Yusuf Fawzy Elnady

Edit Distance (ToP-Down && BoTTom-UP)

Summary:

Given two strings **str1** and **str2**, with the ability to perform only the (**insert**, **remove**, **replace**) **operations**. This program finds the minimum **number of edits** (**operations**) required **to convert 'str1' into 'str2'.** The idea is to process all characters one by one staring from either from left or right sides of both strings.

If we traversed from right corner. There are two possibilities for every pair of character being traversed:
* If **last characters of two strings are same**, **nothing** much **to do.** Ignore last characters and get count for remaining strings. So, we recur for lengths m-1 and n-1.
** If **last characters are not same**, we **consider all operations on 'str1'**, consider all three operations on last character of first string, recursively compute minimum cost for all three operations and take minimum of three values.

Without any memoization, the **time complexity** of above solution is **exponential**. In worst case, we may end up doing **O(3^m) operations**. The **worst case** happens when **none of characters of two strings match.**

The Operations:

- **Insert**: Recur for m and n-1
- **Remove**: Recur for m-1 and n
- **Replace**: Recur for m-1 and n-1

Three Versions of the Function Edit Distance:

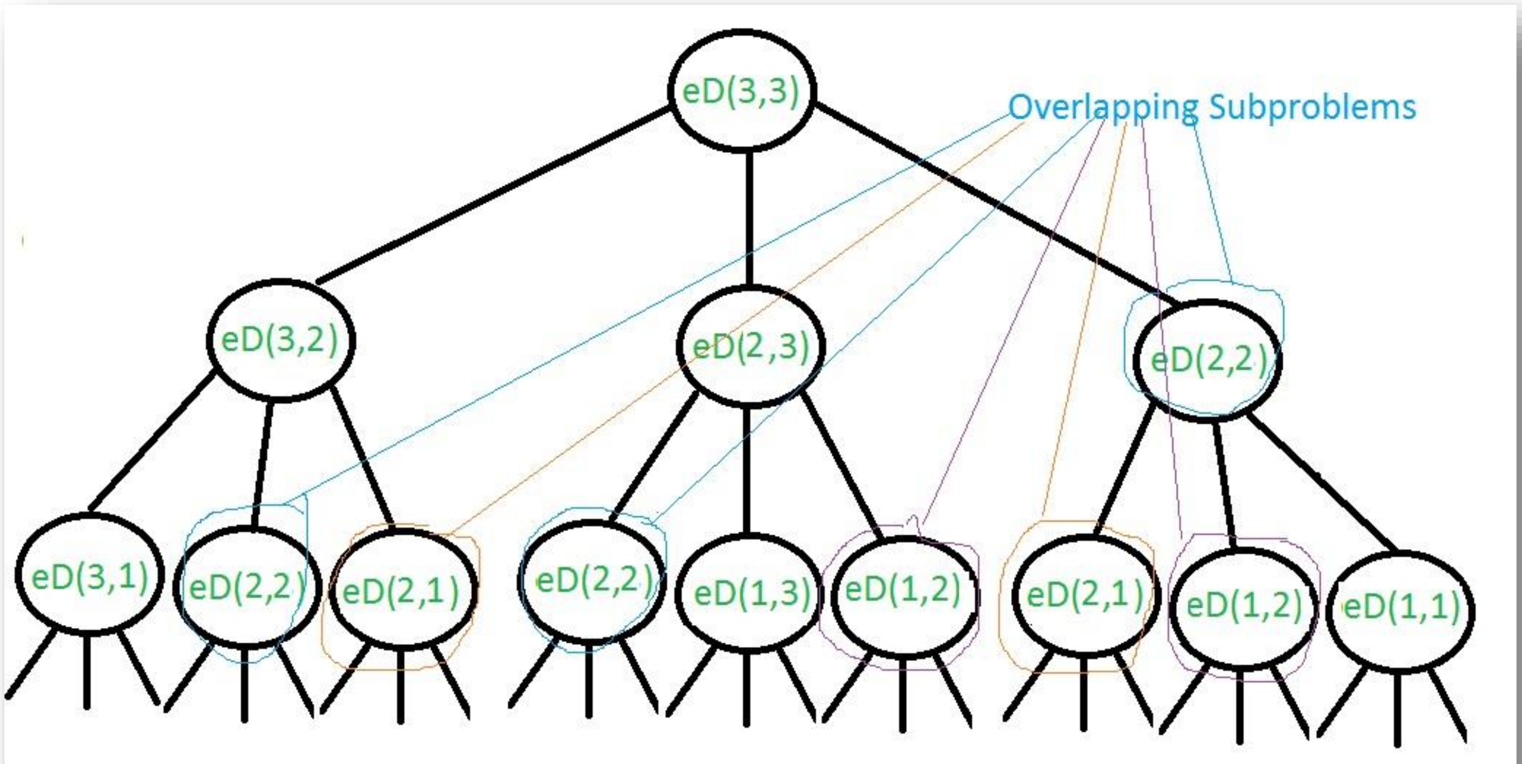
1- Edit Distance - Without DP
2- Edit Distance - DP - Top-Down
3- Edit Distance - DP - Bottom-UP

The program also shows you **the steps** you should follow **to convert str1 into str2** using a minimal number of operations with the help of **printSol()** function.

The project is built using **C++, Visual Studio 2019**.

Yusuf Fawzy Elnady

Below is a recursive call diagram for worst case:



Worst case recursion tree when m = 3, n = 3.
Worst case example str1="abc" str2="xyz"