1 Assignment 3

Submit source code, design and any explanation to EAS[1]. Do it in Java. Do not use java's search or sort methods. Do not use any java pre-built data structures besides arrays! Do not use Java's Collections or Subclasses!

This assignment has a significant design component. You will use UML class diagrams and text to explain your decisions inside the diagrams, as well as any additional decisions. Do the design first! Read the **whole** assignment before you start.

Don't use packages! Using packages is generally good, but is a pain when I want to test all of you easily and you make other minor changes. **Don't deviate from the file names indicated!** Huffman.java and ATree.java That includes renaming the file, but leaving the class inside it the same. You may include other files, and those files may be in packages.

Posted: Monday, June 5^{th}

Due: Friday, June 16^{th}

Grade: 30%

1. Designing a flexible tree structure
   (a) Design a tree structure that you will implement and subclass to solve both the Huffman Coding Problem and either the AVL or Splay Tree problem.
   (b) Include a class diagram showing all the supporting classes of this tree structure.
   (c) Explain your decisions.
   (d) Include a class diagram showing how you Huffman Coding implementation will extend your general tree structure
   (e) Explain your decisions textually
   (f) Include a class diagram showing how either your AVL Tree or Splay Tree implementation will extend your general tree structure.
   (g) Explain your decisions textually
   (h) Explain why you chose either the AVL Tree or the Splay Tree to solve question 3, referencing your diagrams and decisions as is appropriate.
2. Implement a Huffman Coding Tree based off of the design in question 1.
   (a) Your tree should build itself using `https://users.encs.concordia.ca/~sthiel/comp352/Jabberwock.txt`
   (b) Include the characters for spacing as well as all punctuation, but not line breaks.

[1] `https://fis.encs.concordia.ca/eas/`

(c) You should not make encoding for characters not found in the source text, and I will not use any such characters while testing your code.

(d) When Two characters with the same weight are merged into one subtree, your program should choose for the left child, the element that came first in the source material.

(e) When two subtrees with the same weight are merged into one subtree, your program should choose for the left child, the element that came first in the priority queue or ordered list you were using to build the tree.

(f) Using the command line, your program should be called as
java Huffman <source>

(g) **<source>** will be the name of the local file to be loaded to built your Huffman Tree

(h) Upon starting, a frequency table will be built. Your program should listen to stdin (you may use Scanner) and encode any line entered (terminated by a line break) and output the encoded version using 1s and 0s (vs. padded ASCII).

(i) Your program should multiple lines to be entered, encoding each one as it is entered.

3. Implement either an AVL Tree or a Splay tree based off of the design in question 1.

(a) Using the command-line, your program should be called as
java ATree <source>

(b) **<source>** will be the name of the local file to be loaded that contains operations

(c) Your program should output the following statistics after all operations are processed:

- The total number of comparisons.
- The total number of times a node's parent changes (including new nodes getting their first parent, and removed nodes "losing their parent").
- The total number of find operations
- The total number of add operations
- the total number of remove operations.

(d) The operations in the source file will be one-per-line and will consist of the letters a,r or f, representing add, remove or find respectively, followed by 3 digits, the value for the node.

(e) The sample source file that your should run your code against is `https://users.encs.concordia.ca/~sthiel/comp352/Operations.txt`

(f) **note:** The tree starts out empty and is filled by the operations.txt (or similar file). Your program should deal properly with trying to find or delete nods that aren't in the tree.

4. Textual Responses, Keep in a separate file for easy reading by me! All responses should not exceed one page (10pt font, 1inch margin, for you weirdos)

   (a) Huffman

   - Look up the file: `https://users.encs.concordia.ca/~sthiel/comp352/RandomStrings.txt`
   - Encode the string associated with your student id and record the result here.
   - Describe what percentage fewer bits were used than the fixed-length ASCII encoding would have taken.
   - Indicate whether you feel that the encoded string matched the source text frequencies, and provide your reasoning.

   (b) Advanced Trees

   - Provide the output you recorded based on the Operations file provided in question 3
   - Explain whether you feel that your initial design decision was correct when choosing either AVL or Splay trees, given that you have now implemented and tested. Explain anything you have learned through implementing and testing that would affect any future designs.