Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.9, No.4, 2018

www.iiste.org

Index Based Pattern Searching Algorithm Using DNA Sequence

Tania Islam
Computer Science and Engineering Department, University of Barisal, Karnokathi, Barisal, Bangladesh

Abstract
Bioinformatics is the most studied field of computer science. It deals with computer science, biology and information technology. String matching is one of the application fields in Bioinformatics. There are different types of algorithm already developed and several types of research works are running. Index Based Jumping algorithm (IBJ) is the proposed algorithm to find a given pattern into a given string. This algorithm works better than that of IBSPC, IFBMPM, IBKMPM and ISMPMC algorithm in terms of minimizing character comparisons and CPC ratio.
Keywords: String, DNA Sequence, CPC ratio, pattern

1. Introduction

String matching at-times called string searching is a conventional problem in the field of computer science. A part of strings called “Pattern” is to be searched within a well-built string or in a given “Text”. String matching strategies or algorithms provide key role in various real world problems or applications. A few of its imperative applications are Spell Checkers, Spam Filters, Intrusion Detection System, Search Engines, Plagiarism Detection, Bioinformatics, Digital Forensics and Information Retrieval Systems etc (Krishna and et al, 2012). Figure 1 shows the application of sting matching in different fields.

In general, basic biological information is kept in strings of nucleic acids (DNA, RNA) or amino acids (proteins). Aligning sequences assist in exposing their common characteristics, whereas matching sequences could conclude valuable information from them.

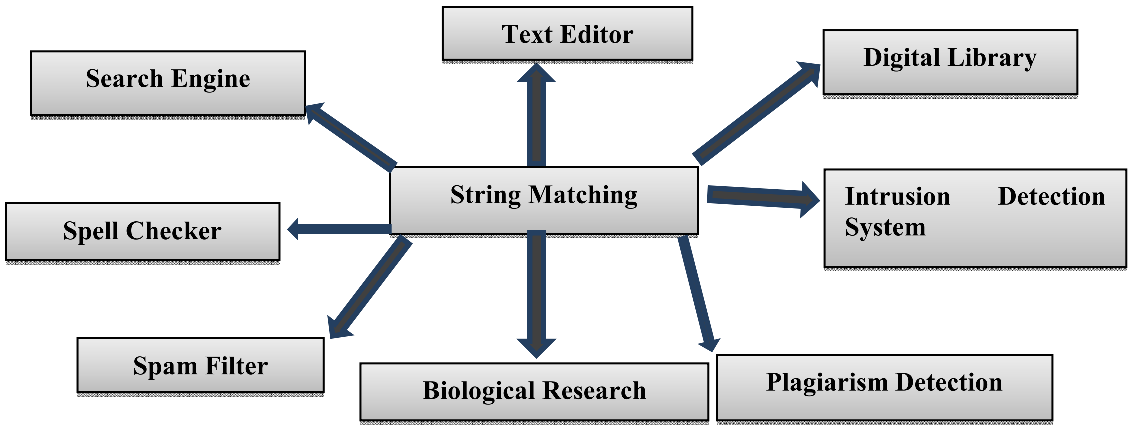

Fig 1: Applications of String Matching

Because of the large amount of DNA data available on websites, matching of nucleotide sequences is becoming an essential application and also there is a growing interest in extremely fast computer technique for data analysis and retrieval (Soni and et al, 2014).

2. Related Work

There are different types of string matching algorithms already developed but still its required efficient algorithm which has less character comparison for pattern matching. Here we work with DNA sequence. DNA or deoxyribonucleic acid is the genetic component in human’s species and almost all other creature. It is the combination of four characters, A (Adenine), G (Guanine), C (Cytosine) and T (Thiamine). A character set ∑={A, C, G, T} is the set of alphabet for DNA sequence which are used in this algorithm. Both given string and pattern are combination of those four characters (R. Bhukya and et al, 2011)(G. Cai and et al, 2009).

The following symbols are used in our proposed algorithm:

- We consider a given string which is declared as S[n], where n is the length of the string.
- It’s possible that sometimes the string does contain any character so it’s a null string and define by φ
- In our algorithm we search for a pattern in our given string and the pattern is defined by P[m], where m is the length of the pattern (R. Bhukya and et al, 2011),(R. Bhukya and et al, 2011).

The most oldest and initial algorithm for string searching is the Brute-force algorithm(C. Charras,2004). But it is the least efficient to check a specific pattern in a given string. This algorithm works by matching with given text that is $T = t_0.t_1.t_2 \ldots\ldots..t_{n-2.}t_{n-1}$ with given pattern that is $P = p_0.p_1.p_2 \ldots\ldots..p_{m-1}$ in character by character. That means when $t[0]$ is not equal to $p[0]$ then it starts it matching with $t[1]$ and $p[0]$. And this process is continued until any mismatch is found. After finding the complete match of the pattern it gives back the position of the pattern and continued its process of next character. The searching phase time complexity of the algorithm is $O(n * m)$, where m is the length of the pattern and n is the length of the given string.

Raju Bhukya and DVLN Somayajulu proposed an algorithm name EPMSPP (R. Bhukya and et al, 2011). They processed the input as pair of character and also the pattern as pair of index. They consider each character only once that means when one character is already in a pair; it is not used in another pair. For given text the pair that is occurred least times compare first. Though DNA sequence contains only four characters A,G,C and T. So it will make 16 pairs among them.

IFBMPM (R.Bhukya and et al, 2010). algorithm works with index value of both pattern character and string character. It works with forward backward techniques and started matching from left to right. At first it stores all index value of all character and check for the first character of the pattern. Which character appears first in the pattern is considered for starting matching position.

ISMPMC(R. Bhukya and et al, 2011) algorithm works for matching pattern using DNA sequences. At first it stores the position of all characters in the string and which character is found minimum number in the string, matching will start from that pattern with the same character. If it makes a match, continue for the rest of the character from left to right order but if it not, go for the next occurrences of that character.

IBMPMP(R. Bhukya and et al, 2011) algorithm works with indexes of DNA sequences characters. Instead of considering each single character it used pair of characters. From the four alphabet of DNA sequence, they make 16 pair of character and this algorithm also maintains the frequency of each pair. The pair which is present least time in the pattern is considered for matching first.

Raju Bhukya and DVLN Somayajulu are proposed another algorithm name IBSPC(R. Bhukya and et al, 2011) algorithm, works for multiple patterns matching on DNA sequence and they emphasized on the reduced the number of character comparisons ratio which is less than 1. They used index table for both string and pattern and used count variable for counting the no of occurrence of each character. The character which appears in minimum numbers in patterns consider as first priority for comparison and the characters which appears in given string in maximum numbers is considered as first priority for comparison's.

3. Proposed Methodology

There are two general parameters which are used to estimate the performance for the string matching algorithm that are used in different types of applications. These two parameters are given below (K. H. Adil, and N. A. Rashid, 2014):

A. Number of character comparisons:
   Number of character comparison which occurred when we match a given pattern with a given pattern. The algorithms with less character comparisons are considered as better.
B. Number of Attempt:
   When any given text is shifted due to any match or mismatch of pattern with the given string is considered as number of attempt. The performance of algorithm is better when the number of attempt is less.

So the motive of this work is to increase the efficiency of the algorithm by reducing the number of character comparisons and also the CPC ratio. Proposed Index based jumping (IBJ) string matching algorithm combines two phases; preprocessing phase and searching phase. This approach will be affecting the number of character comparison of searching processes between the text and the pattern.

The preprocessing phase and searching phase of the proposed Index Based Jumping Algorithm (IBJ) are summarized into the following stage:

3.1 Preprocessing Phase

Preprocessing phase is used to preprocess the pattern or given string. It preprocesses the given text, in order to minimize the number of attempts and number of character comparisons. Proposed algorithm uses the index value of given string/DNA sequence for pre-processing. Pre-processing phase completes in two stages.

3.1.1 Pre-process the given input string

At first it tries to know the all index of given string. We work with DNA sequence and it only contains four alphabets {A, G, C, T}. For this purpose we have to maintain four array index table.

It used ASCII indexing technique to reduce the pre-processing time and comparisons. For each character in Σ it computes array subscript value by using the following technique [(S[i]-64)%5](Raju Bhukya 2011).

SL/No	DNA	ASCII Value	ASCII Value-64	(ASCII Value-64)%5	Array Subscript
1	A	65	1	1	1
2	C	67	3	3	3
3	G	71	7	2	2
4	T	84	20	0	0

So, by using each character ASCII value with this formula we get subscript value for T, A, G and C in the range 0, 1, 2, 3 which is needed for subscripting 2D vector of size [4][n]. So, for each character of string in the function [(S[i]-64)%5] directly refers to its corresponding vector in the 2D vector table stab[4][n].

Let string is S with length of n and Pattern is P with length of m. Suppose, it has a string like AGCCTTTCGACC as our input string. Then it has to construct the following index table.

DNA	Sequence Indices				
T[0]	4	5	6		
A[1]	0	9			
G[2]	1	8			
C[3]	2	3	7	10	11

3.1.2 Compute Substring

In second stage of pre-processing phase it has to divide the given input string into different substrings. The length of substring is equal to the length of pattern. It should follow the following criteria:

The substring length is exactly equal to pattern length. This process starts to compute substring from the text that matches with the pattern 1st character as we already know the index value of text.

Computing of substring ends at substring length= [S[n]-1]-(P[m]-1)].

Suppose for the above given string, the pattern = TTT. The substring for this pattern will be as follows:

DNA substring	Indices of substring
TTT	4
TTC	5
TCG	6

3.2 Searching phase

In this proposed method, searching phase start with matching the last character of pattern with the last character of substring. If any mismatch is found, it is shifted to next substring. If any match found with the last character, it returns to the 2nd character of substring with the 2nd character of pattern. If it matches, continue for the third one and so on.

Unimportant or less significant cases for string matching algorithms in searching phase that may be occurred when search the pattern into the given string:

Case 1: When both string and pattern are not contained any character, the possibility of finding the pattern into the given string is 0.

Case 2: When the given pattern is null but given string is not null then the possibility of pattern into the given text is 0.

Case 3: When given string is null but given pattern is not, then the possibility of occurring of pattern into the given string is 0.

Case 4: If there is any case where pattern length is greater than the given string length, it will not make any exact matching.

Case 5: When given string length is equal to given pattern length then there is possibility of occurring the pattern into the given string is exactly one time.

4. Working Example

In this section, we give a working example to explain our proposed IBS algorithm.

Given string,

S = ATCTAACATCATAACCCTGCAGAGAGGAT

Given Pattern,

P = GCAGAGAG

At first stage, it pre-processes the given string by storing it indices into a two-dimensional array. After storing the indices of all character of given text, the value of indices is given in Table 1.

Table 1. Indices of given string

DNA	Indices										
T	1	3	8	11	17	28					
A	0	4	5	7	10	12	13	20	22	24	27
C	2	6	9	14	15	16	19				
G	18	21	23	25	26						

Now it divides the string into different substrings. Before dividing the text into substrings, it checks the 1st character of pattern. In this given pattern, it starts with character G. So all the substring should start with G and also the length of substring is equal to the length of pattern. By following this procedure, our computed substring is given in table 2.

Table 2. Indices of computing substring

Substring	Indexes
GCAGAGAG	18
GAGAGGAT	21

In this process it is not considered 23, 25, and 26 number of position of character G, because, if we consider them the length of substring will exceed the length of pattern.

Now our pre-processing stage is completed. So we start our matching stage.

1st attempt

At first, we compare from right to left of the pattern and 1st substring last character is matched with last character of 1st substring. It's a match.

G	C	A	G	A	G	A	G
G	C	A	G	A	G	A	G

So continue our matching from left to right direction at the end of the pattern and substring.

G	C	A	G	A	G	A	G
G	C	A	G	A	G	A	G

G	C	A	G	A	G	A	G
G	C	A	G	A	G	A	G

G	C	A	G	A	G	A	G
G	C	A	G	A	G	A	G

G	C	A	G	A	G	A	G
G	C	A	G	A	G	A	G

G	C	A	G	A	G	A	G
G	C	A	G	A	G	A	G

G	C	A	G	A	G	A	G
G	C	A	G	A	G	A	G

Now it's a complete match, so the number of matching indices is 18.

2nd attempt

Compare from right to left of the pattern and substring. In this algorithm, it do not compare the first character because we already know that the both characters are equal.

G	A	G	A	G	G	A	T
G	C	A	G	A	G	A	G

Now, compare with the last character of substring with the last character of pattern that is T ≠ G and skips this substring. So our pattern window shifts to the next the substring.

Finally, there is no more substring left. If there is more substring left, continue our process to find the repetitions of the pattern occur in given string.

So, the number of character comparison is 8 and number of attempt is 2.

5. Experimental Result Analysis

The proposed algorithm is tested using 1024characters as input string and different number of pattern (1 to 20 characters) and compare IBJ algorithm with IFBMP, IBKMPM, ISMPMC and IBSPC algorithm. From table 3, it can be said that IBJ algorithm requires minimum number of character comparison.

Table 3: Comparisons with different algorithms

Pattern	No.of char	No.of Occu	IFBMPM	IBKMPM	ISMPMC	IBSPC	IBJ
A	1	259	518	259	259	259	259
AG	2	53	624	518	300	300	259
CAT	3	11	567	542	542	320	246
AACG	4	5	614	614	318	311	259
AAGAA	5	2	616	607	357	340	259
AAAAAA	6	3	627	620	647	344	259
AGAACGC	7	2	600	613	342	368	259
AAAAAAGG	8	1	634	623	368	346	259
GCTCATTAG	9	1	582	590	399	335	246
CCTTTTCCGG	10	1	562	578	584	347	242
TTTTGCCGTGT	11	1	650	650	350	361	269
TTCTTAATAAAA	12	1	651	634	382	332	268
GGGACCAAAAAAT	13	1	579	582	336	350	245
TTTTGCCGTGTTGA	14	1	638	654	353	351	268
CCTCCAAAAAAGGCT	15	1	578	558	589	354	241
GGCTGTTCAACGCTCC	16	1	598	580	356	597	244
TTTTCGATTGCTCATTA	17	1	643	633	359	359	267
GGGATTTGGCTATACTCC	18	1	598	580	347	355	244
GGCCTTGTCTAAAGGTATG	19	1	579	585	361	344	244
CCTGAGCGCGTCCTCCGTAC	20	1	570	582	592	590	240

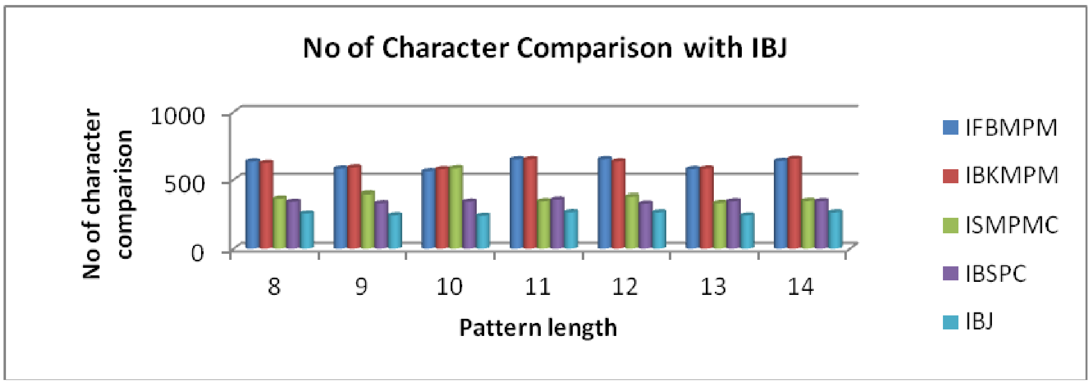

Fig 2: Number of character comparisons with different algorithms

Table 4 Number of CPC with other algorithms

Pattern(P's)	No.of char	No.of Occu	IFBMPM	IBKMPM	ISMPMC	IBSPC	IBJ
A	1	259	.505	0.25	.253	0.25	.253
AG	2	53	.609	0.50	.293	0.29	.253
CAT	3	11	.553	0.52	.529	0.31	.240
AACG	4	5	.599	0.59	.311	0.30	.253
AAGAA	5	2	.601	0.59	.349	0.33	.253
AAAAAA	6	3	.612	0.60	.632	0.33	.253
AGAACGC	7	2	.585	0.59	.334	0.35	.253
AAAAAAGG	8	1	.619	0.60	.359	0.33	.253
GCTCATTAG	9	1	.568	0.57	.390	0.32	.240
CCTTTTCCGG	10	1	.548	0.56	.570	0.33	.236
TTTTGCCGTGT	11	1	.634	0.63	.342	0.35	.263
TTCTTAATAAAA	12	1	.635	0.61	.373	0.32	.262
GGGACCAAAAAAT	13	1	.565	0.56	.328	0.34	.239
TTTTGCCGTGTTGA	14	1	.623	0.63	.345	0.34	.262
CCTCCAAAAAAGGCT	15	1	.564	0.54	.575	0.34	.235
GGCTGTTCAACGCTCC	16	1	.583	0.56	.348	0.58	.238
TTTTCGATTGCTCATTA	17	1	.627	0.61	.351	0.35	.261
GGGATTTGGCTATACTCC	18	1	.583	0.56	.339	0.34	.238
GGCCTTGTCTAAAGGTATG	19	1	.565	0.57	.353	0.33	.238
CCTGAGCGCGTCCTCCGTAC	20	1	.556	0.56	.578	0.57	.234

Again from figure 2, it can be clearly said that number of character comparison in proposed IBJ algorithm is minimum than others algorithm.

Comparisons per character of any algorithm is calculated by total number of character comparison is divided by total number of characters in input string. From table 4, which is represented the number of CPC of IBJ with four different algorithms.

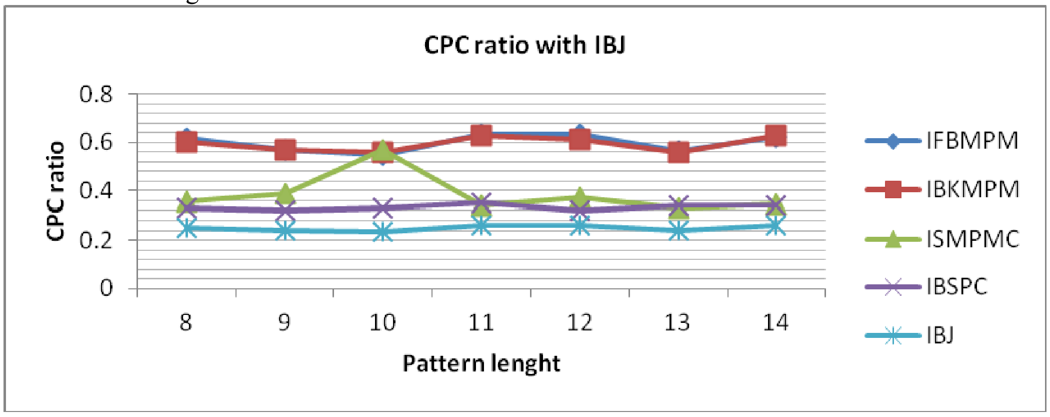

Fig 3: CPC of IBJ algorithm with others algorithm

6. Conclusion

String matching is one of the most studied fields in computer science. The proposed IBJ algorithm will play a vital role in the field of medical science. This algorithm is being capable to reduce number of character comparisons and number of comparison per character (CPC) ratio. In this paper, IBJ algorithm compared with four different algorithms and every time it shows better result than that of others.

References

Krishna V. S., Rasool A. and Khare N(2012)," String Matching and its Applications in Diversified Fields", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012.

Soni K. K., Vyas R., Sinhal A(2012)," Importance of String Matching in Real W orld Problems", International Journal of Engineering and Computer Science ISSN: 2319-7242 Volume 3, Page No. 6371-6375, Issue 6 June, 2014.

R. Bhukya and D. Somayajulu(2011) , "An Index Based Sequential Multiple Pattern Matching Algorithm Using Least Count", International Conference on Life Science and Technology, IACSIT Press, Singapore, vol. 3,

2011.
G. Cai, X. Nie and Y. Huang(2009), "A Fast Hybrid Pattern Matching Algorithm for Biological Sequences," 2nd International Conference on Biomedical Engineering and Informatics, October 2009.
R. Bhukya , D. Somayajulu(2011), "Index based sequential multiple pattern matching algorithm using pair indexing", International Conference on Life Science and Technology, IPCBEE volume 3, 2011.
C. Charras, T. Lecroq(2004), Handbook of Exact String Matching Algorithms, King's College London Publications, 2004.
Raju Bhukya , DVLN Somayajulu(2011) "Exact Multiple Pattern Matching Algorithm using DNA Sequence and Pattern Pair", International Journal of Computer Applications (0975 – 8887), Volume 17– No.8, March 2011.
Raju Bhukya , DVLN Somayajulu(2010) "An Index based Forward Backward Multiple Pattern
Matching Algorithm", International Journal of Biological, Biomolecular, Agricultural, Food and Biotechnological Engineering, Volume:4, No:6, 2010.
R. Bhukya , D. Somayajulu(2011) "Index based sequential multiple pattern matching algorithm using pair indexing", International Conference on Life Science and Technology, IPCBEE volume 3, 2011.
R. Bhukya , D. Somayajulu(2011) "Index based sequential multiple pattern matching algorithm using pair indexing", International Conference on Life Science and Technology, IPCBEE volume 3, 2011.
Raju Bhukya , DVLN Somayajulu (2011)"Index based multiple pattern matching algorithm using DNA sequence and pattern count", International Journal of Information Technology and Knowledge Management ,Volume 4, No. 2, pp. 431-441,July-December 2011.
K. H. Adil, and N. A. Rashid(2014). "Maximum-shift string matching algorithms." Computer and Information Sciences (ICCOINS), 2014 International Conference on. IEEE, 2014.