I.J. Information Technology and Computer Science, **2016, 5, 81-87**
Published Online May 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2016.05.09

Locating All Common Subsequences in Two DNA Sequences

M. I. Khalil
Reactor Physics Dept., Nuclear Research Center, Atomic Energy Authority, Cairo, Egypt
Currently in a sabbatical leave as an Associate Prof. at Princess Nora Bent Abdurrahman University, Faculty of Computer and Information Sciences, Networking and Communication Dept., Riyadh, Kingdom of Saudi Arabia
E-mail: magdi_nrc@hotmail.com, mikhalil@pnu.edu.sa

***Abstract*—**Biological sequence comparison is one of the most important and basic problems in computational biology. Due to its high demands for computational power and memory, it is a very challenging task. The well-known algorithm proposed by Smith-Waterman obtains the best local alignments at the expense of very high computing power and huge memory requirements. This paper introduces a new efficient algorithm to locate the longest common subsequences (LCS) in two different DNA sequences. It is based on the convolution between the two DNA sequences: The major sequence is represented in the linked-list X while the minor one is represented in circular linked-list Y. An array of linked lists is established where each linked list is corresponding to an element of the linked-list X and a new node is added to it for each match between the two sequences. If two or more matches in different locations in string Y share the same location in string X, the corresponding nodes will construct a unique linked-list. Accordingly, by the end of processing, we obtain a group of linked-lists containing nodes that reflect all possible matches between the two sequences X and Y. The proposed algorithm has been implemented and tested using C# language. The benchmark test shows very good speedups and indicated that impressive improvements has been achieved.

***Index Terms*—**DNA similarity algorithms, DNA sequence comparison, DNA analysis, pattern recognition, Longest Common Sequence, Longest Common Subsequence.

I. Introduction

In a DNA sequence, or a molecule of DNA, there are four nucleotide bases: Adenine, Guanine, Cytosine, and Thymine (Fig.1). The knowledge of a DNA sequence and gene analysis can be used in several biological, medicine and agriculture research fields such as: possible disease or abnormality diagnoses, forensics, pattern matching, biotechnology, etc [1-5,7,8,22]. It can be also used to predict the function of a particular gene and compare it with other "similar" genes from same or different organisms. The analysis and comparison studies for DNA sequences connected information technology tools and methods to accelerate findings and knowledge in biological related sciences.

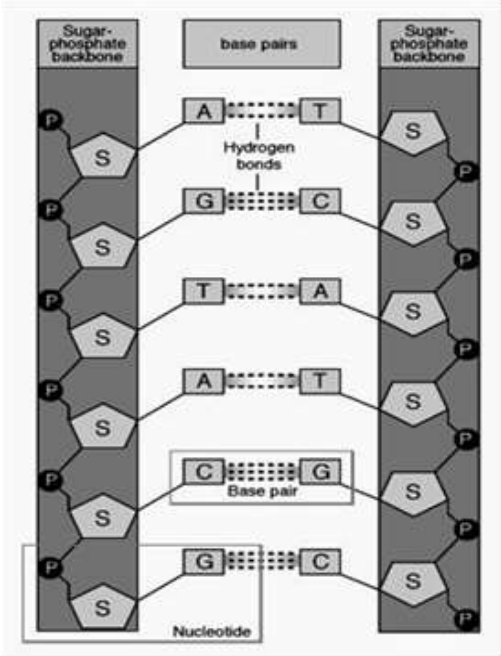

Fig.1. DNA structure

In comparative genomics, comparing genome sequences is one of the main tasks because sequence similarities strongly reflect the evolutionary relationships between the corresponding species. In addition, with the introduction of next-generation sequencing technologies, the demand for rapid comparisons of massive amounts of long sequences has increased in recent years. For years, DNA comparison has been used in biology and forensics to discriminate and compares genes or genomes. Those tools vary in size, complexity and functionality based on several factors. Some small tools or websites are developed as free or open source for research or experimental purposes. Examples of such small size limited purpose tools or applications are: Double Act (http://www.hpa-bioinfotools.org.uk/pise/double_act.html), Genomatix (http://www.genomatix.de), Mobyle (http://mobyle.pasteur.fr), ALIGN, FASTA, etc. BLAST: (Basic Local Alignment Search Tool)[4] is an example of a larger scale. Most of these algorithms uses Smith–Waterman algorithm for performing sequence alignment [6-10,21-25]. This algorithm which is also used in crimes' forensic investigation does not use full DNA to DNA

sequence comparison. It rather selects several segments (e.g. eight segments) selected from the different locations of the DNA. BLAST uses also dynamic programming and "seeding" to find starts of possible matches[11-15]. The goal is to accelerate the process of finding matches between DNA sequences as this can take a significant amount of time and resources.

Another process that can be different from one tool to another is the ranking of the different matches. This can particularly occur when more than a match is in the same size.

This paper addresses the problem of finding the best match between two given DNA sequences or protein strings. The algorithm suggested in this paper aims to minimize both the time of processing and the size of allocated memory. The algorithm detects not only the longest common subsequence but finds all possible common subsequences. The major DNA sequence is represented in the linked-list X while the minor one is represented in circular linked-list Y. An array of linked lists Z is established where each linked list is corresponding to an element of the linked-list X and a new node is added to it for each match between the two DNA sequences. If two or more matches (with same or different length) at different locations in sequence Y share the same location in sequence X, then the corresponding nodes will construct a unique linked-list in Z. Accordingly, by the end of processing, we obtain multiple linked-lists containing nodes that reflect all possible matches between the two sequences X and Y. The array Z of linked lists is then traversed horizontally and vertically retrieving all matches between sequences X and Y.

The rest of the paper is organized as follows: Section 2 demonstrate some of interesting facts about DNA. Section 3 illustrates the suggested algorithm, and the implementation and experimental results are discussed in Section 4. I conclude the paper in Section 5.

II. Interesting DNA Facts

DNA or deoxyribonucleic acid codes for your genetic make-up. There are lots of facts about DNA, but here are some that are particularly interesting, important, or fun [16-20].

- Even though it codes for all the information that makes up an organism, DNA is built using only four building blocks, the nucleotides adenine, guanine, thymine, and cytosine.
- Every human being shares 99% of their DNA with every other human.
- If you put all the DNA molecules in your body end to end, the DNA would reach from the Earth to the Sun and back over 600 times (100 trillion times six feet divided by 92 million miles).
- A parent and child share 99.5% of the same DNA.
- You have 98% of your DNA in common with a chimpanzee.
- Humans share 50% of their DNA with bananas.
- Cells can contain 6-9 feet of DNA. If all the DNA in your body was put end to end, it would reach to the sun and back over 600 times.
- DNA in all humans is 99.9 percent identical. It is about one tenth of one percent that makes us all unique, or about 3 million nucleotides difference.
- DNA can store 25 gigabytes of information per inch and is the most efficient storage system known to human. So, humans are better than computers!!
- In an average meal, you eat approximately 55,000,000 cells or between 63,000 to 93,000 miles of DNA.
- It would take a person typing 60 words per minute, eight hours a day, around 50 years to type the human genome.

DNA sequence analysis can be used to identify possible errors or abnormality in a DNA sequence (e.g. in comparison with a normal one). It can be also used to expect the function of a particular gene and compare it with other "similar" genes from same or different organisms.

If a new DNA sequence is discovered its functionality is specified depending on its similarity with other known DNA sequences. Such technique is used in several medical applications and research studies.

DNA is composed of units called NUCLEOTIDES, which are composed of three sub-molecules:

1. Pentose Sugar (deoxyribose)
2. Phosphate
3. Nitrogen Base (purine or pyrimidine)

DNA is composed of two complimentary strands of nucleotides joined by hydrogen bonds:

Adenine with Thymine (A-T or T-A) They join with 2 hydrogen bonds Cytosine with Guanine (C-G or G-C) They join with 3 hydrogen bonds. DNA twists into a double helix, and conformity of style throughout a conference proceedings. Margins, column widths, line spacing, and type styles are built-in; examples of the type styles are provided throughout this document and are identified in italic type, within parentheses, following the example. Some components, such as multi-leveled equations, graphics, and tables are not prescribed, although the various table text styles are provided. The formatter will need to create these components, incorporating the applicable criteria that follow.

III. The Suggested Algorithm

Given two DNA sequences X and Y of length n and m respectively:

$$X= x\_1 \ x\_2 \ x\_3 \ \ldots\ldots \ x\_n \quad (1)$$

$$Y= y\_1 \ y\_2 \ x\_3 \ \ldots\ldots \ y\_m \quad (2)$$

Where x_i and y_i are chosen from a finite alphabet,

e.g.{A,C,G,T}:

$$x\_i \in \{A,C,G,T\}, \quad y\_i \in \{A,C,G,T\} \qquad (3)$$

The goal is to determine the location and length of all similar subsequences in both X and Y. Achieving this goal, the suggested approach has been divided into two consecutive algorithms:

3.1 The matching algorithm

The matching algorithm compares two DNA sequences for all possible identical matches. The data structure required for the suggested algorithm is shown in Fig.2. The major DNA sequence string is represented in the linked-list X while the minor one is represented in circular linked-list Y. Each element of the linked-list Y contains a data filed along with a single directional pointer to the next element in a circular manner. Each data field of Y holds one of the minor DNA sequence characters (y_i). Each element of the linked-list X consists of three fields described as follows. The first one is a data field holding one of the major DNA sequence characters (x_i), the second field is a pointer to an independent linked-list, where the collection of those independent linked lists are clustered and considered as array of linked-list Z. The array Z will be used to hold the resultant of the matching algorithm as will be illustrated later. The third filed is simply a pointer to the next element of the linked-list X.

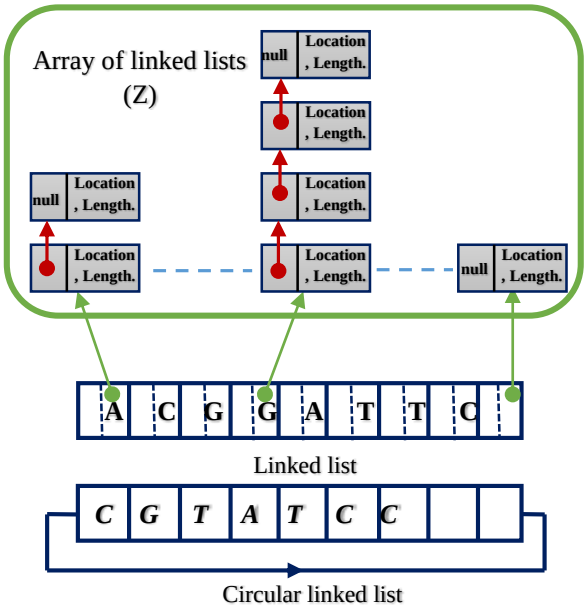

Fig.2. The data structures used in the suggested approach

The suggested algorithm begins by creating and then initializing the prescribed data structure. The linked list X and Y are created with lengths equal to the lengths of the major and minor DNA sequences respectively. The major DNA sequence (the longer one) is streamed and each character is inserted sequentially in the data filed of a cell of linked-list X. The same thing is done with the minor DNA sequence (the shorter one) where its characters are inserted sequentially in the linked-list Y. The linked lists of the array Z is left without creation as they will be created during the matching process.

The matching algorithm aims to determine both the location and length of all possible common subsequences between the two DNA sequences represented in X and Y linked lists respectively. For each match, the location and length will be added to the corresponding node in the array of linked lists Z. The main matching algorithm and its subroutines are listed in List. 1 through List.3.

List.1: Pseudo code of Matching Algorithm

```
//    Input : DNA1  and DNA2 sequences
Get Length_of_DNA1;
Get Length_of_DNA2;

For I = 0; I < Length_of_DNA1
   For j = 0; J < length_of_DNA2
      x  = read character from DNA1[I]
      y  = read character from DNA2[J]

      if ( x == y)
      {
       Perform Get_substring_until_no_match(I,J);
       // L = length of the obtained substring
       Add_ obtained_substring_to_Z(I,J, L);
       }

    Next J;
Next I;
```

As illustrated in List.1, when the two variables x and y are equal, the process named ***Get_substring_until_no_match*** (List 2) will be performed continuously reading new pair of characters as long as the new pair of characters x and y are equal otherwise the process terminates. The length and location of the obtained substring are considered as input to the process named ***Add_obtained_substring_to_Z*** (List 3).

The ***Add_obtained_substring_to_Z*** process is responsible of adding information related to the obtained substring to the array of linked lists Z. Each character of the major DNA sequence, which is represented by X[I], points to a separate linked list in the array Z. The initialization process does not create the linked lists in array Z but leave this task for process ***Add_obtained_substring_to_Z*** to create only the actually needed linked lists to save the allocated memory space. Each separate linked list Z[I] should hold information about all common substrings between sequence X, starting at position I, and sequence Y. Accordingly, the process adds a new node to the corresponding linked list Z[I] writing both the length of the substring (Length**)**, and its location (J) in the second sequence Y to the corresponding fields of this node.

By the end of the matching process, some characters of the DNA sequence X have linked lists in the array Z and

the others do not have.

List.2: Pseudo code Get_substring_until_no_match

```
// Input : DNA1  and DNA2 sequences
//  I, J
m =I;
n = J;
Length =0;
 x  = read character from DNA1[m]
 y  = read character from DNA2[n]

while (x == y)
{
    Length ++;
    m++;
    n++;
    x  = read character from DNA1[m]
    y  = read character from DNA2[n]
 }

Return Length
```

List.3 :Add_obtained_substring_to_Z

```
Input :  I, J, Length
m =I;
n = J;
I  = Length;

if X[m].pointer == null
{
create new linked_list Z[m]
add new cell with ( n, l )
//  n  =  location of substring in DNA2 sequence
// l  =  length of substring
}
Else
Add new cell to linked_list Z[m]
```

The following example illustrates the matching process of the suggested algorithm:

Input: two sequences:

DNA-1 sequence = “ATCAGTTACGT”
DNA-2 sequence = “TATCATG”

The two sequences are placed in linked lists X and Y respectively. The matching process yields the common subsequences and place their locations and lengths in array Z of the linked lists (Red boxes in Fig.3). For example, the first character at position 0 of the first sequence (“A”) has two matches with the second sequence at locations 1 and 4 with lengths 4 and 2 respectively. The first matching has length of 4 where the subsequence “ATCA”exists in both sequences. The second matching has length of 2 where the substring “AT” exists in both sequences.

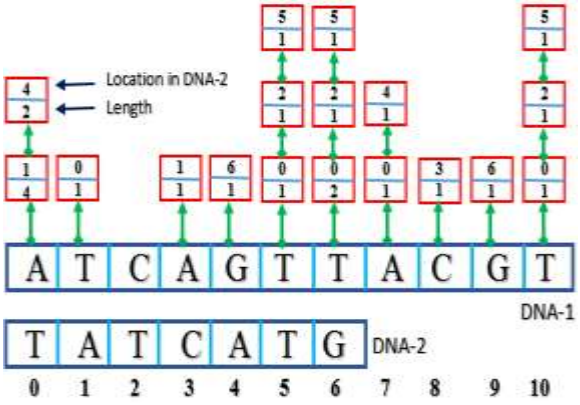

Fig.3. Illustration example of the matching process

Inspecting the linked lists produced by the matching process leads to finding that the longest common substring is “ATCA” at location 0 of the first sequence and location 1 of the second sequence.

3.2 Information retrieving algorithm

List.4: Pseudo code Information_retrieving

```
//   Input : linked list X where each cell consists of:
//            X.P_to_Z    (pointer to a linked list in Z)
//            X.value      (data)
//    Input : linked list array Z where each cell consists
of:
//            Z.Loc_in_Y   (location in DNA2 sequence)
//            Z.length      (length of matching string)

Set pointer P to the first cell of linked_list X

 For I = 0; I < Length_of_DNA1
{
Read cell from linked list X
 Get from this cell pointer P_to_Z
//which points to the corresponding linked list in array
Z

      While (P_to_Z != null)
      {
        Read  from  Z  the  cell  pointed  to  by  P_to_Z
pointer
        L = Z.Loc_in_Y (Get length of matching string)
        y = Get location of matching string in linked list
Y
        Display retrieved information
        P_to_Z   =  P_to_Z.next
      }

P = P.next  (point to next cell in X)
}
```

Performing the matching algorithm yields a massive amount of linked lists in array Z. Every cell of each linked list includes information (location and length) for a single match between the two DNA sequences. To retrieve this information, there is a need for a search

algorithm to crawl through the linked lists included in array Z. The algorithm begins reading the cells of linked list X sequentially where each cell contains a pointer (or null if there is no match at this position) to a single linked list in Z. Each linked list in array Z has number of cells, where each cell holds the length and location of common matching subsequence between the pair of DNA sequences. The algorithm is more clearly explained in List. 4.

IV. Implementation and Experimental Results

Table 1. Relation between DNA sequence length and time of processing (both the major and minor sequences are of the same length)

DNA sequence length (Bytes)	First time component (Seconds)	Second time component (Seconds)	Total processing time (Seconds)
100	0.014	1.115	1.129
150	0.03	2.383	2.413
200	0.04	4.137	4.177
300	0.118	9.25	9.368
400	0.267	16.76	17.027
500	0.482	25.845	26.327
600	0.697	39.124	39.821
700	0.932	49.29	50.222
800	1.683	67.447	69.13
900	1.853	81.893	83.746
1000	2.425	107.085	109.51
1200	4.136	144.045	148.181
1400	7.266	195.367	202.633
1500	8.496	227.435	235.931
1600	9.268	258.286	267.554
1700	12.158	289.917	302.075
1800	13.26	326.128	339.388
1900	15.209	360.33	375.539
2000	18.443	399.903	418.346
2500	34.252	633.684	667.936

The suggested algorithm has been implemented using C# language to perform comparison between two input long DNA sequences and display all possible common subsequences in suitable manner. In the first stage, two DNA sequences with the desired lengths are randomly generated for the test purpose and located in the corresponding linked lists X and Y respectively. The matching algorithm is performed in the second stage of the program. The obtained results are retrieved from the array of linked lists Z and displayed in suitable user interface in the last stage. The program has been applied to several DNA sequences with different lengths and the time of processing is computed and then recorded in each case. The time of processing has been studied in more deep: the total time of processing consists of two components; the first one is consumed in comparing the two DNA sequences, constructing and inserting data in the array of linked lists corresponding to each position in the major sequence. The other time component is consumed in traversing the linked lists to retrieve the locations and lengths of the common subsequences. It has been found that the first time component is trivial compared to the second one (Table-1). The relation between time of processing and its dependence on the length of the major string is considered and is represented in Fig.9.

Applying the Matlab fit function to fit polynomials to the obtained data yields the following polynomial equation:

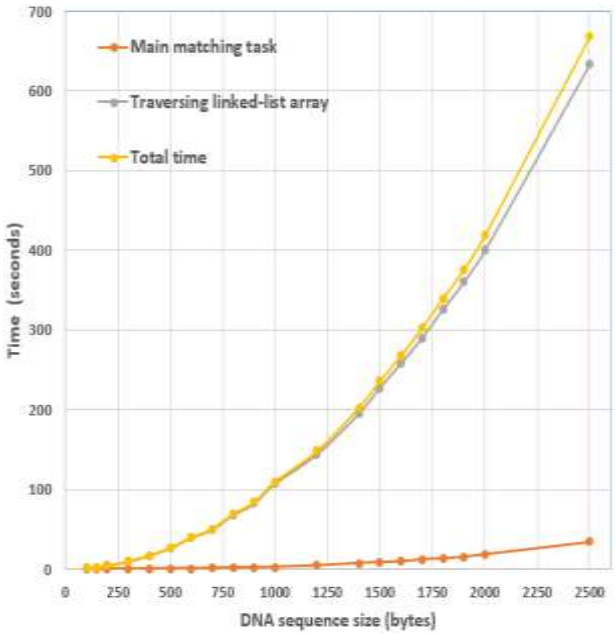

Fig.4. Relation between time consumed in constructing linked lists and time consumed in traversing them in the suggested approach

$$t = 6.205\ x\ 10^{-9}\ S^3 + 8.537\ x\ 10^{-5}\ S^2 + 0.01558\ S - 2.379$$

Where t is the time consumed in seconds and S is the DNA sequence size in bytes.

V. Conclusion

DNA patterns matching is a fundamental and upcoming area in computational molecular biology. The algorithm proposed in this paper addressed not only the problem of locating the longest common subsequences (LCS) in two different sequences but also finds exactly all common subsequences along with their locations and lengths. It is based on the convolution between the two sequences (named major sequence X and minor one Y) and creating a node in the corresponding linked list in array Z for each match between the two sequences. If two or more matches share the same location in sequence X, the corresponding nodes are clustered constructing a single linked-list. The matching process yields a group of linked-lists containing nodes arranged in certain manner representing all possible matches between sequences X and Y. The obtained results, compared with another algorithms [22] presented very good speedups and indicated that impressive improvements has been achieved. The proposed algorithm can be more developed to locate the longest common subsequences between the major DNA sequence and multiple minor sequences. Also, it can be modified to perform the process of alignment between two DNA sequences. Moreover, the algorithm needs to be developed to be able

to run in parallel computing manner to cope with the long time processing problem.

REFERENCES

[1] O. Gotoh, "An Improved Algorithm for Matching Biological Sequences," Journal of Molecular Biology, 162, pp:705~708, 1982.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] S. Grier, "A tool that detects plagiarism in Pascal programs", ACM SIGCSE Bulletin, vol. 13, no. 1, (1981), pp. 15-20.

[3] J. A. W. Faidhi and S. K. Robinson, "An empirical approach for detecting program similarity within a university programming environment", Computers & Education, vol. 11, no. 1, (1987), pp. 11-19.

[4] U. Manber, "Finding similar files in a large file system[C/OL]", In: Proceedings of the Winter USENIX Conference, (1994), pp. 1-10.

[5] BLAST, http://blast.ncbi.nlm.nih.gov/Blast.cgi, (2011) September.

[6] C. Yu, S.-Y. Cheng, R. L. He and S. S. -T. Yau, "Protein map: An alignment-free sequence comparison method based on various properties of amino acids", Gene, vol. 486, (2011), pp. 110-118.

[7] Y. Guo and T. -m. Wang, "A new method to analyze the similarity of the DNA sequences", Journal of Molecular Structure: THEOCHEM, vol. 853, (2008), pp. 62–67.

[8] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins", Journal of Molecular Biology, vol. 48, no. 3, (1970), pp. 443–53. doi:10.1016/0022-2836(70)90057-4. PMID 5420325, http://linkinghub.elsevier.com/retrieve/pii/0022-2836(70)90057-4.

[9] C. S. Iliopoulos and M. S. Rahman, "Algorithms for Computing Variants of the Longest Common Subsequence Problem", Theoretical Computer Science archive Journal, vol. 395, no. 2-3, (2008), pp. 255-267.

[10] C. -P. P. Wu, N. -F. Law and W. -C. Siu, "Cross chromosomal similarity for DNA sequence compression", Bioinformation, vol. 2, no. 9, (2008), pp. 412-416.

[11] C. S. Iliopoulos and M. S. Rahman, "New Efficient Algorithms for LCS and Constrained LCS Problem", In Proceedings of the Third ACiD Workshop Durham, UK, vol. 9 of Texts in Algorithmics. King's College London, (2007), pp. 83-94.

[12] A. F. Klaib, Z. Zainol, N. H. Ahamed, R. Ahmad and W. Hussin, "Application of Exact String Matching Algorithms towards SMILES Representation of Chemical Structure", International journal of computer and information science and engineering, (2007), pp. 497-501.

[13] K. Rieck, P. Laskov and K. -R. M¨uller, "Efficient Algorithms for Similarity Measures over Sequential Data: A Look Beyond Kernels", DAGM 2006, LNCS 4174, (2006), pp. 374–383.

[14] G. Fox, X. Qiu, S. Beason, J. Y. Choi, M. Rho, H. Tang, N. Devadasan and G. Liu, "Case Studies in Data Intensive Computing", Large Scale DNA Sequence Analysis as the Million Sequence Challenge and Biomedical Computing, (2009).

[15] K. Derouiche and D. A. Nicole, "Semantically Resolving Type Mismatches in Scientific Workflows", OTM 2007 Workshops, Part I, LNCS 4805, (2007), pp. 125–135, Springer-Verlag Berlin Heidelberg 2007.

[16] DIALIGN, http://dialign.gobics.de/, (2011) September.

[17] D. Rose, J. Hertel, K. Reiche, P. F. Stadler and J. Hackermüller, "NcDNAlign: Plausible multiple alignments of non-protein-coding genomic Sequences", Genomics, vol. 92, no. 1, (2008), pp. 65-74.

[18] E. Dong, J. Smith, S. Heinze, N. Alexander and J. Meiler, "BCL::Align—Sequence alignment and fold recognition with a custom scoring function online", Gene, vol. 422, no. 1-2, (2008), pp. 41-46.

[19] B. Vishnepolsky and M. Pirtskhalava, "ALIGN MTX—An optimal pairwise textual sequence alignment program, adapted for using in sequence-structure alignment", Computational Biology and Chemistry, vol. 33, no. 3, (2009), pp. 235-238.

[20] P. Kalsi, H. Peltola and J. Tarhio, "Comparison of Exact String Matching Algorithms for Biological Sequences", In: Proc. BIRD '08, 2nd International Conference on Bioinformatics Research and Development (ed. M. Elloumi et al.). Communications in Computer and Information Science 13, Springer (2008), pp. 417-426.

[21] G. Huang, H. Zhou, Y. Li and L. Xu, "Alignment-free comparison of genome sequences by a new numerical characterization", Journal of Theoretical Biology, vol. 281, no. 1, (2011), pp. 107-112.

[22] Izzat Alsmadi, Maryam Nuser, "String Matching Evaluation Methods for DNA Comparison," International Journal of Advanced Science and Technology Vol. 47, October, 2012, p. 13-32.

[23] J. K. Me, M. R. Panigrahi, G. N. Dash and P. K. Meher, Wavelet Based Lossless DNA Sequence Compression for Faster Detection of Eukaryotic Protein Coding Regions, IJIGSP Vol.4, No.7, July 2012.

[24] Mohammed Abo-Zahhad, Sabah M. Ahmed and Shimaa A. Abd-Elrahman. A Novel Circular Mapping Technique for Spectral Classification of Exons and Introns in Human DNA Sequences, IJITCS Vol. 6, No. 4, March 2014, PP.19-29.

[25] G. Sethuraman,Kavitha Joseph, Star Coloring Problem: The DNA Solution, IJITCS Vol. 4, No. 3, April 2012, PP.31-37.

[26] M.I.Khalil, M.A.Hadi, Finding Longest Common Substrings in Documents, IJIGSP Vol. 7, No. 9, 2015, 9, 27-33.

Authors' Profiles

M.I. Khalil: Egyptian, male, has obtained his B.Sc degree in Computer and Automatic Control Engineering from Faculty of Engineering, Ain Shams University, Cairo, Egypt, in 1983, M.Sc degree in Computer Engineering from Faculty of Engineering, Tanta University,Tanta, Egypt, in 2003 and Ph.D degree in Computer Systems Engineering from Faculty of Engineering, Benha University, Cairo, Egypt, in 2005. He is currently working as Associate Professor in Department of Networking and Communication systems at the Faculty of Computer and Information Sciences, Princess Noura Bent Abdulrahman University, Riyadh, KSA. He has 15 years of previous experience at the Reactor Physics Department, Nuclear Research Center (NRC), Egyptian Atomic Energy Authority Cairo (EAEA), Egypt in the field of Data Acquisition and Interface Design. His main research interests focus on: Digital Signal Processing, Wireless Sensor Networks,

Personal and Mobile Communications. So far, he has twelve years of teaching experience and has published more than twenty-five papers in repute journals and proceedings of conferences in fields of the data acquisition, digital signal processing, image processing and neural networks.

How to cite this paper: M. I. Khalil,"Locating All Common Subsequences in Two DNA Sequences", International Journal of Information Technology and Computer Science(IJITCS), Vol.8, No.5, pp.81-87, 2016. DOI: 10.5815/ijitcs.2016.05.09

