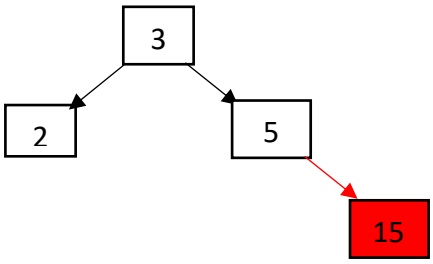
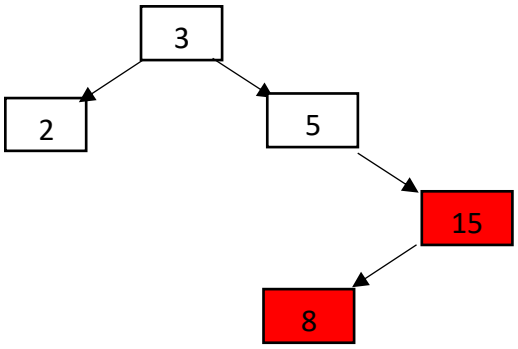
Author: Chakrya Ros
Instructor: Christopher Godley
TA: Shudong Hao
Date: 7/15/2018
Assignment5

1. Inserting a node into a red-black tree, re-balancing, and then deleting, it's depended on a node position that we insert, if insert a node into a red-black tree, a node has two children, and then delete that node, it results to an original tree.

For example, tree < 3, 5, 2, 15>



After insert node 8

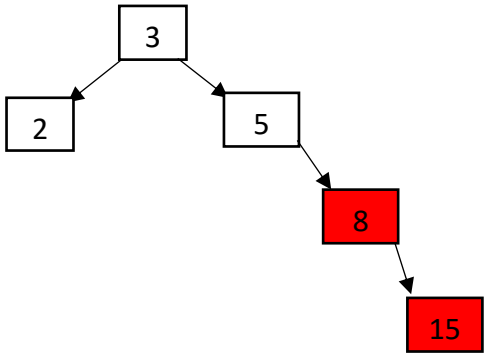


1. Let x = insertRB(8)
2. x.color = red
3. x.leftChild = NULL
4. x.rightChild = NULL
5. while(x!=root) and (x.parent.color == red)
6. if (x.parent == x.parent.parent.right)
7. x = x.parent.parent
8. If(x == x.parent.left)
9. x = x.parent
10. RightRotate(y=15)
    - x.rightChild.parent = y
    - x.parent = y.parent
    - x.rightChild = y
    - y.parent = x

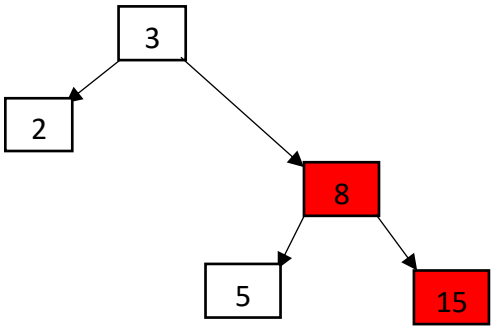
Both node and parent are red, node is rightChild, parent are rightChild

- x = node 5, y = node 8
- Left rotation(x)
  - y = x.rightChild
  - y.parent = x.parent
  - y.leftChild = x
  - x.parent = y
- So recolor node 8 = black, node 5 and 15 are red

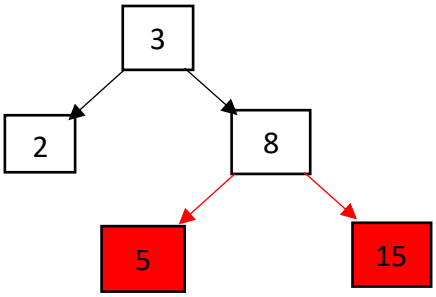
Perform right rotation



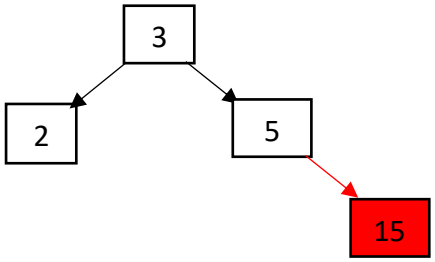
Perform left rotation



Rebalance tree after left and right rotate



After delete the node 8 and rebalance tree

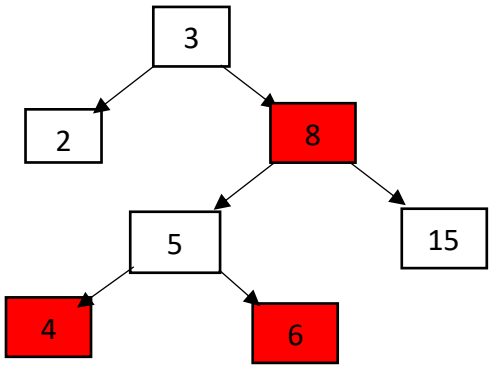


- After delete 8 with two children, it will be replaced with 5
    - x = search(8)
    - If(x!=root)
        - If(x.leftChild != NULL and x.rightChild !=NULL)
            - min = treeMinimum(x.rightChild)
            - x = min.leftChild
            - min.rightChild.parent = min.parent
            - min.parent = x.parent
            - min.rightChild = x.rightChild
            - x.rightChild.parent = min
            - min.color = x.color
            - rbBalacnce(x)
    - delete x

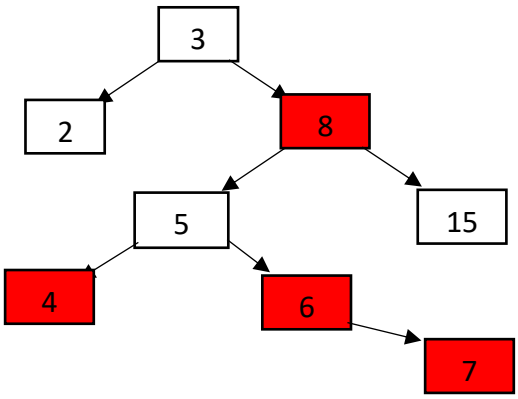
So this tree go back to original tree after delete 8. However, if insert node with no children into the tree and delete it, it does not result to an original tree.

Another example tree <3, 5, 2, 15, 8 , 4, 6> that delete with no children
The root of this tree is 3, after insert 7 into tree, the tree rebalance and the root of the tree is 5 and then delete node 7, it does not result to an original tree, the root of tree is still 5 .

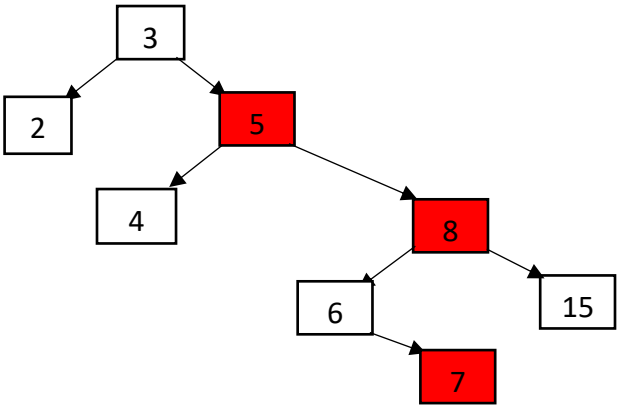
The original tree:



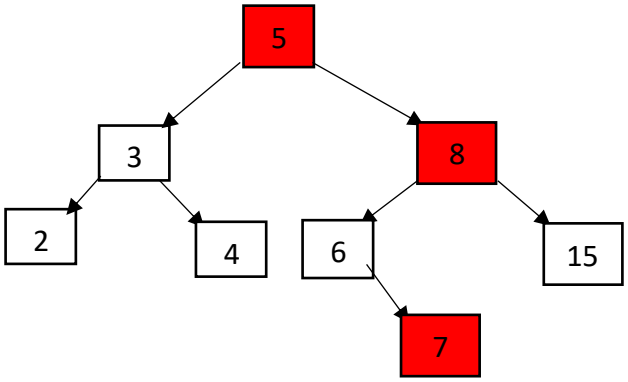
Insert node 7:



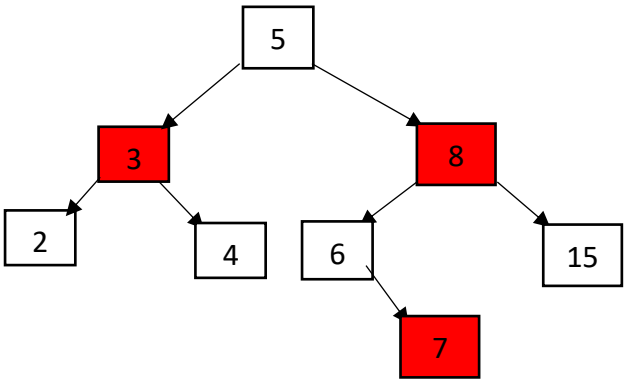
Perform right rotation



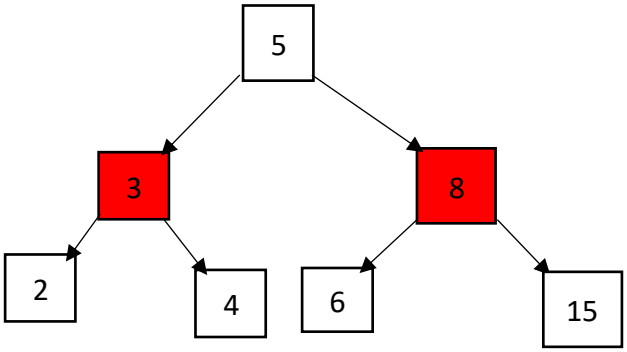
Perform left rotation



After rebalancing the tree:



After delete the node 7 with no children:

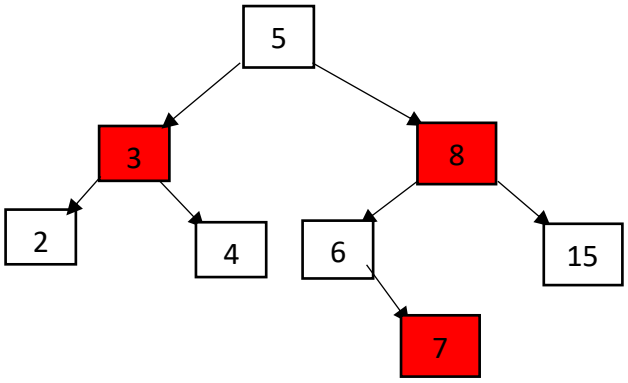


**Insert node 7

- node 7 and parent are both red
- uncle of node is red
- let x = 5
- let y = 8
- perform right rotate(y)
  - x = y.leftChild
  - y.leftChild = x.rightChild
  - if(x.rightChild ! = NULL)
    - x.rightChild.parent =y
  - x.parent = y.parent
  - if(y.parent ==NULL)
    - root =x
  - else if(y==(y.parent.leftChild))
    - y.parent.leftChild = x
  - else
    - y.parent.rightChild =x
  - x.rightChild = y
  - y.parent. x
- perform left rotate (x)
  - x.parent = x
  - x.leftchild = x.parent
  - x.parent.color (node 3) = red
  - x.color = black
- After delete node7, it does not result to an original but the same height.

2. Deleting a node with no children from a red-black tree, re-balancing and then reinserting it with the same key, it depend on the leaf color we delete. For example this red-black tree <3, 5, 2, 15, 8 , 4, 6, 7> .

   The original tree: the root is 5 and black height is one.

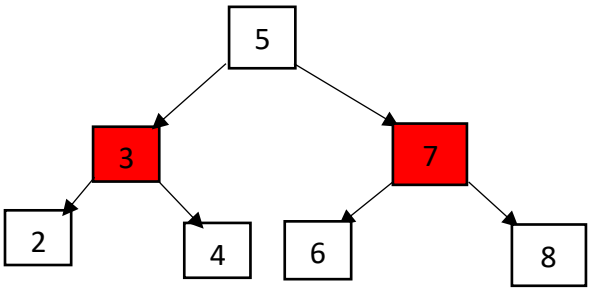


If delete node 7 with color red from this red-black tree and reinsert it with same key, it results to an original tree. However, if delete 15 or 4 with color black from the red-black tree and rebalance the tree and then reinserting it with same key, it does not result the same original tree.

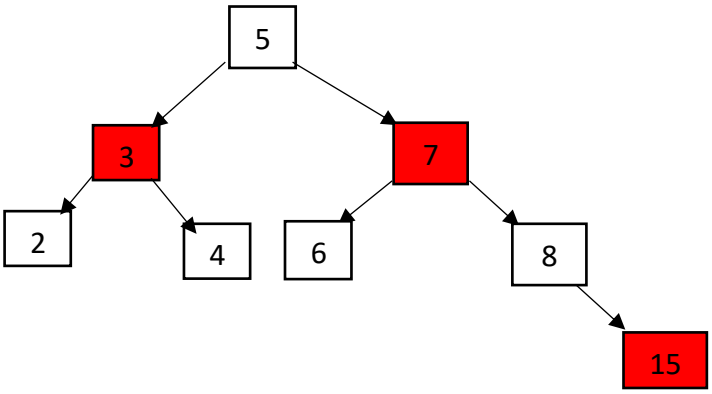
Algorithm for delete 7

1. x = search(7)
2. if(x != root)
   if(x.leftChild == NULL and x.rightChild == NULL)
   x.parent.leftChild = NULL
   x = x.parent.leftChild
3. delete x

For example if delete 15 from the tree



reinsert 15 after deleting



Before delete 15, its color black and its parent is 8 and parent.color is red. After deleting and reinserting 15, its parent.color is black and node15.color is red so the tree does not the same but the black height is the same and the same root.

Algorithm for delete 15 with no children

1. x = search(15)
2. if(x != root)
   if(x.leftChild == NULL and x.rightChild == NULL)
   x.parent..rightChild = NULL
   x = x.parent.rightChild
3. if(x.color == black)
   reBalance(x)
4. delete x

Algorithm for reBalacnc(x)

1. while(x!=root and x.color ==black)
2. if(x ==x.parent.rightChild)

```
3.              S = x.parent.leftChild
4.              If(s.color == red)
5.                 s.color = black
6.                x.parent.color = red
7.                leftRotate(x.parent)
8.                s =x.parent.rightChild
9.              If(s.leftChild.color ==black and s.rightChild.color ==black)
10.              s.color = red
11.              x = x.parent
12.              else if(s.leftChild.color ==red and s.rightChild.color==black
13.                 s.leftChild.color =black
14.                 s.color = red
15.                 rightRotate(s)
16.                 s =x.parent.rightChild
17.              else
18.                s.color = x.parent.color
19.               x.parent.color =black
20.               s.rightChild.color =black
21.               leftRotate(x.parent)
22.               x = root
23.  x.color = black
```