CSCE 221 Assignment 5 Cover Page

Asa Hayes 525003952

User Name: AsaHayes E-mail address: asahayes@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	Lecture Slides on Red-Black and Binary Trees			

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 Asa R Hayes Date 10 April, 2018

Assignment 5 (40 pts)

Program: Due April 10 at 11:59 pm

1 Description and Details: Red-Black Tree

Provide a brief description, reason for building, and applications of the Red-Black tree data structure and its operations.

A red-black tree is a specific type of binary search tree that has rules to its arrangement that facilitate faster average search times than a regular binary search tree. Aside from the regular rules a binary tree must meet, a red-black tree must also meet four rules:

1. All nodes must be either red or black.
2. The root and all leaves are black.
3. If a node is red, both of its children must be black.
4. Every path from any node to one of its descendant leaves contains the same amount of black nodes.

To keep with these new rules, red-black trees need two additional operations past the regular insert and delete functions. One is to change nodes from red to black to keep the black height, and the other is to rotate nodes when just changing colors will no work. This essentially promotes the node it is used on to root and having the old root and other half of the tree be a child of that new root. Both of these have a runtime of O(1).

2 Search Costs

2.1 Provide the upper bound on individual search cost in a Red-Black and binary search tree in the worst case. Express this cost in terms of big-O notation.

The individual search cost of a Red-Black tree in any case is $O(log_2 n)$ due to the nature of the balancing. A regular binary tree does not have any such balancing, so its worst case search cost can get up to O(n), but with an average/best case of $O(log_2 n)$.

2.2 How can you justify that the computed average search costs for some Red-Black trees is higher than for perfectly balanced binary search trees? Does the formula below provide lower bound on the computed average search costs for Red-Black trees? Justify your answer.

While red-black trees are on average much more balanced than regular binary search trees, they are not necessarily always perfectly balanced. Taking for example the sample tree I did by hand from file 3r, It was able to still be a valid red-black tree while being noticeably left-heavy. Despite all the measures a red-black tree uses to keep balance, it is not always more balanced than a tree with no balancing effort, leaving the search cost higher in the rarer situations like where a red-black tree would be less balanced than a regular binary search tree.

As for the formula, it does cover the lower bound cost of red-black trees ($O(log_2 n)$).

$((n+1) * log_2(n+1)/n) - 1$
$((log_2(n+1) * (1/n)) - 1$
$((O(log_2(n+1)) * O((1/n))) - O(1)$
$O(log_2(n)) > O(1/n)$
$\therefore ((n+1) * log_2(n+1)/n) - 1 = O(log_2(n))$

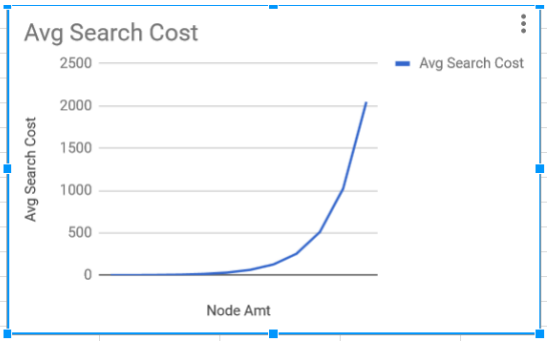
$$\sum_{d=0}^{\log_2(n+1)-1} 2^d(d+1)/n \simeq ((n+1)\cdot \log_2(n+1)/n) - 1$$

3 Tables and Plots

Tables are directly transcribed from terminal after "./RBTree" into tables for better formatting.

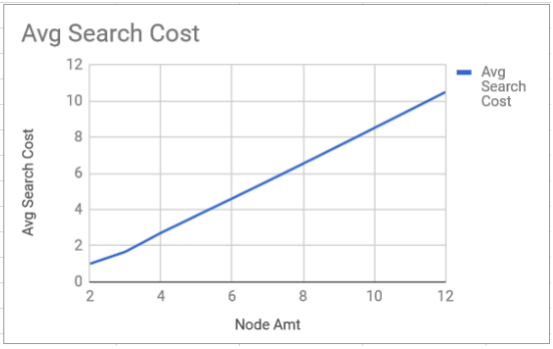
3.1 BTree Linear

	Node Amt	Avg Search Cost
BTree Linear	1	1
	2	2
	3	4
	4	8
	5	16
	6	32
	7	64
	8	128
	9	256
	10	512
	11	1024
	12	2048



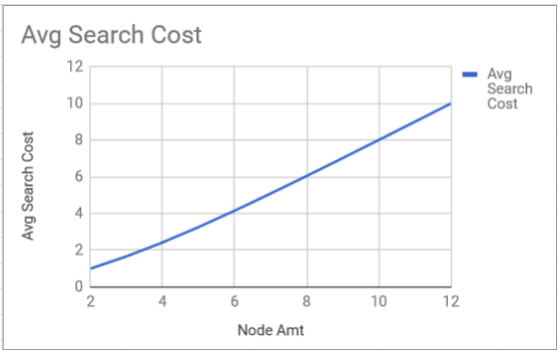
3.2 RBTree Linear

	Node Amt	Avg Search Cost
RBTree Linear	1	1
	2	1.66667
	3	2.71429
	4	3.66667
	5	4.6129
	6	5.57143
	7	6.54331
	8	7.52549
	9	8.51468
	10	9.50813
	11	10.5046
	12	11.5026



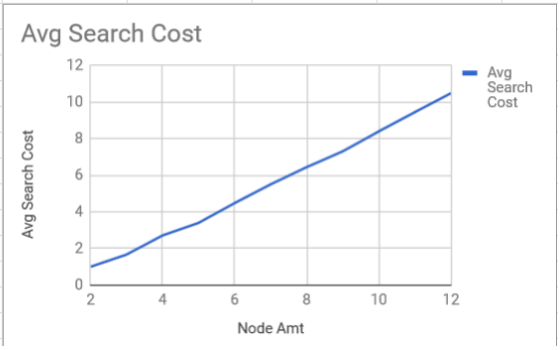
3.3 BTree Perfect

	Node Amt	Avg Search Cost
BTree Perfect	1	1
	2	1.66667
	3	2.42857
	4	3.26667
	5	4.16129
	6	5.09524
	7	6.05512
	8	7.03137
	9	8.01761
	10	9.00978
	11	10.0054
	12	11.0029



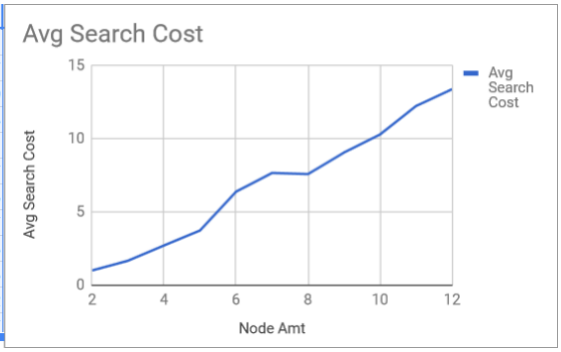
3.4 RBTree Perfect

	Node Amt	Avg Search Cost
RBTree Perfect	1	1
	2	1.66667
	3	2.71429
	4	3.4
	5	4.48387
	6	5.50794
	7	6.44882
	8	7.31373
	9	8.40509
	10	9.45357
	11	10.4919
	12	11.5314



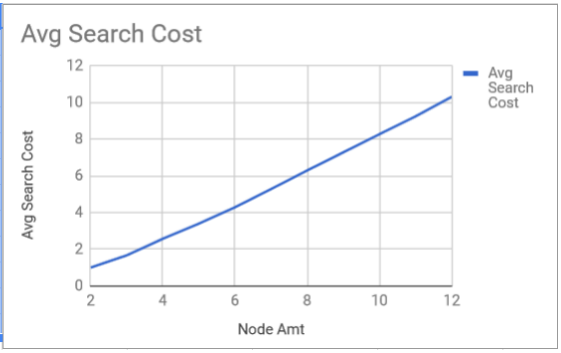
3.5 BTree Random

	Node Amt	Avg Search Cost
BTree Random	1	1
	2	1.66667
	3	2.71429
	4	3.7333
	5	6.3871
	6	7.66667
	7	7.59055
	8	9.06667
	9	10.3033
	10	12.2463
	11	13.3972
	12	14.0237



3.6 RBTree Random

	Node Amt	Avg Search Cost
RBTree Random	1	1
	2	1.66667
	3	2.57143
	4	3.4
	5	4.29032
	6	5.28571
	7	6.29921
	8	7.28627
	9	8.27789
	10	9.25024
	11	10.3166
	12	11.3023



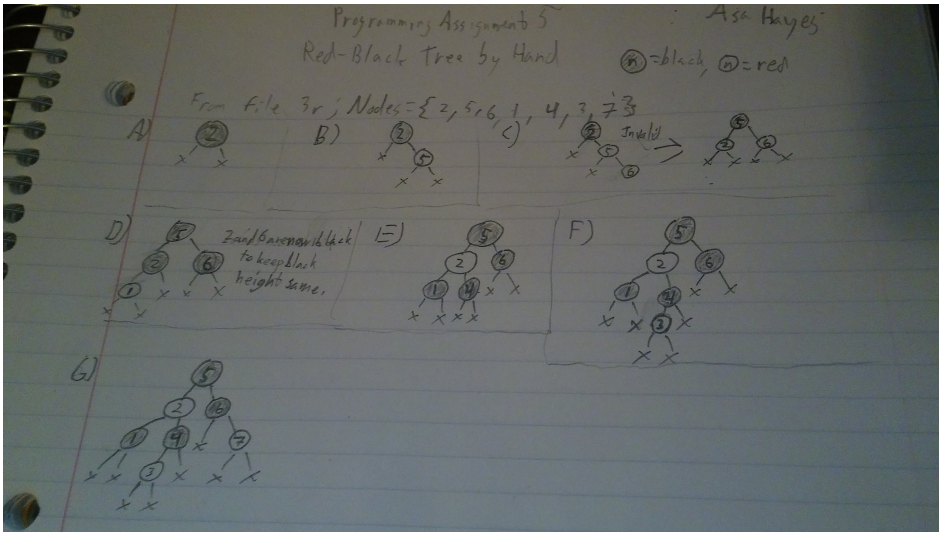
3.7 Conclusion

This fits with the search costs we've learned about. In the worst case (linear), the binary tree is visibly much worse in runtime than the Red-Black tree. For the best case (perfect), the two trees are about the same, which fits with the search costs as well. In the average case (random), they are about equal, as expected, but the binary tree is slightly less efficient from sizes ~5-8.

4 Testing Cases

Include the testing cases for the small input data (the number of nodes less than 16) for the files selected by you in the report.

4.1 Red-Black Tree for file 3r, by hand



4.2 Same Red-Black Tree from "./RBTree -f data-files/3r"

```
[asahayes]@linux2 ~/cs_221/PA5> (18:29:41 04/10/18)
:: ./RBTree -f data-files/3r
custom
BTree average search time       2.71429
2[1]
1[2] 5[2]
X X 4[3] 6[3]
X X X X 3[4] X X 7[4]
RBTree average search time      2.57143
5[1,b]
2[2,r] 6[2,b]
1[3,b] 4[3,b] X 7[3,r]
X X 3[4,r] X X X X X
```

5 Summary

This assignment served to show the advantages of red-black trees over binary search trees. While red-black trees are not always superior to binary trees in theory, in practice it is almost always better to use a red-black tree or other type of balanced tree given that there are no restrictions that would prevent one being used. Even in the rare cases that a binary tree would have a lower search cost, the difference is minimized as the red-black tree can only be off of perfect balance by a maximum of 1 level up or down, which is an acceptable cost for the advantages.