Last Time:

Tree complexity:

$O(\log_2 N)$ - exponentially better than linked list

- only true if tree is "balanced"

Tree Balancing

- different algos. exist
- We will consider the Red-black tree
  - ↳ all same properties as BST, plus builtin self balancing mechanism
    - Nodes now have a binary color choice: red or black
    - Need rotation mechanisms

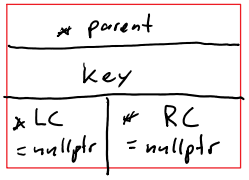
Today:

- Leaf nodes (slightly different approach than a BST)
- Return to "insert" example
  - conceptual
  - algorithm

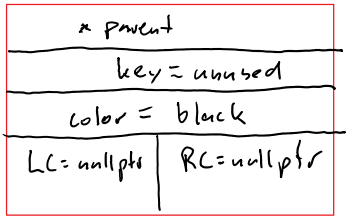
Leaf Nodes

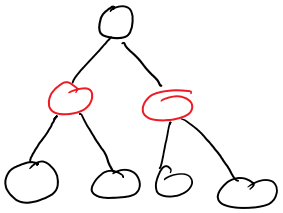
Regular BST

key stores data



RB Tree





Leaf nodes store no data.

Since we still use same node structs we still have "key" value. The contents of "key" as garbage.

Inserting a Node into a RB Tree

1. Set color of new node to red.
2. Terminate branch w/ NULL node (instead of nullptr.)
3. Resolve any RB property violations by using rotations and/or re-coloring.

Example: Building a RB

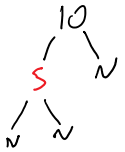
insert (10) into empty tree:

Rule: new node is red
violation: re-color node to be black

Add 5 to tree

insert(5)

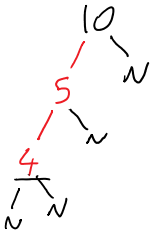
No violations

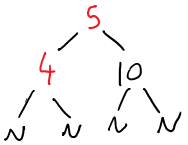


Add 4 to tree

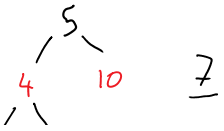
insert (4)

1) rotate right

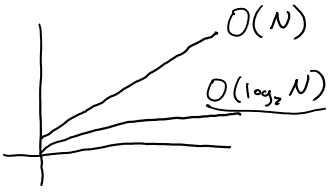


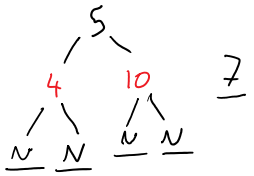


2) re-color



7

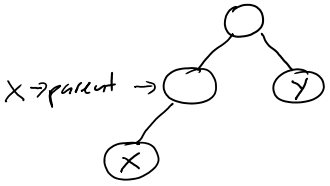




Insert Algorithm Considerations

First, insert node just like you would into a BST.

First identify the "uncle" node in the tree.



x is the new node
y is the uncle

The steps needed to re-balance depend on the color of the uncle node.

6 possible scenarios (configurations) tree can take on after insertion

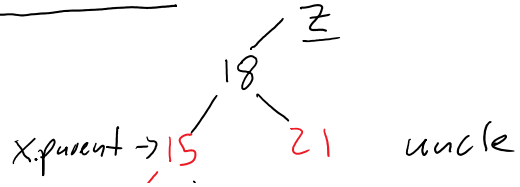
A) Parent of new node is Left Child

1) uncle node is red —
2) uncle node is black AND new node is RC
3) uncle node is black AND new node is LC

B) Parent of new node is RC

1)
2)
3) } symmetrical equivalents of above

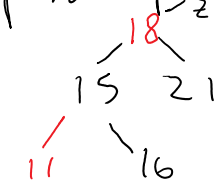
A) Case 1: uncle node is RED



violation: red node must have black children nodes

Resolve case 1

1) Recolor: x.parent and uncle to black
: x.parent.parent to red

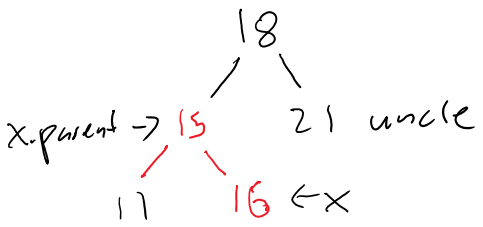


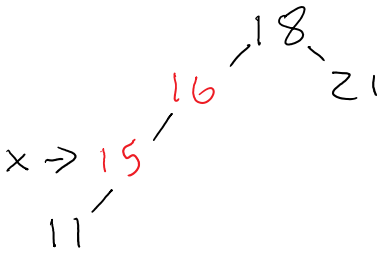
2) Set x = x.parent.parent (move 2 levels up)

3) Repeat 1 and 2 until x is root
or x's parent is black

↳ Once x is root, make sure it is black

Case 2: uncle node is black and new node is right child

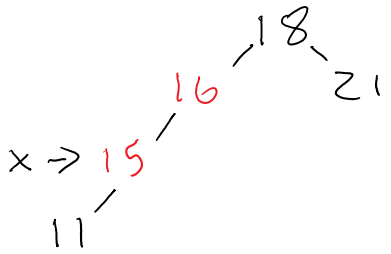




1. Set x = x.parent

2. rotate left around x.

Case 3 uncle node is black and new node is LC



To be continued...