CSCE 221 Cover Page
PA 4

First Name: Mitesh Last Name: Patel UIN: 124002210

User Name: patel221881 E-mail address: patel221881@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 in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources				
People	Ta	HRBB peer teachers		
Web pages (provide URL)	listed below			
Printed material	data structures and algorithm book			
Other Sources	csce 221 tree slides			

http://www.cprogramming.com/tutorial/lesson18.html
http://www.cplusplus.com/forum/general/1551/
http://www.cplusplus.com/forum/general/166187/
http://www.sanfoundry.com/cpp-program-implement-binary-tree-2/
https://en.wikipedia.org/wiki/Brute-force_search
http://algs4.cs.princeton.edu/32bst/

I certify that I have listed all the sources that I used to develop the solutions/code 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 (signature) Mitesh Patel Date 11-1-15

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).

(a) Objective - The objective of this assignment is to create a binary search tree, and use traversals to print them out in such a way in addition to removing a node. We are to create search costs for each node in addition to printing the tree level by level using file functions.

(b) How to compile and run the program. To compile simply use g++ -std=c++14 *cpp. To run: ./a.out . NOTE: my program gives segmentation fault when outputing level by level to the screen (therefore it is commented out), but it works fine on my xCode. Most likely will be seeing you after assignment is graded to show you.

(c) Explanation of program structure: My program is very similar to the one in the slides. I used the BinaryNode class, and the BinarySearchTree class, The BinaryNode class has a friend of the BinarySearchTree. In the BinarySearchTree class I am using BinaryNode to create a root, an insert function, a findmin function, removemin function, and a remove function. In the BinaryNode class I added a int search cost integar and a int key. I deleted the getelem, size, height, and copy function. In the BinarySearchTree class I added an int to store the total nodes, and added functions mentioned above. I added a function to resetSearchcost , and the in order, pre order, and post order traversals. I deleted the destructor, size, height, remove, and find functions and modified insert to set total cost to 0. and added in , post, pre recursive functions. Also added a function outputlevelbylevel which gives segmentation fault on linux server, but works fine on my xcode, and will most likely show you that it works. Also added another function to ssave the input level by level into a file.

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).

   (a) The data structure I created was a binary search tree. I also used a queue to output level by level. A binary search tree (BST) is a binary tree where each node has a Comparable key (and an associated value) and satisfies the restriction that the key in any node is larger than the keys in all nodes in that node's left subtree and smaller than the keys in all nodes in that node's right subtree. The data arrangement was made possible by the use of the BinaryNode class. I also used vectors to help organize the input data from the file so it would be easier to insert into the three.

3. A description of how you implement the calculation of (a) individual search cost and (b) average search cost and (c) updated search cost. Is the implementation associated with the operations find, insert, remove? Analyze the time complexity of (a) calculating individual search cost, (c) updating the search cost of an individual node and (b) summing up the search costs over all the nodes.

   (a) For individual search cost, I implemented it by making another parameter for search cost in the insert function of the BinarySearchTree, I increment it whenever insert is called in the main, then after it is called it will eventually reach a recursive call in which it will increment if there is one or more data in the tree. Time Complexity : O(n) Worst, O(logn) Best.

   (b) For the average search cost, I take the sum of the search costs (totalCost), and I divide it by the number of nodes in the tree (totalNode). Summing up part is done in preorder traversal function in binary search tree. Time complexity of summing up search costs over all the nodes : O(n)

   (c) As for updating the search cost after removal, when the remove function is called it sets root = to another remove function defined in the binarysearchtree cpp file. In this removal we remove the node, and reset the search cost which is defined in the cpp. Time complexity for updating search cost: O(n)

   (d) Yes, the implementation is associated with the operations insert and remove.

4. Give 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 integers).

   (a) For Linear tree: the height of the tree is n-1, and using the formula we know total cost is n(n+1)/2 now dividing by n subproblems we get (n+1)/2 which is O(n).

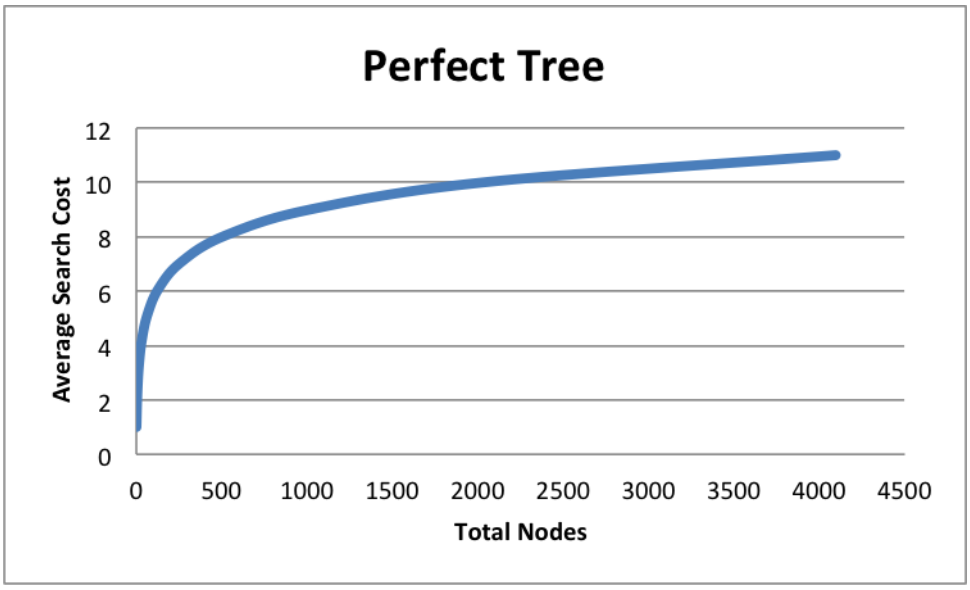
(b) For Perfect tree: the height of the tree is logn. There are 2^n nodes at each level of the tree so the search time would be log(1+1) + 2(log(2+1)) + 2^2log(3+1) +...+ $2^{log(n+1)-1}log(n+1)$ which by using the formula is equal to (n+1)(log(n+1)-n) which is O(logn).

5. Include a table and a plot of average search costs you obtain. In your discussions of the experimental results, compare the curves of search costs with your theoretical analysis results in item 4.

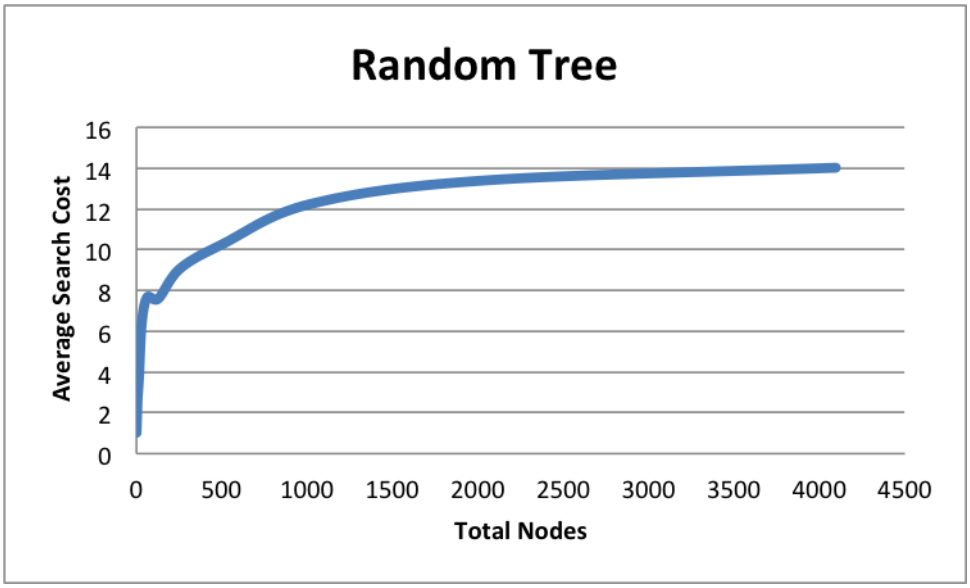
(a)

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

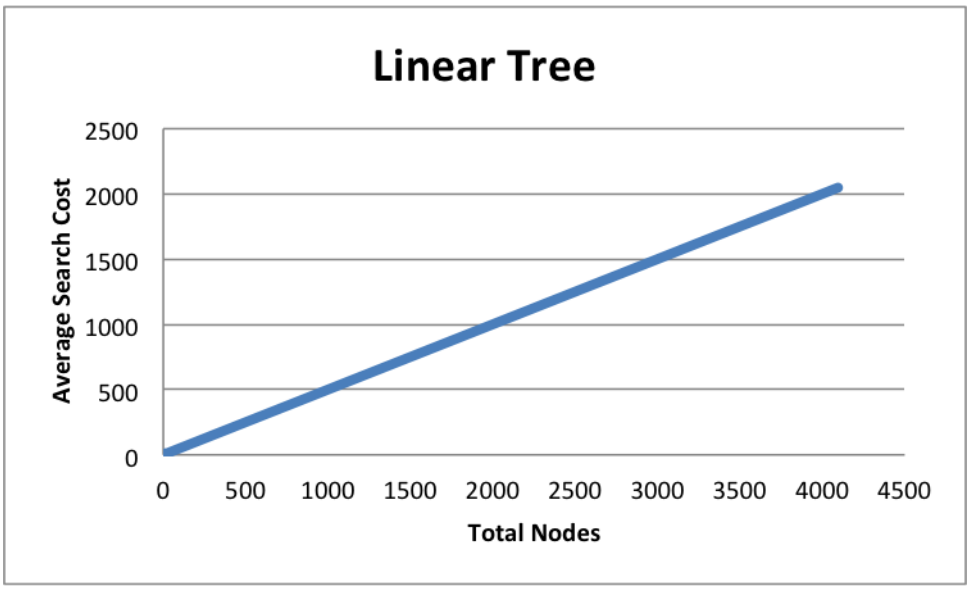
(b)



(c)



(d)



(e) As we can see the Linear tree is truly linear hence the O(n) time complexity is correct. The random and perfect trees are very very similar, and proves the O(logn) time complexity.

6. Input and Output Specifications

(a) Cases in which your program crashes because of wrong input (eg, wrong file name, letter instead of number, or the program expects 10 items and it only finds 9.) : My program will crash if you input a file that has less than 16 nodes, and once you remove a node it removes and prints it in order, but it gets a segmentation fault when trying to print level by level on linux. However, on my xcode it works fine. (NOTE: I have commented out the part where I get the segmentation fault in the main.)

(b) Cases of wrong input that you catch with Exceptions. : If the user inputs a file name that is not in the directory.