**Longest Common Subsequence :-**

Medium Accuracy: 41.68% Submissions: 227K+ Points: 4

Given two strings, find the length of longest subsequence present in both of them. Both the strings are in **uppercase**latin alphabets.

**Example 1:**

**Input:**

A = 6, B = 6

str1 = ABCDGH

str2 = AEDFHR

**Output:** 3

**Explanation:** LCS for input strings “ABCDGH” and “AEDFHR” is “ADH” of length 3.

**Example 2:**

**Input:**

A = 3, B = 2

str1 = ABC

str2 = AC

**Output:** 2

**Explanation:** LCS of "ABC" and "AC" is "AC" of length 2.

**Your Task:**  
Complete the function **lcs()** which takes the length of two strings respectively and two strings as input parameters and returns the length of the longest subsequence present in both of them.

**Expected Time Complexity** : O(|str1|\*|str2|)  
**Expected Auxiliary Space**: O(|str1|\*|str2|)

**Constraints:**  
1<=size(str1),size(str2)<=103

**Code :-**

//{ Driver Code Starts

#include<bits/stdc++.h>

const int mod=1e9+7;

using namespace std;

// } Driver Code Ends

// function to find longest common subsequence

class Solution

{

public:

int func(int i, int j, string &s1, string &s2, vector<vector<int>> &dp){

//base case

if(i>=s1.size() || j>=s2.size())

return 0;

//if answer found, just return it

if(dp[i][j] != -1)

return dp[i][j];

//recursive calls if character matches

if(s1[i]==s2[j])

return dp[i][j] = 1 + func(i+1, j+1, s1, s2, dp);

//recursive calls if character not matches

else

return dp[i][j] = max( func(i+1, j, s1, s2, dp), func(i, j+1, s1, s2, dp) );

}

//Function to find the length of longest common subsequence in two strings.

int lcs(int n, int m, string s1, string s2){

vector<vector<int>> dp(n, vector<int>(m,-1));

return func(0,0,s1,s2,dp);

}

};

//{ Driver Code Starts.

int main()

{

int t,n,m;

cin>>t;

while(t--)

{

cin>>n>>m; // Take size of both the strings as input

string s1,s2;

cin>>s1>>s2; // Take both the string as input

Solution ob;

cout << ob.lcs(n, m, s1, s2) << endl;

}

return 0;

}

// } Driver Code Ends

**T.C :- O(n \* m)**

**S.C :- O(n \* m)**