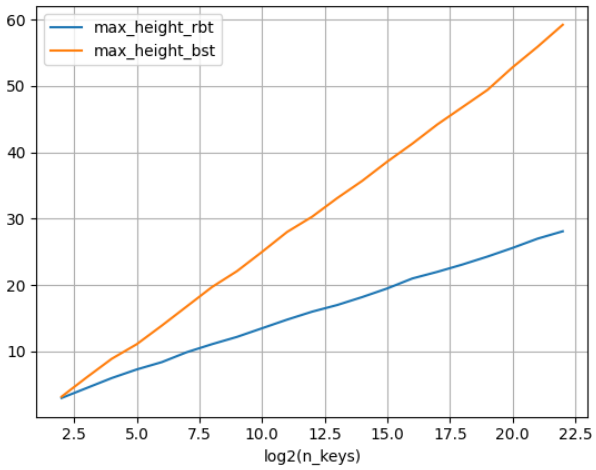
Robert Alvarez
Assignment 5 Report

1. Implement all functions required for Red-Black Trees as stubbed out in the boilerplate section.
   a. Files are attached in the submitted email.

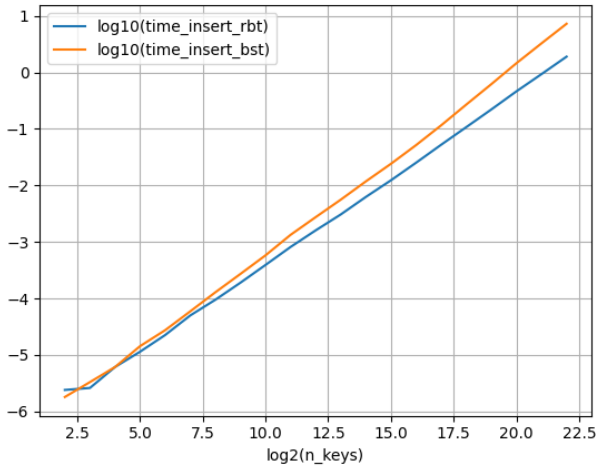
2. Students should test the different functions for correctness thoroughly, which includes testing the red-black properties that make the tree balanced.
   a. The balanced property was tested by creating a new function called “int red_black_tree_is_balanced(const red_black_tree_t *);” which compares the number of black nodes between subtrees. This is because red-black trees only guarantee that the bigger sub-tree is no more than twice the height of the smaller sub-tree.
   b. Such function was called for every insertion made in the red-black tree.
   c. The red-black tree was created with 2^{20} nodes where each key was of length 6 generated at random by selecting a character with replacement from an array with 89 distinct characters.

3. Students must include a description of their testing strategy, with a comparison of classical and red-black binary search trees.
   Description:
   a. Red-black trees and classical BST were created with sizes of $2^2, 2^3, \ldots, 2^{22}$.
   b. Each node was created with keys of size 6 where each character was chosen at random with replacement from a set of 89 distinct characters.
   c. Time comparisons were made for tree height, insertion, search valid keys, search invalid keys, and node removal.
   d. The previous 3 steps were executed 10 times with a different seed per iteration so distinct trees were created based on the key.
   e. The average of all iterations was taken per tree size to make inferences from the results.

   Results:
   a. The red-black trees and BST times for the search with valid keys, the search with invalid keys, and node removal, showed no significant difference in the logarithmic base 10 scale.
   b. Heigh graph:



   c. Insertion time graph:


log10(time_insert_rbt)
log10(time_insert_bst)
1
0
−1
−2
−3
−4
−5
−6
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
log2(n_keys)