**Algorithmic Description:**

**Problem:**

We are given two strings, called the source and the target, which both have the same number of characters n. The goal is to transform the target string into the source string using the smallest number of allowed operations. There are two types of allowed operations, a flip and a substitution.

* A substitution replaces a character in a certain position by another character
* A flip reverses a contiguous substring of the string.

**Algorithm:**

Word processors (MS Word) and Search Engines provide a spelling correction feature. If one types a misspelled word x, then word processor or search engine suggests a correction y.

It is based on the dynamic programming approach to compute the edit distance between two strings.

The edit distance d(source, target) of two strings of text is defined to be the minimum possible cost of a sequence of “transformation operations”.

The cost of a sequence of transformation operations is the sum of the costs of the individual operations on the sequence. The goal of the edit-distance problem is to find a sequence of transformation operations of minimum cost that transforms.

This algorithm is implemented in C#.net using visual studio.net. The complete project file is provided with this file. The Solution file can be opened in Visual Studio.net and can be run from it. One can also run it by directly executing the .exe file in the debug folder.

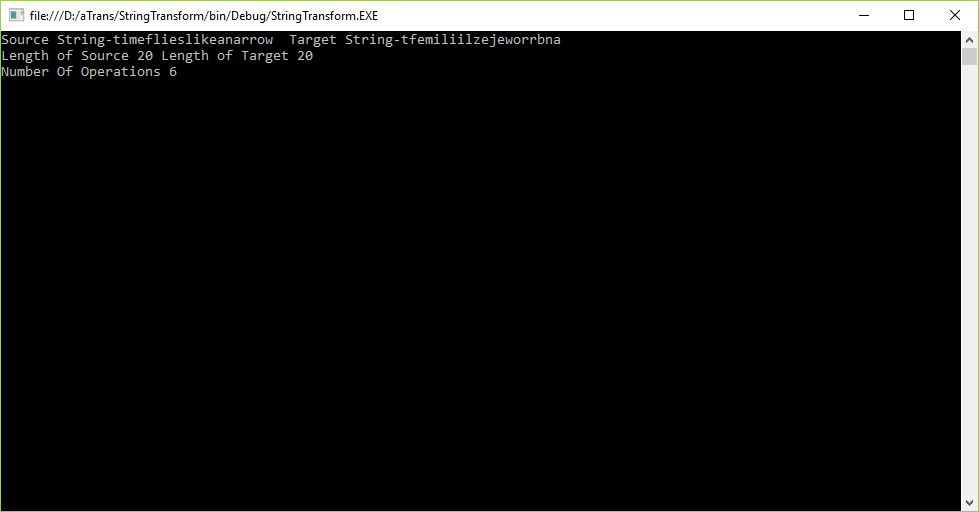
The file “strdata.txt” contains the source and the target string as follows:

1: timeflieslikeanarrow

2: tfemiliilzejeworrbna

One can change this string in the source file for testing for other String

The output:



The program:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace StringTransform

{

class Program

{

static string src = "", dest = "";//String objects to hold source and destination Strings

//A Flip() function reverses a contiguous substring of the string.

public static string Flip(string s,int start,int length)

{

char[] charArray = s.ToCharArray();

Array.Reverse(charArray,start,length);

return new string(charArray);

}

public static string Substitute(string s,int index, char ch)

{

char[] charArray = s.ToCharArray();

charArray[index] = ch;

return new string(charArray);

}

public static int getNoOfOperations()

{

int count = 0;

int m, n,i,j;

m = src.Length-1;

n = dest.Length-1;

i = m;

j = n;

while (true)

{

if (dest[j] == src[i])

{

i = i - 1;

j = j - 1;

}

else

{

if (src.Contains(dest[j]))

{

//flip will occur

int pos = dest.LastIndexOf(src[i]);

dest = Flip(dest, pos, (j - pos + 1));

}

else

{

//substitution will occur

dest = Substitute(dest, j, src[i]);

// j = j - 1;

count = count + 1;

}

if (src.Equals(dest))

break;

}

return count;

} static void Main(string[] args)

{

//Check whether File exits or not

if (!File.Exists("strdata.txt"))

{

Console.WriteLine("Cannot continue!!! File containing Strings Does Not exist");

return;

}

//To read strings from the file strdata.txt

StreamReader sr = new StreamReader("strdata.txt");

string line = sr.ReadLine();

string []s = line.Split();

src=s[1].Trim();

line = sr.ReadLine();

s = line.Split();

dest = s[1].Trim();

Console.WriteLine("Source String-" + src + " Target String-" + dest);

sr.Close();

int length1 = 0, length2 = 0;

length1 = src.Length;

length2 = dest.Length;

Console.WriteLine("Length of Source " + length1 + " Length of Target " + length2);

if (length1!=length2)

{

Console.WriteLine("Cannot continue !!! String are of Different Lengths");

return;

}

int edist = getNoOfOperations();

Console.WriteLine("Number Of Operations " + edist);

Console.ReadLine();

}

}

}

**Time efficiency**: It is a measure of amount of time for an algorithm to execute. The time is measured mostly for the comparisons, because the computer takes maximum time for comparison. The efficiency of an Algorithm depends on the comparison as compared to other operations. This Algorithm is of the order O(n) roughly some multiple of n.

**Algorithm Description:**

The algorithm used to determine the number of operations required is described below:

1. Repeat following steps until the goal to transform the target string into the source string is achieved.
2. Check last character of target string i.e. match it with char of source
3. If Match then increment pointers to next char
4. Else (No Match) Check character of target in source string
5. If Found then Flip
6. Else substitute character of source of that position in target string at that position