CS302 - Programming Assignment 2: Huffman Compression

Motivation

Compression is one of the most important tools in computer science and is also one of the most fundamental ideas in information science. The idea is to take some data, compress (encode) it into fewer bits for storage or transmission, and then decompress (decode) the data to its original form (or very close to its original form) at a later point. Huffman Coding is one of the best ways to encode because it is lossless (you can always recover the original data from the compressed data exactly), optimal (you can't compress the data beyond what Huffman codes can do without additonal information about the data), and easy to encode and decode. We will cover the analysis of Huffman Coding in class as part of our study of greedy algorithms.

Guidelines

Please read and abide by the honor code guidelines in the syllabus.

Please read the rubric so you know how you will be graded. For example, turning in a program that compiles and runs without errors but does nothing will earn you more points than a program that is close to working but does not compile or contains errors on running.

There are two options for this assignment: Standard and Challenge. (You must turn in only one of the two options.) You will be graded using this rubric. Since the rubric is out of 30 points, if you earn X points, then your grade will be $X/30 \times .85$ for a Standard assignment and $X/30 \times .95$ for Challenge. Thus a poorly done Challenge program could give you a worse score than a well executed Standard program.

Put a multi-line comment at the beginning of your program. It should contain:

- Your name
- "Programming Assignment 2"
- "Challenge" or "Standard"
- The name of anyone you worked with and the nature of your collaboration

Standard Assignment

Write a class called `Huffman` in java that has a method `encode` that takes as parameters a .txt file containing lower case English letters and spaces, and outputs two .txt files. One should contain the Huffman encoding of the file (where the letter frequencies should be based on the original file), and one should contain information on how the file was encoded. Your second file should look something like this:

```
a:  01
b:  1001
⋮
```

```
[space]:  111
```

Challenge Assignment

Complete the standard assignment. Next create a method `decode` that takes as input the output of `encode` and recovers the original text. Finally, create a method `createTree` that takes in the decoding file, and outputs a visualization of the binary tree corresponding to that decoder. You should feel free to use external packages such as JGraphX, etc for the visualization.