Program Description / Organization / Purpose & Data Structures Description

The challenge in this assignment is to read in a list of data, transform that data into a binary tree, and to perform various operations on this tree. These operations included different ways of traversing the tree, calculating size, calculating search cost per node, calculating average search cost, and outputting the tree to a file or to the console window.

I made a binaryNode class which was implemented similarly to a linked list. Each node had pointers to left and right children nodes (I also included a pointer to the parent node), a Key which held an integer value, and a search cost value which would hold the value of node height + 1. In relation to the parent node, the left child node is less than the parent node, while the right child node is greater than the parent node.

The average search cost was calculated by using the pre order traversal technique to get the search cost of each node, which was added to a total counter. This number was divided by the number of total nodes in the tree. Both operations took the root node as input and would output integer values. The size of the tree was calculated by recursively following left and right subtrees, with each node that was traversed, a counter was increased by 1.

The purpose of this assignment was to understand and implement the binary tree data structure.

Program Organization and Algorithm Description

In summing search cost, the runtime was O(n) since the tree was traversed once where each node's search cost was added to a total counter.

Calculating the individual search cost was O(log2(n)) since when the node was inserted, the program had to search the binary tree for the correct position to insert the node.

Updating the search cost of an individual node was O(n). This process included removing the chosen node, and destruction the binary tree. This action would require analyzing the children nodes of the removed node and finding the smallest value node.

Compile Instructions

Navigate to directory with the source code. compile by executing: g++ a4.cpp
Run the program using the command: ./a.out

I/O Specifications

The program expects integer data from a plain text file. The program will fail if the input data from the file is not in the expected format, or if the input filename is not correct.

Logical Exceptions

No known logical exceptions.

C++ Object Oriented Programming

The STL templated queue class was utilized to implement outputting the tree to the console/file.

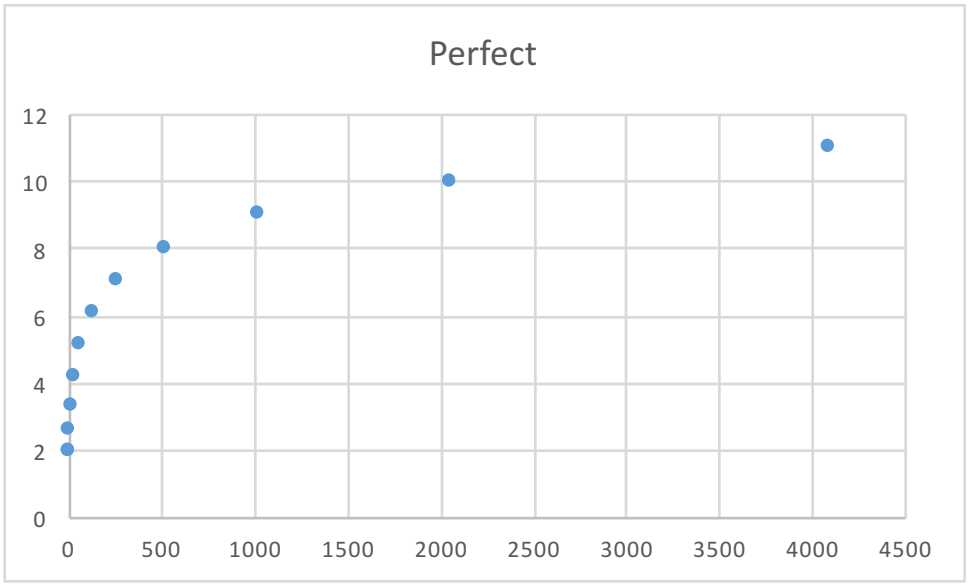
Tests

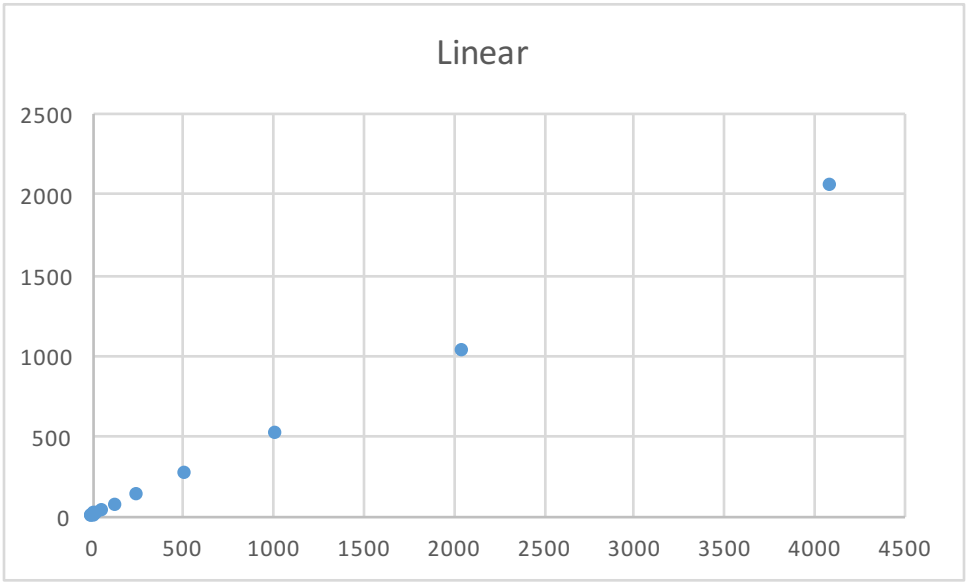
The program was tested with all of the given input data files.

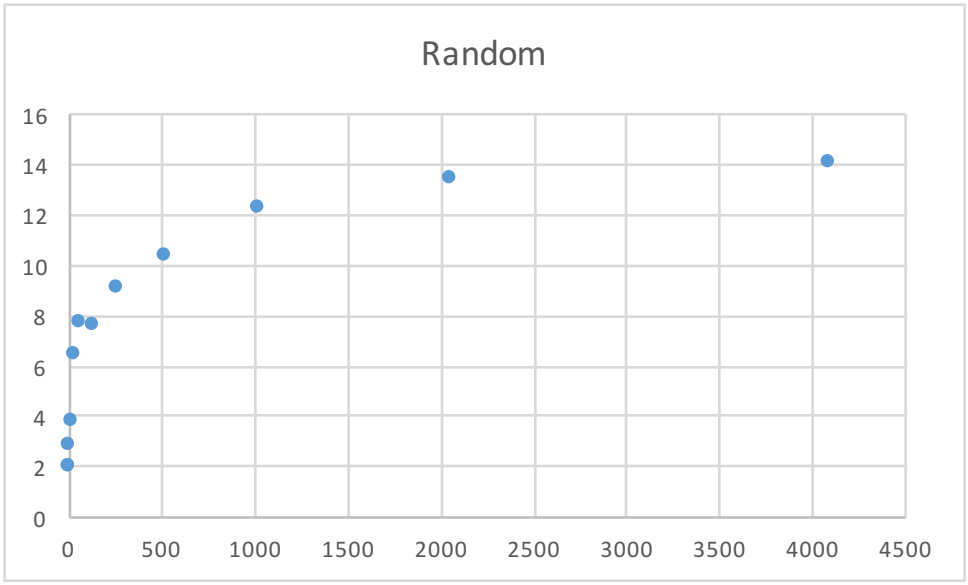
	Perfect		**Random**	
	List	Search Cost	List	Search Cost
1	1p	2	1r	2
3	2p	2	2r	2
7	3p	2.57	3r	2.86
15	4p	3.33	4r	3.8
31	5p	4.19	5r	6.42
63	6p	5.11	6r	7.68
127	7p	6.06	7r	7.6
255	8p	7.04	8r	9.07
511	9p	8.02	9r	10.31
1023	10p	9.01	10r	12.25
2047	11p	10.01	11r	13.4
4095	12p	11	12r	14.02

Linear

List	Search Cost
1l	2
2l	2.33
3l	4.14
4l	8.07
5l	16.03
6l	32.0159
7l	64.01
8l	128
9l	256
10l	512
11l	1024
12l	2048


Perfect
12
10
8
6
4
2
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500


Linear
2500
2000
1500
1000
500
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500


Random
16
14
12
10
8
6
4
2
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500