

DSA Practice

Date: 12/11/2024

Problems:

anagram program

row with max 1s'

Longest consecutive 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:

The screenshot shows the GeeksforGeeks website interface for the 'Anagram' problem. The left sidebar indicates the problem was solved successfully with 1115/1115 test cases passed, 1/2 attempts, 50% accuracy, and a time taken of 0.56. The right pane displays the Java code for the 'areAnagrams' function, which uses two HashMaps to compare character frequencies of two strings.

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:

The screenshot shows the GeeksforGeeks website interface. The top navigation bar includes links for Courses, Tutorials, Jobs, Practice, and Contests. The main content area displays the 'Output Window' for a problem titled 'Maximum no of 1's row'. The problem status is 'Problem Solved Successfully' with a green checkmark. The test cases passed are 20/20, and the accuracy is 100%. The points scored are 2/2, and the time taken is 0.22. The 'Solve Next' section suggests other problems like 'Search in a Row-Column sorted matrix' and 'Bitonic Point'. The code editor on the right shows a Java solution for the problem, which is a function to find the maximum number of 1s in a row of a matrix.

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:

The screenshot shows the GeeksforGeeks online IDE interface. On the left, the 'Output Window' displays the following information:

- Compilation Results:** Problem Solved Successfully ✓
- Test Cases Passed:** 1111 / 1111
- Attempts:** Correct / Total: 2 / 6
- Accuracy:** 33%
- Time Taken:** 0.48
- Solve Next:** Maximum Index, Subarrays with sum K, Next Greater Element in Circular Array

On the right, the Java code for the solution is displayed:

```

1 // } Driver Code Ends
26
27
28 class Solution {
29
30 // Function to return length of longest subsequence of consecutive integers.
31 public int findLongestConseqSubseq(int[] arr) {
32 // code here
33 Arrays.sort(arr);
34 int count = 1;
35 int maxcount=1;
36 for(int i=1;i<arr.length;i++){
37 if(arr[i]-arr[i-1]==1){
38 count++;
39 }else if (arr[i]!=arr[i-1]){
40 if(count>maxcount){
41 maxcount=count;
42 }
43 count=1;
44 }
45 }
46 maxcount=Math.max(maxcount,count);
47 return maxcount;
48 }
49 }

```

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;

```

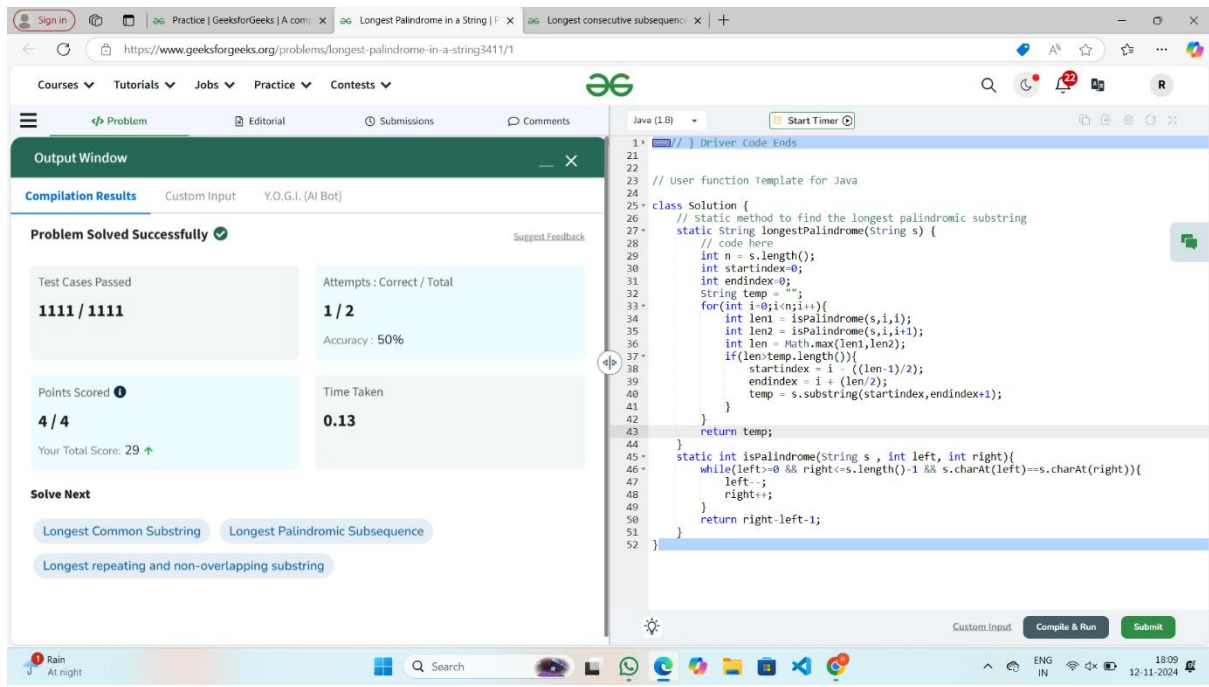
```

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:

Sign in

Practice | GeeksforGeeks | A com X

Rat in a Maze Problem - I | Practi X

Longest Palindrome in a String | F X

Longest consecutive subsequen X

+

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

```
1 // Driver Code Ends
2
3
4
5 class Solution {
6     public ArrayList<String> findPath(int[][] mat) {
7         // Your code here
8         ArrayList<String> ls = new ArrayList<>();
9         if(mat[0][0]==0 || mat[mat.length-1][mat.length-1]==0){
10             return ls;
11         }
12         boolean[][] visited = new boolean[mat.length][mat.length];
13         String temp = "";
14         find(mat,0,0,mat.length,visited,ls,temp);
15         return ls;
16     }
17     public void find(int[][] mat , int x , int y , int N , boolean[][] visited , List<String> ls){
18         if(x==N-1 && y==N-1){
19             ls.add(temp);
20             return;
21         }
22         visited[x][y]=true;
23         if(isSafe(x+1,y,N,visited,mat)){
24             find(mat,x+1,y,N,visited,ls,temp+"D");
25         }
26         if(isSafe(x-1,y,N,visited,mat)){
27             find(mat,x-1,y,N,visited,ls,temp+"U");
28         }
29         if(isSafe(x,y+1,N,visited,mat)){
30             find(mat,x,y+1,N,visited,ls,temp+"R");
31         }
32         if(isSafe(x,y-1,N,visited,mat)){
33             find(mat,x,y-1,N,visited,ls,temp+"L");
34         }
35         visited[x][y]=false;
36     }
37     public boolean isSafe(int x,int y , int N,boolean[][] visited,int[][] mat){
38         if(x<=0 && x<N && y>=0 && y<N && !visited[x][y] && mat[x][y]==1){
39             return true;
40         }
41         return false;
42     }
43 }
```

Custom Input

Compile & Run

Submit

81°F

Mostly cloudy

Search

ENG

IN

18:59

12-11-2024