Homework 6

Daoyu Wang PB21030794

October 21

Contents

1 13.1-3

It will. The proves are as follows:

1. Just blacken the RED root won't bring some unknowed third color.
2. The root now is BLACK.
3. This rule is easily preserved, because no new leaves are introduced.
4. This rule is obvious, as no RED nodes are introduced.
5. This rule is obvious too, because the BLACK height of every nodes including root is equal to themselves' before. The change to root won't be influent to $bh(x)$.

2 13.3-4

1. Originally, when we step into while loop, the node z is not root. So the first loop won't change the T.nil to RED.
2. All risks that change the node color to RED is at **line 7** and **line 13**.
3. At while loop, we only change the color at most two levels of above z. So when z is in a level more than 2, we won't be worry.

   - when z is at depth 0, which means z is root.

     The while loop will stop, because $z.p = T.nil$, and $z.p.color = BLACK$.
   - when z is at depth 1, which means $z.p$ is root.

     The while loop will stop, because $z.p = T.root$ and $z.p.color = BLACK$ which is ensured by the property 2 of RBT and **line 16**.
   - when z is at depth 2, whihc means $z.p.p$ is root.

     At line 7, $z.p.p.color = RED$ will change the color of root to RED, however this will be corrected at **line 16**.

     At line 13, although $z = z.p$ at **line 10** which is ricky for $z.p.p.color = RED$ at **line 13** because z.p.p seems like T.nil, the call of LEFT-ROTATE(T, z) will change z to the level below itself at depth 2. So $z.p.p.color = RED$ is secure.

3 13.4-3

INSERT

1. INSERT 41

Figure 1: INSERT 41

2. INSERT 38

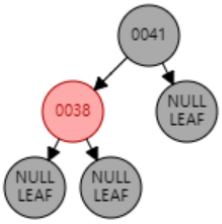

Figure 2: INSERT 38

3. INSERT 31

Figure 3: INSERT 31

4. INSERT 12

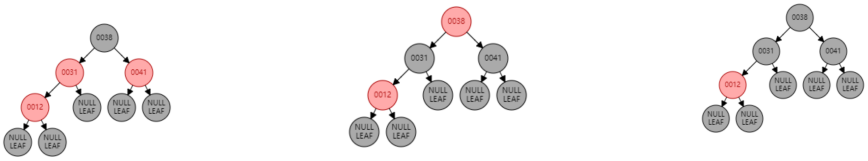

Figure 4: INSERT 12

5. INSERT 19

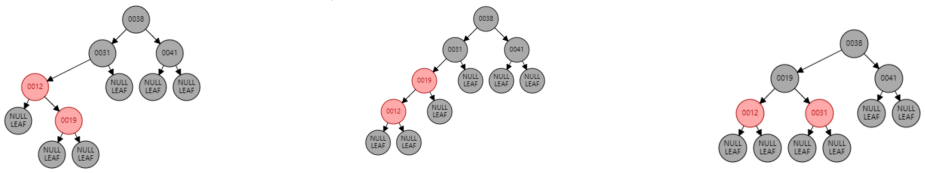

Figure 5: INSERT 19

6. INSERT 8

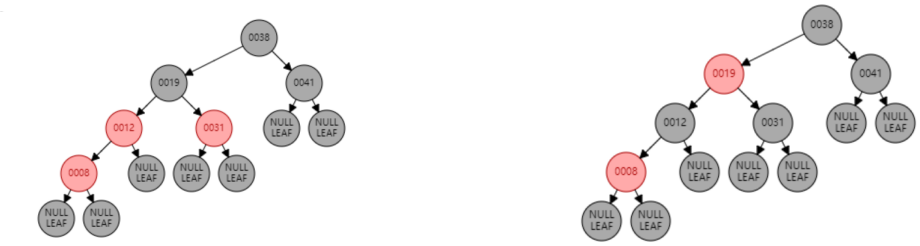

Figure 6: INSERT 8

DELETE

1. DELETE 8

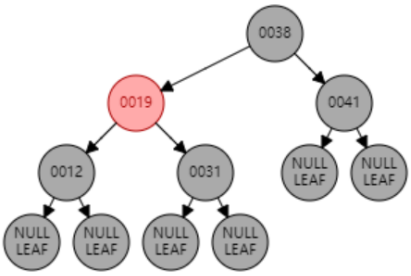

Figure 7: DELETE 8

2. DELETE 12

Figure 8: DELETE 12

3. DELETE 19

Figure 9: DELETE 19

4. DELETE 31

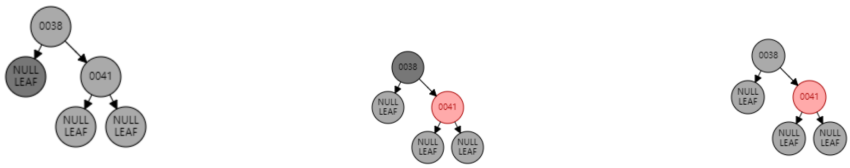

Figure 10: DELETE 31

5. DELETE 38

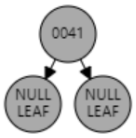

Figure 11: DELETE 38

6. DELETE 41

   The BRT only has T.nil.

4 14.1-5

The method is OS-SELECT(T, OS-RANK(T,x) + i), OS-RANK takes $O(h) = O(log(n))$ while OS-SELECT takes $O(h) = O(log(n))$. So the time complexity is $O(log(n))$.

5 14.2-2

The BLACK height can be maintained in $O(log(n))$. Since BLACK height of RED node is same as its BLACK parent. And the INSERT and DELETE operations all have locality. So change the BLACK height can be finished same as the operations themselves.

However, the depth is opposite, if we delete the root of BRT, the depth of all nodes will be changed potentially. The time complexity is $O(n)$, slowwer than the operation $O(log(n))$.

6 14.3-3

```
x = T.root
min = T.nil
while x != T.nil
    if i overlap x.int
        min = x
    elseif x.left != T.nil and x.left.max >= i.low
        x = x.left
    else
        x = x.right
```