VOLUME - 13, ISSUE - 07, JULY - 2024 • PRINT ISSN No. 2277 - 8160 • DOI : 10.36106/gjra

Original Research Paper | Computer Science

DATA COMPRESSION USE IN GENOME SEQUENCE

Md. Syed Mahamud Hossein	Haldia Institute Of Technology
Neha Vinayak	Haldia Institute Of Technology
Moumita Bhunia	Haldia Institute Of Technology
Payel Chatterjee	Haldia Institute Of Technology
Anwesha Ghorai	Haldia Institute Of Technology

ABSTRACT More and more DNA and RNA sequences are becoming available day by day. Storing and transmitting them may require a huge amount of space. Compression of these genome sequences will help us increase the efficiency of their use. We use the Huffman coding algorithm, implemented in C, to compress the given genome sequence. To design the Graphical User Interface we use Java-Swing.

KEYWORDS : DNA, Compression

INTRODUCTION

Biological sequence compression is a useful tool to recover information from biological sequences. Better compression implies better understanding. Today, the complete DNA sequences of many organisms are already known, and the completion of human genome project is making steady progress. The information of DNA sequences, RNA sequences and amino-acid sequences of proteins are stored in molecular biology databases. It is well known that the sizes of these databases are increasing nowadays very fast. Therefore it is needed to store and communicate data efficiently.

Since DNA sequences contain four symbols 'a,' 't,' 'g,' and 'c,' if these were totally random, the most efficient way to represent them would be using two bits for each symbol. However, only a small fraction of DNA sequences result in a viable organism, therefore the sequences which appear in a living organism are expected to be nonrandom and have some constraints. In other words, they should be compressible.

METHOD

We use Huffman coding to encrypt the nucleotide genome sequence , a greedy algorithm that constructs an optimal prefix code called a Huffman code. The algorithm builds the tree T corresponding to the optimal code in a bottom-up manner. It begins with a set of |c| leaves and perform |c|-1 "merging" operations to create the final tree.

Data Structure used: Priority queue = Q
Huffman I
n = |c|
Q = c
for i = 1 **to** n-1
do z = Allocate-Node ()
x = left[z] = EXTRACT_MIN(Q)
y = right[z] = EXTRACT_MIN(Q)
f[z] = f[x] + f[y]
INSERT (Q, z)
return EXTRACT_MIN(Q)

The total running time of Huffman on the set of n characters is O(n lg n).

MATERIALS

Sweet Potato Virus

1 tcaggcactg aaagaacaaa agacgctgga accccaacac cagcaaaatc agttaagaca
61 agaacaggac aaactcaacc gcttaaagca ccagaaggga gcacggatcc aacagatcca
121 ccacctccaa cagttgaaga gataattgaa gaagaaacac cagcacaaaa agcattgagg
181 gaagcccgtg gcaagcaacc agcaacacaa ccctcataca cttatgggcg agacacagga
241 ccacgtagcc caaagcaagt cacaacaaca agtggagtta gggatagaga tgttaatgct
301 ggaacagtag ggacgtttac agttccaaga cttcaaatca catcaagcaa gaagagatta
361 ccaatagttg acggacgtcc agtaatcaac ctggatcact tggcagttta cgatccagag
421 caaacaaatc ttgcaaatac cagatcaaca caagaacagt ttaaggcatg gtatgaaggt
481 gtgaagggtg attatggggt atctgatgct gaaatgggca tactccttaa tggcctcatg
541 gtttggtgta ttgagaatgg tacatcaccg aatattaatg gaatgtgggt gatgatggac
601 ggagaagaac aagtaactta tccaataaaa cctctattgg atcatgctgt ccccacattt
661 agacagataa tgacacactt cagcgacata gctgaagcgt acattgaaaa gagaaacagg
721 ataaaggcct atatgccaag gtatggccta caggggaatt tgactgatat gagtcttgcg
781 cggtatgcat ttgatttcta tgaactccac tcaaacacac cagtaagagc aagggaggca
841 catatgcaaa tgaaagcagc agctttaaag aatgcacaga accgcctgtt tggtttggat
901 ggaaacgtct ccacgcaaga agaagacacg gagaggcata caacaactga tgttacaaga
961 aatatacata acctgttagg aatgagaggt gtgcagtaaa caatatattg ctcgtacctt
1021 taatttcagt tggtctttaa tttaaattcg tgtctttcag tcccgaagag tgttggttgt
1081 gtgtagtaac tatgtgtggt tgtaccaccg ttgctacata taagaaaacc tctttctatt
1141 acgtatcata agggactctt aaaagtgagt ctttgactcg taagaaaagc ctttttggtt
1201 cgtgatcgag aa

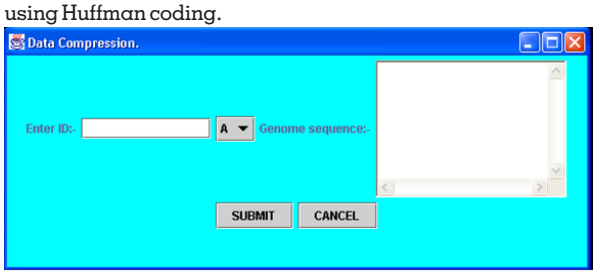
Potato spindle tuber viroid

1 cggaactaaa ctcgtggttc ctgtggttca cacctgacct cctgagcaga aaagaaaaaa
61 gaatggcggc tcggaggagc gcttcaggga tccccgggga aacctggagc gaactggcaa
121 aaaaggacgg tggggagtgc ccagcggccg acaggagtaa ttcccgcaga aacagggttt
181 tcacccttcc ttccttcggg tgtccttcct cgcgcccgca ggtccacccc tcgccccctt
241 tgcgctgtcg cttcggctac tacccggtgg aaacaactga agctcccgag aaccgctttt

301 tctctatctt ccttgcttcg gggcgagggt gtttagccct tggaaccgca gttggttcct

RESULT

The GUI interface is designed using Java Swing. It takes the ID of the genome sequence as input, displays the corresponding genome sequence and on submitting, it encodes the sequence using Huffman coding.



DISCUSSION

Data compression aims at encrypting a nucleotide genome sequence at the source so that we can protect the nucleotide genome sequence from unauthorized use. If the genome sequence is transmitted to the destination and an unauthorized user is able to access the sequence and make changes in it introducing some garbage information, the structure of the genome will totally change and this will lead to loss of potentiality if the genome sequence.

The second most important point we have to keep in mind is Data Compression. The sequence is compressed by means of encryption, but we have to keep in mind that it does not lead to data loss.

Future Work

We have successfully been able to encrypt the genome sequences using Huffman coding. In future we plan to use the information stored in the genome sequence, to omit sending the unwanted data in the sequence to achieve data compression without data loss.

REFERENCES

[1]. Bennett,C., Li,M. and Ma,B. (2000) Linking chain letters. *ScientificAmerican*, to appear Benedetto,D., Caglioti,E. and Loreto,V. (2002) Language trees and zipping. *Phys. Rev. Lett.*, 88, 048702.

[2] Chen, X., Kwong, S., and Li. M., A compression algorithm for DNA sequences and its applicationsin genome comparison, *Genome Informatics*, 10:52–61, December 1999.

[3] Chen,X., Kwong,S. and Li,M. (2001) A compression algorithm for DNA sequences. *IEEE Engineering in Medicine and Biology Magazine*, 20, 61–66.

[4] Cleary, J.G. and Witten, I.H., Data compression using adaptive coding and partial string matching,*IEEE Trans. on Commun.*, COM-32(4):396–402, April 1984.

[7] Nevill-Manning, C.G. and Witten, I.H., Protein is incompressible, *Proc. of IEEE Data Compression Conference*, 257–266, 1999.