CSCE 221 Assignment 5 Cover Page

First Name Hunter Last Name Cleary UIN 625001547

User Name hncleary E-mail address hncleary@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)	Listed Below			
Printed material	Textbook			
Other Sources				

https://www.cs.auckland.ac.nz/software/AlgAnim/red_black.html
https://www.geeksforgeeks.org/red-black-tree-set-1-introduction-2/
https://en.wikipedia.org/wiki/Red%E2%80%93black_tree

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 Hunter Cleary Date 4-8-2018

Assignment 5 (40 pts)

Program: Due April 10 at 11:59 pm

Description: You will be given an executive file, `RBTree`, and you will use it for all testing and running in this assignment therefore you do not need to write any code here. Make sure that you use the TAMU servers (*compute*) to run this assignment.

1. How to run the program
   (a) To run all the test cases, type
   `./RBTree`
   (b) To print out the trees level by level, type
   `./RBTree -o 1`
   In place of `1` it can be any number ≤ 12 (but don't go past 4 because it will print out too much for the terminal to handle). This number is the number of tree levels to print out.
   (c) To test on a custom file, type
   `./RBTree -f file_name`
   where `file_name` must include the (relative) path to this file (say, `data-files/7p`).
2. Use the provided files to test the Red-Black and regular binary search trees created using the same input.
   (a) Use a small input (the number of nodes less than 16) to compare a Red-Black tree generated by the program with a Red-Black tree obtained by hand. Include an evidence of these comparisons (pictures, screenshots, etc).
   (b) Make a table and a plot showing the average search cost (y-axis) versus the number of tree nodes (x-axis) of all Red-Black trees created with the data used.
      i. You should produce a graph with 6 plots: plots for the linear, perfect, and random file type data for both the regular binary trees and the Red-Black trees.
      ii. Your graph plots should include the data points from all 12 input files (from 1 to 12).

Report (40 points)

Write and submit to eCampus a brief report that includes the following:

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

   Red-Black trees are a type of binary search tree that can self balance. Every node that exists in the tree contains a bit which corresponds to a Red or Black value. The red and black values keep tree in an organized state. When data in the tree is modified, color values are changed to restore order in the structure. The self balancing of the tree allows for search, insertion, and deletion times of $O(logn)$. The data structure provides a guarantee on worst case scenarios, which makes it extremely useful in real-time applications.

2. 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. See the lecture notes for more details.

   Upper bound on individual search cost = $O(logn)$.

3. How can you justify that the computed average search costs for some Red-Black trees is higher than for perfectly balanced binary search trees?

   Red-black trees are self-balancing, but do not perfectly balance the tree. The search complexity of a red-black tree is amortized O(logn), while a perfectly balanced tree has a complexity of O(logn). If a red-black tree is built with items that are already sorted, the resulting tree is the most unbalanced it can be.

   Does the formula below provide lower bound on the computed average search costs for Red-Black trees? Justify your answer.

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

   Yes. Red-black trees have a amortized O(logn) for their lower bound. The equation provides this best case value for search.

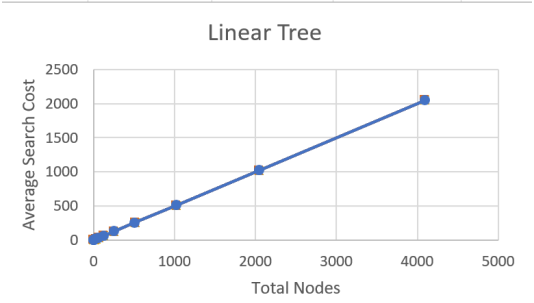
4. Include the table and plots of computed average search costs for Red-Black and regular binary search trees discussed in the item 2 of the **Description** part, together with the comparisons with theoretical lower and upper bounds. Write your conclusion.

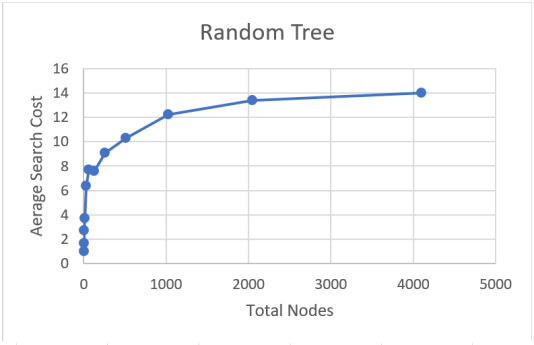
   BinarySearchTree **RedBlackTree**

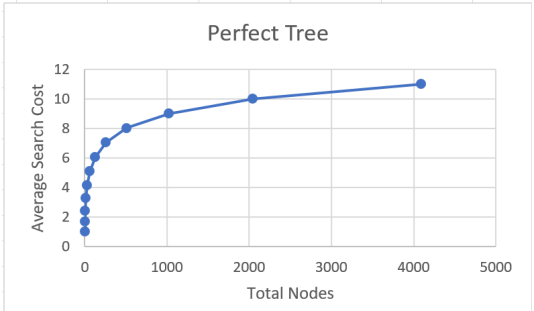
	Search Costs		
Nodes	Linear	Perfect	Random
1	1	1	1
3	2	1.67	1.67
7	4	2.43	2.71
15	8	3.27	3.73
31	16	4.16	6.39
63	32	5.1	7.67
127	64	6.06	7.59
255	128	7.03	9.07
511	256	8.02	10.3
1023	512	9.01	12.25
2047	1024	10.01	13.4
4095	2048	11	14.02

Nodes	Linear	Perfect	Random
1	1	1	1
3	1.67	1.67	1.67
7	2.714	2.714	2.571
15	3.67	3.4	3.4
31	4.613	4.484	4.29
63	5.571	5.508	5.286
127	6.531	6.449	6.3
255	7.525	7.314	7.286
511	8.515	8.405	8.278
1023	9.508	9.454	9.25
2047	10.505	10.492	10.317
4095	11.502	11.531	11.302

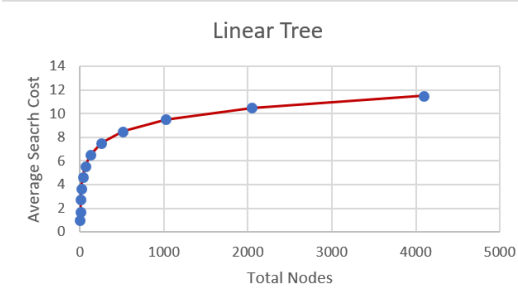
BinarySearchTree

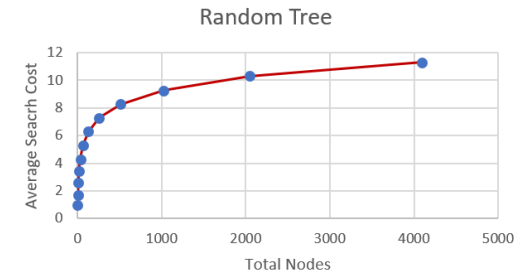


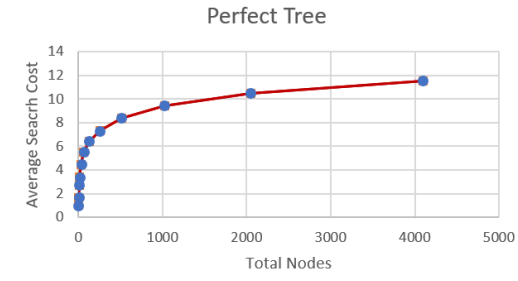




RedBlackTree







Theoretical Upper and Lower Bounds
Red-Black Tree Upper: O(logn) Lower: O(logn) (Amortized)
Binar Seacrh Tree Upper: O(n) Lower: O(logn)

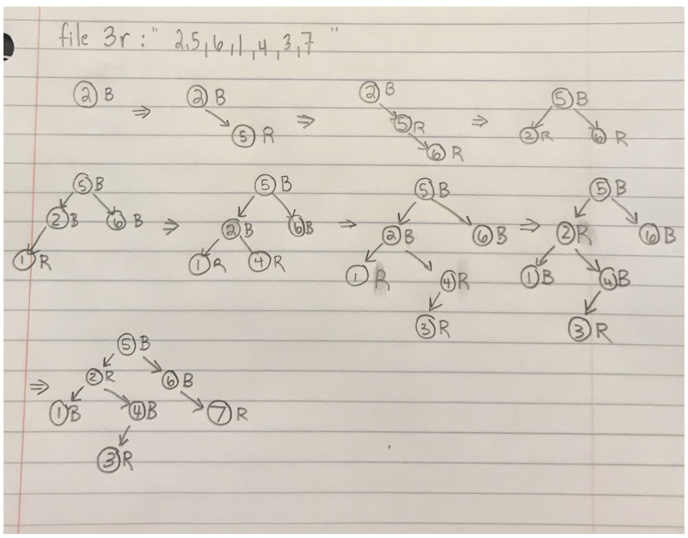
Conclusion

Red-black trees greatly improve the search time for data that is unsorted. The benefit of using this method over a normal binary search tree increases as the data becomes more unsorted. Inversely, data that has previously been sorted will not benefit as greatly or at all from a red-black tree. The results show that perfect sorted data search times are actually negatively impacted by red-black trees.

5. 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.

```
[hncleary]@compute ~/ProgrammingAssignment5> (10:36:54 04/09/18)
:: ./RBTree
BTree average search time linear        1 1
RBTree average search time linear       1 1
BTree average search time perfect       1 1
RBTree average search time perfect      1 1
BTree average search time random        1 1
RBTree average search time random       1 1
**********************************************************
BTree average search time linear        2 2
RBTree average search time linear       2 1.66667
BTree average search time perfect       2 1.66667
RBTree average search time perfect      2 1.66667
BTree average search time random        2 1.66667
RBTree average search time random       2 1.66667
**********************************************************
BTree average search time linear        3 4
RBTree average search time linear       3 2.71429
BTree average search time perfect       3 2.42857
RBTree average search time perfect      3 2.71429
BTree average search time random        3 2.71429
RBTree average search time random       3 2.57143
**********************************************************
BTree average search time linear        4 8
RBTree average search time linear       4 3.66667
BTree average search time perfect       4 3.26667
RBTree average search time perfect      4 3.4
BTree average search time random        4 3.73333
RBTree average search time random       4 3.4
**********************************************************
BTree average search time linear        5 16
RBTree average search time linear       5 4.6129
BTree average search time perfect       5 4.16129
RBTree average search time perfect      5 4.48387
BTree average search time random        5 6.3871
RBTree average search time random       5 4.29032
**********************************************************
BTree average search time linear        6 32
RBTree average search time linear       6 5.57143
BTree average search time perfect       6 5.09524
RBTree average search time perfect      6 5.50794
BTree average search time random        6 7.66667
RBTree average search time random       6 5.28571
**********************************************************
BTree average search time linear        7 64
RBTree average search time linear       7 6.54331
BTree average search time perfect       7 6.05512
RBTree average search time perfect      7 6.44882
BTree average search time random        7 7.59055
RBTree average search time random       7 6.29921
**********************************************************
```

```
**********************************************************
BTree average search time linear        3 4
1[1]
X 2[2]
X X X 3[3]
X X X X X X X 4[4]
X X X X X X X X X X X X X X X 5[5]
X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X 6[6]
X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
X X X X X X X X X X X X X X X X X X X X X X X 7[7]
RBTree average search time linear       3 2.71429
2[1,b]
1[2,b] 4[2,r]
X X 3[3,b] 6[3,b]
X X X X X X 5[4,r] 7[4,r]
BTree average search time perfect       3 2.42857
4[1]
2[2] 6[2]
1[3] 3[3] 5[3] 7[3]
RBTree average search time perfect      3 2.71429
6[1,b]
4[2,r] 7[2,b]
2[3,b] 5[3,b] X X
1[4,r] 3[4,r] X X X X X X
BTree average search time random        3 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 random       3 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
```



6. Summary. What have you learned by doing the assignment?

   Red-black trees are a type of binary search that are self sorted. This guarantees a guaranteed search time of amortized O(logn). The added cost of sorting data upon creation of the tree greatly decreases the search costs of data later on, but if the data has been previously sorted it will negatively impact performance (in a small factor).