**Longest Common Substring**

**Medium**Accuracy: 52.09% Submissions: 34456 Points: 4

Given two strings. The task is to find the length of the longest common substring.

**Example 1:**

**Input:** S1 = "ABCDGH", S2 = "ACDGHR"

**Output:** 4

**Explanation**: The longest common substring

is "CDGH" which has length 4.

**Example 2:**

**Input**: S1 = "ABC", S2 "ACB"

**Output:** 1

**Explanation**: The longest common substrings

are "A", "B", "C" all having length 1.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **longestCommonSubstr()**which takes the string S1, string S2 and their length n and m as inputs and returns the length of the longest common substring in S1 and S2.

**Expected Time Complexity:**O(n\*m).  
**Expected Auxiliary Space:**O(n\*m).

**Constraints:**  
1<=n, m<=1000

class Solution{

    public:

    /\*int substring(int i, int j, string s1, string s2, int ct) {

        if (i<0 or j<0) return 0;

        if (s1[i]==s2[j]) return substring(i-1, j-1, s1, s2, ct+1);

        int x=substring(i, j-1, s1, s2, ct);

        int y=substring(i-1, j, s1, s2, ct);

        return max(ct, max(x, y));

    }\*/

    int longestCommonSubstr (string S1, string S2, int n, int m){

        // your code here

        //return substring(n-1, m-1, S1, S2, 0);

        int dp[n+1][m+1];

        memset(dp, 0, sizeof(dp));

        int ans=0;

        for (int i=1; i<n+1; i++) {

            for (int j=1; j<m+1; j++) {

                if (S1[i-1]==S2[j-1]) {

                    dp[i][j]=1+dp[i-1][j-1];

                    ans=max(ans, dp[i][j]);

                }

                else dp[i][j]=0;

            }

        }

        return ans;

    }

};