Journal of Computational Intelligence in Bioinformatics
ISSN 0973-385X Volume 8, Number 1 (2015) pp. 1-6
© Research India Publications
http://www.ripublication.com

Analysing And Obtaining The Most Efficient Dna Computing Algorithm

Aman Garg[1]*, Misal Choudhary [1]
1. NSIT EMBS Chapter, Department Of Biotechnology, Netaji Subhas Institute of Technology, University of Delhi, Delhi, India
**amangarg81194@gmail.com*

Abstract

There is a growing demand of storage devices in the world. So, there is a need to develop alternate data storage devices that can confer the growing demands of people. Thus, the concept of DNA data storage is being evolved. In this mechanism, first the given information is converted into machine language, which is further converted into the DNA language of A, G, C and T. DNA is a highly compact molecule and according to a research by Harvard University, one gram of DNA can store up to 700 terabytes of information. DNA computing offers large storage capacity along with high accuracy of data retrieval, so there is no doubt in the fact that DNA computing can possibly, in the near future, serve as one of the best alternatives for electronic storage devices. In this paper, we have developed the programs that can be used to convert text data into the DNA language and vice versa. We have designed a number of programs using different logic and assumptions. In order to select the best program which is reliable and cost effective, we compared all the programs on the basis of compilation time and time complexity.

Introduction

DNA is basically a long molecule that codes for various activities of a cell. It also acts like a storage device which passes and transfers information to the new cell during the process of cell division. A DNA is a macromolecule composed of two strands of polynucleotides that are complementary to each other (Watson and Crick, 1953). Each polynucleotide is composed of four nucleotides (A, G, C and T) which are joined together by phosphodiester bonds. Each nucleotide consists of a nitrogenous base, a deoxyribose sugar and a phosphate group.

Adleman described how DNA can be used as a computational system (Adleman, 1994). This was the first instance where DNA was used as a tool to solve a computational problem. Since then, DNA is perceived as a potential device for storing information that can replace current electronic storage devices. After

Adleman's experiment, many scientists proposed and suggested various methods and techniques. Roweis proposed the method of stickers for DNA computing (Roweis et al 1999). Similarly, Schimdt worked on DNA computing using single-molecule hybridization detection (Schmidt et al, 2004).

Research groups like Allenberg and Rotstein came up with a coding method for archiving text, images and music characters in DNA (Allenberg and Rotstein, 2009) but their method could not code for all ASCII numbers. Recently, two groups of scientists managed to encode all the 256 symbols (Church et al, 2012) (Goldman et al, 2013). Church and his group assigned binary digit 0 to A/C and 1 to G/T. They used 8 nucleotides for every 1 byte of information. This method was not efficient and it required high memory utilization. Similarly, Goldman converted the ASCII numbers in to base 3 which meant 5 nucleotides per byte of information; the same nucleotides at adjacent place were avoided. But this resulted in different codes for same character/symbol.

In this paper, we have designed several programs to store and retrieve data from DNA. All possible programs were compared to find the most efficient method of storing and retrieving data.

The ASCII values of all characters in the sample data were first converted to binary/quaternary/ternary/octal form and then converted to A,T,G and C, the four nitrogenous bases. Similarly, we converted information stored in DNA back to the sample data.

The programs were designed keeping in mind various logics and assumptions. All programs were designed to work for all the ASCII values. We used various methodologies to convert text data into DNA language, for example, in one case we assumed 1 byte = 4 nucleotides, which means that each character mentioned in the ASCII table will be represented by four nucleotides of DNA. The ASCII value of character 'A' is 65, we converted that to binary i.e. 01000001 and we assigned nucleotide A for 00, G for 01, C for 10 and T for 11. So, A was converted to GAAG in the DNA language. We thus made different programs with different assumptions. The programs which we designed are:

a) 1 byte = 4 nucleotide (Binary). In this case, we converted the text data (ASCII value) into binary and then binary to DNA language. And the retrieval from the DNA language to text was done by performing the above steps in the reverse order. Assumptions: - 00→ A, 11→ T, 01→ G, 10 → C.

b) 1byte = 4 nucleotide (Quaternary). In this, the text (ASCII value) was converted into base 4 (i.e. quaternary) and then it was converted to the DNA language. Assumptions: - 0→ A, 1→ G, 2→ C, 3→ T.

c) 1byte = 5 nucleotide. In this, the text was converted to the base 3(i.e. ternary) and then to DNA language. Assumptions: - 0→ G, 1→ A/T, 2→ C.

d) 1byte = 6 nucleotide (Math's Method). In this, ASCII value of each character of the data was directly converted to DNA language by using a mathematic formula.

e) 1byte = 6 nucleotide (Triplet Method). In this ASCII value of each character of the data was directly converted to DNA language by taking further assumptions. Assumptions: - 0→ AA, 1→ TT, 2→ GG, 3→ CC, 4→ AT, 5→ AG, 6→ AC, 7→ TA, 8→ TG, 9→ TC.
f) 1byte = 8 nucleotide. In this the text (ASCII value) was converted into binary and then to the DNA language. Assumptions: - 0→ A/T, 1→ G/C.

We prepared a table(Fig 1 and Fig 2) in which all the programs were compared on the basis of time complexity and compilation time.The **time complexity** of an algorithm is the amount of time taken by an algorithm to run as a function of the length of the string representing the input. The time complexity, commonly expressed by big O notation, excludes coefficients and lower order terms, Whereas compilation time is the period of *time* during which translation of program's source code to executable code occurs. The observations were recorded in a table and the graphs(Fig 3 and Fig 4) were plotted.

Results

Data To DNA

Best program in terms of

Time Complexity	:- 6 nt (triplet method)
Compile Time	:- 6 nt (maths Method)
No of nucleotide per byte	:- 4 nt (Binary) , 4 nt (Quaternary)

DNA To Data

Best program in terms of

Time Complexity	:- 6 nt (maths Method)
Compile Time	:- 4 nt (Binary)
No of nucleotide per byte	:- 4 nt (Binary) , 4 nt (Quaternary)

Discussion

According to our observation, the program with the assumption of 1byte = 6 nucleotide is the most preferred program for the text to DNA conversion and vice versa. Based on the compile time and time complexity, this is the best suited program and it can be the most cost effective program for conversion of machine language to DNA and vice versa in future. The less time complexity of this program suggests that it is time efficient and the data conversion will occur faster than the other programs

Supplementary Materials

S. No	Description	Time Complexity f(n)	Time Complexity O(n)	Compile Time (comparative)	Memory
1	4 nt(Binary)	40n + 1	O(n)	2.20 s	1 byte = 4 nt
2	4 nt (Quaternary)	28n + 1	O(n)	2.33 s	1 byte = 4 nt
3	5 nt	37n + 1	O(n)	2.20 s	1 byte = 5 nt
4	6 nt (maths method)	33n + 3	O(n)	1.33 s	1 byte = 6 nt
5	6 nt (triplet method)	14n + 1	O(n)	2.23 s	1 byte = 6 nt
6	8 nt	60n + 1	O(n)	2.63 s	1 byte = 8 nt

Figure 1: Program details (DNA to data)

S. No	Description	Time Complexity f(n)	Time Complexity O(n)	Compile Time (comparative)	Memory
1	4 nt(Binary)	35n + 2	O(n)	2.11 s	1 byte = 4 nt
2	4 nt (Quaternary)	30n + 2	O(n)	2.70 s	1 byte = 4 nt
3	5 nt	37n + 2	O(n)	2.13 s	1 byte = 5 nt
4	6 nt (maths Method)	12n + 1	O(n)	2.11 s	1 byte = 6 nt
5	6 nt (triplet method)	45n + 2	O(n)	2.23 s	1 byte = 6 nt
6	8 nt	59n + 1	O(n)	2.34 s	1 byte = 8 nt

Figure 2: Program details (data to DNA)

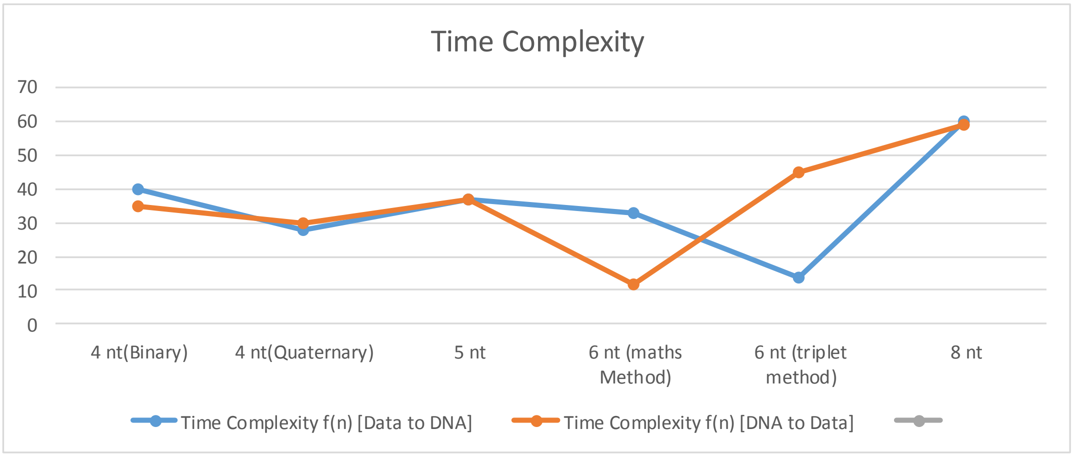

Figure 3: Time Complexity Comparison

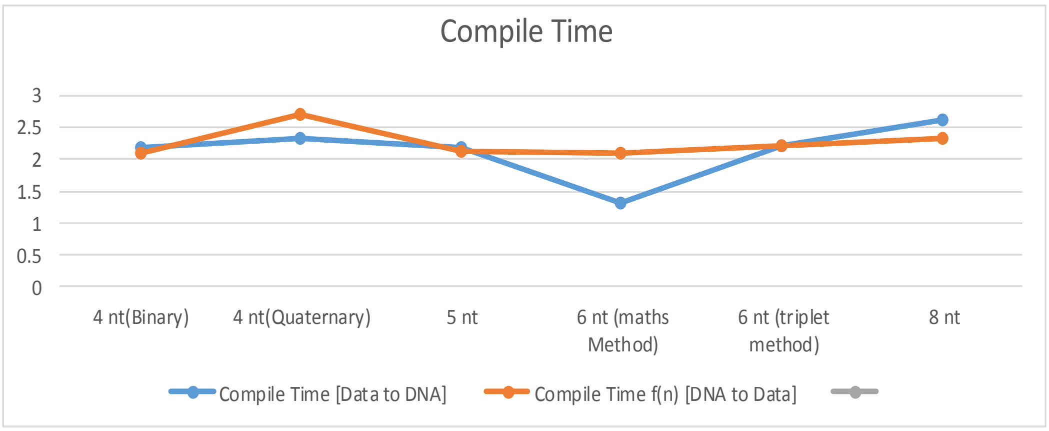

Figure 4: Time Complexity Comparison

Acknowledgments

We would like to express our gratitude to **Dr. Shilpa Sharma, Assistant Professor, Netaji Subhas Institute of Technology, University of Delhi** for guiding us at various stages throughout out research. We would also like to thank our other professors, who helped us in our research. We shall always cherish the guidance and encouragement endowed by them.

References

[1] Goldman, Nick, et al. "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA." *Nature* (2013).

[2] Adleman, Leonard M. "Molecular computation of solutions to combinatorial problems." *Science-AAAS-Weekly Paper Edition* 266.5187 (1994): 1021-1023.

[3] Benenson, Y.; Paz-Elizur, T.; Adar, R.; Keinan, E.; Livneh, Z.; Shapiro, E. (2001). "Programmable and autonomous computing machine made of biomolecules". *Nature* **414**(6862)

[4] Shu, Jian-Jun; Wang, Q.-W.; Yong, K.-Y. (2011). "DNA-based computing of strategic assignment problems". *Physical Review Letters* **106** (18): 188702

[5] Church, G. M., Gao, Y. & Kosuri, S. Next-generation digital information storage in DNA. Science 337, 1628 (2012).

[6] Chen, Junghuei, et al. "DNA computing implementing genetic algorithms."*Evolution as Computation* (1999): 39-49.

[7] Chen, Junghuei, and John Reif, eds. *DNA Computing: 9th International Workshop on DNA Based Computers, DNA9, Madison, WI, USA, June 1-3, 2003, Revised Papers*. Vol. 9. Springer, 2004.

[8] Pisanti, Nadia. "A survey on DNA computing." *EATCS BULLETIN*. 1997.

[9] Watson, James D., and Francis HC Crick. "Molecular structure of nucleic acids." *Nature* 171.4356 (1953): 737-738.

[10] Hagiya, Masami, and Azuma Ohuchi, eds. *DNA Computing: 8th International Workshop on DNA Based Computers, DNA8, Sapporo, Japan, June 10-13, 2002, Revised Papers*. Vol. 8. Springer, 2003.

[11] Staden, R. "A strategy of DNA sequencing employing computer programs."*Nucleic acids research* 6.7 (1979): 2601-2610.

[12] Adleman, Leonard M., et al. "On applying molecular computation to the data encryption standard." *Journal of Computational Biology* 6.1 (1999): 53-63.

[13] Burgoyne, Leigh Alexander. "Solid medium and method for DNA storage." U.S. Patent No. 5,807,527. 15 Sep. 1998.

[14] Staden, R. "A mew computer method for the storage and manipulation of DNA gel reading data." *Nucleic Acids Research* 8.16 (1980): 3673-3694.

[15] Fritz, Markus Hsi-Yang, et al. "Efficient storage of high throughput DNA sequencing data using reference-based compression." *Genome research* 21.5 (2011): 734-740.

[16] Mir, Kalim U. "A restricted genetic alphabet for DNA computing." *DNA Based Computers II* (1999): 243-246.

[17] Roweis, Sam, et al. "A sticker-based model for DNA computation." *Journal of Computational Biology* 5.4 (1998): 615-629.

[18] Deaton, R., et al. "Reliability and efficiency of a DNA-based computation."*Physical Review Letters* 80.2 (1998): 417.

[19] Gehani, Ashish, Thomas LaBean, and John Reif. "DNA-based cryptography."*Aspects of Molecular Computing*. Springer Berlin Heidelberg, 2004. 167-188.

[20] Schmidt, Kristiane A., et al. "DNA computing using single-molecule hybridization detection." Nucleic acids research 32.17 (2004): 4962-4968.