Kyle Maclean
Red-Black Trees (RBTs)
November 2021

1 Design of `median t`

The median of a BST with an odd number of nodes is the middle in-order key; with an even number of nodes, it is the average of the two middle in-order keys. Non-Leaf Nodes are augmented to maintain the number of non-Leaf Nodes contained in each of their subtrees as *Counts*. Each (sub)tree root therefore contains the information about how many nodes are in the (sub)tree and in which direction to descend to visit the median node(s). After inserting or deleting a key, the number of levels required to descend to the location of the inserted/deleted key is counted and stored in whichever child of the root is its ancestor. Successively decremented values of this *Count* are distributed level-by-level to all its descendants. We chose this strategy because the median can be descended to in $\mathcal{O}(height(t))$ time and the extra two descents of the tree after insertion/deletion are also in the same time class. Since the strong RBT property is maintained, this class is $\mathcal{O}(\log n)$, where n is the number of nodes in the tree.

2 Design of `intersection t1 t2` and its time complexity

We first perform in-order traversals of each tree, yielding two sorted lists of their respective keys. This first stage takes $\Theta(length(t1) + length(t2))$ time because every node must be visited once in both trees. Then, to find the intersection of these two lists, we use an algorithm which, in the worst case, visits each element once in both lists. A better case is when the algorithm terminates early because one list is exhausted before the other. This second stage attempts to "synchronously" walk through the two lists, dropping elements in one that are less than the current element in the other and "saving" elements that occur in both. This second stage takes $\mathcal{O}(length(t1) + length(t2))$ time because the lengths of the lists are the same as the number of nodes in the RBTs. Therefore, the total time complexity of these two stages is $\Theta(length(t1) + length(t2))$, as determined by the first stage, since its complexity class is a subset of the second's.

3 How to run the program

The `rbt.hs` script can be compiled/interpreted with recent versions of GHC/GHCi. Test RBTs are provided at the bottom of the script.

```
>ghci rbt.hs
GHCi, version 8.10.7: https://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Main             ( p1.hs, interpreted )
Ok, one module loaded.
*Main> tc
Node Black (Node Black Leaf 0 8.0 Leaf 0) 1 16.0 (Node Black Leaf 0 32.0 Leaf 0) 1
*Main> traverse' tc
[8.0,16.0,32.0]
*Main> median tc
16.0
*Main> median $ insert 9 tc
12.5
*Main> (delete 9 $ insert 9 tc) == tc
True
*Main> intersection (delete 8 tc) (delete 16 tc)
[32.0]
```