Huffman Warmup (skip if you're feeling comfortable with Huffman coding). How many bits are in the Huffman encoding of the following message?

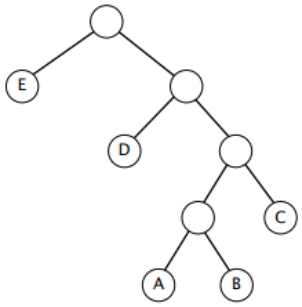
a b a a b a c a b a a b a c d a b a a b a c a b a a b a c d e

The frequencies for this message are given for the table below:

a	b	c	d	e
16	8	4	2	1

Huffman Best/Worst Case. Assuming the input is a sequence of 8-bit characters, give a best case input for Huffman coding. What is the compression ratio? What is the compression ratio for a worst case input?

Tougher Huffman Analysis. Consider the following Huffman trie of a message over the 5-character alphabet {A, B, C, D, E}.



For each statement below, determine which of the three conditions applies: true for all messages, false for all messages, or its veracity depends on the message.

1. The frequency of A is strictly less (i.e. not equal) than the frequency of B.

2. The frequency of C is greater than or equal to the frequency of A.

3. The frequency of D is strictly greater than the frequency of A.

4. The frequency of D is greater than or equal to that of A, B, and C combined.

5. The frequency of E is strictly less than that of A, B, and C combined.

LZW Encoding Warmup (skip if you feel comfortable with LZW basics). Use LZW to compress the string ABAAABABA. Assume 8 bit codewords and that the first new codeword starts at 81. A corresponds to 41, and B corresponds to 42.

LZW Decoding Warmup (skip if you feel comfortable with LZW basics). Use LZW to decompress 42 41 42 83 82 82 80. Assume 8 bit codewords and that the first new codeword starts at 81.

LZW. What is the best-case compression ratio for N characters using 12-bit codewords? Recall that no new codewords are added to the table if it already has $2^{12} = 4096$ entries.

What is the compression ratio in the worst case?

Extra: Given an input bistream of length L and a policy where the most-recently-used-codeword is replaced, what is the best case order of growth of the compression ratio as a function of L?

Tries. The problem with tries is that they are very memory hungry. Suppose we wanted to address this by replacing the `next[]` array with an LLRB. What would be the worst-case order of growth for a search hit, assuming a query of length L, an alphabet of size R, and that there are N string keys in the Trie.

Prefix-Free Codes. In data compression, a set of binary strings is prefix free if not string is a prefix of another. For example {01,10,0010,1111} is prefix free, but {01,10,0010,10100} is not because 10 is a prefix of 10100. Design an efficient algorithm to determine if a set of binary strings is prefix-free. The running time of your algorithm should be proportional the number of bits in all of the binary stings.

Extra problem: Autocomplete Enhanced. On assignment 3, we designed an autocomplete data structure that used binary search to return the list of words starting with a given prefix. For an array of length N and a number of matches M, our algorithm took M + log N time to return all matching strings. Suppose we want to return only the top Q matches. Design a Q + log N data structure.

Extra problem #2: Kolmogorov Complexity. Given a string S, the Java-Kolmogorov complexity $K_J(S)$ is defined by the number of bytes in the shortest Java program that prints S to the screen. Let $|S|$ be the length of the string.

a. Prove that $K_J(S) \leq |S| + c$ for some constant c.

b. Give an example of an S such that $K_J(S)$ is much less than$|S|$.

c. Suppose we use LZW compression on S and get an output stream B. Explain how B and $K_J(S)$ are related.

d. Explain why $K_J(S)$ is greater than S for almost all strings.

e. Let the Python-Kolmogorov complexity $K_P(S)$ be defined as the number of bytes in the shortest Java program that prints S to the screen. Prove that for all S $K_J(S) \leq K_P(S) + c$ for some constant c. Why is this result interesting?