Areege Chaudhary
10197607

CISC235 Assignment 2 Discussion

Experiment #1:

I expect that the number of rotations between checkpoints will remain fairly constant. I think this because between each checkpoint, there are always 100 values added to the Red-Black Tree. There would be a very small difference in values (accounted for by the randomness of tree generation), but overall it should remain fairly constant since it is the same four frequent scenarios that occur during each 100 inserts and cause a rotation. In this hypothesis, I am also assuming that a double rotation counts as only 1 rotation.

Checkpoint	Average Number of Rotations
100	28.112
200	28.89
300	28.504
400	28.22
500	28.772
600	28.494
700	28.794
800	29.22
900	28.208
1000	28.38

As predicted, the average number of rotations (for 500 trials) per 100 inserts remains constant across checkpoints, ranging between 28.112 and 29.22. This does make sense since the order of inserting 100 different values per trial should not affect the number of rotations. Rotations occur based on the comparison of the colours of the current node and at most, a node within two nodes of the current. And, each value is being inserted into an already balanced tree. Therefore, the overall size of the tree would not change whether or not a rotation occurs because the only thing that would cause a rotation would be an exception being met when a new value is being inserted.

Areege Chaudhary
10197607

Experiment #2:

In this experiment, total depth values of a binary search tree and a red black tree were compared in order to determine whether or not red black trees are in fact more efficient and compact than binary search trees. For a set {1, … , n} (where n was either 1000, 2000, 4000, 8000 or 16000), 500 different, random permutations were created. The values in these unordered sets were then added to both an empty binary search tree and an empty red black tree. The total depths for both trees were calculated and a ratio between the two were determined. The results are summarized in the table below (where R = Total Depth of Binary Search Tree / Total Depth of Red Black Tree).

n	$R < 0.5$	$0.5 \leq R < 0.75$	$0.75 \leq R \leq 1.25$	$1.25 < R \leq 1.5$	$R > 1.5$
1000	0%	0%	88%	12%	0%
2000	0%	0%	91%	9%	0%
4000	0%	0%	92%	8%	0%
8000	0%	0%	97%	3%	0%
16000	0%	0%	97%	3%	0%

In my data, there were no instances of a ratio being a value below 1. The average ratio of each n value was calculated as approximately 1.17. This means that there was no instance where the total depth of a red black tree was greater than the total depth of a binary search tree. And, as summarized in the table above, most of the R values ranged between 0.75 and 1.25 while the other few ranged between 1.25 and 1.5. Although, the frequency of R values between 1.25 and 1.5 decreased as the n value got larger (there was less of a difference between total depths as n increased). This might be because as n increases, the number of branches increase as well, causing the total depth to increase less quickly than it did for a smaller tree (with less values in it).

In every case, the total depth of the red black tree was smaller than the total depth of a binary search tree. Total depth refers to the sum of the depths of all the vertices in the tree. The longer the tree is vertically, the greater the value for its total depth is going to be. The assumption was that a red black tree would maintain balance as values were inserted, leading to a minimized total depth value while the binary search tree would be carefree about balance and insert values wherever they fit, potentially resulting in a tree with very long branches and a greater total depth. This is exactly what the results showed: red black trees were in every case “shorter” than its equivalent binary search tree therefore being more efficient.

I confirm that this submission is my own work and is consistent with the Queen's regulations on Academic Integrity.