CSCE 221 Cover Page

Homework Assignment # PA-3

First Name: Justin Last Name: Lee UIN: 727000824

Username: 727000824 E-mail address: jlee232435@gmail.com

Please list all sour es in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you a get a lower number of points or even zero, read more in the Aggie Honor System O- e http://aggiehonor.tamu.edu/

Type of sources	Textbook	Lecture Slides		
People				
Web pages (provide URL)		Dr. Leyk's Trees Slides		
Printed material	Data Structures and Algorithm Analysis in C++ by Mark A. Weiss			
Other sources				

I certify that I have listed all the sour es that I used to develop the solutions/codes to the submitted work. *On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*

Your Name: Justin Lee Date: 3/29/20

1. A description of the assignment objective, how to compile and run your programs, and an explanation of your program structure (i.e. a description of the classes you use, the relationship between the classes, and the functions or classes in addition to those in the lecture notes).

In this assignment, we are tasked to create a binary search tree in order to calculate the average search cost of each node in the tree as well as output the entire tree, level by level. There are two structures, where one is for the nodes in the search tree and the other is in the tree itself. The search tree has nodes that contain data and pointers to other nodes in the tree. In addition, the tree uses functions for finding the average search cost and reading the input from files as well as a few overloaded operators for output.

The files included in this project are the BSTree.cpp, BSTree.h, BSTree_main.cpp, and the makefile. The makefile will compile all the programs together, using ./test.

2. A brief description of the data structure you create (i.e. a theoretical definition of the data structure and the actual data arrangement in the classes).

The data structure implemented in this assignment is a binary search tree, which is node-based binary tree that gives quick access to data using keys. Although there is a large initial cost of creating the tree, there is a tradeoff as the operations for searching items within the tree is significantly reduced in cost. The tree's properties consist of a left subtree and a right subtree. The left subtree of a node only has nodes with keys lesser than the node's key, while the right subtree is for nodes with keys greater than the node's key. In this way, any key comparisons made within the subtrees helps operations to avoid checking through about half the tree itself. The actual data arrangement in the classes was done through the use of the Node struct and vectors were also used to help store the input data so that it could be easily inserted into the tree when needed.

3. A description of how you implemented the calculation of (a) individual search cost and (b) average search cost and explain which tree operation (e.g. find, insert) was helpful. Analyze the time complexity of (a) calculating individual search cost and (b) summing up the search costs over all the nodes.

(a) The calculation of the individual search cost was implemented with an update_search_time() function that used a DFS algorithm which incremented the search times of all the nodes as it traversed through the tree. The time complexity for calculating the individual search cost is O(n).

(b) The calculation of the average search cost was implemented by calculating for the total cost and then dividing it by the number of nodes in the tree using the get_total_search_time() helper function. The time complexity for summing up the search costs over all the nodes is O(n).

4. Give an individual search cost in terms of n using big-O notation. Analyze and give the average search costs of a perfect binary tree and a linear binary tree using big-O notation, assuming that the following formulas are true (n denotes the total number of input data).

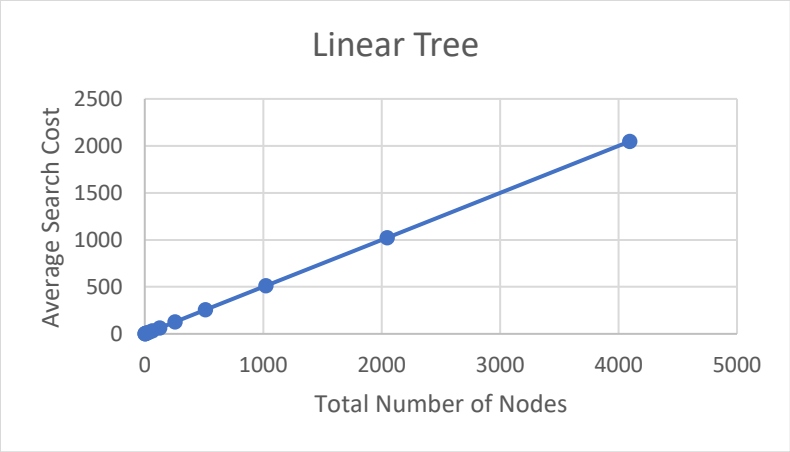
We know that the individual search cost of each item has a big-O of O(logn).

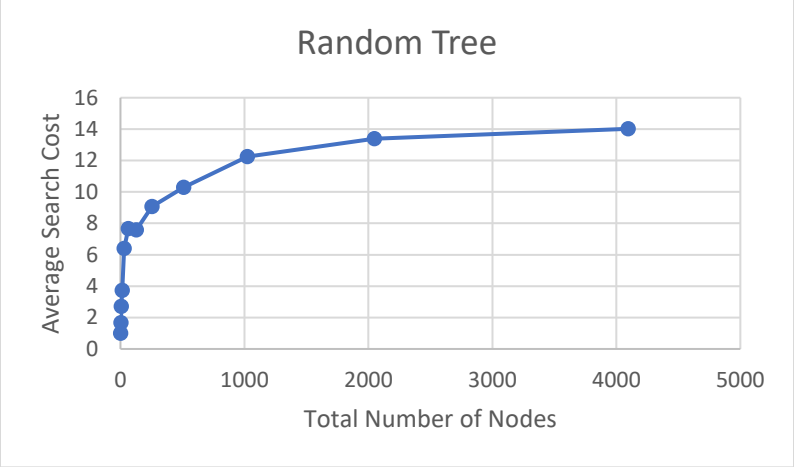
In a linear tree, the height of the tree is n-1. With the given formula, the total cost is shown as n(n+1)/2. After dividing by n subproblems, it would give (n+1)/2, which is just O(n) when simplified.

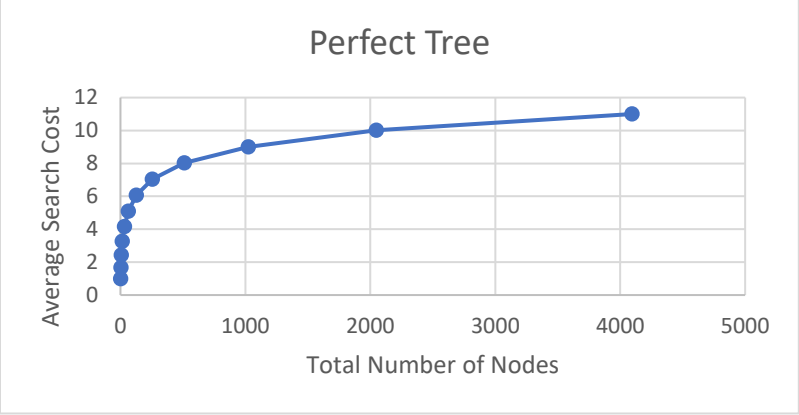
In a perfect tree, the height of the tree is log(n), where there are 2^n nodes in each level of the binary tree. With the formula, it would give 2(log(n + 1) – 1)log(n + 1), which is just O(logn) when simplified.

5. Include a table and a plot of an average search costs you obtain. In your discussions of experimental results, compare the curves of search costs with your theoretical analysis results in item 3.

Nodes	Linear	Random	Perfect
1	1	1	1
3	2	1.67	1.67
7	4	2.71	2.43
15	8	3.73	3.27
31	16	6.39	4.16
63	32	7.67	5.1
127	64	7.59	6.06
255	128	9.07	7.03
511	256	10.3	8.02
1023	512	12.25	9.01
2047	1024	13.4	10.01
4095	2048	14.02	11







The experimental results are quite representative of the theoretical analysis given earlier. The random and perfect trees are very close to each other, thus proving that both have a time complexity of O(logn). In addition, the linear tree is also indeed linear, thus proving that it has a time complexity of O(n).