CPSC 319
Assignment 3

Aleksander Berezowski
Tutorial Section: T03
TA Name: Shopon
Email: aleksander.berezowsk@ucalgary.ca

Contents

Figures

ReadMe

How to compile and run program

Assign3:

~ 1. Compiling the program ~

Prior to compiling the program, one must cd into the directory where the .java files are saved. Ensure that the files Assign3.java, BinaryTree.java, Data.java, Node.java, and Queue.java are present. Then to compile the files, the following command should be ran:

javac *.java

~ 2. Running the program ~

Once again, prior to running the program one must check that the input text file(s) are in a "src" subfolder present in the same directory as where the .java files are saved. Why do they need to be in a "src" subfolder? No idea. I can't explain this, and the string "src" is no where in my program. I am very confused, but if the input text file(s) are in a "src" subfolder then it works. To run the program, the following command should be used:

java Assign3 inputFileName depthFirstFileName breadthFirstFileName

Note: In time of actual use, replace the word inputFileName, depthFirstFileName, and breadthFirstFileName with the appropriate text file name that you are wishing to use. The extension .txt should not be added to the end of the text file name.

Assign3b (Bonus Component)

~ 1. Compiling the program ~

Prior to compiling the program, one must cd into the directory where the .java files are saved. Ensure that the files Assign3b.java, BinaryTree.java, AVLTree.java, Data.java, Node.java, and Queue.java are present. Then to compile the files, the following command should be ran:

javac *.java

~ 2. Running the program ~

Once again, prior to running the program one must check that the input text file(s) are in a "src" subfolder present in the same directory as where the .java files are saved. Why do they need to be in a "src" subfolder? No idea. I can't explain this, and the string "src" is no where in my program. I am very confused, but if the input text file(s) are in a "src" subfolder then it works. To run the program, the following command should be used:

java Assign3b inputFileName depthFirstFileName breadthFirstFileName

Note: In time of actual use, replace the word inputFileName, depthFirstFileName, and breadthFirstFileName with the appropriate text file name that you are wishing to use. The extension .txt should not be added to the end of the text file name.

Output files generated by program

Below are 4 outputs. Figure 1 and Figure 2 are both generated by the Binary Tree data structure, whereas Figure 3 and Figure 4 are done by an AVL Tree data structure.

java Assign3 a3input2 outputFile1 outputFile2

outputFile1.txt

1	8503881	Appleford	0251	EST	1
2	8500453	Banks	0251	EST	1
3	8301164	Bannister	0251	EST	1
4	8499120	Black	0341	RST	1
5	8422991	Card	0045	JA	2
6	8399000	Doyle	0334	ENT	1
7	8508883	Fisher	0341	TXT	1
8	8400912	Green	0045	RFM	1
9	8322189	Hopper	0251	CT	1
10	8422911	Johnston	0341	RST	1
11	8306700	Jones	0251	CT	2
12	8399129	Jordan	0334	ENT	1
13	8400342	LaPorte	0045	JA	1
14	8300500	Last	0251	CT	1
15	8322889	Morrison	0045	JA	3
16	8322188	Newman	0251	CT	2
17	8412094	Phillips	0251	EST	1
18	8400001	Smith	0251	CT	2
19	8255999	Sykes	0451	WET	2
20	8300000	Walker	0341	TXT	1
21	8499341	Waters	0341	TXT	1
22	8400000	Watson	0334	ENT	2
23	8500060	Watts	0341	RST	1
24	8500000	West	0334	ENT	1
25	8367091	White	0341	TXT	1
26	8388231	Woods	0251	CT	1

Figure 1 - Assign3 outputFile1.txt

outputFile2.txt

1	8322889	Morrison	0045	JA	3
2	8400342	LaPorte	0045	JA	1
3	8400001	Smith	0251	CT	2
4	8499120	Black	0341	RST	1
5	8300500	Last	0251	CT	1
6	8412094	Phillips	0251	EST	1
7	8367091	White	0341	TXT	1
8	8301164	Bannister	0251	EST	1
9	8400912	Green	0045	RFM	1
10	8322188	Newman	0251	CT	2
11	8255999	Sykes	0451	WET	2
12	8388231	Woods	0251	CT	1
13	8500453	Banks	0251	EST	1
14	8422991	Card	0045	JA	2
15	8422911	Johnston	0341	RST	1
16	8300000	Walker	0341	TXT	1
17	8503881	Appleford	0251	EST	1
18	8508883	Fisher	0341	TXT	1
19	8322189	Hopper	0251	CT	1
20	8306700	Jones	0251	CT	2
21	8500000	West	0334	ENT	1
22	8399000	Doyle	0334	ENT	1
23	8399129	Jordan	0334	ENT	1
24	8400000	Watson	0334	ENT	2
25	8499341	Waters	0341	TXT	1
26	8500060	Watts	0341	RST	1

Figure 2 - Assign3 outputFile2.txt

java Assign3b a3input2 outputFile1 outputFile2

outputFile1.txt

1	8503881	Appleford	0251	EST	1
2	8500453	Banks	0251	EST	1
3	8301164	Bannister	0251	EST	1
4	8499120	Black	0341	RST	1
5	8422991	Card	0045	JA	2
6	8399000	Doyle	0334	ENT	1
7	8508883	Fisher	0341	TXT	1
8	8400912	Green	0045	RFM	1
9	8322189	Hopper	0251	CT	1
10	8422911	Johnston	0341	RST	1
11	8306700	Jones	0251	CT	2
12	8399129	Jordan	0334	ENT	1
13	8400342	LaPorte	0045	JA	1
14	8300500	Last	0251	CT	1
15	8322889	Morrison	0045	JA	3
16	8322188	Newman	0251	CT	2
17	8412094	Phillips	0251	EST	1
18	8400001	Smith	0251	CT	2
19	8255999	Sykes	0451	WET	2
20	8300000	Walker	0341	TXT	1
21	8499341	Waters	0341	TXT	1
22	8400000	Watson	0334	ENT	2
23	8500060	Watts	0341	RST	1
24	8500000	West	0334	ENT	1
25	8367091	White	0341	TXT	1
26	8388231	Woods	0251	CT	1

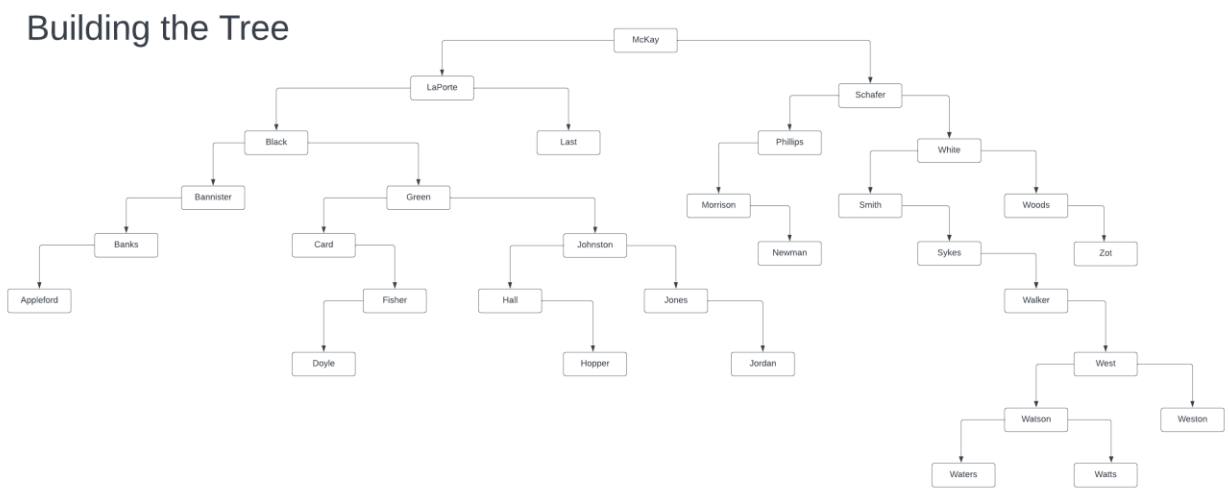
Figure 3 - Assign3b outputFile1.txt

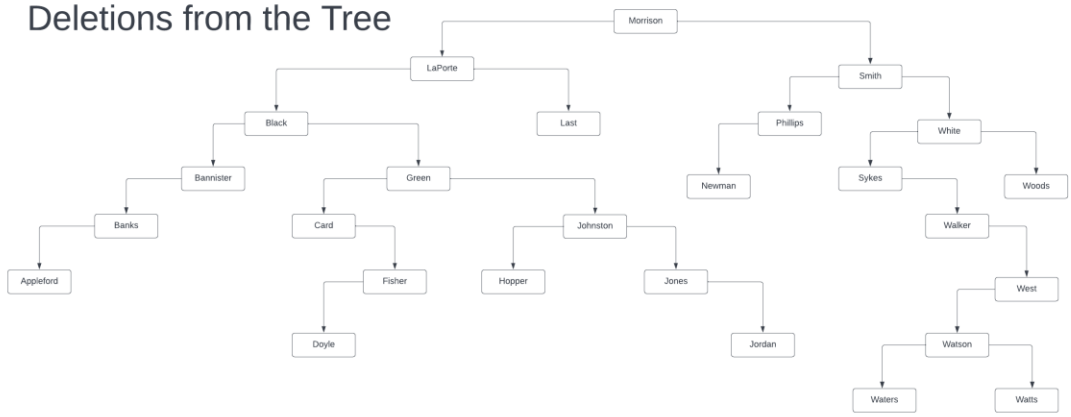
outputFile2.txt

1	8322889	Morrison	0045	JA	3
2	8400342	LaPorte	0045	JA	1
3	8400001	Smith	0251	CT	2
4	8499120	Black	0341	RST	1
5	8300500	Last	0251	CT	1
6	8412094	Phillips	0251	EST	1
7	8367091	White	0341	TXT	1
8	8301164	Bannister	0251	EST	1
9	8400912	Green	0045	RFM	1
10	8322188	Newman	0251	CT	2
11	8255999	Sykes	0451	WET	2
12	8388231	Woods	0251	CT	1
13	8500453	Banks	0251	EST	1
14	8422991	Card	0045	JA	2
15	8422911	Johnston	0341	RST	1
16	8300000	Walker	0341	TXT	1
17	8503881	Appleford	0251	EST	1
18	8508883	Fisher	0341	TXT	1
19	8322189	Hopper	0251	CT	1
20	8306700	Jones	0251	CT	2
21	8500000	West	0334	ENT	1
22	8399000	Doyle	0334	ENT	1
23	8399129	Jordan	0334	ENT	1
24	8400000	Watson	0334	ENT	2
25	8499341	Waters	0341	TXT	1
26	8500060	Watts	0341	RST	1

Figure 4 - Assign3b outputFile2.txt

Picture of binary search tree





Complexity analysis

Question 1

Assuming that the records are inserted into the tree in random order, what is the height of your tree expressed using big-O notation?

With n as the total number of items and the insertion being a random order, and that random order does not happen to be alphabetically or reverse-alphabetically, the height of the tree will have the big-O notation of O(log n).

If going breadth first, the number of nodes will be 1, 2, 4, etc. until reaching the bottom of the tree. If a binary tree has a height of h, then it has 2^h leaf nodes. When taking the sum of this series, you will find it is equivalent to 2^{h+1}-1. This logic is shown in the following equations:

$$n_{nodes} = 1 + 2 + 4 + \cdots + 2^h = 2^0 + 2^1 + 2^2 + \cdots + 2^h = 2^{h+1} - 1$$

When solving in respect to h, you get a logarithmic equation. When applying big O notation rules, you get a big-O notation of O(log n).

$$h_{tree} = \log_2(n+1) - 1 = O(\log n)$$

Due to each level being double the size of the last level, this intuitively confirms this conclusion to be correct.

Question 2

What is the worst-case height of the tree? What input gives the worst case?

The worst-case height of the tree is when every single item has one parent and one child, meaning the tree has deteriorated into a linked list. This most commonly happens when the items start in either alphabetical or reverse-alphabetical order. For example, if the items are inserted alphabetically then each item only has a right child, resulting in a linked list structure. Therefore, if the items are inserted reverse-alphabetically then each item only has a left child, also resulting in a linked list structure. Therefore, in the worst-case scenario, the height of the tree will be O(n).

Question 3

What is the worst-case space complexity of the depth-first, in-order traversal and breadth-first traversal? Compare your implementation of these two methods: is there one that will outperform another in terms of memory usage for a specific data set? Discuss.

Depth-First, In-Order Traversal

Within the algorithm for this type of traversal there is no dependence on the number of records, therefore each function in and of itself is O(1). Depth-first, in-order traversal is a recursive method that, at maximum, generates a copy of itself one time for each level of the tree. If we continue to define the height of the tree as h, then the instances of the algorithms can be characterized as O(h). Using the product rule, the space complexity of this function is $O(h) * O(1) = O(h)$

Breadth-First Traversal

This traversal method implements a queue to keep track of all the nodes at each level. The queue will be the largest at the last level of the binary tree with all the leaf nodes, and the number of leaf nodes will therefore be $2^{h+1} - 1$ in a binary tree, as shown above. A queue space is needed for each node, therefore the space complexity of breadth-first traversal will be characterized as $O(2^h)$.

Conclusion

Depth-first, in-order traversal has a space complexity of O(h) and breadth-first traversal has a complexity of $O(2^h)$, therefore depth-first, in-order traversal is by far more space efficient.

Bibliography

- “Avl Tree: Set 1 (insertion),” GeeksforGeeks, 19-Feb-2022. [Online]. Available: https://www.geeksforgeeks.org/avl-tree-set-1-insertion/.
- “Binary search tree: Set 2 (delete),” GeeksforGeeks, 01-Feb-2022. [Online]. Available: https://www.geeksforgeeks.org/binary-search-tree-set-2-delete/.
- “Binary tree: Set 1 (introduction),” GeeksforGeeks, 22-Nov-2021. [Online]. Available: https://www.geeksforgeeks.org/binary-tree-set-1-introduction/.
- “Dynamic queue implementation in java - w3schools,” w3schools. [Online]. Available: https://www.w3schools.blog/java-dynamic-queue-implementation.
- “Java create file,” w3schools. [Online]. Available: https://www.w3schools.com/java/java_files_create.asp.
- “https://www.geeksforgeeks.org/implementing-a-linked-list-in-java-using-class/,” GeeksForGeeks. [Online]. Available: https://www.geeksforgeeks.org/implementing-a-linked-list-in-java-using-class/
- “Java tutorial,” TutorialsPoint. [Online]. Available: http://www.tutorialspoint.com/java/.
- “Level order binary tree traversal,” GeeksforGeeks, 30-Dec-2021. [Online]. Available: https://www.geeksforgeeks.org/level-order-tree-traversal/.
- “Queue poll() method in Java,” GeeksforGeeks, 20-Oct-2021. [Online]. Available: https://www.geeksforgeeks.org/queue-poll-method-in-java/.
- “Queue: Set 1 (introduction and array implementation),” GeeksforGeeks, 09-Sep-2021. [Online]. Available: https://www.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/.
- “Read file into an array in Java,” GeeksforGeeks, 21-Feb-2022. [Online]. Available: https://www.geeksforgeeks.org/read-file-into-an-array-in-java/.
- S. Woltmann, “AVL tree (with java code),” HappyCoders.eu, 07-Dec-2021. [Online]. Available: https://www.happycoders.eu/algorithms/avl-tree-java/.
- “Tree traversals (Inorder, preorder and postorder),” GeeksforGeeks, 30-Nov-2021. [Online]. Available: https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/.
- “Tree traversals (Inorder, preorder and postorder),” GeeksforGeeks, 30-Nov-2021. [Online]. Available: https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/.