Text Compression

Efficiently encode string S into a small binary string B

- The problem we want to solve is, given a string S defined over some alphabet Σ, (example, ASCII or Unicode character sets), and we want to efficiently encode S into a small binary string B (using only the characters 0 and 1).

- Reason is to increase throughput

- The Huffman-coding algorithm begins with each of the d distinct characters ($|\Sigma|$) of the string D to encode being the root node of a single-node binary tree

- Explained: A fixed size encoding approach assigns the minimal number of bits to encode the entire Σ
    - This requires $\lceil log_2(|\Sigma|) \rceil$ bits.
    - A decompression map is usually sent with the message which requires a common binary representation of the characters of Σ (8 bits for ASCII) + their encoded version.
    - Total bits sent are: $(|S| \times \lceil log_2(|\Sigma|) \rceil) + (|\Sigma| \times |c|)$, where c is the common encoding for each character c in Σ

- Example: Let $\Sigma = \{A, C, G, T\}$ and S = "CGTAAATGCTTTAATCAAACGC"

- We begin by finding the frequency of each character in S and order it in ascending order by frequency

G = 3
C = 5
T = 6
A = 8

- Build a binary tree by combining the frequency minimums. all two minimums
  22
  / \
  14 \
  / \ \
  8 \ \ \
  / \ \ \
  G = 3 C = 5 T = 6 A = 8