CSCE 221 Cover Page

First Name Clayton Last Name Kristiansen UIN 328003173

User Name kristiansenc E-mail address kristiansenc@tamu.edu

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

Type of sources				
People	none			
Web pages (provide URL)	none			
Printed material	none			
Other Sources	none			

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.

Your Name Clayton Kristiansen Date 03-28-2021

Report (20 points)
Write a brief report and submit it to Canvas. The report should include the following sections:

1. A description of the assignment objective, how to compile and run your program, and an explanation of your program structure (i.e. a high level description of the functions or classes in your code).

   The objective of this assignment was to create a Binary Search Tree implementation. This implementation was not supposed to be self balancing, but had to include basic insert, search, search cost, and print functionality. The insert function used a recursive method to find the correct place in the tree for inserted node. The search function operated in a similar way to the insert function in that it recursively searched for a certain value by branching left if the desired value was less then the current node, or branching right otherwise. This would continue until either the current node was equal to thedesired value, or the end of a branch was reached. The search cost method simply traveled to each noderecursively and assigned a cost value of 1 plus depth to each node it visited. The inorder print method used an inorder traversal recursive function to print each node in its place in the recursion. Finally, the print level by level method used an interative method and a queue to print the tree level by level with emtpy space represented by X.

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 in this assignment is a Binary Tree. This means that data is organized in a form where each node has two child nodes, who in turn may have child nodes of their own, or nullptr in place of actual data. This format means that there is a root node, that branches out into every node. The data is inserted into this tree structure such that a nodes left child is always less, and its right child is always greater. In this way, data can be searched for by comparing the desired value to the current node and recursively continuing through the appropriate child. If balanced correctly, this means that a binary tree can have an average search time of logn. The Binary Tree structure created in this assignment also kept track of the search costs of each node. Each node kept track of its cost. This value was updated when a node was looked up or the Tree was being moved or copied.

3. A description of the implementation of (a) individual search cost and (b) average search cost. Analyze the time complexity of the functions that (a) calculate the individual search cost and (b) sum up the search costs over all the nodes. The recurrences/runtime functions should be from your functions (insert() for an individual search cost).

   (a) The insert function has a recurance relation of:

   $$T(n) = T(n/2) + O(1) \tag{1}$$

   This means that, following the general form of this relation, the insert function is O(log(n)).

   (b) The update search costs function goes to each node and calculates its respective search cost to be 1 + its depth. To calculate the average search cost, the only extra step would be modify the update search costs function to recursively add up the search costs of each node and then divide the total resuly by the number of nodes. This process would not change the time complexity of this function which is O(n) because it is required to visit each node exactly once.

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). To be clear, part 3 asks you to analyze the running time of the functions implemented to compute the individual and average search cost, while here you must analyze the asymptotic behavior of the values of the search cost itself (Hint: depth of the tree affects the individual search cost.)

   Formula for perfect binary tree: $\sum_{d=0}^{\log_2(n+1)-1} 2^d(d+1) \simeq (n+1)\cdot\log_2(n+1) - n$

   Formula for linear binary tree: $\sum_{d=1}^{n} d \simeq n(n+1)/2$

   where d represents the depth of the tree.

The formula for the perfect binary tree clearly shows that, for a tree with n elements, the total search cost for every node is:

$$(n+1) * log(n+1) - n \tag{2}$$

To find the average search cost function, the total search cost function must be divided by n. The result of this is:

$$log(n+1) + log(n+1)/n - 1 \tag{3}$$

Converting this function to its Big-O yields O(log(n)) for perfect binary tree.

The formula for the linear binary tree clearly shows that, for a tree with n elements, the total search cost for every node is:

$$n(n+1)/2 \tag{4}$$

To find the average search cost function, the total search cost function must be divided by n. The result of this is:

$$n/2 + 1/2 \tag{5}$$

Converting this function to its Big-O yields O(n) for linear binary tree.

5. Use BSTree_main.cpp to run your code to analyze the data files provided. In case you are unable to compile and run code outside of Mimir, contact your TA for assistance.

   (a) The files *1p*, *2p*, ..., *12p* contain $2^1 - 1$, $2^2 - 1$,..., and $2^{12} - 1$ integers respectively. The integers make 12 **perfect binary trees** where all leaves are at the same depth. Calculate and record the average search cost for each perfect binary tree.

   (b) The files *1r*, *2*r, ..., *12r* contain same number of integers as above. The integers are randomly ordered and make 12 **random binary trees**. Calculate and record the average search cost for each tree.

   (c) The files *1l*, *2l*, ..., *12l* contain same number of integers as above. The integers are in increasing order and make 12 **linear binary trees**. Calculate and record the average search cost for each tree.

   (d) Include a table and a plot of the average search costs you obtain. The x axis should be the size of the tree and the y axis should be the average search cost. In your discussions of experimental results, compare the curves of search costs with your theoretical analysis that is derived above.

```
average search time linear 1 1
average search time perfect 1 1
average search time random 1 1

average search time linear 2 2
average search time perfect 2 1.66667
average search time random 2 1.66667

average search time linear 3 4
average search time perfect 3 2.42857
average search time random 3 2.71429

average search time linear 4 8
average search time perfect 4 3.26667
average search time random 4 3.73333

average search time linear 5 16
average search time perfect 5 4.16129
average search time random 5 6.3871

average search time linear 6 32
average search time perfect 6 5.09524
average search time random 6 7.66667

average search time linear 7 64
average search time perfect 7 6.05512
average search time random 7 7.59055

average search time linear 8 128
average search time perfect 8 7.03137
average search time random 8 9.06667

average search time linear 9 256
average search time perfect 9 8.01761
average search time random 9 10.3033

average search time linear 10 512
average search time perfect 10 9.00978
average search time random 10 12.2463

average search time linear 11 1024
average search time perfect 11 10.0054
average search time random 11 13.3972

average search time linear 12 2048
average search time perfect 12 11.0029
average search time random 12 14.0237
```

Figure 1: Results

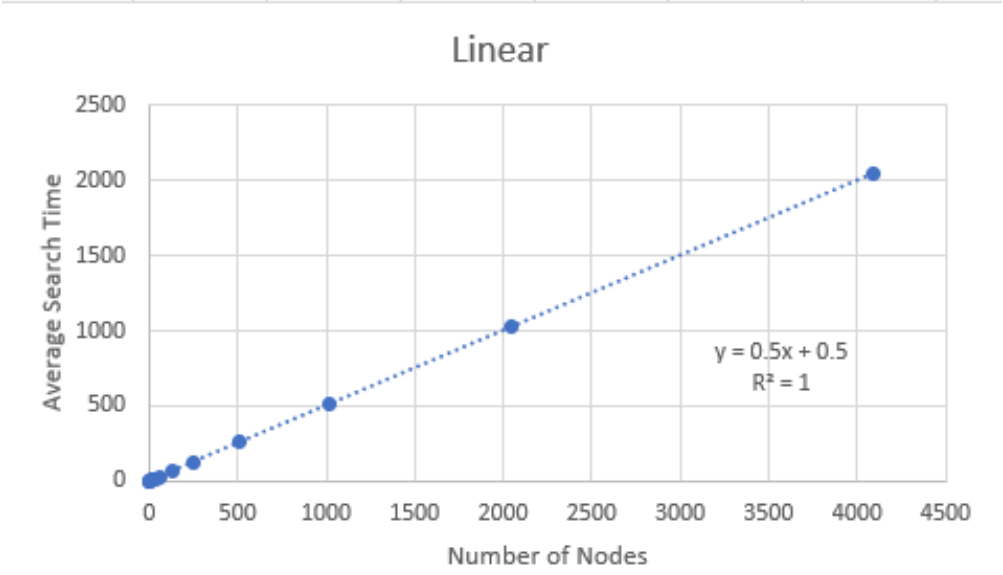

Figure 2: Linear costs graph

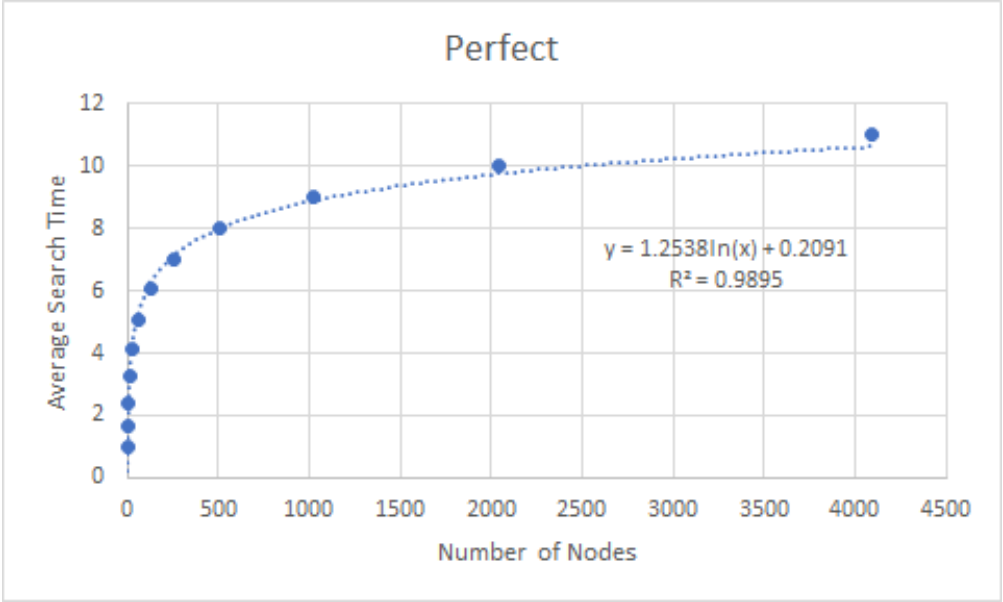

Figure 3: Perfect costs graph

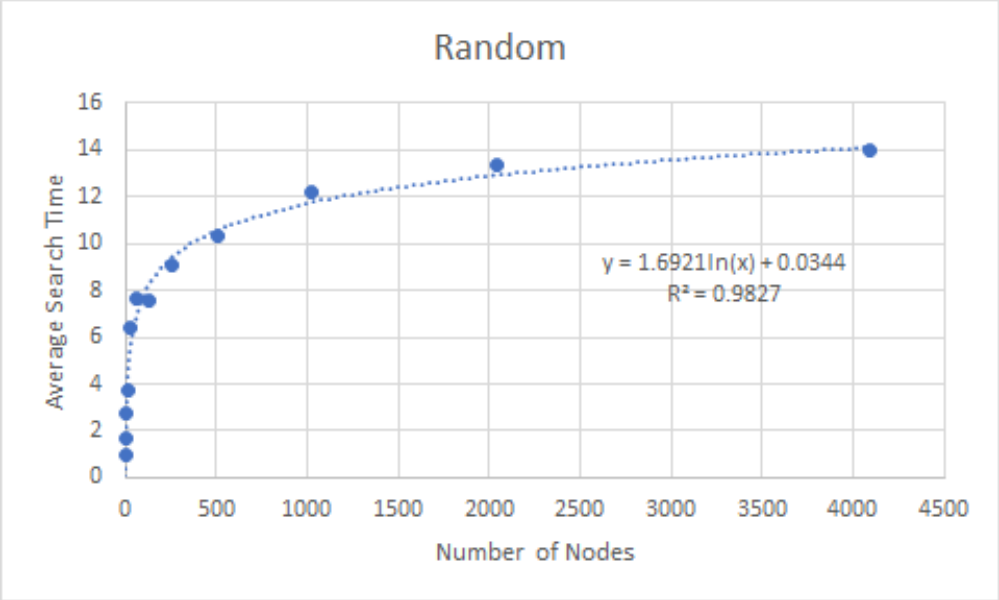

Figure 4: Random costs graph

The theorectical analysis determined that the average search cost for a linear Binary Tree should be O(n). This is supported by the data and the graph as it can be seen that a linear equation fits the data with an r value of 1.

The analysis also determined that the average search cost for a perfect Binary Tree should be O(log(n)). This is supported by the data and the graph since the plot of search costs vs size shows a logarithmic relationship.

The random Binary Tree also showed a logarithmic average search cost growth function. This makes sense as in previous assignments we have derived the average search cost for Binary Trees and they have been O(log(n)).