CS/SE 2XB3 Lab 5 Report
Enrolled in CSL02

Wang, Mingzhe
400316660
wangm235@mcmaster.ca

Li, Xing
400292346
li64@mcmaster.ca

Moon, Hyosik
400295620
moonh8@mcmaster.ca

March 5, 2021

Contents

1 Implementing Insert

We implemented the three functions only (`rotate_left()`, `rotate_right()`, and `fix()`) as provided by the professor. No any other codes are modified. We tested the algorithm with multiple data sets to cover all possible rotations. The example below is the one to test the extreme of right leaning by inserting the number from 1 to 9, and we compared the result (Figure 2) with the expected result (Figure 1) as demonstrated at `https://www.cs.usfca.edu/~galles/visualization/`. They are exactly the same, so we are sure our algorithm is correct.

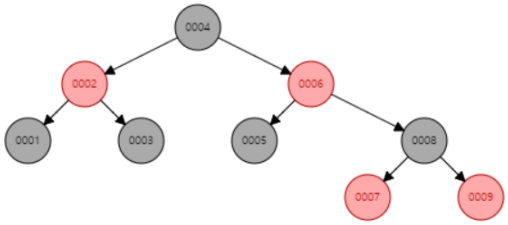


Figure 1: RBT insert result (1 to 9)

```
Insert: 1
[[(1,B)]]

Insert: 2
[[(1,B) -> [(2,R)]]]

Insert: 3
[[[(1,R)] <- (2,B) -> [(3,R)]]]

Insert: 4
[[[(1,B)] <- (2,B) -> [(3,B) -> [(4,R)]]]]

Insert: 5
[[[(1,B)] <- (2,B) -> [[(3,R)] <- (4,B) -> [(5,R)]]]]

Insert: 6
[[[(1,B)] <- (2,B) -> [[(3,B)] <- (4,R) -> [(5,B) -> [(6,R)]]]]]

Insert: 7
[[[(1,B)] <- (2,B) -> [[(3,B)] <- (4,R) -> [[(5,R)] <- (6,B) -> [(7,R)]]]]]

Insert: 8
[[[[(1,B)] <- (2,R) -> [(3,B)]] <- (4,B) -> [[(5,B)] <- (6,R) -> [(7,B) ->
   [(8,R)]]]]]

Insert: 9
[[[[(1,B)] <- (2,R) -> [(3,B)]] <- (4,B) -> [[(5,B)] <- (6,R) -> [[(7,R)]
   <- (8,B) -> [(9,R)]]]]]
```

Figure 2: Python implementation result

2 Red-Black Tree Height

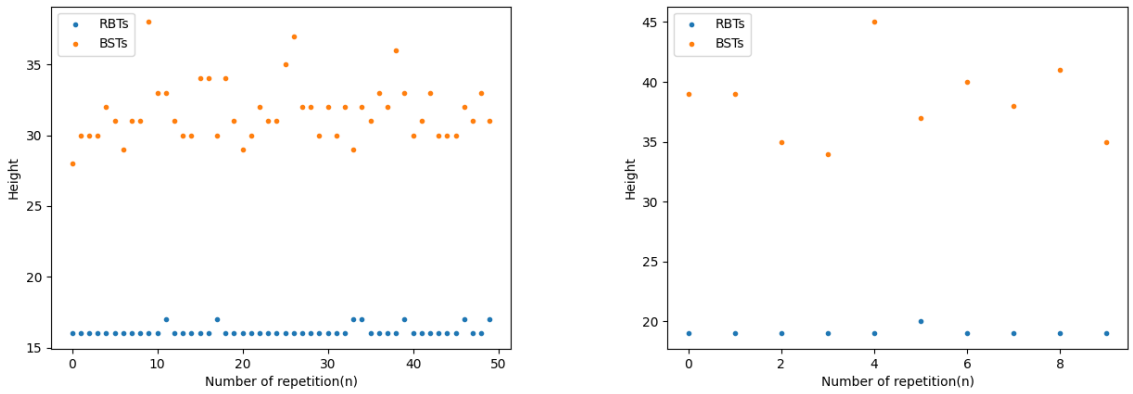
2.1 Insert the numbers from 1 to 10,000

When we repeated inserting the numbers 1 to 10,000, every time the height of the result was 24. We tested more than 20 times, but the height was always 24. If the implementation is correct, and the order of inserted number is not changed, then the height of the RBT tree should be always the same. This is because the construction of a RBT is constrained by its specific rules, such as "Red nodes cannot have red children.", "All simple paths from the root to a leaf must contain the same number of black nodes." . As mentioned in the question. However, if the implementation is incorrect, for example, if the root is not reset properly, the height could be wrong. In our implementation, we knew that after node.parent.parent is rotated, the root might be changed. We added a checker to reset the root accordingly. Because there are many ways to implement RBT tree, it's hard to guess the exact reason of the height change. We need to analyze the exception specifically to identify the root cause.

2.2 Average heights between BSTs and RBTs

We ran the test to know the average heights of insertion between RBTs and BSTs. Figure 3a represents the each result of inserting 10,000 randomized elements to RBTs and BSTs. We ran the test 50 times, and the average height is 16.14 for RBTs, and 31.6 for BSTs. The result shows that the RBTs's height is approximately half of BST's height.

We tested further to know the trend between the number of elements and height. When we inserted 50,000 and 100,000 (10times, 5times respectively), it showed that the average length of RBTs 19.1, 20.0 and 38.3, 40.6 for BSTs respectively. With the results, we know that the hight of RBTs is approximately half of the BSTs. This big difference between the two trees are shown in Figure 3b. So, we can simply know that the time complexity of RBTs will be better than BSTs when searching an item in the tree. However, if in a specific scenario, there are very few searching and many insertions, the additional cost of rebalancing RBT might not be paid off by the gain from searching. In that case, BST would be preferred over an RBT.



(a) Height of insertion (10,000(n), 50times) (b) Height of insertion (50,000(n), 10times)

Figure 3: Red Black Tree insert result

2.3 Average heights between BSTs and RBTs for sorted input

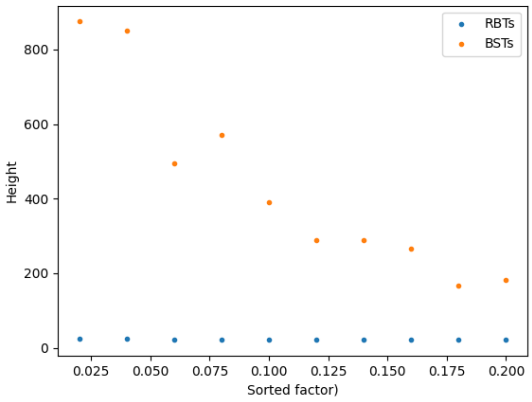


Figure 4: Heights of RBTs and BSTs (sorted lists)

We tested the average heights of BSTs and RBTs based on sorted lists. We ran the test changing the factor from 0.02 to 0.20. The main observations are listed below:

- The average heights of the trees were 22.6 for RBTs and 437.7 for BSTs.
- As the number of sorted items in the list increases, the BSTs' performance becomes worse, while the RBTs' performance almost remains the same.

- We can induce that in the worst case, when the list is totally sorted, the height of the BSTs will be nearly 10000, which is the length of the list.

References

[1] Red Black Tree,
`https://www.codesdope.com/course/data-structures-red-black-trees-insertion/`