Edit distance

Edit distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another, by counting the minimum number of operations required to transform one string into the other. The distance operations are the removal, insertion, or substitution of character in the string. All of the above operations are of equal cost.

Examples:

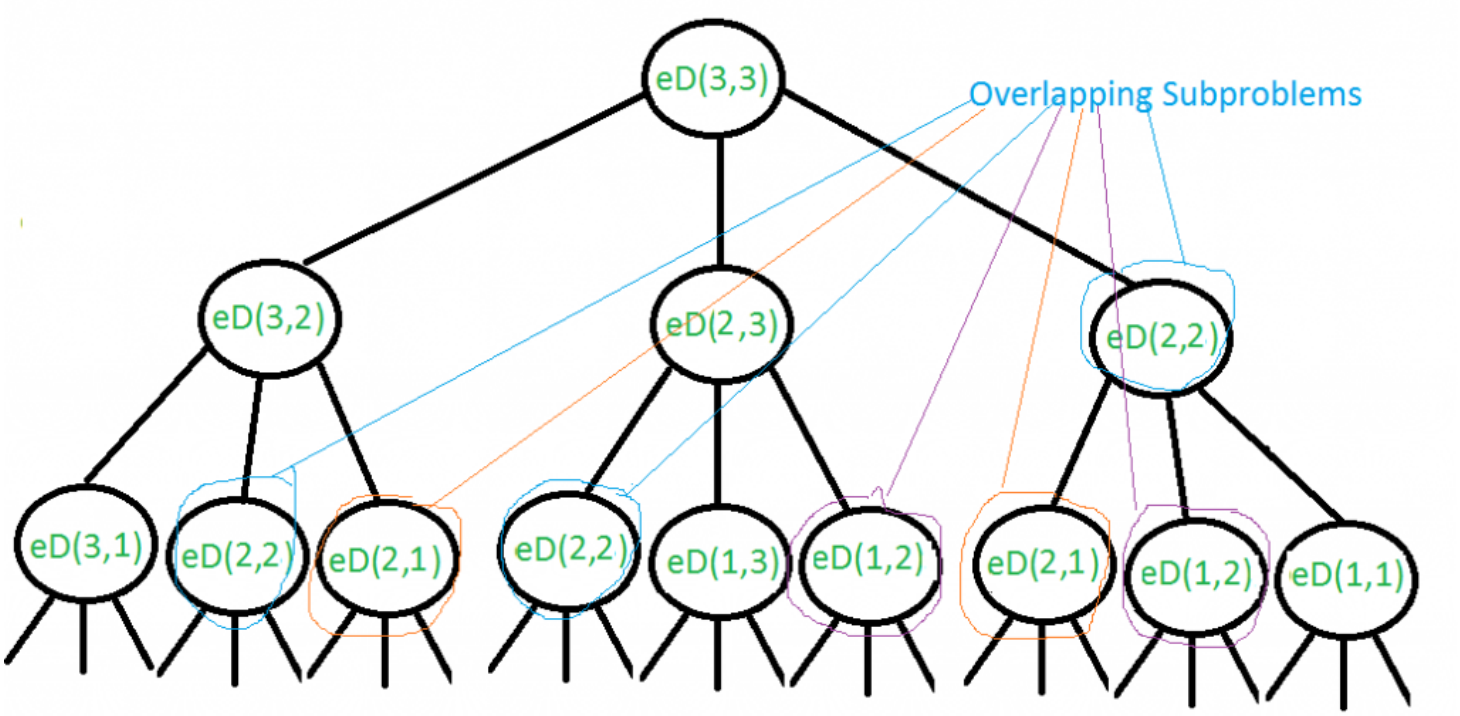
```
Input:   str1 = "geek", str2 = "gesek"
Output:  1
We can convert str1 into str2 by inserting a 's'.

Input:   str1 = "cat", str2 = "cut"
Output:  1
We can convert str1 into str2 by replacing 'a' with 'u'.

Input:   str1 = "sunday", str2 = "saturday"
Output:  3
Last three and first characters are same.  We basically
need to convert "un" to "atur".  This can be done using
below three operations.

Replace 'n' with 'r', insert t, insert a
```

Sub-problems in this case?

The idea is process all characters one by one staring from either from left or right sides of both strings.

```
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.
Else (If last characters are not same), we consider all operations on 'str1', consider all three
        Insert: Recur for m and n-1
        Remove: Recur for m-1 and n
        Replace: Recur for m-1 and n-1
```

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. 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"

We can see that many subproblems are solved, again and again, for example, eD(2,2) is called three times. Since same suproblems are called again, this problem has Overlapping Subprolems property. Like other typical Dynamic Programming(DP) problems, recomputations of same subproblems can be avoided by constructing a temporary array that stores results of subproblems.

Time complexity

Time Complexity: O(m x n)
Auxiliary Space: O(m x n)