CS256 Lab 8 - Popularity with trees

Value: 30 points, plus 10 points extra credit
Due: Friday 1 November @ 23:50 (design document due for class on Wednesday 30 October)
10% bonus for submitting it before Friday @ 18:00!
Submission Format: Design document (image) and a Python script, lab_08.py, optionally lab_08_ec.py.

This lab uses a tree data structure to do the same work as Lab 7 did with the same input data. This will give us more practice designing and writing code, and comparing how different ADTs can be used to solve the same problem. In doing so we will learn more about the strengths and weaknesses of each approach.

Setup

The CS department supports a number of web servers (Apache), one of them hosts a bunch of virtual domains such as http://cs.earlham.edu, http://fieldscience.cs.earlham.edu, and others. The department would like to know how many visits were made to all of the domains, listed in alphabetical one list in alphabetical order (ascending).

Tasks

Design, write and test:

- A class for a tree node to be used in a binary search tree with a constructor and fields for a domain name, a reference count, and pointers to its parent, left, and right children.
- A class for a binary search tree with a constructor, a field for the root pointer, an addOrIncrementDomainNameNode() method that either adds the domain to the BST at the appropriate place (by domain name) with a reference count of 1, or increments the reference count if the domain name is already there, and a printTree() method that displays the domain name and reference count in domain name order (ascending). Note that this is a property of the way you visit nodes (inorder, preorder, etc.)
- A method that given an input log file populates a BST tree with the domain names and references found in the log file, returning the tree to the caller (your main()). Your algorithm will need to read one line of the file at a time, parse-out the domain name (the first column, space separated), remove the port number from it, and then process it.
- Your main() function should read the log file name from the command line, check to see that it exists, call your populate method, and then print the returned populated tree using the printTree() method.
- Your solution should print the results to stdout in this format (these are the correct values):

```
cs.earlham.edu 17938
datascience.cs.earlham.edu 303
dpr.cs.earlham.edu 12
energy.cs.earlham.edu 2
fieldscience.cs.earlham.edu 619
greenscience.cs.earlham.edu 37
hip.cs.earlham.edu 3390
hop.cs.earlham.edu 15
i4inclusion.cs.earlham.edu 34
lists.cs.earlham.edu 36
mail.cs.earlham.edu 4
portfolios.cs.earlham.edu 7537
staging.cs.earlham.edu 127
wiki.cs.earlham.edu 741
```

Implementation Notes

- Design your solution on paper first, there is more of your design in this lab than in previous ones. Most people would start by reviewing the whole, and then designing the low-level elements (the class/method definitions) and then move to the higher level tasks. Update it as you develop and refine your thinking. Make sure your design covers all of the assignment, you will be turning-in an image of it on Wednesday and Friday.
- Your script should have one command line argument, -i for the input logfile.
- Your script should use the main() technique, it helps organize the code and it makes reuse much easier.
- Read the input file one line at a time rather than slurping it up in one go.
- We will provide both small and large input files to test with, you should use all of them, volume matters and you need to show that your solution works across the range of reasonable input sizes.

Extra Credit

- (10 points) Extend your solution so that it also displays the domain names in reference count order (descending) after the the alpha list. Make a copy of your base solution first so that you can turn-in both versions. The format of the output for this section should be:

```
reference-count domain name
reference-count domain name
…
```

- One way to accomplish this is to make a second tree class whose method for adding nodes uses the reference count as the key rather than the domain name. This can be built from a tree that has domain names as the key (which you will already have) by traversing it and one at a time adding the nodes you encounter to the new tree using the total reference count as the key.
- The printTree() method of your second tree class will need to traverse it in the appropriate way to generate an ordered, descending list.