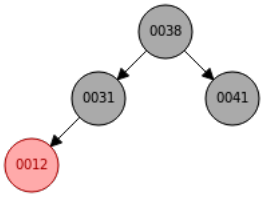
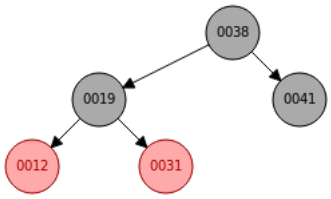
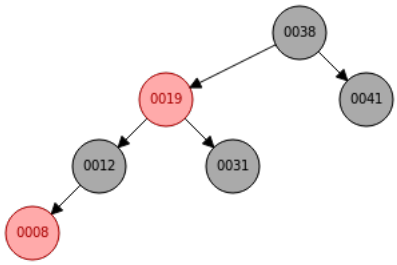
Emma Steinman and Colin Smith
CS 271 Project 8
December 17, 2017

1. Insert {41,38,31,12,19,8}







2. Hypothesis: The longest path from a node x in a red-black tree to a descendant leaf has length at most twice that of the shortest path from x to a descendant leaf.
   Proof:

   By definition of a red-black tree the number of black nodes on the path from a node to a descendant leaf must be the same for all paths.
   By definition of a red-black tree the children of a red node must be black.
   The shortest possible path for a given red-black tree is one where every node is black because there are no reds adding to the length.
   The longest possible path for a given red-black tree is one where every black node has a red child. This causes there to be the maximum number of red nodes, because every black node will be followed by a red node. This adds the maximum number of reds to the length.
   If every black node is followed by a red node there are as many red nodes as black nodes and therefore the path is twice the length of a path consisting only of black nodes. We know the path consisting of only black nodes is the shortest possible path for a given red-black tree. So, the longest possible path for a given red-black tree is twice the shortest possible path for the red-black tree. Because all paths in any given red-black tree are within these bounds, any given longest path in a red-black tree is at most twice the shortest path in the same red-black tree.

3. see code

4. see code

5. table The Binary Search Tree (BST) takes the longest because the movies are inserted in

Data Structure	time
BST	1.91219
HashTable	0.083683
RBT	0.032025

alphabetical order. This creates a completely unbalanced tree where all left children are null and makes run times linear. This is the worst case for a BST due to insertion and take linear time for each insertion. The Hash Table is much faster because it uses a hash function to negate any need for insertion into a list, just a memory location. There is an infrequent event where items must be inserted in a linked list but this is done in constant time, the most time consuming part is computing the hash, which is based on the size of the item being input. Assuming the hash function operates in constant time (the items being inserted don't get bigger) the Hash Table does each insertion in constant time. The Red-Black Tree (RBT) was the fastest because it is a tree which self balances, making it take logarithmic time, and it did not have to compute a hash which allowed it to move very quickly. This does insertions in logarithmic time.

If we were to use a faster hash function or have a much larger data set the Hash Table should be the fastest out of the three but for this hash function and data set the RBT came out on top.