# Algorithms Assignment – Implementation

Most of us use text editors and whenever we make a spelling error ,it immediately shows us that the typed word contains an error. How does this happen. In the world of computer science, string algorithms that involves the difference between 2 strings become very important. In this, we will be trying to implement an important algorithm in this category called Wagner-Fischer algorithm that gives the Levenshtein distance between two strings. Levenshtein distance between two strings of characters is the minimum number of character operations – either insertion, deletion or substitution required to convert the first string into the second or the second to first. So this also becomes our problem statement

**PROBLEM STATEMENT** : Given two strings of characters as input, find the minimum number of character operations – either insertion,deletion or substitution required to convert one string to the other.

**SAMPLE OUTPUT** :
Given the two strings as KITTEN and SITTING.
The minimum number of character operations required to convert one to the other is 3 as:
1) KITTEN – SITTEN ( substitute K by S)
2) SITTEN – SITTIN ( substitute E by I)
3) SITTIN – SITTING ( insert G at the end of the string)

**ALGORITHM** :

The folowing is an efficient dynamic programming solution to the above stated problem. This returns the number of operations needed directly in O(MN) time where M is the length of the first string and N is the length of the second string.

```
   Int findSeq(char s[0..m-1] , char t[0..n-1],int m,int m)
 {
   //declare a 2d matrix d[m+1][n+1]
```

```
//for all i,j ,d[i,j] will hold the levenshtein distance
// between the first i characters of string S and the first
// j characters of string T

   let d be a 2-d array of int with dimensions [0..m, 0..n]

  for i in [0..m]
     d[i, 0] ← i // the distance of any first string to an
empty second string
                 // (transforming the string of the first
i characters of s into
                 // the empty string requires i deletions)
   for j in [0..n]
     d[0, j] ← j // the distance of any second string to
an empty first string

   for j in [1..n]
     for i in [1..m]
       if s[i] = t[j] then
         d[i, j] ← d[i-1, j-1]        // no operation
required
       else
         d[i, j] ← minimum of
                    (
                      d[i-1, j] + 1,  // a deletion
                      d[i, j-1] + 1,  // an insertion
                      d[i-1, j-1] + 1 // a substitution
                    )

   return d[m,n]

}
```

**APPLICATIONS OF THE ALGORITHM :**

1) **Spell Checking Systems :** In programs that always run in the background,say text editors for example, for each word you type it compares this string with the most relatable word in its in built dictionary to find the number of character edits required. If this number happens to be 0,then the word has correct spelling.

2) **File checking systems :** This uses the algorithm on a large scale. Assuming two files contains some sequences of characters, maybe code or simply a text document, then using this algorithm for the different sequences of characters, we can find by how much one file differs to the other. This is very useful when comparing two code files to find the difference. This is exactly what the **'diff'** command in UNIX does. It takes 2 files as parameter and produces the lines where a difference in the two files occur.

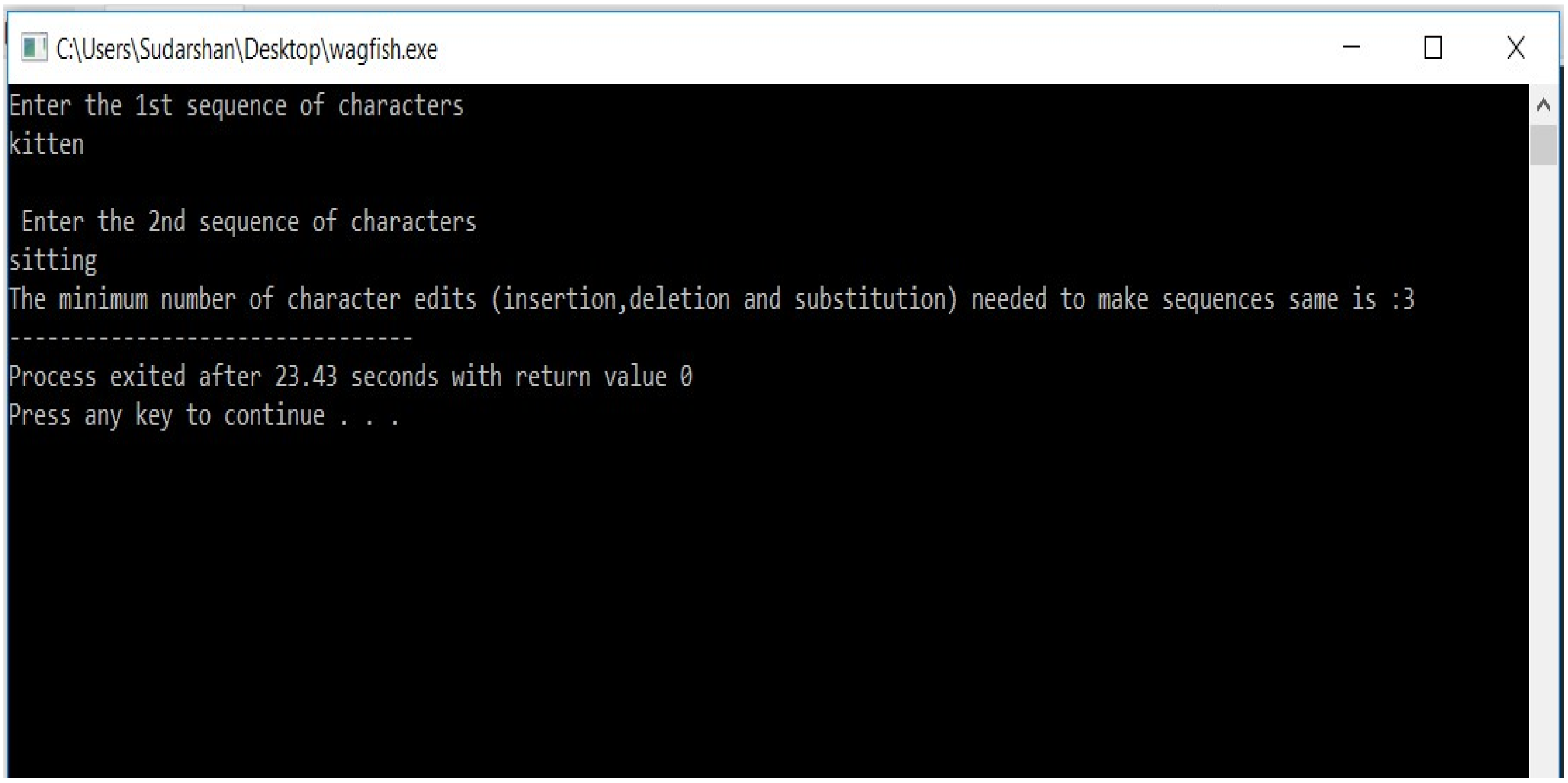
## IMPLEMENTATION CODE IN C++

```
#include<bits/stdc++.h>
using namespace std;
int minimum(long long int a ,long long int b,long long int
c)
{
    if(a<b && a<c)
    return a;
    else if(b<c)
    return b;
    else
    return c;
}
int findSeq(char s1[], char s2[],int m, int n)
{
    //matrix d[m+1,n+1] is declared.
    // d[i,j] holds the number of single character edits
needed to make 2 sequences equal between the first i
characters of s1
```

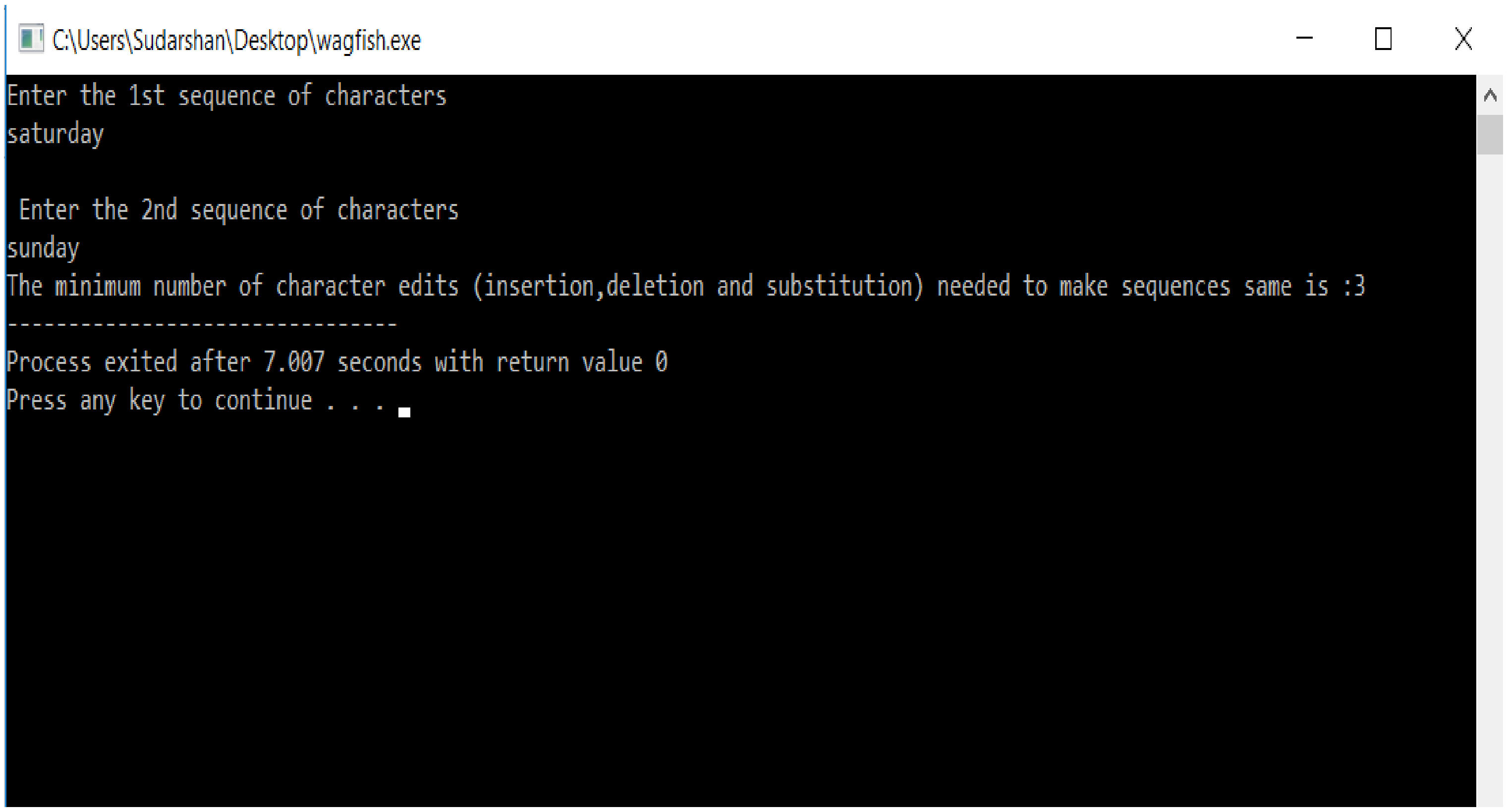
```
    // and the first j characters of s2
  long long int d[m+1][n+1];
    int i,j;
    for(i=0;i<=m;i++)
    {
        d[i][0] = i;
    }
    // transforming one string of i charcters to a string
 //with 0 characters requires i deletions.
    for(j=0;j<=n;j++)
    {
        d[0][j] = j;
    }

    for(j=0;j<=n;j++)
    {
        for(i=0;i<=m;i++)
        {
            if(s1[i-1] == s2[j-1])
            d[i][j] = d[i-1][j-1]; // both sequences are
same, no operation required.
            else
            d[i][j] = minimum(d[i-1][j]+1,  d[i][j-1]+1,
d[i-1][j-1]+1 );
        }
    }
    return d[m][n];
}
```

```
int main()
{
    char s1[1000],s2[1000];
    cout<<"Enter the 1st sequence of characters\n";
    cin>>s1;
    cout<<"\n Enter the 2nd sequence of characters\n";
    cin>>s2;
    int m = strlen(s1);
    int n = strlen(s2);
    cout<<"The minimum number of character edits
(insertion,deletion and substitution) needed to make
sequences same is :"<<findSeq(s1,s2,m,n);
    return 0;
}
```

**OUTPUTS OF DIFFERENT TEST CASES:**

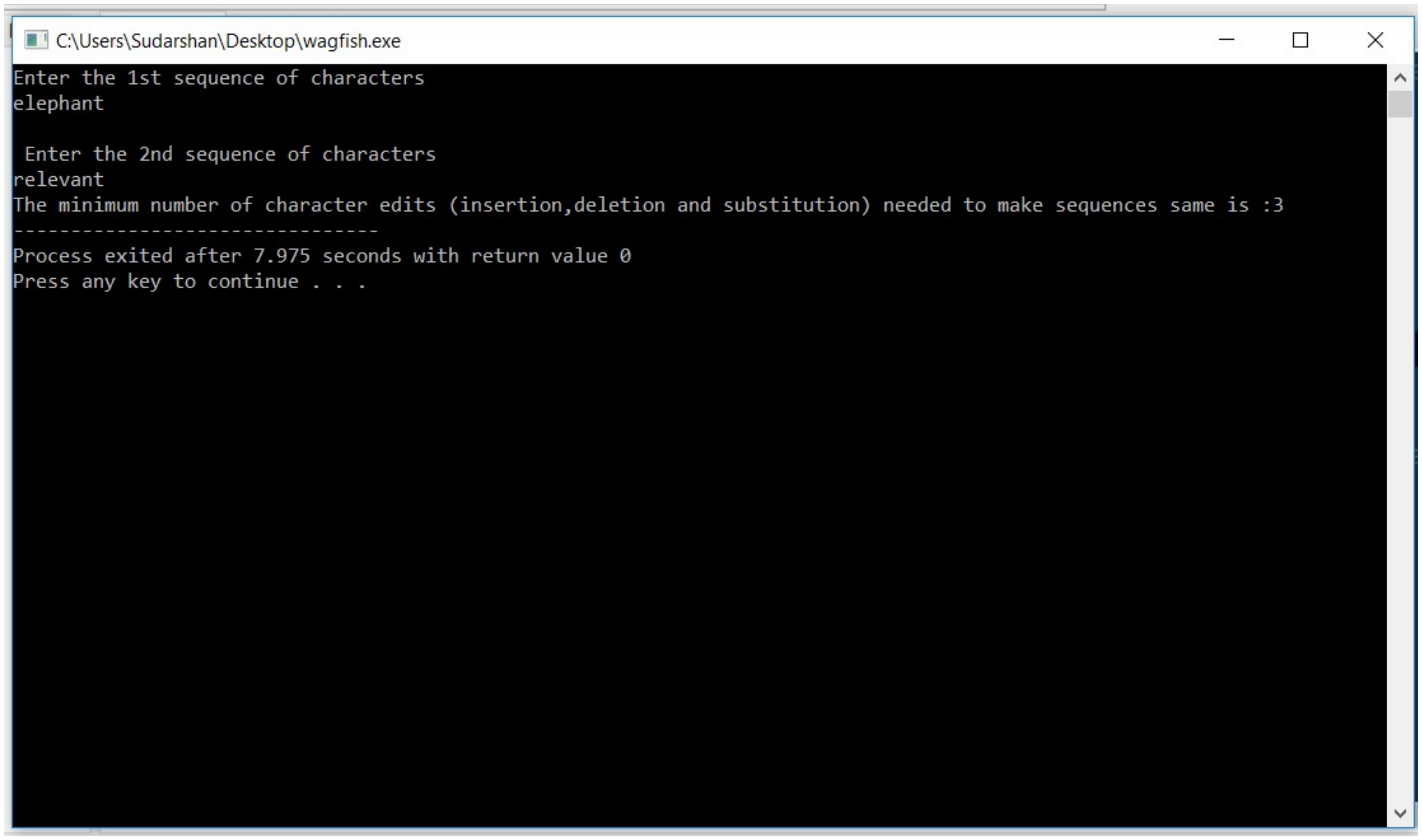


```
Enter the 1st sequence of characters
kitten

 Enter the 2nd sequence of characters
sitting
The minimum number of character edits (insertion,deletion and substitution) needed to make sequences same is :3
--------------------------------
Process exited after 23.43 seconds with return value 0
Press any key to continue . . .
```



C:\Users\Sudarshan\Desktop\wagfish.exe

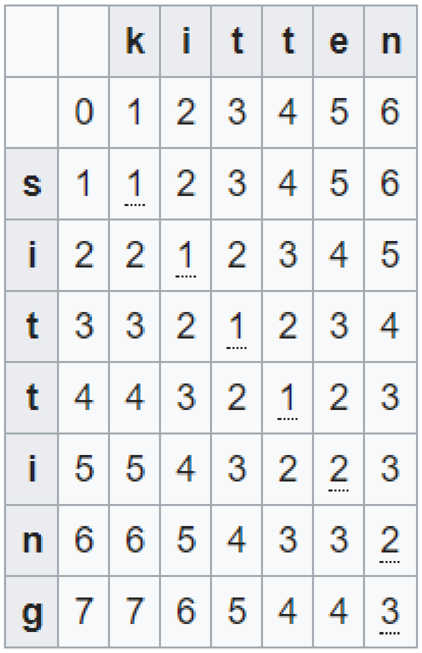
```
Enter the 1st sequence of characters
saturday

 Enter the 2nd sequence of characters
sunday
The minimum number of character edits (insertion,deletion and substitution) needed to make sequences same is :3
--------------------------------
Process exited after 7.007 seconds with return value 0
Press any key to continue . . .
```



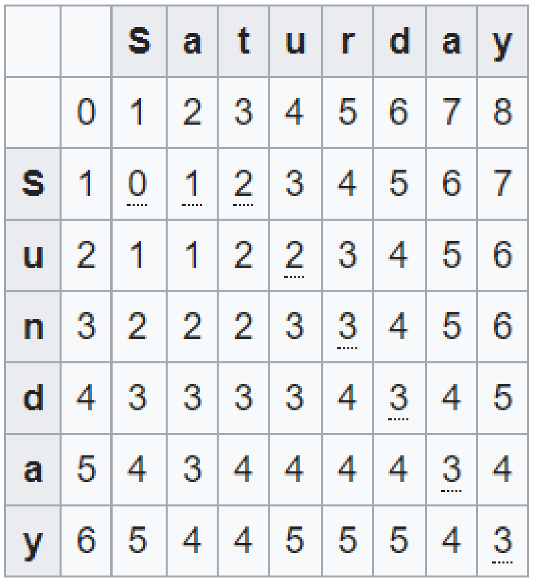
C:\Users\Sudarshan\Desktop\wagfish.exe

```
Enter the 1st sequence of characters
elephant

 Enter the 2nd sequence of characters
relevant
The minimum number of character edits (insertion,deletion and substitution) needed to make sequences same is :3
--------------------------------
Process exited after 7.975 seconds with return value 0
Press any key to continue . . .
```

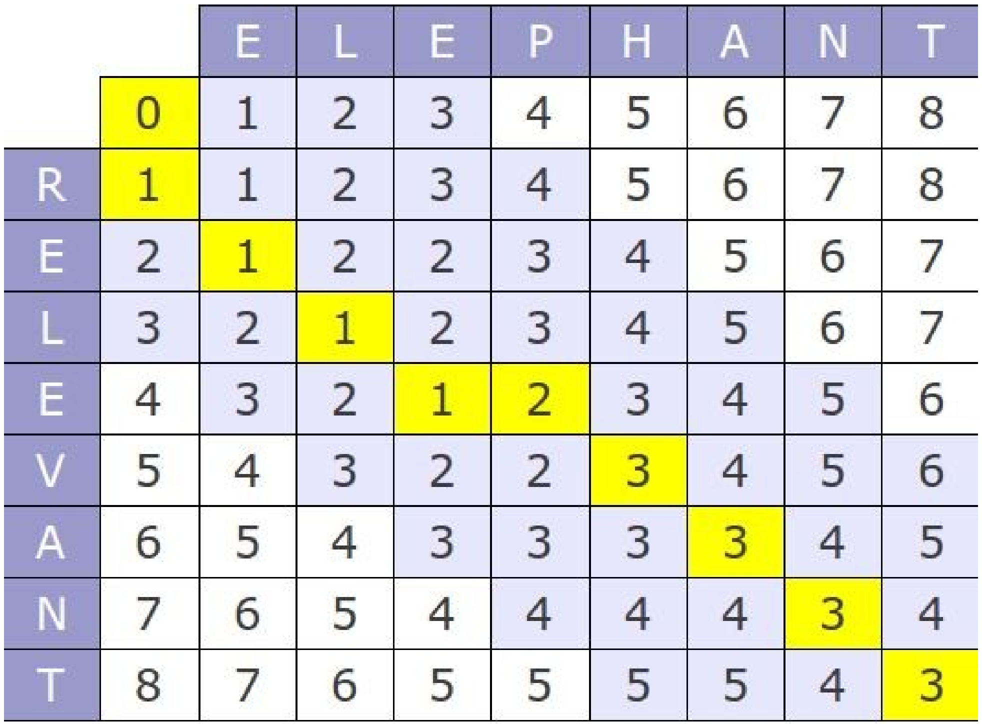
The Matrix Allocations For each Of the 3 different test cases :



| | | k | i | t | t | e | n |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| s | 1 | 1 | 2 | 3 | 4 | 5 | 6 |
| i | 2 | 2 | 1 | 2 | 3 | 4 | 5 |
| t | 3 | 3 | 2 | 1 | 2 | 3 | 4 |
| t | 4 | 4 | 3 | 2 | 1 | 2 | 3 |
| i | 5 | 5 | 4 | 3 | 2 | 2 | 3 |
| n | 6 | 6 | 5 | 4 | 3 | 3 | 2 |
| g | 7 | 7 | 6 | 5 | 4 | 4 | 3 |



| | | S | a | t | u | r | d | a | y |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| S | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| u | 2 | 1 | 1 | 2 | 2 | 3 | 4 | 5 | 6 |
| n | 3 | 2 | 2 | 2 | 3 | 3 | 4 | 5 | 6 |
| d | 4 | 3 | 3 | 3 | 3 | 4 | 3 | 4 | 5 |
| a | 5 | 4 | 3 | 4 | 4 | 4 | 4 | 3 | 4 |
| y | 6 | 5 | 4 | 4 | 5 | 5 | 5 | 4 | 3 |



| | | E | L | E | P | H | A | N | T |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| R | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| E | 2 | 1 | 2 | 2 | 3 | 4 | 5 | 6 | 7 |
| L | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| E | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 |
| V | 5 | 4 | 3 | 2 | 2 | 3 | 4 | 5 | 6 |
| A | 6 | 5 | 4 | 3 | 3 | 3 | 3 | 4 | 5 |
| N | 7 | 6 | 5 | 4 | 4 | 4 | 4 | 3 | 4 |
| T | 8 | 7 | 6 | 5 | 5 | 5 | 5 | 4 | 3 |

**DONE BY,**

**SUDARSHAN P, CSE A, 106116095**