Bases to Bits: An Analysis of *Biocompress 1*

Gavin Saxer, Jared Arroyo Ruiz, Ryan Son, Advised by Layla Oesper
Department of Computer Science, Carleton College

Compressing DNA

DNA sequences require vast amounts of storage. As of 2021, the Sequence Read Archive had 16.5 petabytes of DNA [1], and that number continues to grow.

Storing this enormous volume efficiently requires specialized compression algorithms designed for DNA.

This poster focuses on the implementation and analysis of the first DNA compression algorithm, *Biocompress 1*, created in 1993 [2].

Biocompress 1

Biocompress 1 exploits the limited letters that form DNA, called bases (A, C, T, G) and the repetition of sequences of these bases compress DNA more efficiently than general text compressors.

Each base can be encoded as two bits, rather than the eight bits used in ASCII. This creates a compression ratio of 0.25.

A ⟶ 11 G ⟶ 01

C ⟶ 10 T ⟶ 00

The repetition of strings of bases in DNA can be used to compress the genome beyond the simple base encoding.

There are two types of repetition: factors and palindromes. Factors represent exact repeats, while palindromes encode complement repeats, both common in DNA.

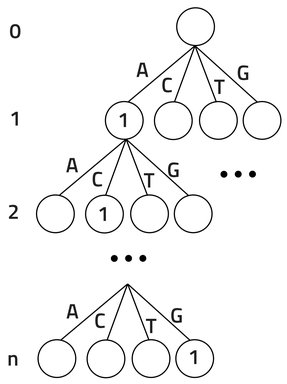
Complements:
A ⟷ T C ⟷ G

Each repetition can be encoded as a position to copy from, the type of repetition, and a length to copy.

Factor encoding:	Palindrome encoding:
ACTG… ACTG	ACTG… TGCA
↓	↓
ACTG… + (0, factor, 4)	ACTG… + (0, palindrome, 4)

Repeated sequences are found using a 4-ary tree, where each node has four branches representing the bases. Each node stores the positions of previously seen sequences.

To test whether a new sequence has appeared before, the algorithm follows the corresponding path in the tree. If the final node contains a stored position, that sequence is a repeat.



Example structure of 4-ary tree with height n. Positions are recorded for sequence AC…T at position 1.

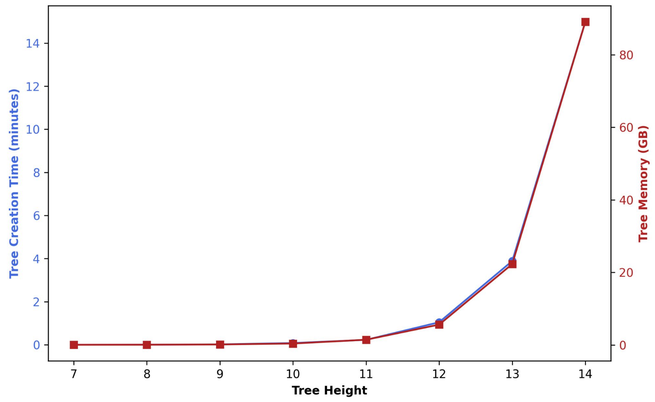
Tree Creation

The number of nodes in the tree is determined by the following equation, where n is the height of the tree:

$$\text{nodes} = \sum_{i=0}^{n} 4^i$$

This increasing number of nodes in the tree for increasing heights is reflected in the memory consumption of the tree and the time it takes to create the tree.

The changes in memory and creation time for each height are proportional, as reflected in the graph.

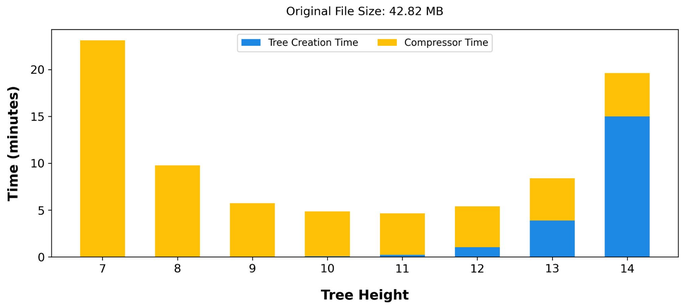


Runtime Comparison

As a genome is compressed, the tree will progressively get filled with position values. As each base is read in, if it finds a match in the tree at a leaf node, the algorithm will do an inefficient extended search.

So large trees means less matches at leaf nodes, therefore doing less extended searches, which improves the runtime, but the large tree size means the memory requirement is large. Small trees trees have a small memory requirement, but the number of extended searches makes the runtimes considerably longer.

The tree gets so large with increasing heights that the creation time is $O(4^n)$ where n is the height of the tree. This creates a tradeoff: larger trees speed up compression but take longer to build, and the optimal height depends on genome size.



For smaller genome sequences, such as a single chromosome, a height of 11 proved the most efficient.

For smaller trees, the algorithm had to do significantly more extended searches, and so had a large compression time.

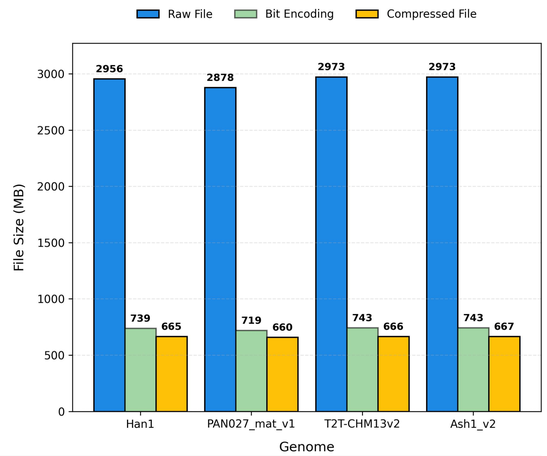
For larger heights, the tree was large enough to significantly increase the tree creation time.

For larger genome sequences, such as a full human genome, larger heights proved most efficient.

The increasing tree creation time was unable to offset the decreasing compression time for any height this project was able to test.

Larger tree sizes proved too big to run, but in analyzing compression time and tree creation time trends, a height of 15 is estimated to be the most efficient for full human genomes.

Compression



	Han1	PAN027_mat_v1	T2T-CHM13v2	Ash1_v2
Compression Ratio	0.225	0.229	0.224	0.224
Space Savings	0.775	0.771	0.776	0.776

Data for *Biocompress 1* on full genomes in above graph

Biocompress 1 was able to achieve a compression ratio of 0.229 or better for full human genomes, reducing the size of each file by at least 77%.

Notably, this is only slightly better than the 0.25 compression rate achieved by encoding each base as two bits as described in the first step in the *Biocompress 1* algorithm.

Future Work

In the future, this project would likely expand to implement and analyze a more modern compression algorithm. By doing this for multiple algorithms, the project could expand to become an analysis on how DNA compression algorithms have changed over time.

Acknowledgements

We'd like to thank the original authors for their foundational work, Layla Oesper for her help throughout the process, and Mike Tie for his technical assistance.

Works Cited

[1] Council of Councils Working Group on Sequence Read Archive Data, "Final Report," National Institutes of Health, Bethesda, MD, USA, Sept. 2021.
[2] S. Grumbach and F. Tahi, "Compression of DNA sequences," in Proc. Data Compression Conf., 1993, pp. 340–350, doi: 10.1109/dcc.1993.253115.