Documentation

By anthonyCameron- 19035457

Tree Profiler-
The Tree profiler works by checking if the users command line parameters are more then 1 if it isn't it calls a function displaying the usage of the class. If the user enter –I it triggers interactive mode which also requires user to enter what tree they want to user interactive mode on using a case statement on parameter 1 which than calls the appropriate menu. If the user was to enter –p into the command line, they require to entering which tree they want to use and size of degrees and the file they want to process. Each tree returns the stats and overall speed.

UI-
Each Tree has their own unique menu that declares and initialises their specified tree. The class reuses the methods for inputting string or integer however has there own unique method for construction and object for their intended tree. The menu options are load data, insert, find, delete, stats, display, save and exit option that utilises the methods within each class. Both save and load option has additional options if they require reading or saving a file as a text, binary or serialised file.

Tree IO-
Contains the methods for loading and writing files, has a unique method for writing each class

Binary Search Tree-
Contains a private inner class which contains the detail for each node, the classfields inside the Binary Search Tree declares the node to be the root. The class as a default constructor and an alternate constructor that uses the file Input method. The file Input just passes each line to the processStock method that parses each line assigning the value to a variable associated with stock and constructs and object out of them. The method also sends the key and object into the insert method.

Insertion
Increment the count for displaying the amount of variables inside the tree and updates the root to bed the new node. This function has two methods, one is a wrapper and the other is a recursive method. This function has a base case that stops the recurse when the node is null and creates a new node and setting update to become the new node. The method checks if the key is less then or greater then the current node and recurses left if less than or right if greater. If its equal it throws an exception if the key is the same as current

Delete
Follows the same process above for insert however it checks for the key being equal to any node. If the key equals to current node than it calls the delete function. If the key is null it sets the node as null and updates the child on the left or right if there is only one child. When there are two it calls the promote successor method that adopts the right most node.

Find
Uses a wrapper method that calls a resursive method recFind. This function navigates through the entire tree checking if the key is equal to the node key returning that value if found, if not it recalls itself until node is null

Height
Another recursive function that calculates the height of the left hand side and the height of the right hand side. The function recuses until node is null and compares the heights of left and right children. The highest value determines the trees height

Traverse
Recurses through the tree and adds the values into the queue, this function has different orders, postOrder, inOrder and preOrder. Post order recurses adds to queue last inOrder adds inbetween and preOrder adds at start. The store function uses the same method as inOrder

B-Tree
Contains a private inner class that stores size, data[], key[], child[] and leaf which is different from Binary Search Tree as key, data and child are arrays. Shares the same file IO method as Binary Search Tree. The insertion Method checks if node exists if not it creates one else it checks if the node is full. If the node is full it splits the node and inserts into the new nodes empty leaf . if the node isn't full it normally inserts the value by calling insertNonFull

InsertNonFull
This function checks if the node has a leaf if it does it goes through the node inserting into the empty space. If the leaf doesn't exist it checks if the node is full and splits it into 2 locating the new location to insert the key. This method recurses until node is null

splitChild
This method checks if the node is not full and if the node is we split it into two nodes. First the function creates a new child node with the size of 1 and set the key of that to the middle key. It then copies over all the values into the new node and searches for the location of the new key. This then shuffles the array of the node into ascending order and assigns the reference of the new node to the child of the parent increasing the size of parent by 1.

Search
Has a wrapper method which calls the recursive search function. This function finds the first key of that is less than or equal to the current node and checks if any of the values are equal to the key. If the child can't be found it returns null else it keeps recursing until it does

Display
Prints out the values with in the node by going through each child and printing that data and moving to the next child

TwoFourTree-
A type of Btree set to the max degree of four and instead of splitting after insertion. The children are split when they detect a child is full.