CSCE 221 [redacted]
PA3 [redacted]

First Name: Cristian Last Name: Avalos [redacted]

[redacted]

Type of Sources			
People	Friends (discussion)		
Web pages (provide URL)	http://www.cplusplus.com/reference/list/list/		
Printed material			
Other Sources			

I certify that I have listed all the sources 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."

Cristian Avalos March 31, 2020

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).

   The objective of this assignment is to create a binary tree in C++ that will be created be reading in data from a file and then outputting information about the tree, such as size and average search cost, and print out the tree itself. My program consists of three files:

   - BSTree.h
   - BSTree.cpp
   - BSTree_main.cpp
   - makefile

   The program can compile and run using the following commands:

   - make all (a command that will prompt the makefile to compile the programs)
   - ./run-bss (will run the program)

   In the file BSTree.h, there are two structures.

   (a) Node
   - This struct is essentially the foundation of the tree. Without nodes you cannot create a tree
   - This struct creates a Node with the following values:
     i. It contains a constructor that will create the Node
     ii. An integer value named ”value” that will hold the value of that specific node
     iii. An integer value named ”search_time” that will keep track of the search time of the binary search tree
     iv. Two *Nodes, named ”left” and ”right”, that will essentially be the children of the nodes
   - All of these components will create a node, or leaf, for the tree

   (b) BSTree
   - This is the largest struct and is responsible for creating the binary search tree
   - This is created by Nodes coming together
   - It contains many functions that can be used for the tree
   - Because of how extensive this struct is, I will only name a few
     i. Private variables
       A. An integer value named ”size” that will keep track of the size of the tree
       B. A Node named ”root” that will be the root, or foundation, of the tree

C. A total of six helper functions that I created in order to assist me in creating the other functions in the Public section of the struct

ii. Public variables

A. This section contains constructors, a destructor, copy assignments and move assignment for the tree

B. A total of 10 functions that can be used in order to get information about the tree, such as: getsize(), insert(), search(), get_total_search_time(), etc.

(c) Lastly, it contains input and output operators

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).

   In this programming assignment I created a binary search tree consisting of nodes and a set of tree operations. A binary search tree is a binary tree that contains nodes in which each node has an associated value that assigns an identity to it that will assign smaller values to the left of a node and larger values to the right of it. My implementation does this by having a node with a left and right child, that are used in order to create the binary search tree. I have functions inside of the BSTree that will insert values into the tree accordingly that will satisfy the definition of a binary search tree.

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.

   The way that search cost was kept track of is simple. This was because of the recursive insert function that I implemented. Since each node can store a value and a search time, every time that a node is inserted it automatically increments the search time for that specific node based on its position in the tree. If a node needs to go two levels deep then the search time would increment twice for that node, so that would be 3 because the root has a search time of 1. If a node needs to be inserted a hundred levels in then the same thing will apply. To find the search cost of a tree is also simple. It is a recursive function that starts from the root and goes through the tree and adds up the search times for every node. A size of the tree is also kept track of and it increments every time a node is inserted using the insert function. To find the average search time a function is called and that function returns the search time for the tree divided by the size, assuming that the size is not zero. Calculating the search cost of an individual node is not challenging. If it is in the tree you simply iterate through a path to get to it. This has a Big-O of $O(log_2(n))$. If you want to find the sum of every node in a tree you will have to traverse throughout the whole tree in order to calculate the total search cost, this causes the Big-O of this function to be $O(n)$ in the worst case.

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). $\sum_{d=0}^{log_2(n+1)-1} 2^d(d+1) \simeq (n+1) \cdot log_2(n+1) - n$ and $\sum_{d=1}^{n} d \simeq n(n+1)/2$

   I couldn't show that the first is true. I'll try the second one though.

   $\sum_{d=1}^{n} d \simeq n(n+1)/2$

   Assume $\sum_{n=1}^{n} n = n(n+1)/2$

   Show $\sum_{n=1}^{n+1} n = (n+1)(n+2)/2$

   $\sum_{n=1}^{n} n + (n+1) = (n+1)(n+2)/2$
   $n(n+1)/2 + n + 1 = (n+1)(n+2)/2$
   $(n^2+n)/2 + 2(n+1)/2 = (n+1)(n+2)/2$
   $(n^2+n)/2 + (2n+2)/2 = (n+1)(n+2)/2$
   $(n^2+3n+2)/2 = (n^2+3n+2)/2$
   True.

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 4.
   The following data was gathered from the provided files and outputted into an excel sheet. From there, I created the table below and created the graphs down below.

Table 1: Nodes and Average Search Time

Nodes	Linear	Perfect	Random
1	1	1	1
3	2	N/A	1.66667
7	4	2.42857	2.71429
15	8	3.26667	3.73333
31	16	4.16129	6.3871
63	32	5.09524	7.66667
127	64	6.05512	7.59055
255	N/A	7.03137	9.06667
511	256	8.01761	10.3033
1023	512	9.00978	12.2463
2047	1024	10.0054	13.3972
4095	N/A	11.0029	14.0237

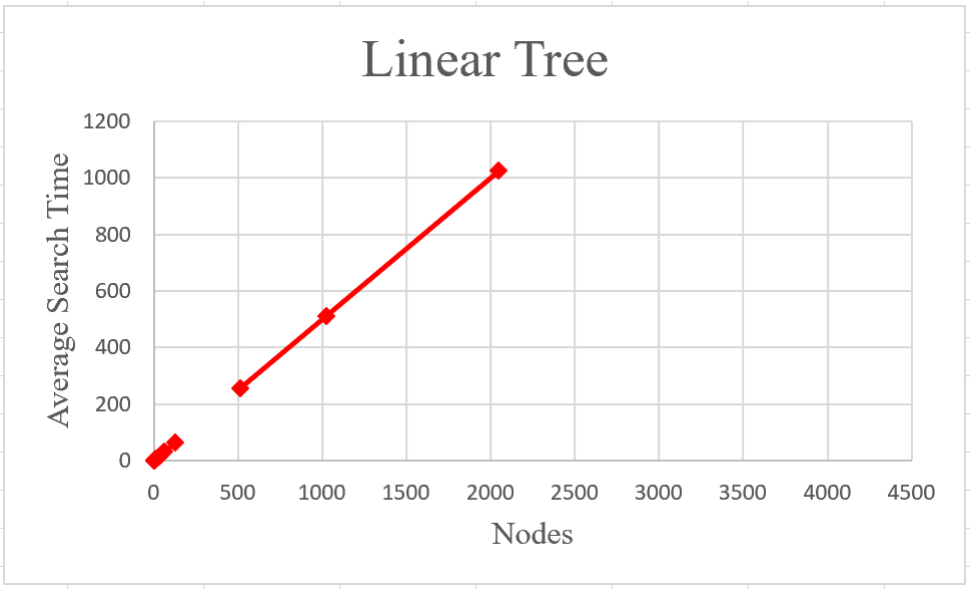

Figure 1: Graph for Linear Tree

This graph behaved as expected. The theoretical analysis states that the big-o of a linear tree is $O(n)$. Based on the image above, I would say that it is pretty linear.

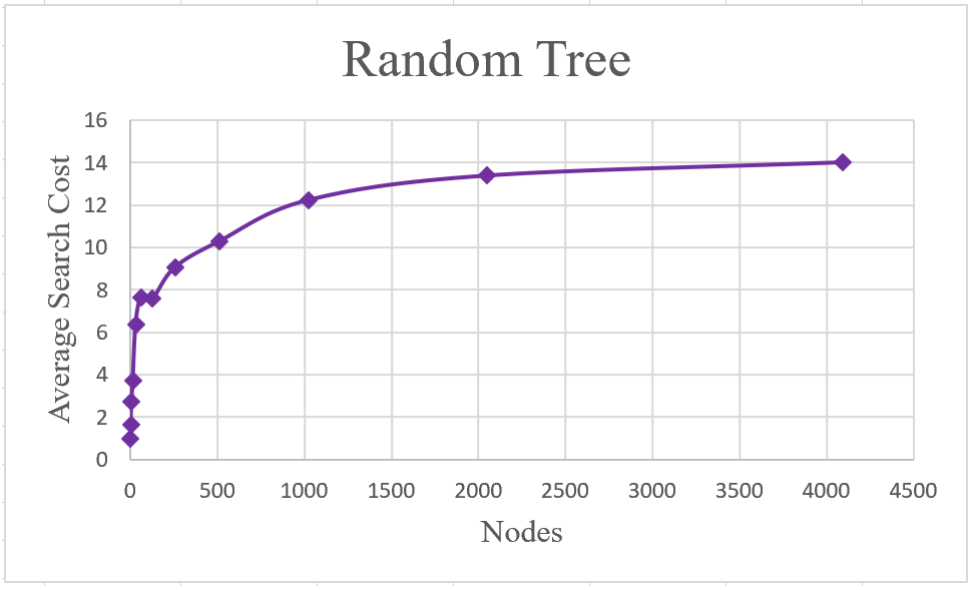

Figure 2: Graph for Random Tree

The random tree graph is very similar to a perfect tree.

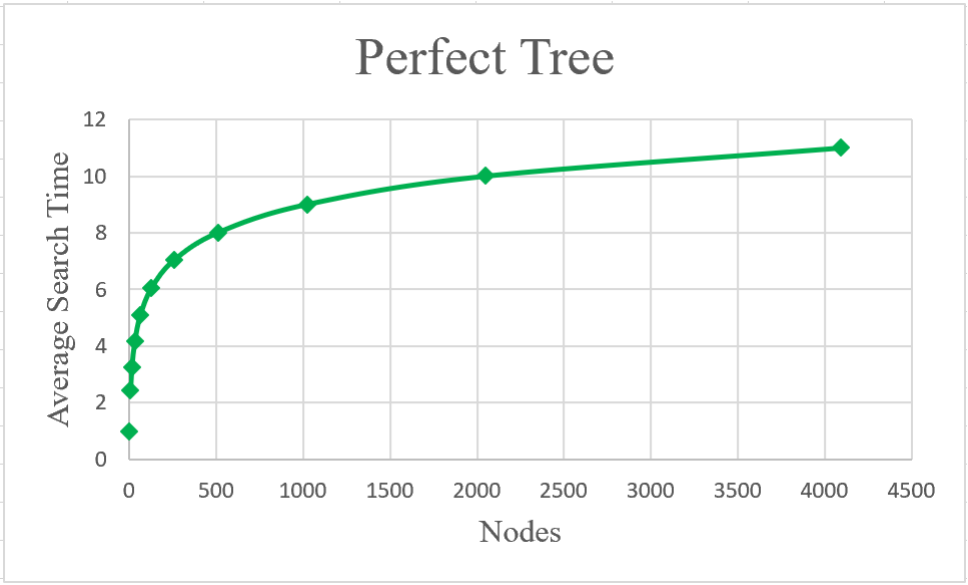

Figure 3: Graph for Perfect Tree

This graph also behaved as expected. The theoretical analysis states that the big-o of a perfect tree is $O(log_2(n))$. From the image above, it looks like a logarithmic function.