A New DNA Sequencing Alignment Using Longest Common Subsequence Method

Mr. Hirak Mondal Mr. Debkumar Chowdhury

University of Engineering & Management, Kolkata- 700156, India

Abstract--- The goal of this project is to perform DNA sequence alignment using the Longest Common Subsequence Algorithm to obtain the longest common subsequence from the two strings ‘X’ and ‘Y’ for matching DNA in molecular biology etc. This research would seek to establish the differences between the 2 species DNA and thereby conclude on their genetic similarity. The project will be implemented in terms of space and time using the optimized longest common subsequence algorithm. This project acts as a bridge between computer science and genomics, exploring the interdisciplinary area of computational biology and helping us to consider the very fundamental concepts of life and our relationship with other species

Keywords--- *Computational Biology, Genetic Engineering, Medical Diagnosis, Bioinformatics*

I. INTRODUCTION

DNA, or deoxyribonucleic acid, is the human genetic stuff, and virtually all other species. In a person’s body, virtually every cell has the same DNA. Much DNA is in the nucleus of the cells (where it is called nuclear DNA), but in the mitochondria, there is also a significant amount of DNA (where it is called mitochondrial DNA or mtDNA). In cells, mitochondria are structures that transform food energy into a form that cells can utilize.

DNA data is stored as a file consisting of four chemical bases: adenine (A), guanine (G), cytosine (C) and thymine (T). DNA of a person is composed of approximately 3 billion bases, and more than 99% of those bases are the same in all humans. The order of these bases, or sequence, determines what information for construction is available

One essential role in molecular biology is to compare two sequences, known as sequence comparison, either from the same organism or from a separate organism. It helps to provide solutions to many biological problems, such as:

- predicting the structure and function of proteins
- inferring evolutionary history and species relatedness
- identifying specific subsequences in genes/proteins to classify specific motifs,
- as a sub-problem in DNA sequencing genome assembly
- mutating the genomes to reproduce a whole species

What is a Sequence Alignment?

- Sequence alignment is a way of putting the correspondence between related characters or substrings one series over series. This may occur in deoxyribonucleic acid (DNA), ribonucleic acid (RNA), or protein sequences
.
- Sequences of various species can have varying sizes. Alignment involves the addition of spaces in the series in random places so that both are of the same size. 'Spaces' or 'gaps' are either added at the start or the end of the series.

Let's take an example here.

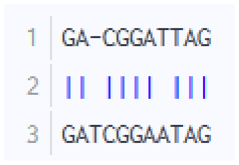


Fig 1: Base Comparison

II. LITERATURE SURVEY

SL No.	Year Propos e	Name	Implementation
1	1955	H.Rick	Rick in 1955 introduced an algorithm based on advancing from contour to contour which was compared with other existing algorithms and proved to be better among them.
2	1974	Wagner & Fischer	Wagner and Fischer in the year 1974 introduced an algorithm using the concept of a matrix to get the solution for this problem with dynamic programming. This algorithm just gives the LCS length but not the LCS
3	1975	Hirschberg	Fully use the divide and conquer technique and complex method, In 1975 Hirschberg developed a method for finding LCS.

SL No.	Year Propose	Name	Implementation
4	1977	Hirschberg	Dominant matches was another approach given by Hirschberg in 1977.
5	1977	J.W. Hunt & T.G.Szymanski	J.W. Hunt, T.G. In 1977, Szymanski's claimed that computing LCS from two strings is tantamount to finding the longest monotonically increasing path in the graph, where $x_i = y_j$.
6	1987	Apostolico, A. & Guerrain	Another algorithm was published by Apostolico, A. & Guerrain 1987, an alternative to support the framing of the LCS.
7	1990	Wu ET	In the year 1990 Wu ET made an effort to decrease the issue of editing distance to reduce edit distance and to apply it to finding LCS. The time complexity of this algorithm is O(n (m-r)).

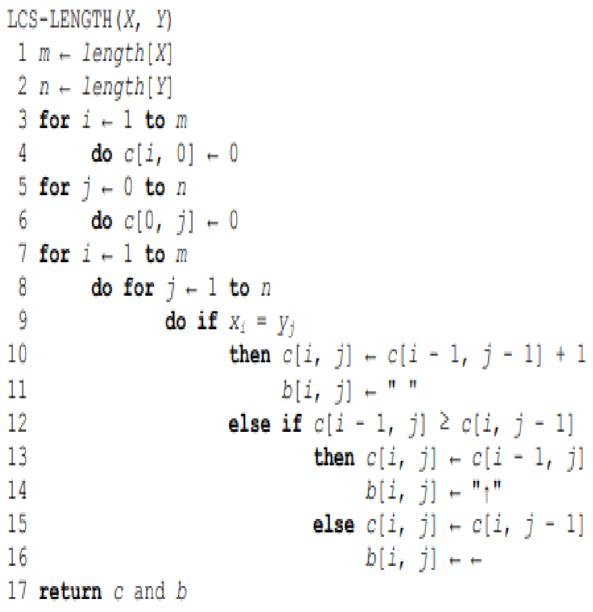
III. PROPOSED METHODOLOGY

Longest Common Subsequence (Naïve Approach)

Find the length of the longest subsequence present in both sequences, given two parts. A subsequence is a string that exists but not necessarily contiguous in the same relative order. "abc," "bdf," "aeg," "acefg," "bdf," "bdf," "bdf." .. Subsequences of "abcdefg" are, etc.

For determining the difficulty of the brute force method, the number of differing Length subsequences of a string n must first be identified. Remember from permutation theory and the combination that numerous combinations with 1 variable are nC1. The combination number of 2 elements is nC2 and so on. ${}^nC_0$ + ${}^nC_1$ + ${}^nC_2$ + … ${}^nC_n$ = 2^n.. A string of length n, therefore, has 2^n-1 separate potential subsequences, since we do not find the subsequence of length 0. This means that this approach's time complexity would be **O(n * 2^n)**. Note it has O(n) time to search for both strings to have a subsequence common. With DP this runtime complexity can be decreased.

ALGORITHM



```
LCS-LENGTH(X, Y)
 m ← length[X]
 n ← length[Y]
 for i ← 1 to m
     do c[i, 0] ← 0
 for j ← 0 to n
     do c[0, j] ← 0
 for i ← 1 to m
     do for j ← 1 to n
            do if $x_i = y_j$
                  then c[i, j] ← c[i - 1, j - 1] + 1
                       b[i, j] ← " "
                  else if c[i - 1, j] ≥ c[i, j - 1]
                          then c[i, j] ← c[i - 1, j]
                               b[i, j] ← "↑"
                          else c[i, j] ← c[i, j - 1]
                               b[i, j] ← ←
 return c and b
```

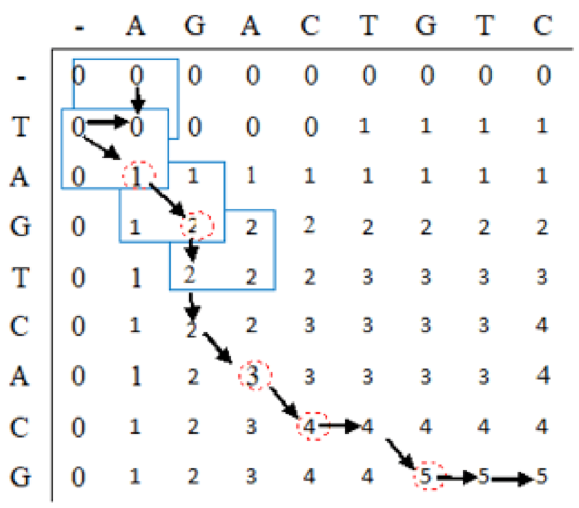
```
PRINT-LCS(b, X, i, j)
if i = 0 or j = 0
    then return
if b[i, j] = " "
    then PRINT-LCS(b, X, i - 1, j - 1)
         print $x_i$
elseif b[i, j] = "↑"
   then PRINT-LCS(b, X, i - 1, j)
else PRINT-LCS(b, X, i, j - 1)
```

,

The simplistic approach to this problem would be to generate all the subsequences of the two components and also to seek the resulting longest subsequence. This solution is exponential in terms of time complexity. Let us see how this problem possesses both important properties of a Dynamic Programming (DP) Problem

Optimal Substructure (DP SOLUTION)

Suppose we have two Sl and S2 sequences of lengths m and n respectively, where SI = al a2 an, and S2 = bl b2 bn. We'll create a matrix A where Aij denotes the length of al a2 ai and bl b2 bj, the longest common subsequence.



	-	A	G	A	C	T	G	T	C
-	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	1	1	1	1
A	0	1	1	1	1	1	1	1	1
G	0	1	2	2	2	2	2	2	2
T	0	1	2	2	2	3	3	3	3
C	0	1	2	2	3	3	3	3	4
A	0	1	2	3	3	3	3	3	4
C	0	1	2	3	4	4	4	4	4
G	0	1	2	3	4	4	5	5	5

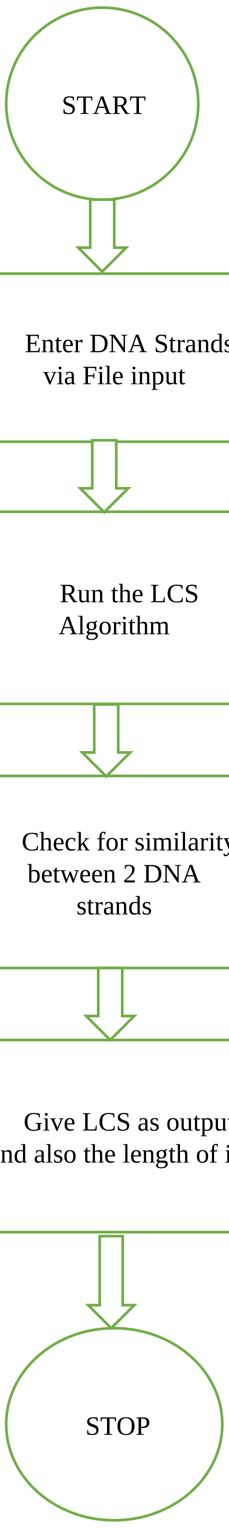
Fig 2: Tracing path in LCS

Sequence SI is written vertically, and S2 is written horizontally. Compare every symbol expressing the rows to the column expressing every letter. We must continue Row by Row, Column by Column. 1. If ai = bj, we did consider a match. For the current match, we get a score of 1 and from the rest of the LCS, we've already received substrings al all and bl b(j-l).

$$A_{i,j} = \begin{cases} A_{i-1,j-1} + 1 & \text{if } a_i = b_j \\ max(A_{i-1,j}, A_{i,j-1}) & \text{if } a_i \neq b_j \end{cases}$$

It takes **O(nm)** time to fill in the m by n matrix A. This approach is called dynamic programming.

FLOWCHART


START
Enter DNA Strands via File input
Run the LCS Algorithm
Check for similarity between 2 DNA strands
Give LCS as output and also the length of it
STOP

IV. EXPERIMENTAL SETUP AND RESULT ANALYSIS

Till this date of project progress report submission, all the works have been carried out in C++ language using TDM-GCC 4.9.2 64-bit profiling compiler. The IDE used is Dev C++. The computation machine used is HP Pavilion AU 003tx.

Our Approach vs Brute Force Approach

To evaluate the complexity of the brute force solution, we must first know the number of possible different Remember from permutation theory and the combination that numerous combinations with 1 variable are nC1. The combination number of 2 elements is nC2 and so on. ${}^nC_0 + {}^nC_1 + {}^nC_2 + \ldots {}^nC_n = 2^n$..

A string of length n, therefore, has 2^n-1 separate potential subsequences, since we do not find the subsequence of length 0. This means that the brute

force approach's time complexity would be **O(n * 2^n)**. Note it takes O(n) time to search for both strings to have a subsequence common. With DP this runtime complexity can be decreased.

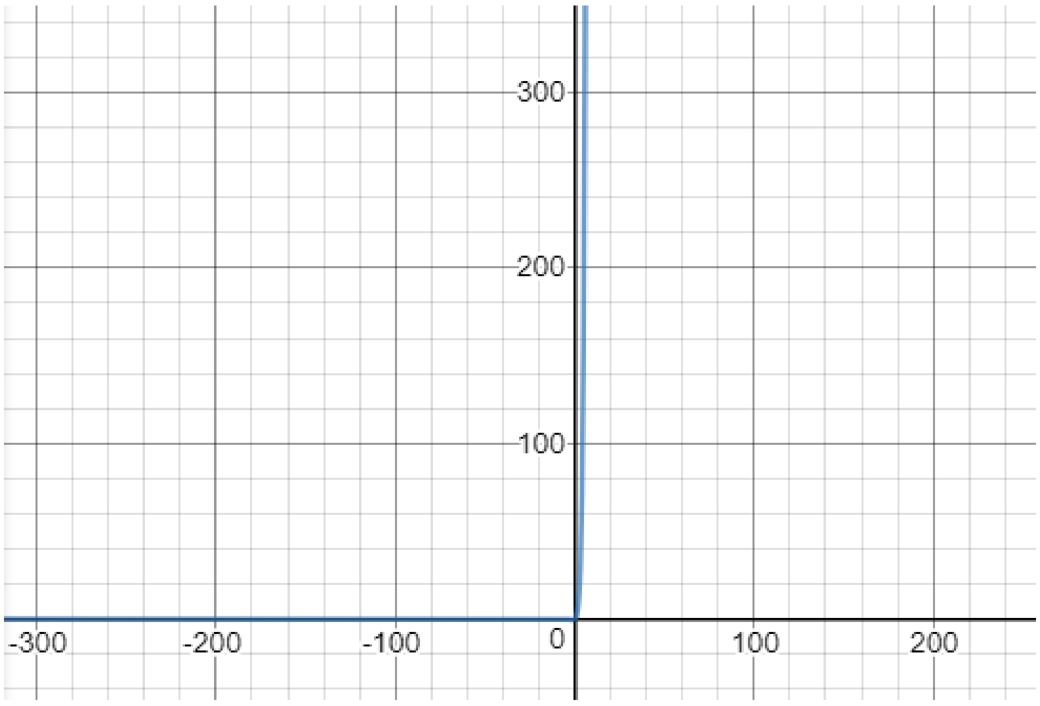


Fig 4: Graph showing LCS using Brute Force

With the help of Dynamic Programming, the time complexity is **O(m.n)**

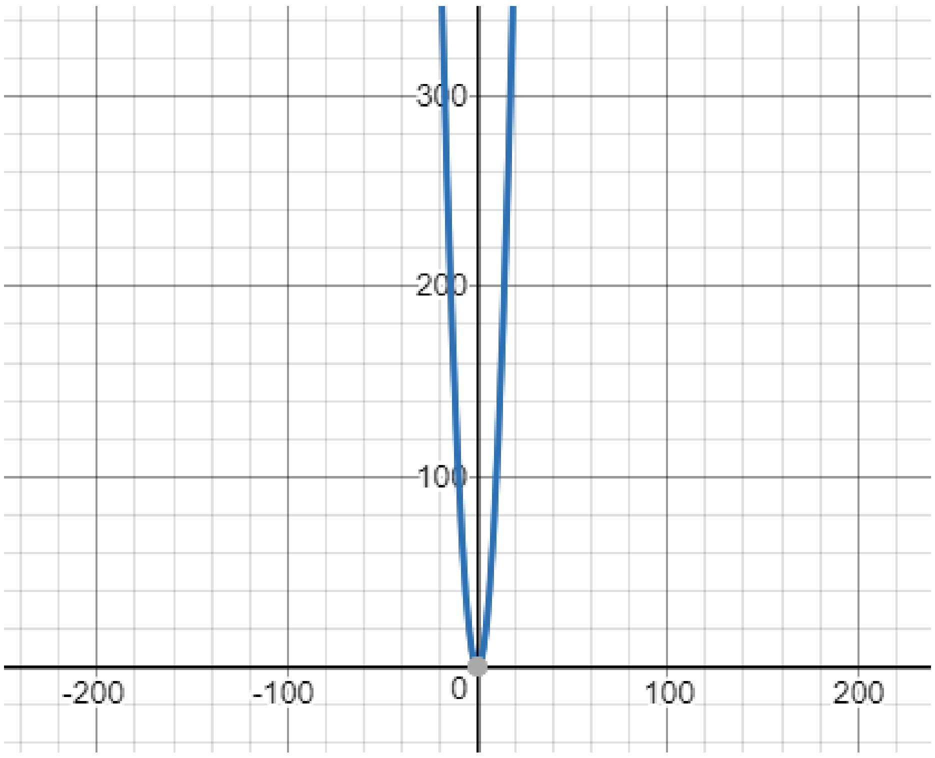


Fig 5: LCS using DP solution

We can see how this method has helped DNA sequencing instead of just normal brute forcing method and thereby decrease the overall time complexity

V. CONCLUSION

In this scheduled outline we have shown how the Longest Common Subsequence algorithm can be used to compare 2 DNA strands and thereby showing its importance In interdisciplinary fields too. We have used the DP solution for our work which reduces the complexity to a great extent and thereby is more efficient than other methods based on recursive or brute force approach. With the help of experimental results and analysis, we have demonstrated its performance as compared to other naïve approaches. With COVID-19 here in the news, this algorithmic approach can also be used to detect the protein sequence of viruses and compare its similarity with other viruses and thereby increasing the chances to find its similarity with other known viruses for which we already have a vaccine.

VI. REFERENCES

1. S.L. Sheetlin, L. Mariño-Ramírez, and J.L. Spouge
   "Searching for Repeats, as an Example of Using the Generalized Ruzzo-Tompa Algorithm to Find Optimal Subsequences with Gaps"
   (2014)

2. S.L. Sheetlin, Y. Park, and J.L. Spouge
   "An objective method for estimating asymptotic parameters, with application to biological sequence alignment"
   (2011)

3. Commonlounge.com blogs on sequence alignment

4. *Introduction to Algorithms*, Second Edition. Thomas H. *Cormen*. Charles E. Leiserson. Ronald L. Rivest. Clifford Stein. The MIT Press.