# Optimized Edit distance problem Report

## Problem :

The problems faced in the standard edit distance algorithm was of time and space complexity issues.

for example the edit distance between words (fondam, albanian) made a 6*8 matrix of the memory. + 2 initialization arrays.

Which held place in memory.
Specially if words or sequences grow large.

| | | a | l | b | a | n | i | a | n |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| f | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| o | 2 | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| n | 3 | 3 | 3 | 3 | 4 | 4 | 5 | 6 | 7 |
| d | 4 | 4 | 4 | 4 | 4 | 5 | 5 | 6 | 7 |
| a | 5 | 4 | 5 | 5 | 4 | 5 | 6 | 5 | 6 |
| m | 6 | 5 | 5 | 6 | 5 | 5 | 6 | 6 | 6 |

## Proposed Solution :

An algorithm which is space efficient compared to the standard.

Notice that each cell in the matrix **ONLY** need 3 surrounding cells to obtain the minimum result wanted using the minimum function

E(i, j) = min( [E(i-1, j) + D], [E(i, j-1) + I], [E(i-1, j-1) + R if i,j characters are not same] )

So if we managed to do a linear computation of for each cell

A one dimensional array the holds the previous needed characters.

This array's length is determined by the length of the smaller phrase+2.

For example in this edit distance between BOLT and ALT

| | B | O | L | T |
|---|---|---|---|---|
| A | 0 | 3 | 6 | 9 |
| L | 1 | 4 | 7 | 10 |
| T | 2 | 5 | 8 | 11 |

if we presented the matrix linearly.

If we look at cell 4 or cell 8

we need just three cells for each computation

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|

| 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|

so working with a **one Dimensional array of length = string length+1** holding the last 5 (for this example) entries of the sequence

**Minimum function ( array[0] + diff, array[0] +1, arr[length-1] +1 )**

and swapping array elements after each iteration.

## Problems faced in the proposed solution:

1- The initialization arrays [ the first column and first row in the standard matrix ] .

2- The traceback of the shortest edit. (Next Phase )

## Solution of the first problem was by also using 2 one dimensional arrays.

## The initialization arrays : Array A & Array B :

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | arrA[4] |
| | | | | |
| | | | | |
| | | | | |
| 0 | 1 | 1 | 0 | arrB[4] |

Both of length 4. used to solve the initialization problem and compute the minimum and feed them to the main array in time and place when and where needed.

## Cost of standard vs Cost of enhanced:

If we take **(Alt, Bolt)** example **(3,4)**

we find that as the **standard matrix** held 4x5 matrix = **20**

the **enhanced costs** is 4x2 + 5 = **13**

almost insignificant when comparing three letters with a four letters words. But when the difference is compared with a longer example.

**(10,10) :**

**standard** : 11x11 = **121**

**enhanced**: 4x2 + 12 = **20**