CSCE 221 Assignment 4 Cover Page

First Name Zhiyang Last Name Zeng UIN 720005338

User Name zhiyangzeng E-mail address zhiyangzeng@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)	stackoverflow.com	cplusplus.com	cppreference	
Printed material	lecture slides			
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 Date 11/1/2015

1. Program Description

The objective of this assignment is to implement a binary search tree data structure. The main.cpp file is dependent on BinarySearchTree header file and implementation, which is dependent on BinaryNode header file and implementation. In addition to the structure provided in the class note, the following functionalities has been added:

- There is a new printTree method in BinaryNode header file to print the tree with sizes<16 to a .txt file.
- Search cost and total cost calculation has been added to the BinaryNode class.
- The definition of find, remove, and insert has been redefined in the private function of BinarySearchTree header file.
- Exeption handling is added.
- Main.cpp file has been added to handle user input and importing external data.

To compile, go to current directory and type "make all". To run, type "./main". Once the program is running, type 1-12(p/r/l) to process the correct perfect/random/linear tree. The program will run forever until user types "quit".

2. Data Structure

A BinaryNode consists of 4 fields. Element is to store the data inside the node, the cost is an added field to calculate the cost for searching its position when building a tree. A left and right pointer to elements smaller and larger than its element. TreeOperation is a class for handling printing BinaryNodes and adding up the total build cost. A BinarySearchTree is made of interconnected BinaryNodes. The root is the first node of the tree. Every BinaryNode after is first traversed through the root node. A perfect BinarySearchTree can insert, delete, and find individual element at $\mathrm{O}(\log_2 n)$ time. Randomly built BST can also handle the above operations at $\mathrm{O}(\log_2 n)$ time. Linear BST insert, delete, and find elements at O(n) time.

3. Search Cost

The cost for individual node is stored as its private member in the class. The cost for searching is updated by the insert method inside the BinarySearchTree class implementation. After insertion, the individual searching cost is added to a total cost during tree traversals. When printed out, the total cost will be divided by the tree size to get the average search cost. Tree size is calculated by traversal and increment from node to its children/leaves.

4. Theoretical analysis

4.1) Perfect tree

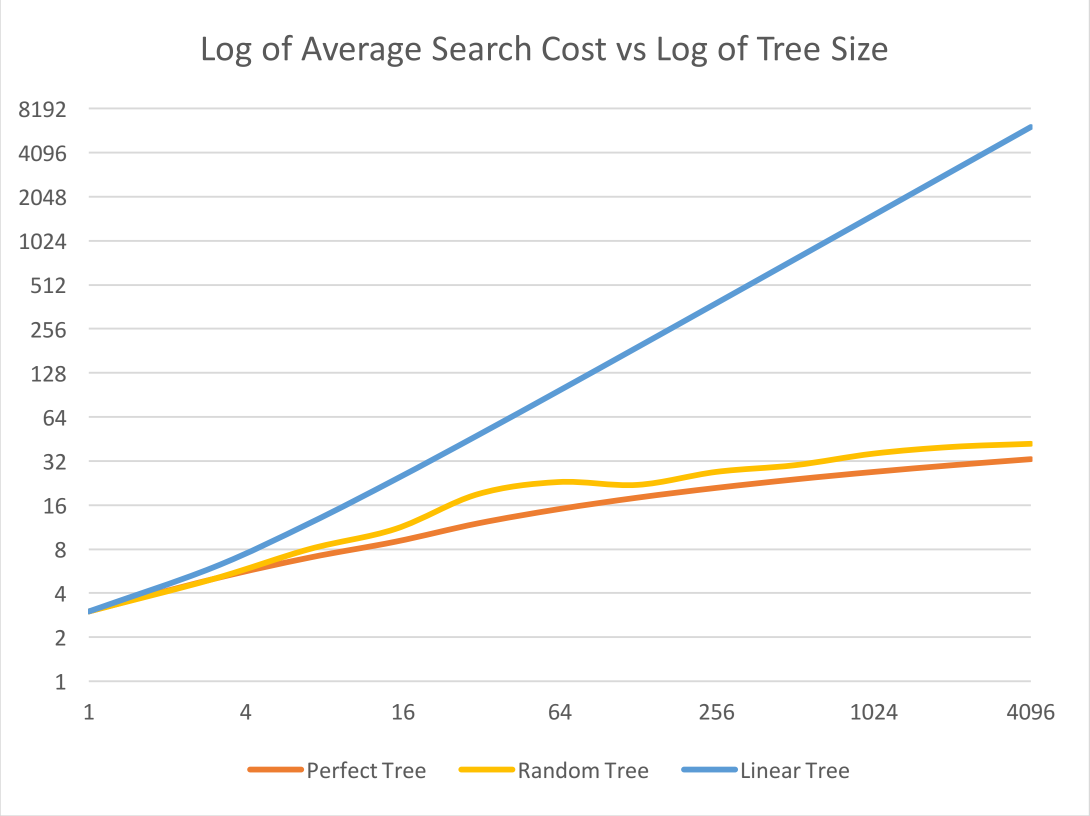
For a perfect tree, the total search cost is $\mathrm{O}(\mathrm{n}\log_2 n)$. The individual/average search cost is then $\mathrm{O}(\mathrm{n}\log_2 n)/n$= $\mathrm{O}(\log_2 n)$

4.2) Linear tree

For linear tree, the total search cost is $O(n^2)$. The individual/average search cost is $O(n^2)/n = O(n)$.

5. Experimental Analysis

Size	Average Cost-Perfect	Average Cost-Linear	Average Cost-Random
1	3	3	3
3	5	6	5
7	7	12	8
15	9	24	11
31	12	48	19
63	15	96	23
127	18	192	22
255	21	384	27
511	24	768	30
1023	27	1536	36
2047	30	3072	40
4095	33	6144	42



Both axis of the graph is $\log_2 n$ scaled for better representation. It is cleared that both the perfect tree and the random tree have average cost of $O(\log_2 n)$, because their logarithmic graph is relatively linear. While the linear tree has average cost increase at the same rate as the data size, which is O(n). The experimental results coincides with the theoretical analysis.