Project 4 Design Rubric

CS 2413 – Fall 2018

Description

*"While this description suggests that using the AVL tree in place of the linked list for reading in files and checking for duplicates is more time-efficient, you should consider whether that is really the case. For this reason, you should include a design document with your submission. Please make this a PDF file and name it "**design.pdf**" in your submission. In this document, you should analyze the time efficiency of reading into an AVL rather than a linked list while checking for duplicates. In this document, you should also analyze the time and space efficiency of the processes described above for merging and purging data by reading the new file into its own tree, then using the two existing trees to generate a new tree."*

Important Note on Purge Algorithm

The purge requirement was later revised to have the purged items iteratively removed from the main tree via an in-order traversal of the purge tree. The above verbiage is correct for merge, but your analysis should be on the purge algorithm described in revision 2 of the project description (also described at the beginning of this paragraph).

Submission Expectations

Submissions receiving full credit will provide the algorithm's complexity in Big-O notation along with a brief (< 1 paragraph) explanation. Note that this does not need to describe every function of the program, nor does it need to cover any points that are not listed below.

Points Breakdown

Note: For clarity, the problem size "n" is explicitly stated in each of the following

Additional Note: Each of the following are worth 1 point (0.5 for correct Big-O complexity, 0.5 for explanation) for a total of 5 points on Project 4.

1. Time complexity of inserting n objects into an AVL tree while checking for duplicates, contrasted with time complexity of inserting n objects into a sorted linked list while checking for duplicates.
2. Time complexity of merge algorithm, where m is the size of the initial tree and n is the size of the tree containing the file data.
3. Space complexity of merge algorithm, where m is the size of the initial tree and n is the size of the tree containing the file data.
4. Determine time complexity of purge algorithm, where m is the size of the initial tree and n is the size of the tree containing the file data.
5. Determine space complexity of purge algorithm, where m is the size of the initial tree and n is the size of the tree containing the file data.