International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/IJSRCSEIT

Substitution based DNA Sequences Compression-Encryption Method

Syed Mahamud Hossein
Department of Computer Science, Vidyasagar University, Midnapore, West Bengal, India

Article Info
Publication Issue :
Volume 8, Issue 4
July-August-2022

Page Number : 63-70

Article History
Accepted: 05 July 2022
Published: 14 July 2022

ABSTRACT

Now a day's research is being carried out on minimizing the executing time and rate of lossless compression as DNA sequence size are increasing in large amount. The protection of DNA sequence database from hackers is a challenging question. To solve this question, a technique known as lossless DNA sequence compression is developed which is based on searching for exact Repeat and Palindrome (RP). One of the hidden characteristic of DNA sequence is approximate repeats, this feature-RP (Repeat & Palindrome) has been consider in this work. This algorithm can be used to minimize the storage capacity and reduce the cost of transmission. The DNA sequence compression is optimized by encoding exact repeats and palindromes in match position. There must not be overlapping of the repeat and palindrome technique in DNA sequence compression. In this technique after compression two files are produced compressed and library file. This library file act as a signature and provides security. The group of characters of repeat and palindrome technique are also act as a private key and provide strong data security. This algorithm attains the greater compression rate & ratio, compared to the prevailing DNA based compression techniques and provides the strong information security. The difference between cellular DNA and artificial sequence of same length is observed. The complexity of this algorithm is O(N2) where n is the set of characters. We can get compression rate of 3.2076 bits/base by using this technique.

Keywords : DNA, Compression, Encryption, Repeat, Palindrome, rate, ratio & Security.

I. INTRODUCTION

Every year the size of DNA sequences increases tremendously [1-9]. This sequence procession is very difficult task[10-11], because this DNA sequences have same logical organization[12]. For storing purpose some special techniques have been designed. We cannot apply the marketable compression techniques [13] on DNA sequences because this sequence has some special characteristics [14]. The two bit encoding and Huffman compression techniques are inapplicable on cellular DNA sequences because these sequences are non random[10,14]. The cellular DNA sequences contained so many repetitions within the sequence[10]. The researchers developed so many compression techniques using the special structure[10,15-16] of DNA sequences. In a long DNA sequence, we cannot find out the exact match position of RP so easily. This RP search engine basically worked on fast and sensitive homology search [17-18] technique. Here the match substring is replaced by the ASCII code and match substring is placed in library file.

Also developed here another algorithm for string matching, changing string orientation and calculating file size, etc.

The definition of substring, file format and the generation of substring forming the input sequence are mentioned in paper [19]. Our algorithm is a substitution based compression technique where library file is a key.

II. METHOD

2.1 Method of Repeat & Palindrome technique (RP)

Consider a DNA sequence as tattgtagtaatgtacatatgcatatgtat . In repeat and palindrome technique, the principal idea is s_1=tat sub-string (consider length of sub sequence is 3) is repeated in how many places is shown by red color. The s_2=tat (Palindrome of tat) sub-string is repeated in how many places is shown by green color and so on.

Replace maximum number of repetitions of repeat and palindrome sub string by corresponding ASCII code.

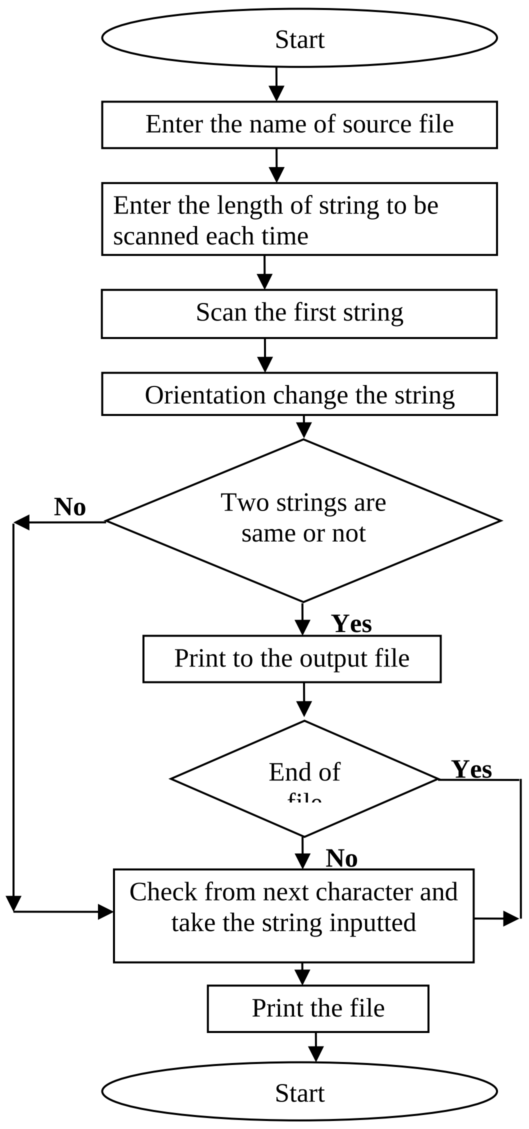
2.2 Repeat & Palindrome (RP) technique introduction

One of the hidden regularities of DNA sequence is approximate repeats. Repeat with Palindrome is taken in this work. This lossless technique is based on exact Repeat-palindrome match techniques. The exact Repeat-palindrome technique optimally compress the sequence by encoding the match position and matched position must not overlap one another.

There are two phases in this algorithm i) Repeat-palindrome exact match position finding ii) exact Repeat-palindrome regions is encoded and non-repeat & palindrome etc regions.

This technique basically worked by the combinations of Repeat-palindrome substring substitution and produced dynamic Library file. The substring size is user dependable.

2.3. Flowchart


Start
Enter the name of source file
Enter the length of string to be scanned each time
Scan the first string
Orientation change the string
Two strings are same or not
No
Yes
Print to the output file
End of file
Yes
No
Check from next character and take the string inputted
Print the file
Start

2.4 Basic terminology of Repeat & Palindrome

Searching for exact repetitions

The DNA sequence is consider as a finite sequence and composed only four alphabets a, t, g & c.

An exact all Repeated sub string of Repeat with Palindrome is a substrings in s & s_1 (where s represent the repeat and s_1 represent the palindrome sub string) and produced another substring S by edit operations (repeat, insertion). The maximum repeated position is encoded for better compression result.

Example :

Let

s=atgggtaatagtatatgtacatgcatgtagtattataggata……..n.

Where atg substring repeat on four places, gta repeat on three places (shown by red color) and so, on and ata palindrome is ata is repeated on three places (shown in green color) and so on. Highest match score is replaced first of repeat and palindrome simultaneously and atg and ata is replace by ASCII character and insert ASCII code in every match position i,e ith & jth position

B=!gtagta!tac! { where B is the intermediate step}, continue this process

o=!""!tac![compressed output file is represent by o]

2.5 *Working Principal*

i. File type: All DNA sequences are text format, the file extension is dot txt.

ii. Sub sequences repeat-palindrome are auto generated by breaking the query sequence into words

iii. Encoding by Edit process

This approximate matching technique have three edit processes

These are:

1) Matching--Find out the match position of repeat & palindrome substring

1) Replacing-- Replacing the substring (R) by the character (char) at the position (p) is define as (R, p, char).

2) Inserting- Inserting ASCII symbol S at position (p) by the character (char) is expressed as (S, p, char).

2.6 Algorithm

Repeat and Palindrome DNA sequence compression algorithm

```
INITIALIZATION OF INPUTS:
i. DNA sequence & Artificial sequence in text format
ii. Word size of length l
iii. S=w_ri

ESTIMATED OUTPUT :
i. Compressed output file and Library file

START
i. Define ASCII code start position
ii. Word size 1 to <10 and count
iii. Product of different word
iv. Match word with the DNA sequence
v. Request to store output in two separate file

ITERATION

Step 1: Here use three text file. First is for take input of Genome sequence i.e, any combination of {a,c,g,t}. Second is for Dynamic look up table. And last one for out put.
Step 2: Store the input form the first text file to a buffer, say s.
Step 3: We have to check whether the first input sequence is {a,t,g,c} or not. If true do step 4 to
        step8 else increment the position by one.
Step 4: We have to match the whole input sequence according to the first four taking sequence.
Step 5: If the number of matching sequence found greater than one then do step 6 to step 8 else increment the position by one.
Step 6: Write the ASCII character with its corresponding matching sequence into the dynamic look up table.
Step 7: Replace the sequences with corresponding ASCII character in the input string where the matching sequences are found.
Step 8: Increment the value of the ASCII counter by one.
Step 9: Now we have to write the input buffer
```

```
into the output file. After doing those steps successfully the input buffer will be the compressed genome sequence.
Step 11:  Stop
```

DNA sequence decompression algorithm based on Repeat and Palindrome method

```
INITIALIZATION OF INPUTS:
i.  Enter the compressed text file and library file
ESTIMATED OUTPUT :
i.  Exact original  sequence

START
iv. Replace ASCII code by DNA sub sequence

ITERATION
Step 1 : Declare three FILE pointer fp, fp1, fs and five character variable say ch1,ch2,ch3,ch4,ch5.
Step 2 : fp is required to point the encrypted file & retrive the encrypted genom sequece.
Step 3 : fs is required to point lookup table & retrive the genom sequeuence and croessponding ASCII character.
Step 4 : fp1 required to point the decripted file & used to store the decrypted genom sequence.
Step 5 : Store the encryped genom sequence in a temporary buffer say s.
Step 6 : Determine the number of neucleotide exist in the encrypted file (say len).
Step 7 : Initialize a counter (say i) is equal to 0 (zero). Repeat step 8 to step 15 until i lessthan number of neucleotide
        exist in the encrypted file i.e, (i<len).
Step 8 : Seek the FILE pointer fs to the first neucleotide in the lookup table by rewind(fs).
Step 9 : Do step 10 to step 14 while(!feof(fs)).
Step 10: Initielize the following :
            ch1=fgetc(fs) i.e, ch1 holds the first neucleotide present in the lookup table.
            ch2=fgetc(fs) i.e, ch2 holds the second
ide present in the lookup table.
            ch3=fgetc(fs) i.e, ch3 holds the thired
ide present in the lookup table.
            ch4=fgetc(fs) i.e, ch4 holds the fourth
ide present in the lookup table.
            ch5=fgetc(fs) i.e, ch1 holds the first neucleotide
in the lookup table.
Step 11: If s[i]=ch5 then print ch1,ch2,ch3,ch4 into the decrypted output file.
Step 12: else if(s[i]=='a' || s[i]=='t' || s[i]=='g' || s[i]=='c') then print s[i] into the decrypted output file.
Step 13: else continue.
Step 14: end if.
Step 15: end for.
Step 16: The number of neucleotide present in the encrypted file is equal to len.
Step 17: Calculate the total estimated time to decompress the encrypted file.
Step 18: Check that the decompression is lossless. If true then decompression is successful.
Step 19: Stop
```

III. ALGORITHM EVALUATION

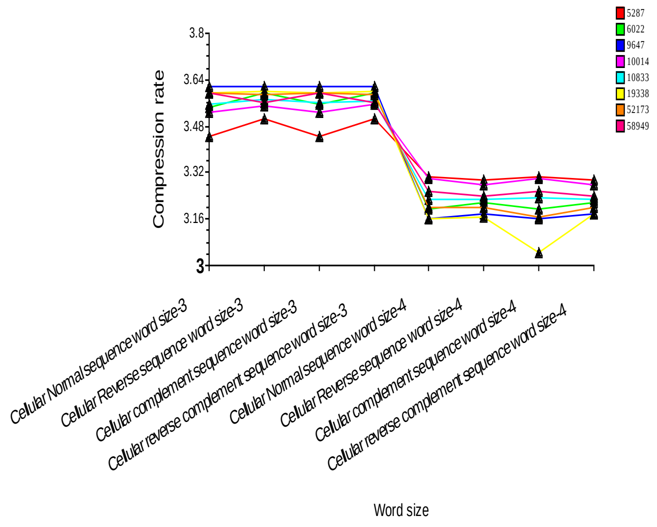
It is not permissible to get any type of error either in the compression or in the decompression phase. Hence, for accuracy purpose a string matching algorithm is developed, which checks character one by one. The proposed algorithm is efficient because DNA sequence compresses from substring length (l) into single ASCII character and the output file contains less characters than the input file. This algorithm requires very small memory space because it reads the input file and stores them immediately into the destination file. Hence, the required space is constant.

IV. RESULTS & DISCUSSION OF REPEAT & PALINDROME TECHNIQUE

Testing purpose used one data set and compression rate & ratio is mentioned in paper [19]

Table-1 Cellular DNA sequences Compression ratio and rate shown in the table using RP technique. From top to bottom, each column displays the result for a single algorithm showing the compression ratio and rate in bits per bases for each sequence.

			Cellular DNA Sequences							
			Normal Sequences		Reverse Sequences		Complement Sequences		Reverse Complement Sequences	
Sequence Size	**Sequence Name**	**Base pair/ File size**	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)
Sub string Size 3	atatsgs	9647	-0.80906	3.61812	-0.80740	3.61480	-0.80740	3.61480	-0.80740	3.61480
	atef1a23	6022	-0.7735	3.54699	-0.79608	3.59216	-0.77881	3.55762	-0.79608	3.59216
	atrdnaf	10014	-0.76473	3.52946	-0.77591	3.55182	-0.76472	3.52945	-0.77751	3.55502
	atrdnai	5287	-0.7212	3.44241	-0.75146	3.50293	-0.72120	3.44240	-0.75297	3.50595
	celk07e12	58949	-0.79701	3.59402	-0.78113	3.56226	-0.79701	3.59402	-0.78113	3.56226
	hsg6pdge	52173	-0.79825	3.5965	-0.79334	3.58668	-0.79824	3.59649	-0.79334	3.58668
	mmzp3g	10833	-0.7779	3.5558	-0.78676	3.57352	-0.78011	3.56023	-0.78380	3.56761
	xlxfg512	19338	-0.79708	3.59417	-0.80080	3.60161	-0.79708	3.59416	-0.80080	3.60161
	Average rate			3.55968		3.57323		3.56115		3.57327
Sub string Size 4	atatsgs	9647	-0.58184	3.1637	-0.5893	3.1786	-0.58184	3.1637	-0.5893	3.1786
	atef1a23	6022	-0.59681	3.1936	-0.60877	3.2175	-0.59681	3.1936	-0.60877	3.2175
	atrdnaf	10014	-0.64889	3.2978	-0.6393	3.2786	-0.64889	3.2978	-0.6393	3.2786
	atrdnai	5287	-0.65311	3.3062	-0.6463	3.2926	-0.65311	3.3062	-0.6463	3.2926
	celk07e12	58949	-0.62866	3.2573	-0.62052	3.241	-0.62866	3.2573	-0.62052	3.241
	hsg6pdge	52173	-0.59914	3.1983	-0.60136	3.2027	-0.5845	3.169	-0.60136	3.2027
	mmzp3g	10833	-0.61396	3.2279	-0.61285	3.2257	-0.61543	3.2309	-0.61285	3.2257
	xlxfg51	19338	-0.58184	3.1637	-0.583	3.166	-0.52115	3.0423	-0.5892	3.1784
	Average rate			3.22606		3.22534		3.2076		3.22689



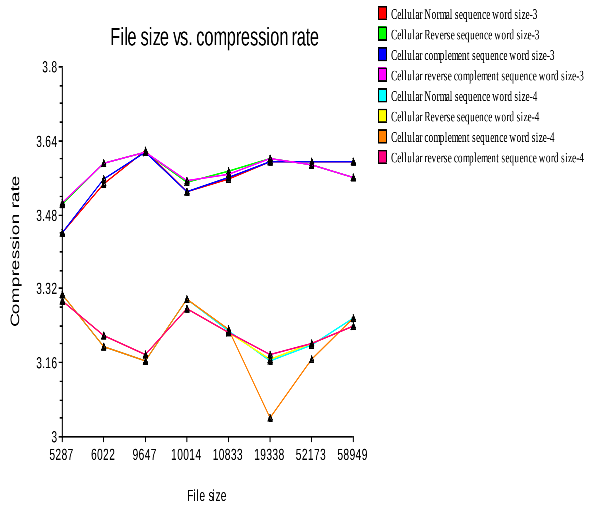


Fig.1 Compression rate vs. word size on basis of file size

Fig. 2 File size versus compression rate based on word

Table-2 Artificial sequences compression ratio and rate shown in the table using RP technique. From top to bottom, each row displays the result for a single algorithm showing the compression ratio and rate in bits per bases for each sequence

Sequence Size	Sequence Name	Base pair/ File size	Artificial sequences							
			Normal Sequences		Reverse Sequences		Complement Sequences		Reverse Complement Sequences	
			Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)
Sub string Size 3	XX1	9647	-0.80574	3.61148	-0.80076	3.60153	-0.80574	3.61148	-0.80076	3.60153
	XX2	6022	-0.79475	3.58950	-0.81600	3.63201	-0.79475	3.58950	-0.81600	3.63201
	XX3	10014	-0.80307	3.60615	-0.79988	3.59976	-0.80547	3.61094	-0.79988	3.59976
	XX4	5287	-0.80593	3.61187	-0.80291	3.60582	-0.80442	3.60885	-0.80291	3.60582
	XX5	58949	-0.81411	3.62822	-0.81682	3.63364	-0.81411	3.62822	-0.81682	3.63364
	XX6	52173	-0.80284	3.60569	-0.80453	3.60907	-0.80284	3.60569	-0.80468	3.60937
	XX7	10833	-0.81704	3.63408	-0.81334	3.62669	-0.81704	3.63408	-0.81187	3.62374
	XX8	19338	-0.79915	3.59830	-0.81114	3.62229	-0.79915	3.59830	-0.81114	3.62229
	Average rate			3.610661		3.616351		3.610883		3.61602
Sub string Size 4	XX1	9647	-0.8057	3.61149	-0.5893	3.1786	-0.6129	3.22587	-0.6179	3.23582
	XX2	6022	-0.7948	3.58951	-0.6546	3.3092	-0.6367	3.27333	-0.6546	3.3092
	XX3	10014	-0.8167	3.63331	-0.6269	3.25384	-0.6369	3.27382	-0.6269	3.25384
	XX4	5287	-0.815	3.63004	-0.6304	3.26083	-0.6304	3.26083	-0.6304	3.26083
	XX5	58949	-0.8141	3.62822	-0.6126	3.2253	-0.6181	3.23615	-0.6126	3.2253
	XX6	52173	-0.8108	3.62164	-0.6186	3.23723	-0.6063	3.21254	-0.6036	3.20718
	XX7	10833	-0.8244	3.64885	-0.604	3.20798	-0.5959	3.19173	-0.604	3.20798
	XX8	19338	-0.8057	3.61149	-0.624	3.24791	-0.6122	3.22433	-0.624	3.24791
	Average rate			3.621819		3.240111		3.237325		3.243508

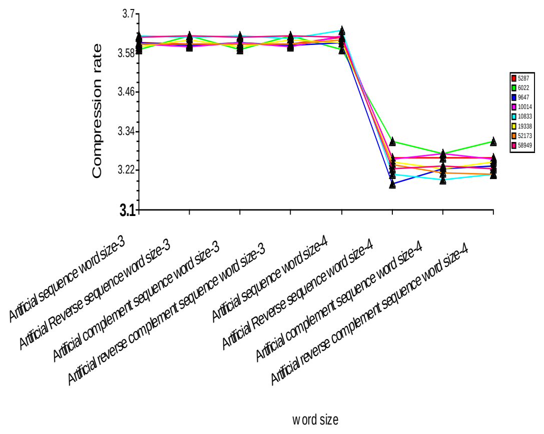


Fig.3 compression rate versus word size based on file size

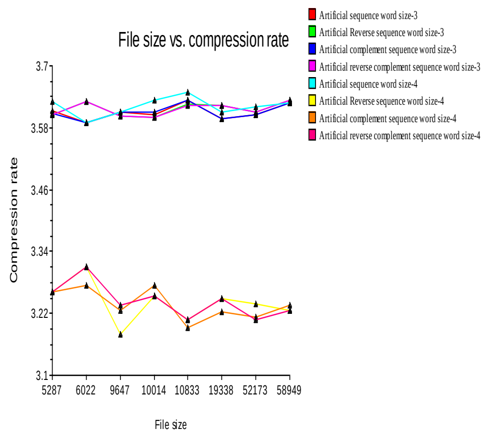


Fig.4 compression rate vs. file size based on word size

The result is presented in Table-1 for repeat and palindrome of cellular sequences & Table-2 for artificial sequences. This value is graphically presented in fig.-1 & 2 for cellular sequence and fig.-3, 4 & 5 for artificial sequences. The figure shown the compression rate & ratio varies with word size and independent of file size. The fig.-1 and 3 shows the more or less compression rate & ratio same of the different orientation. If word size is 4, the minimum compression rate is 3.2076 bits/base for cellular sequence & 3.2373 bits/base for artificial sequence where size of word is 4 and complement sequence orientation. If the word size is increased, the compression rate is decreased. The fig. 5 shows the artificial compression rate and cellular sequence compression rate are completely different, also graphical nature is different. It is also observed that the compression rate is similar in a particular word size because the sequence comes under different species and matching pattern are same but in case of artificial data the compression rate is dissimilar because this data is random.

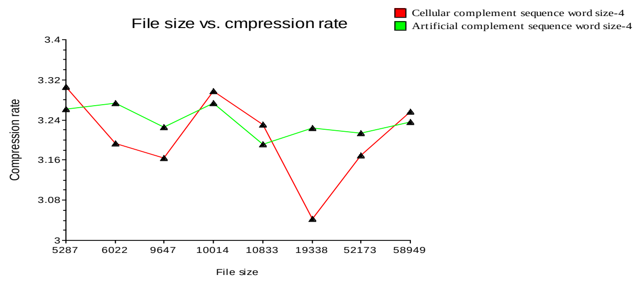


Fig.5 compression rate versus file size of cellular & artificial sequence

This algorithm is very much effective on normal cellular sequences than reverse, complement and reverses complement. The compression ratio and rate are noticeably distinct because the sequences come from different organisms. But in case of artificial DNA sequences the compression ratio and rate are equal in all cases. Important observations are:

1. Higher compression rate & ratio is obtained when the substring length varies from 2 to 5 whereas the substring length of more than 5 is inapplicable with respect to compression rate and ratio.
2. The compression rate & ratio is high when substring length is 3 as compared to the substring length 2, 4 and 5.
3. This DNA compression algorithm overcomes the limitations of binary coding and Huffman's coding.

V. CONCLUSION

This lossless compression technique is a good model for compressing and retrieving the DNA sequences in their original characteristics and is also very useful in database
storing. We presented an algorithm which detects repeats-palindromes within the DNA sequence, with restriction either on the length or on the spacing of the occurrences. Our lossless DNA sequence algorithm uses less number of bit/base for storing and execution time is saved. Out algorithm is fruitful for short pattern and inapplicable for long pattern. The result analysis of the application of our algorithm shows slightly high compression ratio to other exiting Biological Sequence Compression. All compression rate are similar, it also suggests a highly similar sequences. The auto created library file act as a signature for the time of transmission, it provides the information security.

VI. FUTURE WORK

In future we will find out the effect of compression on actual sequence by using this technique with the help of Levenshtein distance (LD).

VII. REFERENCES

[1]. International nucleotide sequence database collaboration, (2013),Online]. Available: http://www.insdc.org.

[2]. A.Jahaan, Dr. T. N. Ravi, Dr. S. Panneer Arokiaraj" Bit DNA Squeezer (BDNAS) : A Unique Technique for Dna Compression" International Journal of Scientific Research in Computer Science, Engineering and Information Technology,pp-512-517,2017

[3]. Kirill Kryukov, Mahoko Takahashi Ueda, So Nakagawa and Tadashi Imanishi, 'Nucleotide Archival Format (NAF) enables efficient lossless reference-free compression of DNA sequences' ,Bioinformatics, pp 1-3,2019

[4]. Karsch-Mizrachi, I., Nakamura, Y., and Cochrane, G., 2012, the International Nucleotide

Sequence Database Collaboration, Nucleic Acids Research, 40(1), 33–37.

[5]. Deorowicz, S., and Grabowski, S., 2011, Robust relative compression of genomes with random access, Bioinformatics, 27(21), 2979–2986.

[6]. Brooksbank, C., Cameron, G., and Thornton, J., 2010, The European Bioinformatics Institute's data resources, Nucleic Acids Research, vol. 38, 17-25.

[7]. Shumway, M., Cochrane, G., and Sugawara, H., 2010, Archiving next generation sequencing data, Nucleic Acids Research, vol. 38, 870-871.

[8]. Kapushesky, M., Emam, I., Holloway, E., et al, 2010, Gene expression atlas at the European bioinformatics institute, Nucleic Acids Research, 38(1), 690-698.

[9]. Jahaan A, Ravi TN, Panneer Arokiaraj S (2017) Bit DNA Squeezer (BDNAS): a unique technique for DNA compression. Int J Sci Res Comput Sci Eng Inf Technol 2:512–517

[10].Nour S. Bakr1, Amr A. Sahrawi, 'DNA Lossless Compression Algorithms: Review ', American Journal of Bioinformatics Research, 2013 pp 72-81

[11].Nahida Habib, Kawsar Ahmed, Iffat Jabin and Mohammad Motiur Rahman, Modified HuffBit Compress Algorithm – An Application of R, Journal of Integrative Bioinformatics, pp 1-13. 2018

[12].Mr Deepak Harbola1 et al. State of the art: DNA Compression Algorithms, International Journal of Advanced Research in Computer Science and Software Engineering, 2013, pp 397-400.

[13].K. Kryukov, M. T. Ueda, S. Nakagawa, and T. Imanishi, ``Nucleotide archival format (NAF) enables efficient lossless reference-free compression of DNA sequences,'' Bioinformatics, vol. 35, no. 19, pp. 3826-3828,Oct. 2019.

[14].Matsumoto, T., Sadakane, K., and Imai, H., 2000, Biological Sequence Compression Algorithms, Genome Informatics 11: 43–52 (2000).

[15].Giancarlo, R., Scaturro, D., and Utro, F., 2009, Textual data compression in computational biology: a synopsis, Bioinformatics, 25(13), 1575–1586.

[16].Ozkan U. Nalbantoglu, David J. Russell and Khalid Sayood,Data Compression Concepts and Algorithms and their Applications to Bioinformatics, Entropy 2010, 12, 34-52; doi:10.3390/e12010034.

[17].Deloula Mansouri, Xiaohui Yuan and Abdeldjalil Saidani, A New Lossless DNA Compression Algorithm Based on A Single-Block Encoding Scheme, Algorithms, pp 1-18,2020

[18].Tapasi Bhattacharjee and Santi P. Maity, An image-in-image communication scheme using secret sharing and M-ary spread spectrum watermarking, Microsystem Technologies, 2017, pp 4263–276

[19].Syed Mahamud Hossein et al., Comparison Of Compression Algorithm For DNA Sequences With Information Security Using Exact Matching Of Repeat, Reverse, Complement & Palindrome Technique On DNA Sequences and Apply On Others Orientation Also, International Journal of Information Technology & Management Information System,2013, pp 25-46

Cite this article as :

Syed Mahamud Hossein, "Substitution based DNA Sequences Compression-Encryption Method", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 8 Issue 4, pp. 63-70, July-August 2022.
Journal URL : https://ijsrcseit.com/CSEIT22848