+"D");
        }
        if(isSafe(x,y+1,N,visited,mat)){
            find(mat,x,y+1,N,visited,ls,temp+"R");
        }
        if(isSafe(x,y-1,N,visited,mat)){
            find(mat,x,y-1,N,visited,ls,temp+"L");
        }
        visited[x][y]=false;
    }
    public boolean isSafe(int x,int y, int N,boolean[][] visited,int[][] mat){
        if(x>=0 && x<N && y>=0 && y<N && !visited[x][y] && mat[x][y]==1){
            return true;
        }
        return false;
    }
}
```

Output:


Practice | GeeksforGeeks | A comp
Rat in a Maze Problem - I | Practi
Longest Palindrome in a String | F
Longest consecutive subsequenc
https://www.geeksforgeeks.org/problems/rat-in-a-maze-problem/1
Courses
Tutorials
Jobs
Practice
Contests
Problem
Editorial
Submissions
Comments
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
162 / 162
Attempts : Correct / Total
1 / 3
Accuracy : 33%
Points Scored
4 / 4
Your Total Score: 33
Time Taken
0.45
Solve Next
Tower Of Hanoi
Black and White
Rat Maze With Multiple Jumps
Java (1.8)
Average Time: 25m
Start Timer
// } Driver Code Ends
class Solution {
public ArrayList<String> findPath(int[][] mat) {
// Your code here
ArrayList<String> ls = new ArrayList<>();
if(mat[0][0]==0 || mat[mat.length-1][mat.length-1]==0){
return ls;
}
boolean[][] visited = new boolean[mat.length][mat.length];
String temp = "";
find(mat,0,0,mat.length,visited,ls,temp);
return ls;
}
public void find(int[][] mat , int x , int y , int N , boolean[][] visited , List<Strin
if(x==N-1 && y==N-1){
ls.add(temp);
return;
}
visited[x][y]=true;
if(isSafe(x-1,y,N,visited,mat)){
find(mat,x-1,y,N,visited,ls,temp+"U");
}
if(isSafe(x+1,y,N,visited,mat)){
find(mat,x+1,y,N,visited,ls,temp+"D");
}
if(isSafe(x,y+1,N,visited,mat)){
find(mat,x,y+1,N,visited,ls,temp+"R");
}
if(isSafe(x,y-1,N,visited,mat)){
find(mat,x,y-1,N,visited,ls,temp+"L");
}
visited[x][y]=false;
}
public boolean isSafe(int x,int y, int N,boolean[][] visited,int[][] mat){
if(x>=0 && x<N && y>=0 && y<N && !visited[x][y] && mat[x][y]==1){
Custom Input
Compile & Run
Submit
81°F
Mostly cloudy
Search
ENG
IN
18:59
12-11-2024