Binary Search Trees

Tuesday, November 10, 2015 11:33 AM

- Lookup
- Tree Rotations
- AVL Trees

Next Week

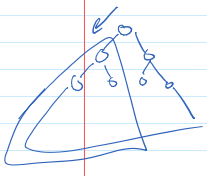
- Tuesday (17th) Intel Speaker
  SWGN 2A14 6:00–7:00 (Also pizza?)
- Friday (20th) Code-a-thon ID11/ID15
  3 tiers 145, (146) 240+ (pizza)
  12 hrs 8 pm – 8 am
  Tentative 1st hour → 10 pt
  Each completed problem → 10 pts

Will Hoskins 3D11

BSTs

```
class BST<E extends Comparable<E>> {
    private class Node {
        : left, right, elem, (parent?)
    }
    size, root
    void insert(E elem) {
        :
    }
    (bool, E) lookup
```

depending on implementation



check that root is not null

```
public E lookup(E element) {
    return lookup(element, root);
}
```

E, Node<E> — type of Node

```
public E lookup(E element, Node(E) sRoot) {
    if (element.equals(sRoot.elem)) {
        return sRoot.elem; //NOT ELEMENT
    }
```

Example

Element
int index;
Obj payload;

```
equals(this, that)
    ret this.index == that.index;
}

compareTo(this, that) {
    return this.index - that.index;
}
```

lookup(Element index = 3 payload = null)

return

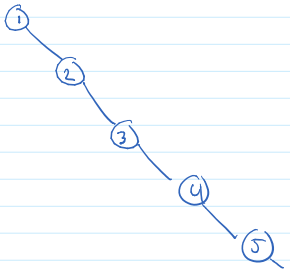
Element index = 3 payload = some object

Check if exists

```
if (sRoot == null) { // if sRoot.left == null && element.compareTo(sRoot.elem) < 0) return null
    return null; //return false  else if (.... (right case))
}
if (element.compareTo(sRoot.elem) < 0) {
    return lookup(element, sRoot.left);
```

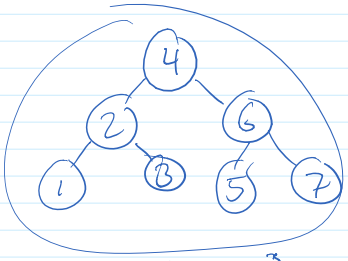
```
}
if (element.compareTo(sRoot.elem) < 0) {
        return lookup(element, sRoot.left);
} else {
        return lookup(element, sRoot.right);
}
}
```

insert ([1, 2, 3, 4 ...])



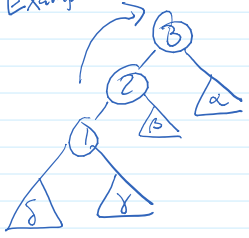
Find some way of "balancing" the tree ...

Example, 1 ... 7

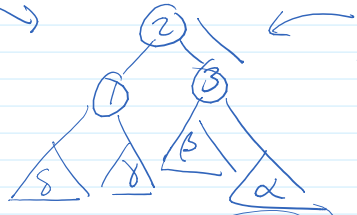


[4, 2, 6, 1, 3, 5, 7]

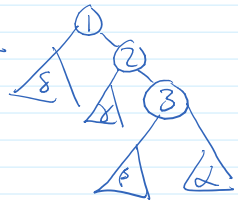
Example



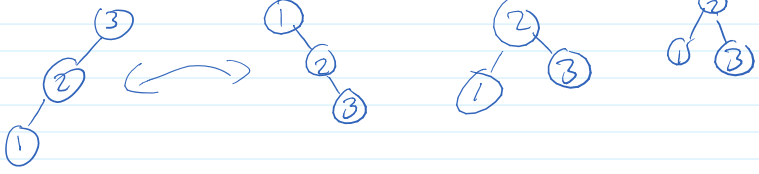
right rotate on 3



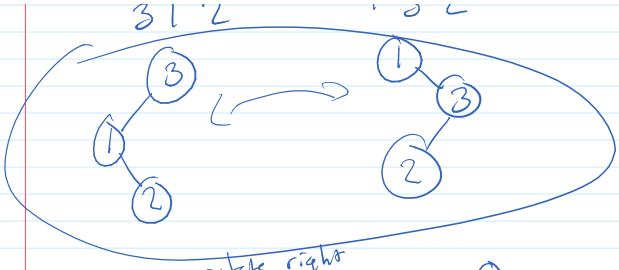
left rotate on 1

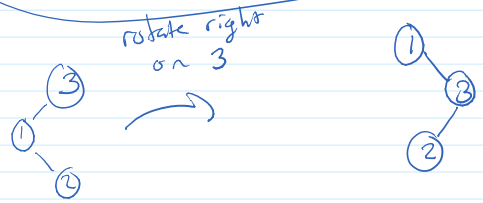


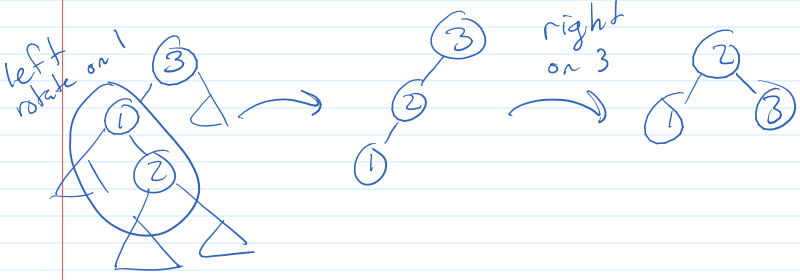
3 2 1 | 1 2 3 | 2 1 3 | 2 3 1



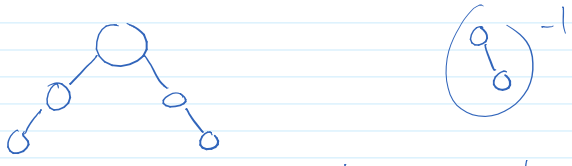
3 1 2 | 1 3 2





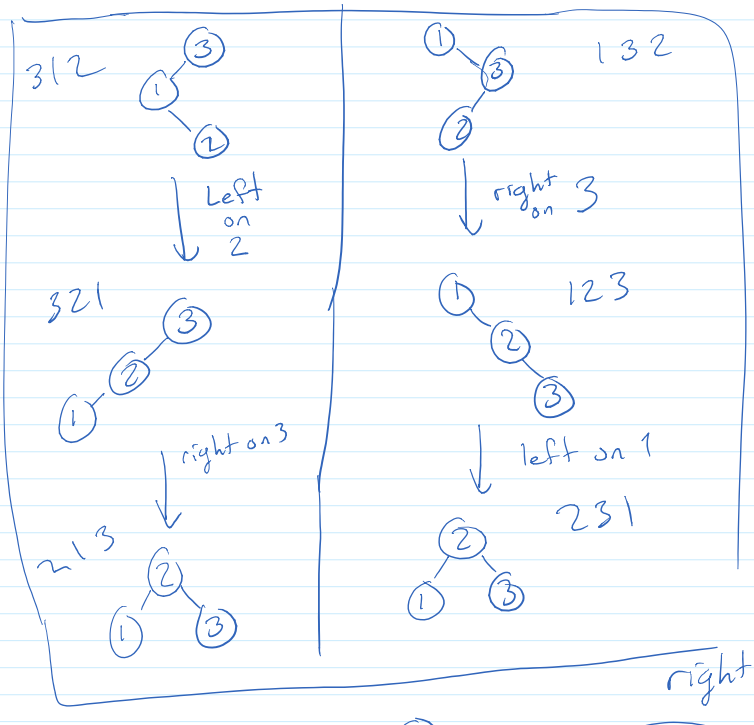


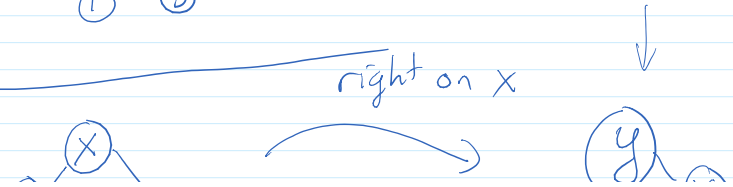
balance Factor = depth(left) - depth(right)

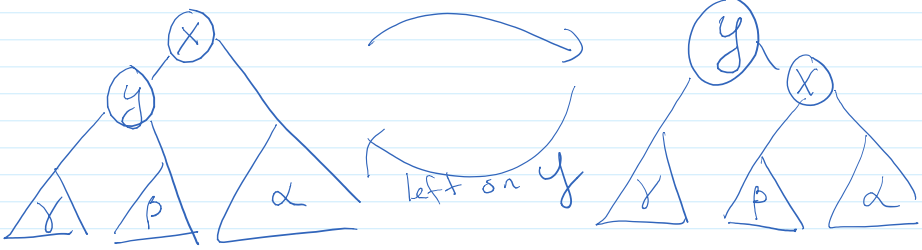


A "balanced" tree has a |balance Factor| ≤ 1

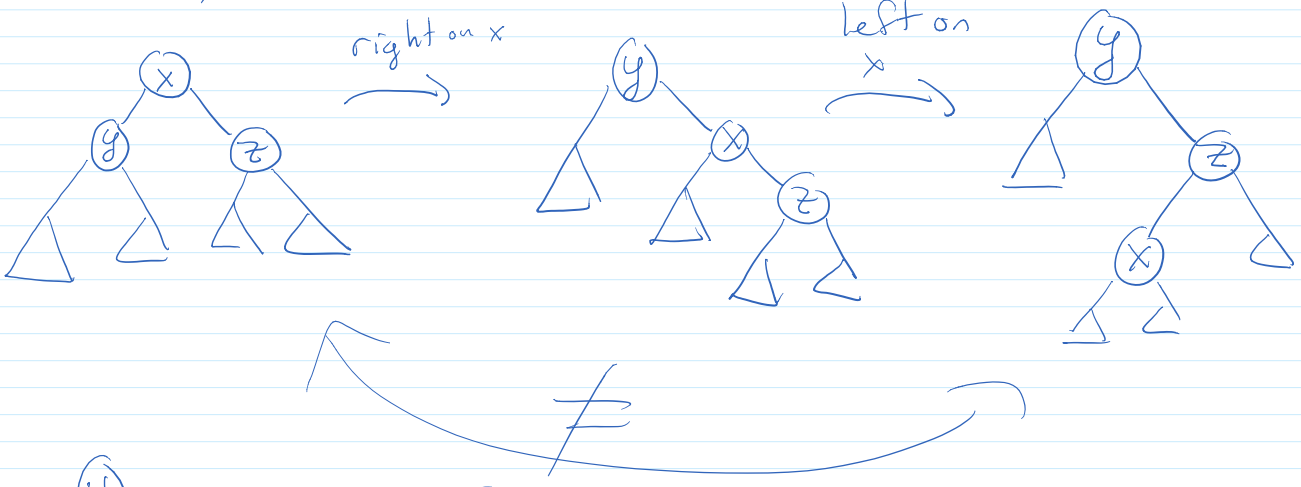
Tree rotations

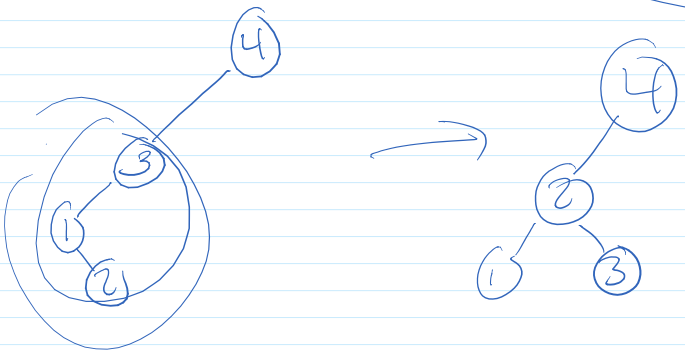


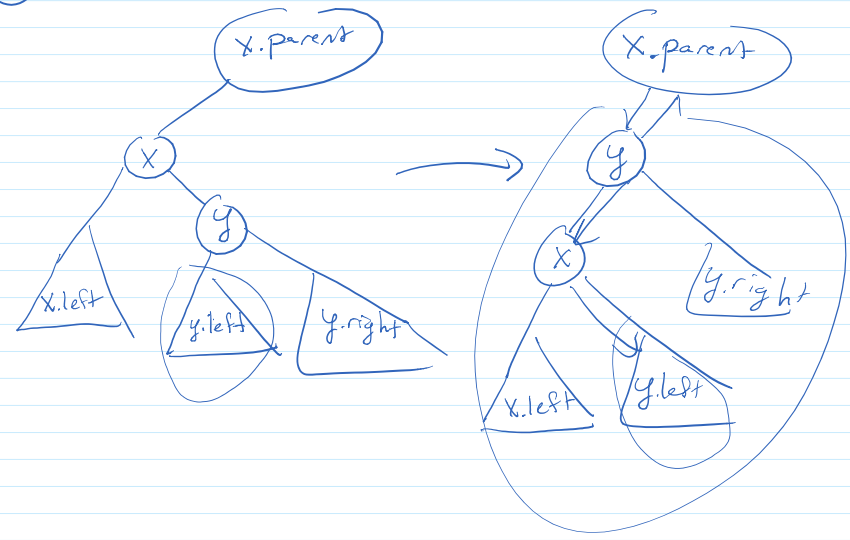




(left x) ∘ (right x)





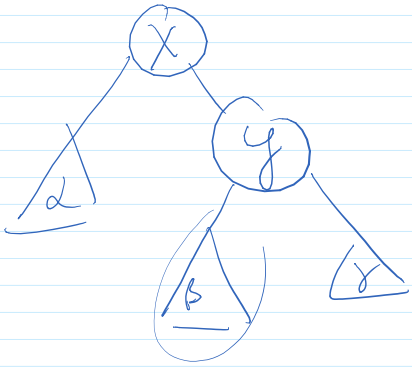


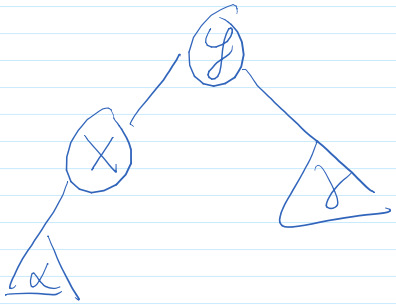

```
rotateLeft(node) {
    x = node
    y = node.right

    x.right = y.left
    y.left.parent = x
    y.left = x

    x.parent.left = y
    y.parent = x.parent

    x.parent = y
}
```


x
y
α
β
γ


y
x
γ
α