Report for implementation assignment 2
Group Member: Hao Wang

## *1.Pseudo-code*

```
Def Align(str1, str2):
    m = len(str1) + 1
    n = len(str2) + 1
    D[0, 0] = 0
    for i in range(1, m):
        D[i, 0] = costDel(str1[i - 1]) + D[i - 1, 0]
        ptr[i, 0] = Left
    for j in range(1, n):
        D[0, j] = costInsert(str2[j - 1]) + D[0, j - 1]
        ptr[0, j] = Down
    for i in range(1, m):
        for j in range(1, n):
            cost = costSub( str1[i - 1], str2[j - 1] )
            option1:
                D[i, j] = D[i, j -1] + costInsert(str2[j - 1])
                ptr[i, j] = Down
            option2:
                D[i, j] = D[i - 1, j] + costDel(str1[i - 1])
                ptr[i, j] = Left
            option3:
                D[i, j] = D[i - 1, j - 1] + cost
                ptr[i, j] = Diag
            D[i, j] = min(option1, option2, option3)

    minCost = D[i, j]
    while 1:
        if(i == 0 and j == 0):
            break
        if(ptr[i ,j] == Down):
            finalStr1 += "-"
            finalStr2 += str2[j - 1]
            j -= 1
        else if(ptr[i , j] == Left):
            finalStr1 += str1[i - 1]
            finalStr2 += "-"
            i -= 1
```

```
else:
        finalStr1 += str1[i - 1]
        finalStr2 += str2[j - 1]
        i -= 1
        j -= 1
```

### *2.Asymptotic Analysis of run time*

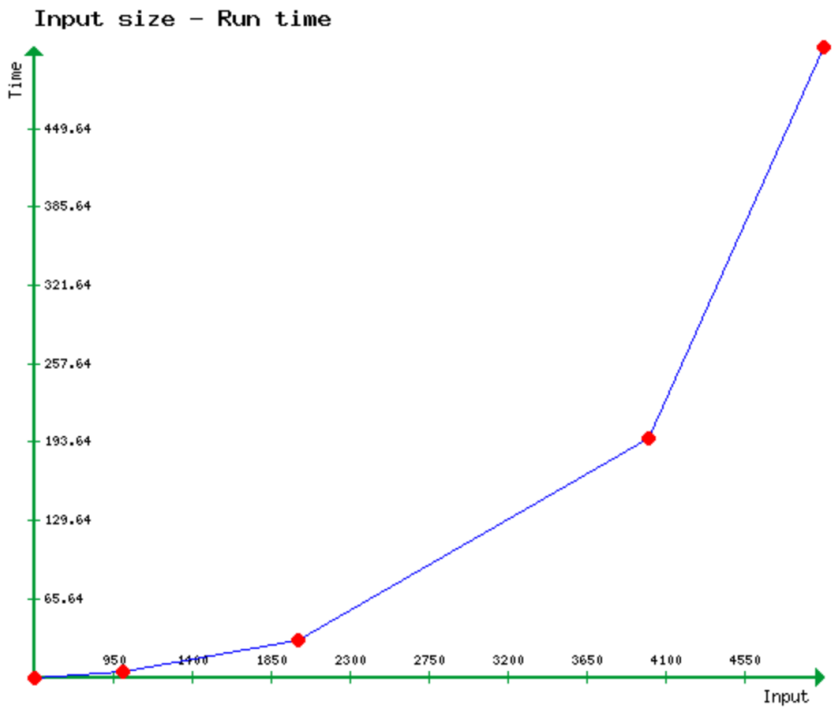
This is a program that use dynamic programing, D[i,j] = min(D[i - 1, j] + cost1, D[i, j - 1] + cost2, D[i - 1, j - 1] + cost3)
Finding the optimal cost should run in O(mn) m is the length of first string, n is the length of second string.
Finding the alignment should run in O(m + n)
overall, program should run in O(mn)

### *3. Reporting and plotting the run time*

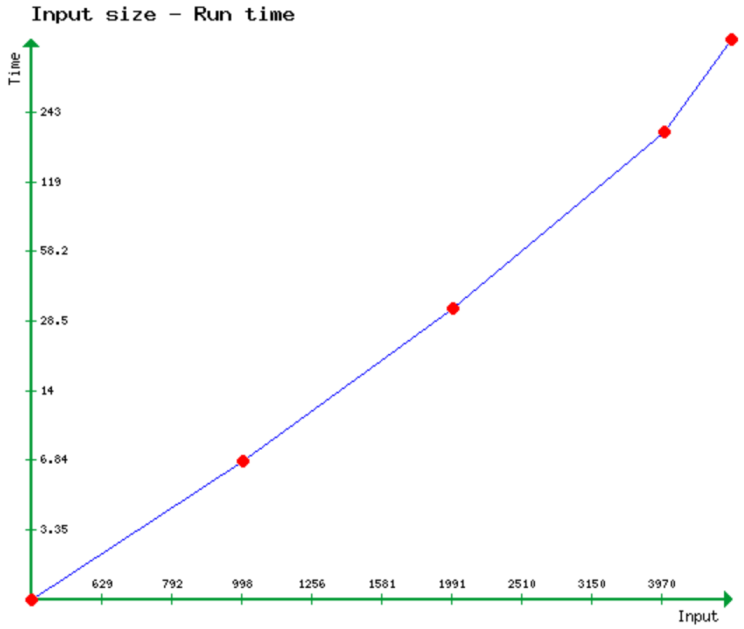
The run time I am measuring is the running time of whole program. I used a Macbook Pro running in local.

It runs 1.6s when length = 500, 6.74s when length = 1000, 32.78s when length = 2000, 197.33s when length = 4000, and 516.77s when length = 5000

Graph:



Log-Log graph:




Slope on log-log graph is 2, which agree the analysis above, because the input grows in x^2

## *4.Interpretation and discussion*

The graph match the expectation on part 2, it's O(x^2) because when running my program, I input two same length string, so m = n O(mn) = O(m^2). Additionally, it makes sense because by the table on edit-distance problem, our program actually fill out a table of m * n, and each iteration running in a constant time, O(mn) make sense.