Implementing a Binary Search Tree and a Binary Tree Search Algorithm

Homework #4

By Logan Miles

1. Objectives

The goal of this assignment was to write a program that would read a CSV file and implement an algorithm that would add the items from the CSV into a binary search tree with respect to each item's unique UPC key and the binary search tree properties. Another algorithm was required to traverse the binary search tree using in-order traversal, printing the attributes of the item at each node including its UPC key, amount, and name. The last algorithm that was to be implemented was supposed to use the binary search tree properties to locate an item in the tree with a specific UPC key. Each item's UPC key, along with the item's amount and name, could be found in another file titled "input.dat" which also needed to be read in order to compare the key of the target item to keys of those in the binary search tree.

2. Program Design

Because this assignment required the implementation of a binary search tree and Java is an object oriented programming language, the first thing implemented was a TreeNode class that represented the attributes of the items in the CSV file. A BinaryTree class was then created that contains the necessary functions to create the binary search tree, print each node using in-order traversal, and search the tree for a particular key. Lasty, the HW4 class

contains the main() function that is used to call the functions from the other classes and test the algorithms. The following functions are contained within these classes:

TreeNode()

TreeNode() is a basic constructor for the TreeNode class that establishes each object's key, amount, name, and parent. The amount and name are of the String type, while the key is of the type Long, and the parent (or p) of the node is of the type TreeNode as well. The amount is a String instead of an integer because the amounts of each item in the csv file are of different types and most of them contain alphabetical characters. These amounts also do not need to be operated on, only printed. Lastly, all keys within these classes are of the type Long because there are items in the csv file with a UPC key larger than the maximum value possible for a signed 32-bit integer. The type of unsigned integer also would have worked as a solution, as it also can represent values above 2,147,483,647.

insert()

Constructs the binary search tree by inserting each TreeNode object created based on the attributes read from the UPC.csv file. Each node is inserted with respect to its unique UPC key and the binary search tree properties. An iterative solution had to be implemented using while loop's because a recursive solution would result in a stack overflow at runtime due to the function hitting recursion depth. The function determines the correct place for the node to be placed in the tree by iterating through the tree with a while loop, testing an if condition with each loop. The loop continues if the current node is not null, meaning the loop will stop once it has reached a leaf. The if condition tests if the key of the node to be added is less than that of the current node. If so, it sets the current node to the left node,

and if not, it sets it to the right node. A second set of if statements outside the while loop tests determine if the node to be added is the left or right child of this node. The first if statement tests if the parent of the node to be added is null, making it the root if so. The second tests if the parent's key is less node's key, making it the left child if so and the right child if not. This ensures that less than the node to be added is to the left and any node greater than the node is on the right, following the binary search tree property.

printInOrderTraversal()

Uses in-order traversal to visit every node contained in the binary search tree. This code uses a stack to simulate the recursive in-order traversal of a binary search tree. The reason that a recursive solution could not be implemented is because it would result in a stack overflow at runtime due to the function hitting recursion depth. The function starts from the leftmost node and works its way up to the root while printing the nodes in ascending order of their keys. Before the in-order traversal, an if statement tests if the tree is empty and returns null if so, preventing the code from crashing. The function first initializes this stack then continues a while loop if the current node is not null and the stack is greater than zero. A nested while loop traverses the leftmost subtree and pushes it onto the stack. After the nested while loop exits the current node is popped from the stack, and a series of if statements print the node's key, amount if applicable, and name. Finally, the current node is set to the right child to ensure that the right children nodes are printed as well.

searchTree()

Takes advantage of the already established binary search tree properties to locate a specific node using its unique UPC key. The function iterates through the tree using a while loop, continuing if the current node is not null and the target key does not match that of the current node's key. An if statement tests if the target key is less than the node's key, assigning the current node to the left node for the next iteration if so, and assigning it to the right node if not. Due to the binary search tree properties, the key must be to the left if it is less and to the right if it is greater, as each nodes' left child is less and right child is greater. Once the while loop exits, the target key must match the key of the current node, and the function returns this node.

main()

This function reads the UPC.csv and input.dat files and calls the functions necessary to build the binary search tree, print each node using in-order traversal, and search for a particular node's key and print the results. First, the function creates a BinaryTree object so that the other functions can be called on it later. Then a scanner object is initialized and called on the CSV file. A while loop iterates through each line of the CSV file. The resulting string is split at each comma using a regex, and each element of the string array is assigned to an attribute of the TreeNode object and constructed. The key attribute is first converted to a long using the parseLong() function. The TreeNode object is then added to items array list. While the insert() function could have been called on each object directly, it was found to be much slower in testing. The arraylist is then iterated over using a for loop, and the insert() function is called on each object. The printInOrderTraversal() is then called on the

tree to print every node's attributes. Next, a new scanned is instantiated and called on the input.dat file. Using a similar method to before, the resulting string is split at each comma using a regex, the first element of each array is converted to a long, and then added to an array list. A for loop then iterates through this array list, calling the searchTree() function on each key and the root of the tree. A series of print statements then print out the result's key, amount if applicable, and name. An if statement was also added to test if the result is null, ensuring that the code does not crash if the key was not found. The function also records the search and insert functions' execution time and prints it in nanoseconds, milliseconds, and seconds, then prints the results.

Code Screenshots:

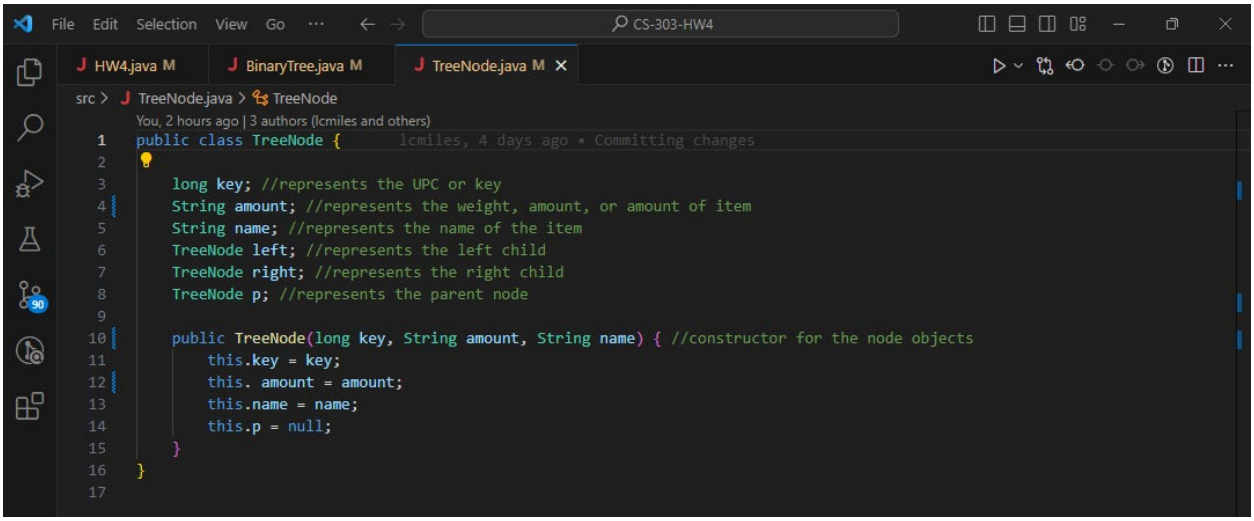


Figure 1: TreeNode.java

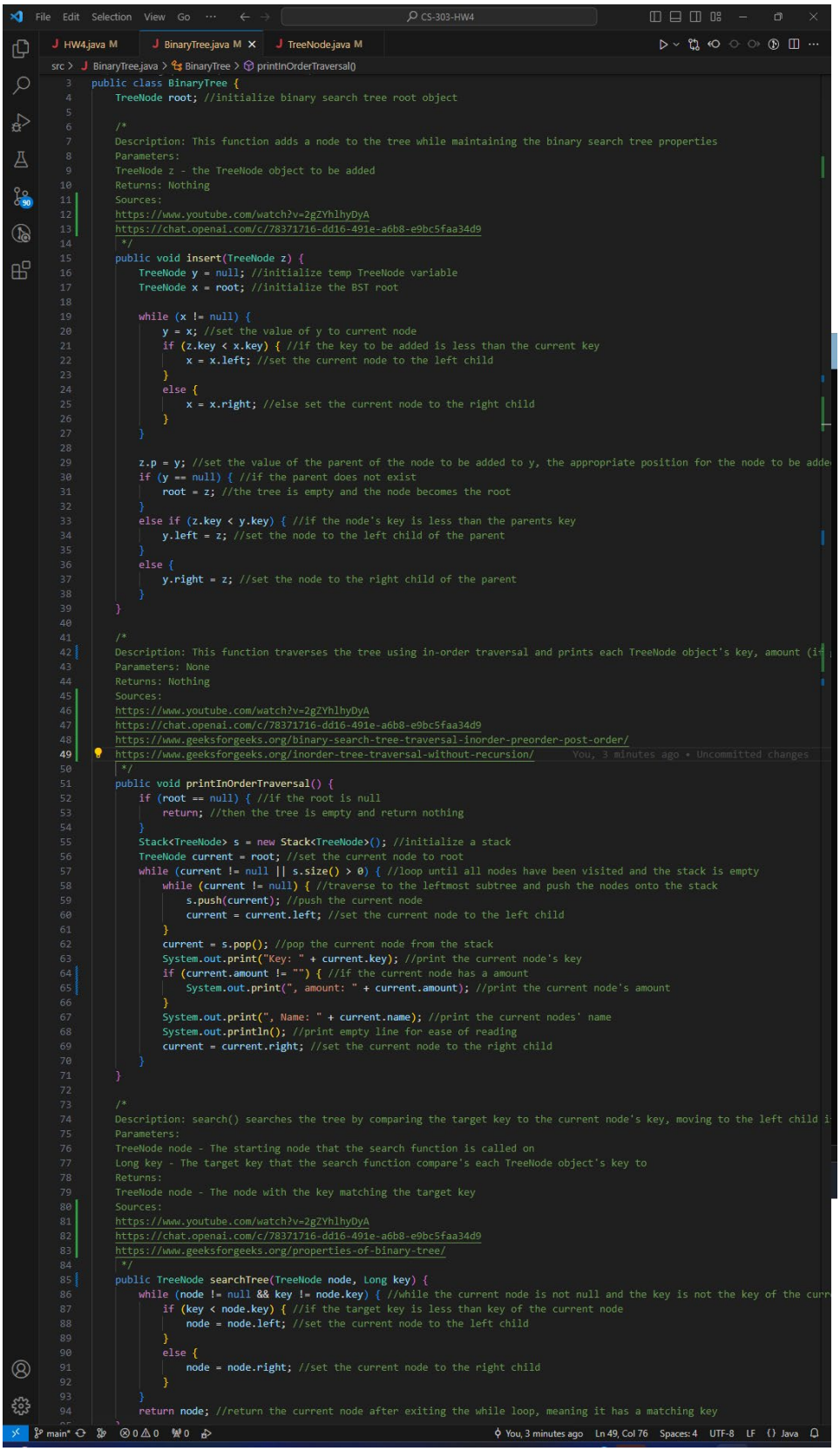
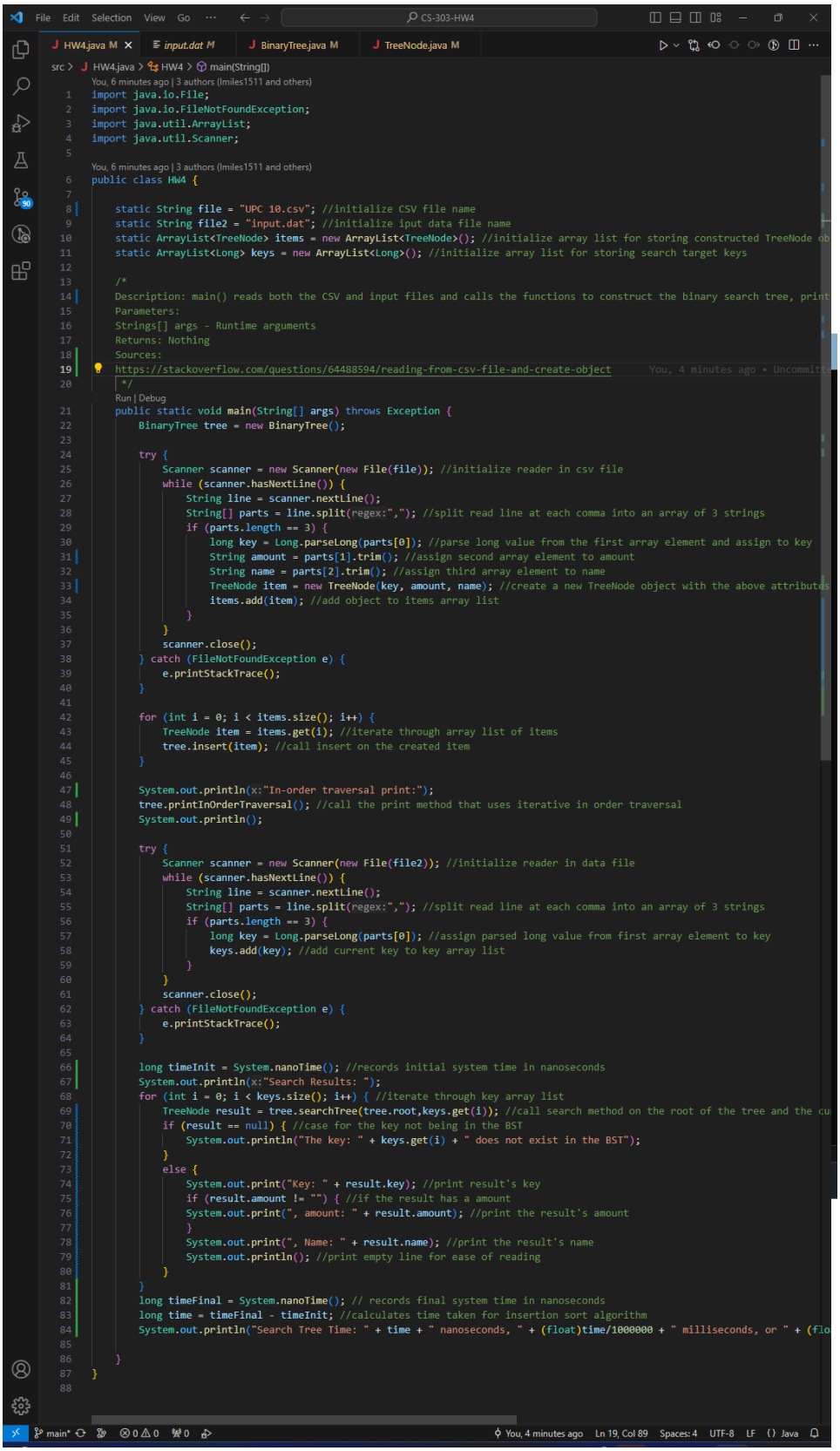


Figure 2: BinaryTree.java



```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

public class HW4 {

    static String file = "UPC 10.csv"; //initialize CSV file name
    static String file2 = "input.dat"; //initialize iput data file name
    static ArrayList<TreeNode> items = new ArrayList<TreeNode>(); //initialize array list for storing constructed TreeNode ob
    static ArrayList<Long> keys = new ArrayList<Long>(); //initialize array list for storing search target keys

    /*
    Description: main() reads both the CSV and input files and calls the functions to construct the binary search tree, print
    Parameters:
    Strings[] args - Runtime arguments
    Returns: Nothing
    Sources:
    https://stackoverflow.com/questions/64488594/reading-from-csv-file-and-create-object
    */
    public static void main(String[] args) throws Exception {
        BinaryTree tree = new BinaryTree();

        try {
            Scanner scanner = new Scanner(new File(file)); //initialize reader in csv file
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                String[] parts = line.split(",", -1); //split read line at each comma into an array of 3 strings
                if (parts.length == 3) {
                    long key = Long.parseLong(parts[0]); //parse long value from the first array element and assign to key
                    String amount = parts[1].trim(); //assign second array element to amount
                    String name = parts[2].trim(); //assign third array element to name
                    TreeNode item = new TreeNode(key, amount, name); //create a new TreeNode object with the above attributes
                    items.add(item); //add object to items array list
                }
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        for (int i = 0; i < items.size(); i++) {
            TreeNode item = items.get(i); //iterate through array list of items
            tree.insert(item); //call insert on the created item
        }

        System.out.println("In-order traversal print:");
        tree.printInOrderTraversal(); //call the print method that uses iterative in order traversal
        System.out.println();

        try {
            Scanner scanner = new Scanner(new File(file2)); //initialize reader in data file
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                String[] parts = line.split(",", -1); //split read line at each comma into an array of 3 strings
                if (parts.length == 3) {
                    long key = Long.parseLong(parts[0]); //assign parsed long value from first array element to key
                    keys.add(key); //add current key to key array list
                }
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        long timeInit = System.nanoTime(); //records initial system time in nanoseconds
        System.out.println("Search Results: ");
        for (int i = 0; i < keys.size(); i++) { //iterate through key array list
            TreeNode result = tree.searchTree(tree.root,keys.get(i)); //call search method on the root of the tree and the cu
            if (result == null) { //case for the key not being in the BST
                System.out.println("The key: " + keys.get(i) + " does not exist in the BST");
            }
            else {
                System.out.print("Key: " + result.key); //print result's key
                if (result.amount != "") { //if the result has a amount
                System.out.print(", amount: " + result.amount); //print the result's amount
                }
                System.out.print(", Name: " + result.name); //print the result's name
                System.out.println(); //print empty line for ease of reading
            }
        }
        long timeFinal = System.nanoTime(); // records final system time in nanoseconds
        long time = timeFinal - timeInit; //calculates time taken for insertion sort algorithm
        System.out.println("Search Tree Time: " + time + " nanoseconds, " + (float)time/1000000 + " milliseconds, or " + (flo

    }
}
```

Figure 3: HW4.java

3. Testing

The algorithms in this program that were tested include the insert() function, which is the algorithm that builds the binary search tree, and the searching algorithm contained in searchTree(). Both functions were tested using several different versions of the same CSV file, each containing a different number of elements:

1) UPC 10 - 10 elements
2) UPC 100 - 100 elements
3) UPC 1K - 1000 elements
4) UPC 10K - 10000 elements
5) UPC 100K - 100000 elements
6) UPC - The Original CSV file containing 177650 elements

Screenshots of testing outputs:

```
PS C:\Users\logan\OneDrive - UAB - The University of Alabama at Birmingham\UAB Files\FA 2023\CS-303\HW\CS-303-HW4>  c:; cd 'c:\Use
rs\logan\OneDrive - UAB - The University of Alabama at Birmingham\UAB Files\FA 2023\CS-303\HW\CS-303-HW4'; & 'C:\Program Files\Jav
a\jdk-19\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\logan\OneDrive - UAB - The University of Alabama
at Birmingham\UAB Files\FA 2023\CS-303\HW\CS-303-HW4\bin' 'HW4'
In-order traversal print:
Key: 79, Name: INDIANA LOTTO
Key: 93, Name: treo 700w
Key: 123, Name: Wrsi Riversound cafe cd
Key: 161, Name: Dillons/Kroger Employee Coupon ($1.25 credit)
Key: 178, Name: Outdoor Bag
Key: 1090, amount: VAR, Name: Rountech Asset 1
Key: 1205, amount: 1 gal, Name: GIANT NATURAL MOUNTAIN SPRING WATER
Key: 1243, Name: CVS Photo 1-Hour 4x6 Finishing
Key: 1601, Name: Sainsbury's Red Pepper
Key: 2288, amount: 18 oz, Name: Winn Dixie Hand Lotion
Build Tree Time: 5918900 nanoseconds, 5.9189 milliseconds, or 0.0059189 seconds
```

Figure 4: Insert() UPC 10

```
PS C:\Users\logan\OneDrive - UAB - The University of Alabama at Birmingham\UAB Files\FA 2023\CS-303\HW\CS-303-HW4>  c:; cd 'c:\Use
rs\logan\OneDrive - UAB - The University of Alabama at Birmingham\UAB Files\FA 2023\CS-303\HW\CS-303-HW4'; & 'C:\Program Files\Jav
a\jdk-19\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\logan\OneDrive - UAB - The University of Alabama
at Birmingham\UAB Files\FA 2023\CS-303\HW\CS-303-HW4\bin' 'HW4'
In-order traversal print:
Key: 79, Name: INDIANA LOTTO
Key: 93, Name: treo 700w
Key: 123, Name: Wrsi Riversound cafe cd
Key: 161, Name: Dillons/Kroger Employee Coupon ($1.25 credit)
Key: 178, Name: Outdoor Bag
Key: 1090, amount: VAR, Name: Rountech Asset 1
Key: 1205, amount: 1 gal, Name: GIANT NATURAL MOUNTAIN SPRING WATER
Key: 1243, Name: CVS Photo 1-Hour 4x6 Finishing
Key: 1601, Name: Sainsbury's Red Pepper
Key: 2288, amount: 18 oz, Name: Winn Dixie Hand Lotion
Key: 2745, Name: sunglasses
Key: 4138, amount: 100 pack, Name: Taiyo-Yuden Value Line 8x DVD-R
Key: 4145, amount: 7.6 cm x 1.8m, Name: Conforming Bandages
Key: 5210, Name: maggi
Key: 6217, amount: 1 gal, Name: Trader Joe's 1% Lowfat Milk
Key: 7511, amount: 44 oz, Name: Maverik Plastic Soda Cup
Key: 8570, Name: jewerly
Key: 9997, amount: 32 FL OZ, Name: Down to Earth - Goji Berry Juice
Key: 10016, amount: 16.9 oz, Name: Coupon
Key: 10023, amount: 1.3 - 2.2 oz, Name: Coupon
Key: 10344, amount: CD, Name: WNCI Morning Zoo - Oops... We Did It Again!
Key: 14946, amount: 6.5mm thick - 3m roll, Name: Florist Ribbon - Lemon Yellow
Key: 29438, amount: 16 FL OZ. (1 pt) 473ML, Name: Trader Joe's Energy Drink Wild Berry
Key: 40112, amount: Variable Weight, Name: Bananas
Key: 42215, amount: Small, Name: Avacados - Green
Key: 43816, amount: 1.76 oz, Name: Altoids/ Peppermint
Key: 46268, Name: JICAMA
Key: 49887, amount: 1 x 8 oz, Name: Coca Cola:Diet Coke
Key: 50340, Name: 12057 2X72X150 POULTRY NETTING
Key: 52559, amount: 17.5 oz, Name: Raid Ant & Roach Spray
Key: 54607, Name: Engizer e2 Batteries
Key: 56731, Name: 3/8X1.5X48 POINT LATH STAKE
Key: 56748, Name: 3/8X1.5X48 WD PLASTER LATH
Key: 58322, amount: 10 oz (284g), Name: Trader Joe's Dark Chocolate Covered Macadamia Nuts
Key: 61339, Name: 4 Way Smoothing Block Nail Buffer
Key: 66419, Name: 854000 LUMBER CRAYON HOLDER
Key: 68529, Name: Gian Eagle $5 Off seafood Purchase
Key: 72342, Name: WAVELINE NON/ASBESTOS SHINGLE
Key: 72359, Name: STRT EDGE NON ASBESTOS SHINGLE
Key: 89906, amount: 16 oz (1 lb) 454g, Name: Trader Joe's Pacific Northwest Super Sweet Cut White Corn (frozen)
Key: 93699, Name: 8.fHmg.C3PWDxP6Ca.
Key: 98601, Name: 00414 9GA GALV 50# SMOOTH WIRE
Key: 98632, Name: 00426 12GA 50# GLV SMOOTH WIRE
Key: 98649, Name: 00434 14GA 50# GLV SMOOTH WIRE
Key: 98656, Name: 00442 16GA 50# GLV SMOOTH WIRE
Key: 98670, Name: 00314 9GA 50# BLAK SMOOTH WIRE
Key: 98687, Name: 00326 12GA 50# BL SMOOTH WIRE
Key: 98694, Name: 00342 16GA 50# BL SMOOTH WIRE
Key: 98724, amount: 8.5 oz (241g), Name: Trader Joe's Julienne Sliced Sun Dried Tomatoes in Olive Oil
Key: 98823, Name: 36 1X20 50 HEX NETTING
Key: 98830, Name: 48 1X20 50 HEX NETTING
Key: 99288, Name: 2X1X24X100 14GA WELD WIRE
Key: 99325, Name: 2X1X72X100 14GA WELD WIRE
Key: 99394, Name: 36X50 2X2 HDWR CLOTH
Key: 99400, Name: 48X50 2X2 HDWR CLOTH
Key: 99530, amount: 6 oz., Name: Trader Joe's Canned Crab Meat
Key: 99622, Name: 3X3 96 PAINT METAL LATH
Key: 102148, Name: CMH-4 4D MASONRY CUT NAILS
Key: 102155, Name: CMH-6 6D MASONRY CUT NAILS
Key: 102162, Name: CMH-8 8D MASONRY CUT NAILS
Key: 102179, Name: CMH-10 10D MASONRY CUT NAILS
Key: 102186, Name: CMH-16 16D MASONRY CUT NAILS
Key: 105156, Name: 19803 4-1/2 BAR TIE
Key: 105163, Name: 19804 5 BAR TIE
Key: 105170, Name: 19805 5-1/2 BAR TIE
Key: 105187, Name: 19806 6 BAR TIE
Key: 105200, Name: 19810 8 BAR TIE
Key: 114936, amount: 1 litre, Name: Indian Tonic Water
Key: 125628, amount: 454g (serves 4), Name: Sainsburys christmas pudding
Key: 129725, amount: 13.75 oz (390 g), Name: Trader Joe's Artichoke Hearts
Key: 130530, amount: 860gm, Name: Sainsbury's 48 wholewheat biscuits
Key: 140003, Name: Kroger $5.00 in Free Groceries w/ $20.00 Min. Purchase
Key: 156103, amount: 547268|0|20|CD|4|35.98, Name: CDMSIM
Key: 157148, amount: 750 ml, Name: DOM LEFLAIVE BATARD MONTRACHET BIENVENUES
Key: 165105, amount: 16 oz., Name: Trader Joe's Mini Ravioli With Cheese Filling
Key: 179744, amount: 16 oz (1 lb) 454g, Name: Trader Joe's Marinated Mushrooms with Garlic
Key: 188920, amount: 1 lb. 8oz. (681g), Name: TRADER JOE'S WHITE CREAMER POTATOES SKU#18892
Key: 200318, amount: 16 oz., Name: Trader Joe's Green Protein
Key: 210805, amount: Net Wt. 2LB 8 oz (1135g), Name: TRADER JOE'S  RUSSET BAKER POTATOES SKU#21080
Key: 235525, amount: 5.5 Oz., Name: """Trader Joe's"" Tuna for Cats"
Key: 259941, amount: 25.3 FL OZ, Name: Trader Joes Natural Mountain Spring Water
Key: 267304, amount: 280g, Name: Marks & Spencer Teacakes (16) Milk Chocolate
Key: 270427, amount: 907g, Name: organic sugar
Key: 298803, Name: 7 7X1/4 FLOAT BALL
Key: 310222, Name: 1/2X21 T&C BLACK IMP PIPE
Key: 310239, Name: 3/4X21 T&C BLK IMP PIPE
Key: 310246, Name: 1X21 T&C BLACK IMP PIPE
Key: 310253, Name: 1-1/4X21 T&C BLACK IMP PIPE
Key: 310260, Name: 1-1/2X21 T&C BLACK IMP PIPE
Key: 310277, Name: 2X21 T&C BLACK IMP PIPE
Key: 313452, Name: 1/2X21 T&C GALV IMP PIPE
Key: 313469, Name: 3/4X21 T&C GALV IMP PIPE
Key: 313476, Name: 1X21 T&C GALV IMP PIPE
Build Tree Time: 25126900 nanoseconds, 25.1269 milliseconds, or 0.0251269 seconds
```

Figure 5: Insert() UPC 100

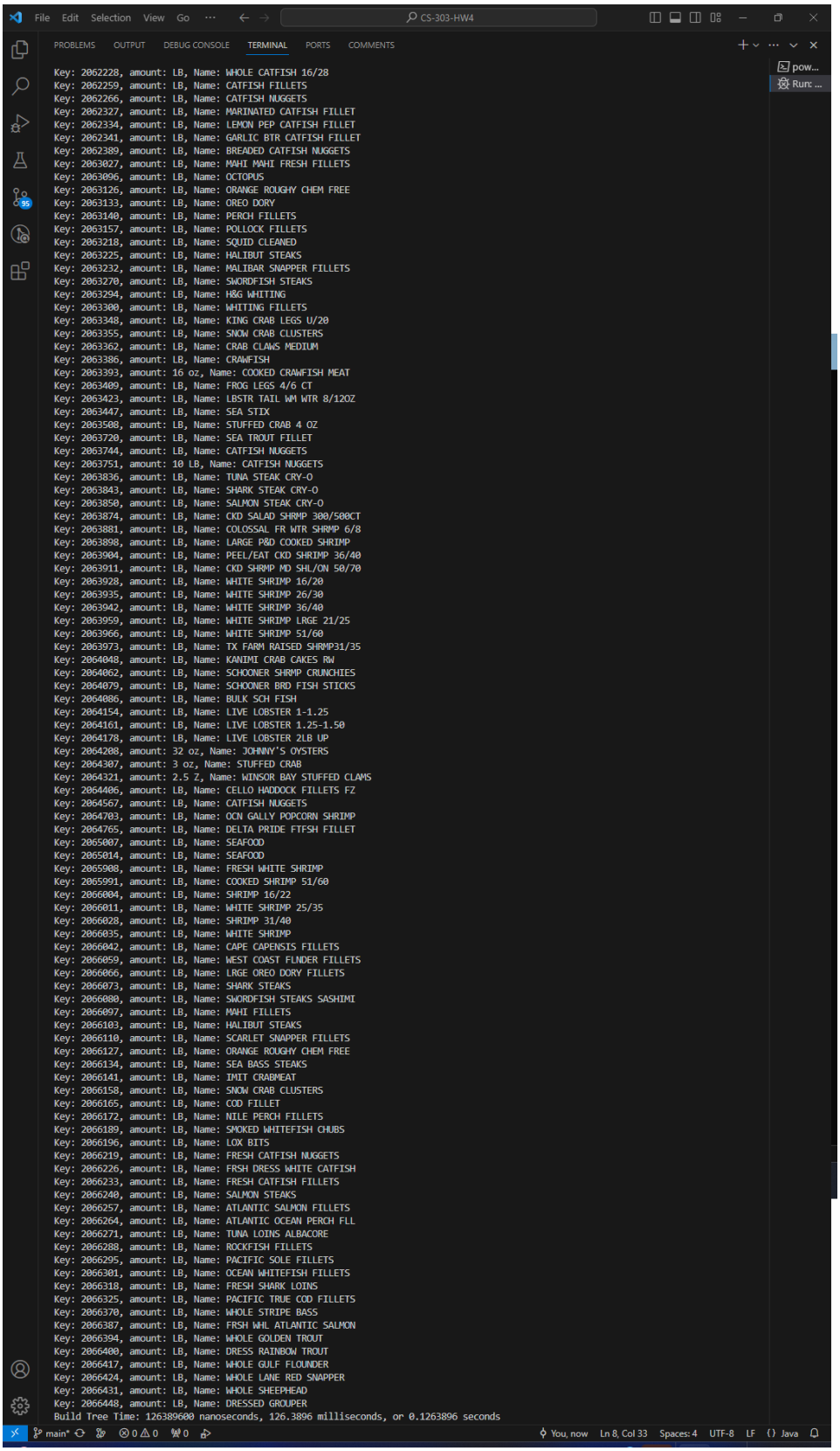


Figure 6: Insert() UPC 1K

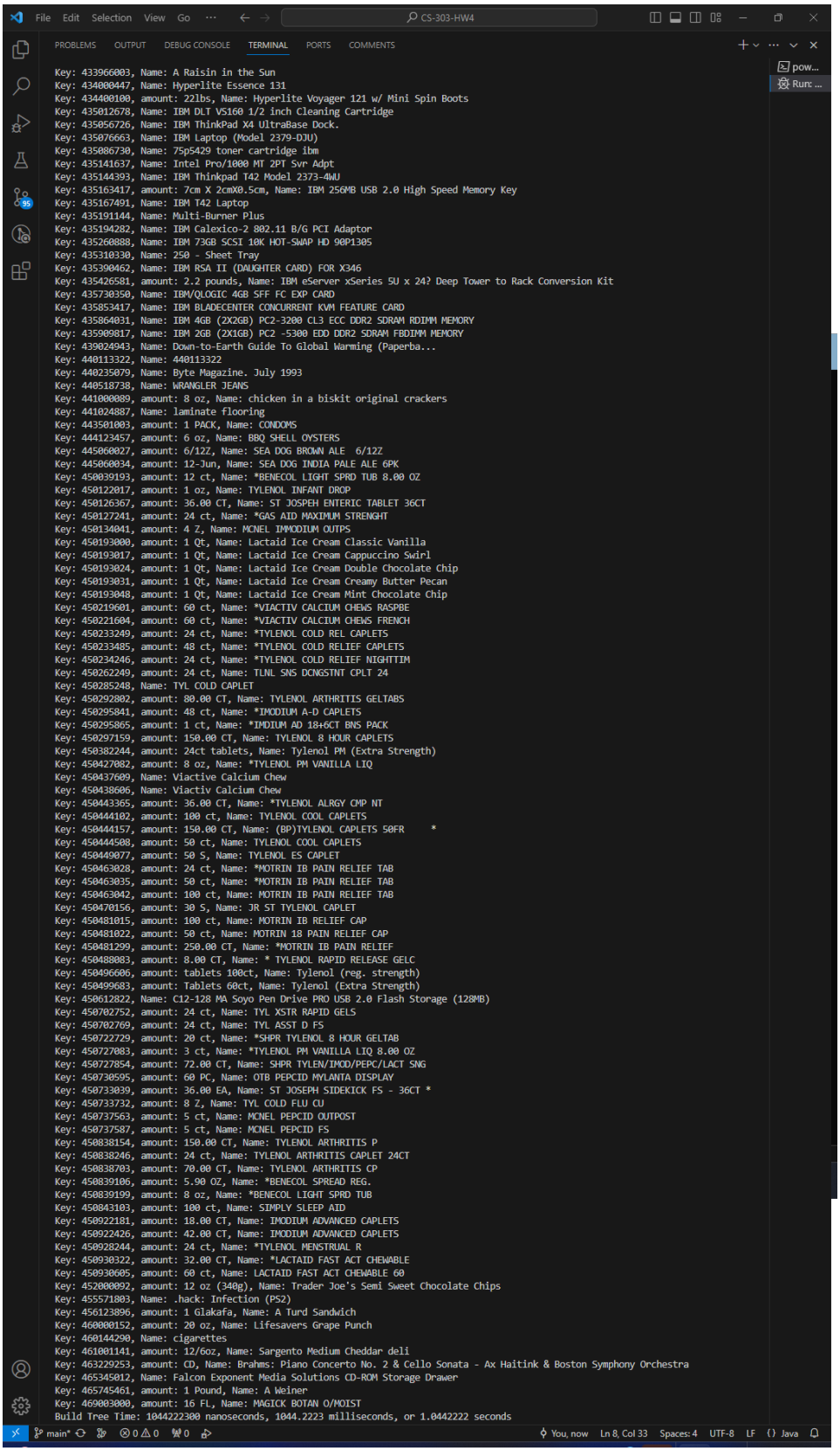


Figure 7: Insert() UPC 10K

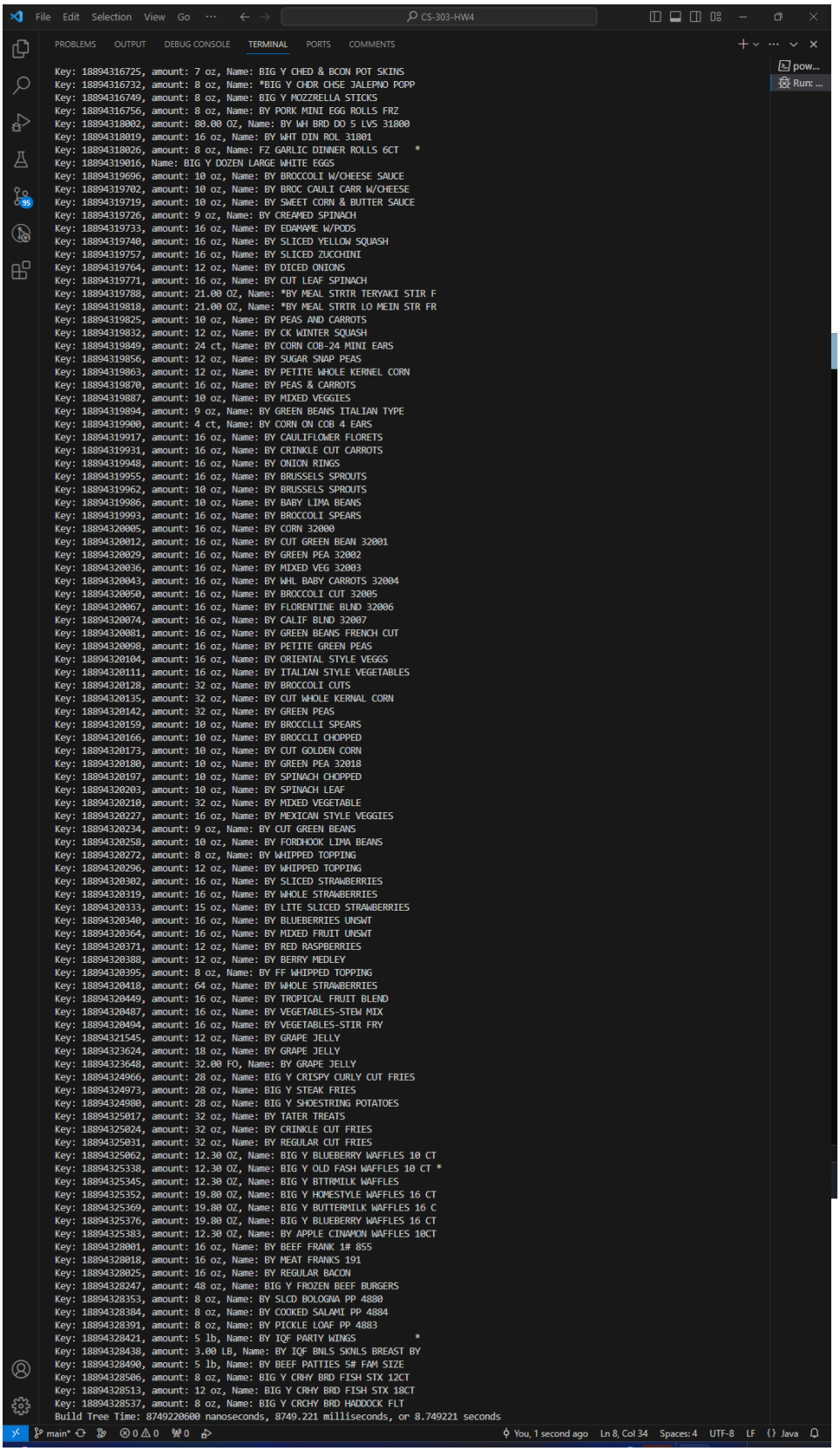


Figure 8: Insert() UPC 100K

Figure 9: Insert() UPC

```
Search Results:
Key: 79, Name: INDIANA LOTTO
Key: 93, Name: treo 700w
Key: 123, Name: Wrsi Riversound cafe cd
Key: 161, Name: Dillons/Kroger Employee Coupon ($1.25 credit)
The key: 2140000070 does not exist in the BST
The key: 2140118461 does not exist in the BST
The key: 2144209103 does not exist in the BST
The key: 2144622711 does not exist in the BST
The key: 2147483647 does not exist in the BST
The key: 2158242769 does not exist in the BST
The key: 2158561631 does not exist in the BST
The key: 2158769549 does not exist in the BST
The key: 2160500567 does not exist in the BST
The key: 2172307284 does not exist in the BST
The key: 2177000074 does not exist in the BST
The key: 2184000098 does not exist in the BST
The key: 2187682888 does not exist in the BST
Search Tree Time: 7550800 nanoseconds, 7.5508 milliseconds, or 0.0075508 seconds
PS C:\Users\logan\OneDrive - UAB - The University of Alabama at Birmingham\UAB Files\FA 2023\CS-303\HW\CS-303-HW4>
```

Figure 10: searchTree() UPC 10

```
Search Results:
Key: 79, Name: INDIANA LOTTO
Key: 93, Name: treo 700w
Key: 123, Name: Wrsi Riversound cafe cd
Key: 161, Name: Dillons/Kroger Employee Coupon ($1.25 credit)
The key: 2140000070 does not exist in the BST
The key: 2140118461 does not exist in the BST
The key: 2144209103 does not exist in the BST
The key: 2144622711 does not exist in the BST
The key: 2147483647 does not exist in the BST
The key: 2158242769 does not exist in the BST
The key: 2158561631 does not exist in the BST
The key: 2158769549 does not exist in the BST
The key: 2160500567 does not exist in the BST
The key: 2172307284 does not exist in the BST
The key: 2177000074 does not exist in the BST
The key: 2184000098 does not exist in the BST
The key: 2187682888 does not exist in the BST
Search Tree Time: 4204200 nanoseconds, 4.2042 milliseconds, or 0.0042042 seconds
PS C:\Users\logan\OneDrive - UAB - The University of Alabama at Birmingham\UAB Files\FA 2023\CS-303\HW\CS-303-HW4>
```

Figure 11: searchTree() UPC 100

```
Search Results:
Key: 79, Name: INDIANA LOTTO
Key: 93, Name: treo 700w
Key: 123, Name: Wrsi Riversound cafe cd
Key: 161, Name: Dillons/Kroger Employee Coupon ($1.25 credit)
The key: 2140000070 does not exist in the BST
The key: 2140118461 does not exist in the BST
The key: 2144209103 does not exist in the BST
The key: 2144622711 does not exist in the BST
The key: 2147483647 does not exist in the BST
The key: 2158242769 does not exist in the BST
The key: 2158561631 does not exist in the BST
The key: 2158769549 does not exist in the BST
The key: 2160500567 does not exist in the BST
The key: 2172307284 does not exist in the BST
The key: 2177000074 does not exist in the BST
The key: 2184000098 does not exist in the BST
The key: 2187682888 does not exist in the BST
Search Tree Time: 3876400 nanoseconds, 3.8764 milliseconds, or 0.0038764 seconds
PS C:\Users\logan\OneDrive - UAB - The University of Alabama at Birmingham\UAB Files\FA 2023\CS-303\HW\CS-303-HW4>
```

Figure 12: searchTree() UPC 1K

```
Search Results:
Key: 79, Name: INDIANA LOTTO
Key: 93, Name: treo 700w
Key: 123, Name: Wrsi Riversound cafe cd
Key: 161, Name: Dillons/Kroger Employee Coupon ($1.25 credit)
The key: 2140000070 does not exist in the BST
The key: 2140118461 does not exist in the BST
The key: 2144209103 does not exist in the BST
The key: 2144622711 does not exist in the BST
The key: 2147483647 does not exist in the BST
The key: 2158242769 does not exist in the BST
The key: 2158561631 does not exist in the BST
The key: 2158769549 does not exist in the BST
The key: 2160500567 does not exist in the BST
The key: 2172307284 does not exist in the BST
The key: 2177000074 does not exist in the BST
The key: 2184000098 does not exist in the BST
The key: 2187682888 does not exist in the BST
Search Tree Time: 6478300 nanoseconds, 6.4783 milliseconds, or 0.0064783 seconds
PS C:\Users\logan\OneDrive - UAB - The University of Alabama at Birmingham\UAB Files\FA 2023\CS-303\HW\CS-303-HW4>
```

Figure 13: searchTree() UPC 10K

```
Search Results:
Key: 79, Name: INDIANA LOTTO
Key: 93, Name: treo 700w
Key: 123, Name: Wrsi Riversound cafe cd
Key: 161, Name: Dillons/Kroger Employee Coupon ($1.25 credit)
Key: 2140000070, Name: Rhinestone Watch
Key: 2140118461, Name: """V"": Breakout/The Deception  VHS Tape"
Key: 2144209103, amount: VHS, Name: Tintorera - Tiger Shark
Key: 2144622711, Name: Taxi : The Collector's Edition VHS
Key: 2147483647, Name: Toshiba 2805 DVD player
Key: 2158242769, amount: 288/1.12Z, Name: GREEN SUGAR COOKIES4276
Key: 2158561631, Name: HOT COCOA W/BKMK
Key: 2158769549, amount: njhjhn, Name: gjfhjbgkj
Key: 2160500567, amount: 2.25 oz (64)g, Name: Dollar Bar Rich Raspberry
Key: 2172307284, Name: Mixed seasonal flower bouquet
Key: 2177000074, Name: 4 way 13 AMP Extension Lead (Wilkinson UK)
Key: 2184000098, amount: 21 oz, Name: Christopher's Assorted Fruit Jellies
Key: 2187682888, Name: fairway
Search Tree Time: 4739400 nanoseconds, 4.7394 milliseconds, or 0.0047394 seconds
PS C:\Users\logan\OneDrive - UAB - The University of Alabama at Birmingham\UAB Files\FA 2023\CS-303\HW\CS-303-HW4>
```

Figure 14: searchTree() UPC 100K

```
Search Results:
Key: 79, Name: INDIANA LOTTO
Key: 93, Name: treo 700w
Key: 123, Name: Wrsi Riversound cafe cd
Key: 161, Name: Dillons/Kroger Employee Coupon ($1.25 credit)
Key: 2140000070, Name: Rhinestone Watch
Key: 2140118461, Name: """V"": Breakout/The Deception  VHS Tape"
Key: 2144209103, amount: VHS, Name: Tintorera - Tiger Shark
Key: 2144622711, Name: Taxi : The Collector's Edition VHS
Key: 2147483647, Name: Toshiba 2805 DVD player
Key: 2158242769, amount: 288/1.12Z, Name: GREEN SUGAR COOKIES4276
Key: 2158561631, Name: HOT COCOA W/BKMK
Key: 2158769549, amount: njhjhn, Name: gjfhjbgkj
Key: 2160500567, amount: 2.25 oz (64)g, Name: Dollar Bar Rich Raspberry
Key: 2172307284, Name: Mixed seasonal flower bouquet
Key: 2177000074, Name: 4 way 13 AMP Extension Lead (Wilkinson UK)
Key: 2184000098, amount: 21 oz, Name: Christopher's Assorted Fruit Jellies
Key: 2187682888, Name: fairway
Search Tree Time: 4361600 nanoseconds, 4.3616 milliseconds, or 0.0043616 seconds
PS C:\Users\logan\OneDrive - UAB - The University of Alabama at Birmingham\UAB Files\FA 2023\CS-303\HW\CS-303-HW4>
```

Figure 15: searchTree() UPC

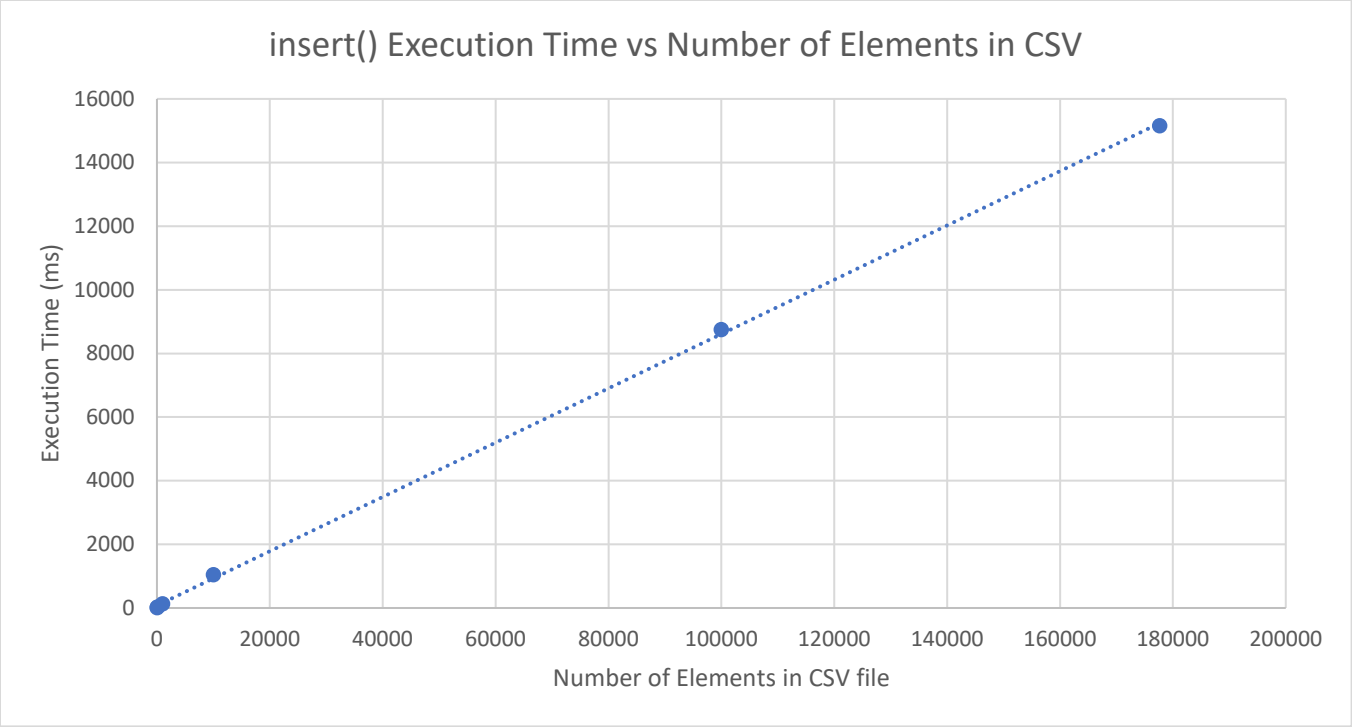


Figure 16: insert() Execution Time vs Number of Elements in CSV

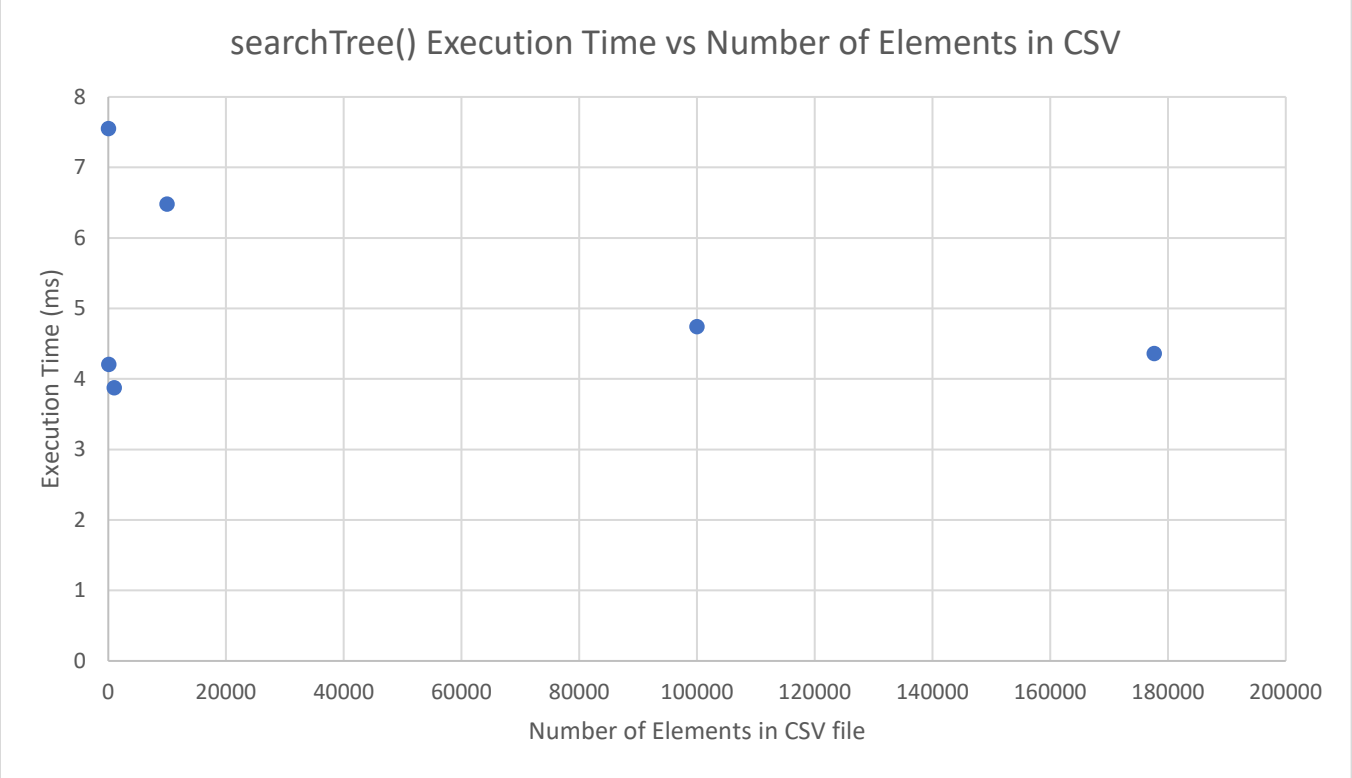


Figure 17: searchTree() Execution Time vs Number of Elements in CSV

The data above clearly shows that the time complexity of the insert() function is linear, or O(n). The time complexity of the search() function is not so easily discernable, however, as just by looking at the graph it would seem there is no connection between the execution time of the algorithm and the number of elements operated on. This isn't reflective of the algorithm's actual time complexity, which is O(log(n)). This is because the algorithm is similar to binary search, in that it divides the tree that needs to be searched in half with each iteration, resulting in a time complexity of O(log(n)) on average and O(n) in the worst case. The reason that the graph poorly reflects the actual time complexity of the algorithm could be due to the scaling of the datasets or the fact that the same search keys were used for each test. In the future, it could be more effective to randomize the keys with each test.

4. Sources

https://stackoverflow.com/questions/64488594/reading-from-csv-file-and-create-object

https://www.youtube.com/watch?v=2gZYhlhyDyA

https://www.geeksforgeeks.org/binary-search-tree-traversal-inorder-preorder-post-order/

https://www.geeksforgeeks.org/properties-of-binary-tree/

https://www.geeksforgeeks.org/inorder-tree-traversal-without-recursion/

https://chat.openai.com/c/78371716-dd16-491e-a6b8-e9bc5faa34d9