CSCE 221 Cover Page

Assignment #4

First Name Alan Last Name Achtenberg UIN 321000608

User Name alan76 E-mail address alanachtenberg@neo.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				
Web pages (provide URL)				
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.

Your Name Alan Achtenberg Date 10/25/2013

Assignment 4

due oct 25 at 11:59 pm.

Program Description:

Construct a binary tree from file input and output total number of nodes and average search cost of tree in a file.

File input and output is specified on the command line with the -f option for example. -f 3p.txt

Purpose:

To learn how to inplement and use recursive functions as well as determine the ability and usefullness of the binary tree data type.

Data Stuctures:

Binary Tree consisting of a BinaryNode root and a set of tree operations. BinaryNode consisting of two integers containing element and search cost of the element as well as two node pointers to other nodes that are the childeren left or right.

Algorithms:

Calculation of the search cost can be definied simply as follows height(root)-height(node)+1 or it can alsi be caluculated as the number of recursive calls of insert during creation of the list. The second method was perfered during creation because the height of the root is constantly changing and the recursive calls cause the lower values to be overwritten. But when recalculation of the ssearch cost is required the first method is better. The height() function is O($\log_2$n)

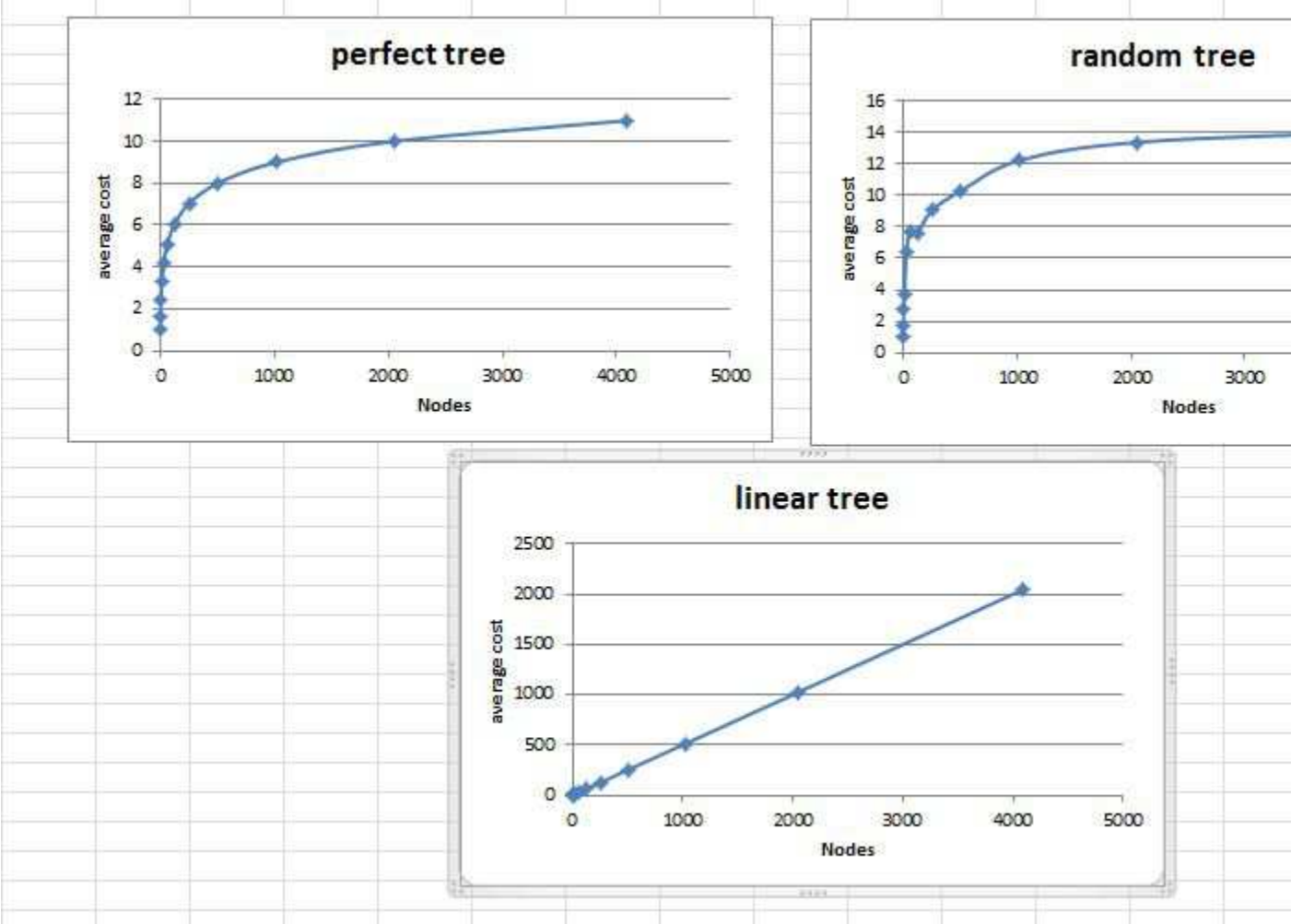
Average search cost is simply determined by summing the number of nodes and total search cost as you traverse the tree, then dividing the latter by the former to determine the Average. Total calculation time can be considered O(n) because it is 2n sums + 1 divide operation.

For the random and perfect case Update and calculate new search cost average can be described as O(n) because it is $\log_2$n to find the node +2$\log_2 n$ to recalculate search cost and 2n+1 to find average therefore the log functions are bounded by the linear summation.

Search cost for the perfect binary tree is always O($\log_2$n) and due to pigeon hole theory there must be a search cost for every integer between 0 and $\log_2 n$. therefore the average search cost is approx $(n+1)(log_2(n+1)) - n$ or the sum from 0 to $log_2(n+1) - 1$of $2^d(d+1)$

Search cost for the linear binary tree is always O(n) therefore its average is the sum from 1 to n which is approx $\frac{(n+1)n}{2}$

As shown in the graphs below the experimental results closely follow the theoretical.



Program:

The program constructs a binary tree based off of an input file that is read in the command line (defaults to

input 3p) output is then directed to results/filename.txt. during the recursive construction of the tree each node is

assigned its correpsonding search cost. After constuction of the binary tree the program then request the user to

remove an int from the binary tree. The program then rearanges the tree elements without changing the position of

the nodes so that the calculation of the search cost does not need to be done except for the special linear case.

Because of sheer size of the output files. The program then only outputs the trees to the files if the have a height

less than 17. everything else only outputs the node count and average search cost.

To compile and run the program without user input and removing a node simply run the shell script by typing

./run.sh this shell script automatically compiles and runs the program with every input file. To test individual

input files and to have the option to remove a node. compile the program with g++-4.7 -std=c++11 main.cpp and

run it with ./a.out -f sample.txt the sample.txt must be in the test_files folder and the output will be in the results

folder of the same name sample.txt

Input specs.

For char input on the yes or no question simply input a lower case y or n otherwise the program continues to

ask for inputThe number to remove must be greater than 0 and less than 100,000 for simple error checking, but it

also must contain an existing elem. the program simply outputs not found and continues in this case.

For command line input the file option must be as such -f sample.txt.