Lab 4: Binary Search Trees

Binary Search Trees (BSTs) are used to quickly sort and search through data. Applications of BSTs or their variations are numerous in the fields of Information Retrieval, Artificial Intelligence, Computer Vision, and 3D rendering. The "downside" is that the data being stored must have a "natural order" to them which allows the data to be compared and sorted.

Overview

In this lab, you will implement a searchable inventory system. The main program will read a file of items from the hard disk and insert them into one or more Binary Search Trees. If you encounter duplicate items, then combine their stock and take the average of their prices. If the stock gets below or equal to zero, then remove the node from the tree. Finally, you will create an output file which follows the format below.

Input File The input file, called *inventory.txt*, is a text file where each line contains a JSON with the following information:

- Item Name
- Item Cost
- Number of Items in Stock

Output File The output file must be named *storeData.txt* and contain the following information:

- Number of unique items in stock
- A list of items sorted by name (as JSONs, one per line)

Concepts

- Generics with Requirements
- Binary Search Trees
- File Input and Output

Hints and Tips

- BSTs sort on **one** attribute only. If you need another sorting, use another tree.
- To combine duplicates, adjust the stock and price, remove the node, and then re-add the node to the tree.
- Break the input file into smaller chunks so that you can test your program more easily.

Program Details

- The program may be console or graphic based, but it may **not** switch between the two.
- You may not use any other data structures besides a BST in your program, except as returned by your travsersal functions (return a list).
- There must be a clear separation of presentation (file I/O, console I/O, etc) and logic (BST, BinNode, etc). Specically, you may not output to or read from a file from your logic.
- The input file is *inventory.txt* and the output file must be *storeData.txt*.

Grading

- /4 Separation of logic and presentation
- /3 Handling Duplicates correctly
- /10 Implementation of a Binary Search Tree
  - /3 Add method
  - /4 Remove method
  - /3 Traversal
- /5 Implementation of File Input and Output

- /3 Documentation and style (class and method documentation, name in program, etc.)