**Problem Statement:**

﻿Imagine you are developing a software application for an educational tool that helps students learn basic arithmetic. One of the features of this application is to calculate the sum of the first n natural numbers. The application needs to handle large values of n efficiently.

To ensure the application works correctly, you need to implement a function that computes the sum of the first n natural numbers using recursion. This function will be used to provide feedback to students and verify their answers in real-time.

Your task is to write a recursive function that takes an integer n as input and returns the sum of the first n natural numbers. The function should be optimized to handle values of n up to 1,000,000 efficiently.

**Example:**

Input:

10

Output:

55

**Explanation:** The sum of the first 10 natural numbers is

1+2+3+...+10=55

**Problem Title: Document Comparison using Longest Common Subsequence**

**Topic:** Dynamic Programming  
**Subtopic:** Longest Common Subsequence (LCS)  
**Difficulty Level:** Medium

**Problem Statement:**

**Scenario:**

John is working on a document comparison tool that helps authors track the similarity between different versions of their documents. He needs a program to determine the **Longest Common Subsequence (LCS)** between two different versions of the document. The LCS is the longest sequence that appears in the same relative order in both documents but does not need to be consecutive.

For example, given two document versions:

* Version 1: abcde
* Version 2: ace

The longest common subsequence between these two versions is "ace", which has a length of 3. The subsequence appears in the same order in both documents, even though the characters are not contiguous.

**Input format:**

The first line of input consists of a string str1, representing the first version of the document.

The second line consists of a string str2, representing the second version of the document.

**Output format:**

* The first line of output displays: "The LCS of the two documents is X", where X is the Longest Common Subsequence.
* The second line of output displays: "The length of the LCS is Y", where Y is the length of the LCS.

**Example:**

**Input 1:**

Copy code

abcde

ace

**Output 1:**

csharp

Copy code

The LCS of the two documents is ace

The length of the LCS is 3

**Explanation:**

The longest common subsequence between abcde and ace is "ace", and the length is 3.

**Input 2:**

javascript

Copy code

document

comentary

**Output 2:**

csharp

Copy code

The LCS of the two documents is coment

The length of the LCS is 6

**Constraints:**

* 1 <= str1.length, str2.length <= 1000
* str1 and str2 consist of lowercase English letters only.

**Test Cases:**

**Easy Test Case:**

1. **Input:**

Copy code

abc

aebc

**Output:**

csharp

Copy code

The LCS of the two documents is abc

The length of the LCS is 3

**Medium Test Case:**

1. **Input:**

Copy code

abcxyz

axcybz

**Output:**

csharp

Copy code

The LCS of the two documents is acz

The length of the LCS is 3

**Hard Test Case:**

1. **Input:**

Copy code

encyclopedia

archaeological

**Output:**

csharp

Copy code

The LCS of the two documents is eol

The length of the LCS is 3

**Solution Code in Java:**

java

Copy code

import java.util.Scanner;

class LongestCommonSubsequence {

public static String findLCS(String str1, String str2) {

int m = str1.length();

int n = str2.length();

int[][] dp = new int[m + 1][n + 1];

// Fill dp array

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

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

if (str1.charAt(i - 1) == str2.charAt(j - 1)) {

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

} else {

dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);

}

}

}

// Reconstruct LCS

StringBuilder lcs = new StringBuilder();

int i = m, j = n;

while (i > 0 && j > 0) {

if (str1.charAt(i - 1) == str2.charAt(j - 1)) {

lcs.append(str1.charAt(i - 1));

i--;

j--;

} else if (dp[i - 1][j] > dp[i][j - 1]) {

i--;

} else {

j--;

}

}

return lcs.reverse().toString();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input

String str1 = scanner.nextLine();

String str2 = scanner.nextLine();

// Find LCS

String lcs = findLCS(str1, str2);

int lcsLength = lcs.length();

// Output

System.out.println("The LCS of the two documents is " + lcs);

System.out.println("The length of the LCS is " + lcsLength);

}

}

**Solution Code in C++:**

cpp

Copy code

#include <iostream>

#include <vector>

#include <string>

using namespace std;

class LongestCommonSubsequence {

public:

string findLCS(string str1, string str2) {

int m = str1.size();

int n = str2.size();

vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));

// Fill dp array

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

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

if (str1[i - 1] == str2[j - 1]) {

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

} else {

dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);

}

}

}

// Reconstruct LCS

string lcs = "";

int i = m, j = n;

while (i > 0 && j > 0) {

if (str1[i - 1] == str2[j - 1]) {

lcs = str1[i - 1] + lcs;

i--;

j--;

} else if (dp[i - 1][j] > dp[i][j - 1]) {

i--;

} else {

j--;

}

}

return lcs;

}

};

int main() {

LongestCommonSubsequence solution;

string str1, str2;

// Input

cin >> str1 >> str2;

// Find LCS

string lcs = solution.findLCS(str1, str2);

int lcsLength = lcs.length();

// Output

cout << "The LCS of the two documents is " << lcs << endl;

cout << "The length of the LCS is " << lcsLength << endl;

return 0;

}