CSCE 221 Cover Page
Programming Assignment #4

First Name: Devin Last Name: Tuchsen UIN: 121000270

User Name: devint1 E-mail address: devint1@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	
Web pages (provide URL)	http://courses.cs.tamu.edu/teresa/csce221/ cplusplus.com http://en.wikipedia.org/wiki/Binary_search_tree stackoverflow.com http://leetcode.com/2010/09/printing-binary-tree-in-level-order.html
Printed material	
Other Sources	

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) Date

Problems

Perfect Tree Average Cost

1p	2p	3p	4p	5p	6p	7p	8p	9p	10p	11p	12p
1	1.66667	2.42857	3.26667	4.16129	5.09524	6.05512	7.03137	8.01761	9.00978	10.0054	11.0029

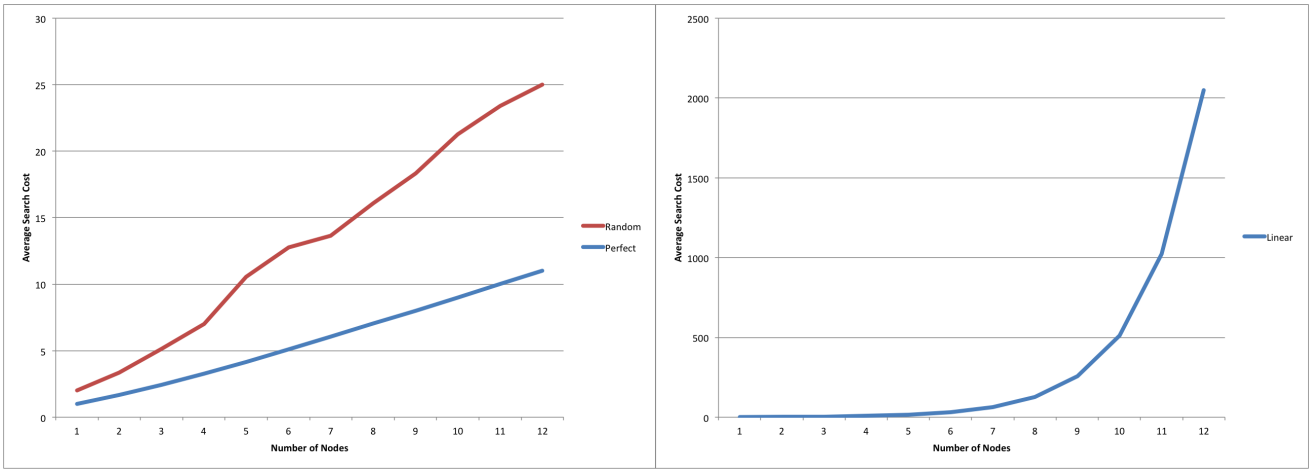
Random Tree Average Cost

1r	2r	3r	4r	5r	6r	7r	8r	9r	10r	11r	12r
1	1.66667	2.71429	3.73333	6.3871	7.66667	7.59055	9.06667	10.3033	12.2463	13.3972	14.0237

Linear Tree Average Cost

1l	2l	3l	4l	5l	6l	7l	8l	9l	10l	11l	12l
1	2	4	8	16	32	64	128	256	512	1024	2048

Plots



Output can be found in out1.txt (for original) or out2.txt (one element removed)

Report

The objective of this assignment was to explore the binary tree data structure and its traversals. To compile and run the program:

1. In a Unix terminal, change the current directory by typing `cd PATH_TO_SOURCE`, where `PATH_TO_SOURCE` is the path to all the source files.
2. Type `make` then press enter.
3. Type `./binarytree` then press enter.
4. The program will ask for a file. You can specify a file path, such as `test-files/4p`.
5. The program will output the given file to the screen, followed by the binary tree (or a file if the size of the tree is greater than 16). It will then ask for an element to remove. Type an integer and press enter.
6. The program will output the new binary tree, either to a file or the screen.

This program uses two classes: `BinaryTree` and `BinaryNode`. `BinaryTree` represents one binary tree as a whole, while `BinaryNode` represents a single node on a tree, with links to the left and right nodes.

The data structure in this program is based on a binary tree, although some elements of a binary search tree are also present. A binary tree data structure technically only needs one data member, the root. The root would then contain pointers to a left and right node, and each node would also contain these pointers. In this example there are several helper functions to perform insertion, output, and node deletion, to name a few. Insertion and deletion use pre-order tree traversal. Output uses a queue to achieve level-in-order output.

Individual search cost is calculated when an element is inserted into the binary tree, using the number of comparisons. Each node therefore has an individual search cost equivalent to its depth + 1. Finding the average search cost relies on a pre-order traversal to sum up the total search cost of all nodes in a tree, then returns the average by dividing this by the tree's size. These require $O(log_2 n)$ operations, where n is the number of nodes in the binary tree.

The formulas $\sum_{d=0}^{log_2(n+1)-1} 2^d(d+1) \simeq (n+1) \cdot log_2(n+1) - n$ and $\sum_{d=1}^{n} d \simeq n(n+1)/2$ represent the sum of all search costs in a binary tree. Dividing these by n gives us equations that are $O(log_2 n)$ and $O(n)$ for perfect and linear trees, respectively.

From the data obtained, one can see that for perfect and random trees, the average search cost is roughly linear, $O(n)$. For linear trees, it is exponential, $O(2^n)$.