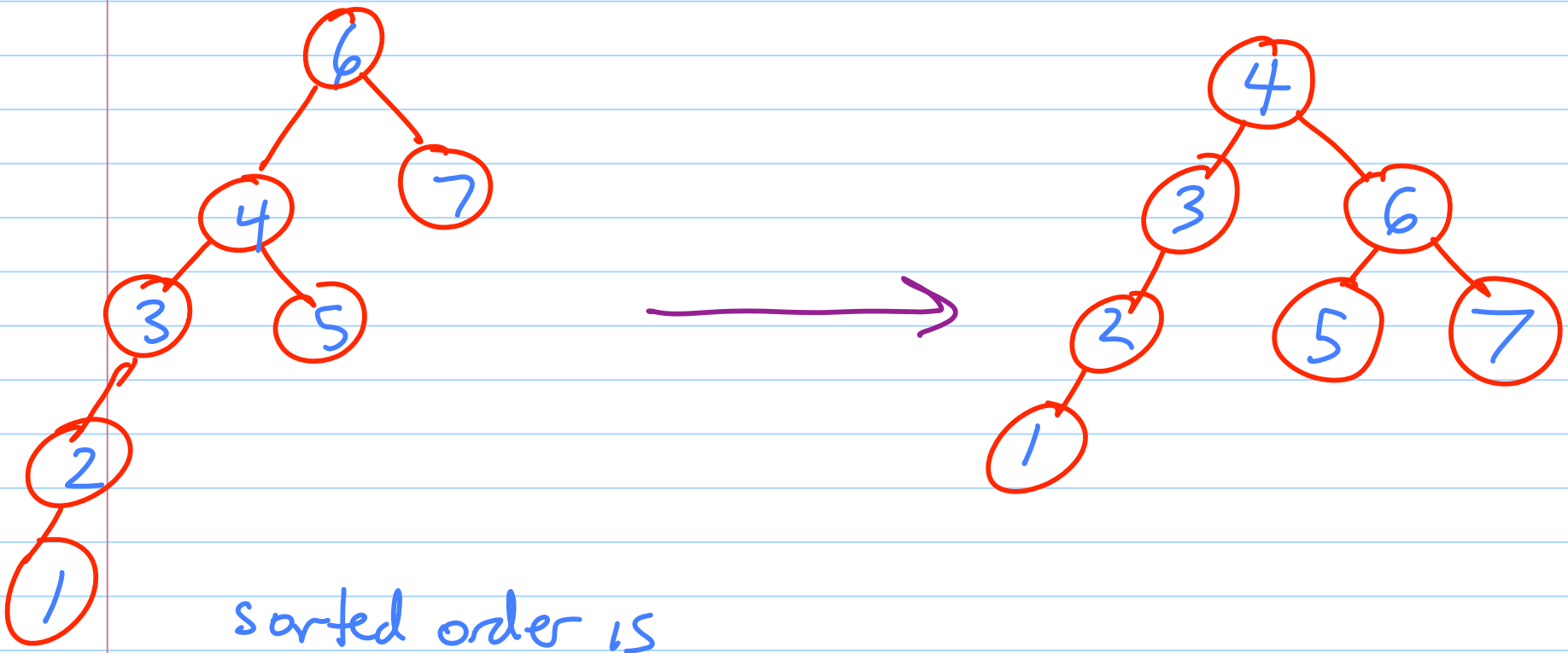
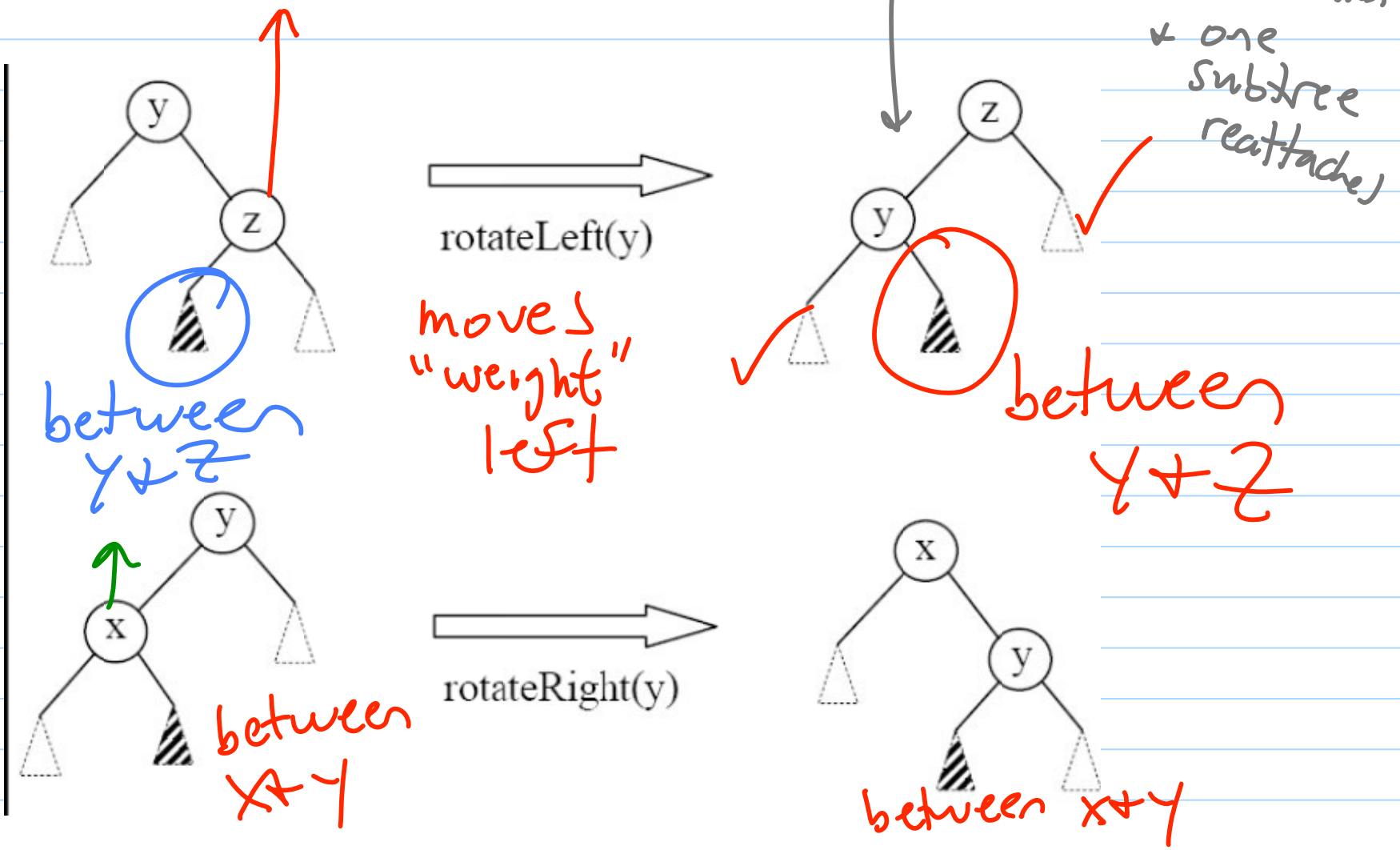
Balanced Binary Search Trees

Note Title 10/12/2007



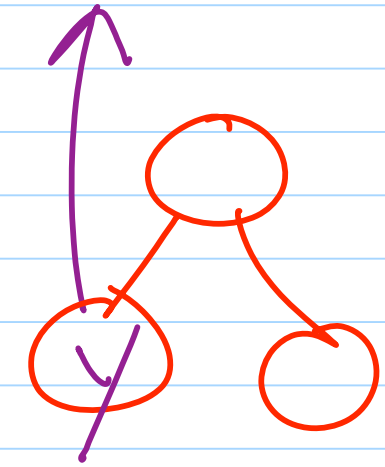
sorted order is given by an inorder traversal

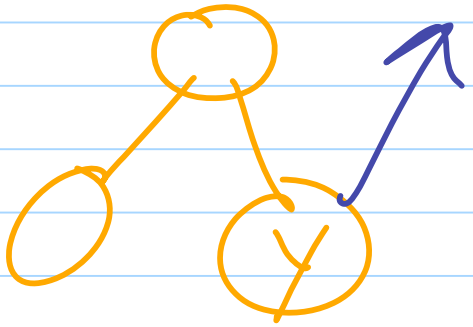

y
z
between y & z
rotateLeft(y)
moves "weight" left
y
z
between y & z
y
x
between x & y
rotateRight(y)
x
y
between x & y

Requires parent is not null

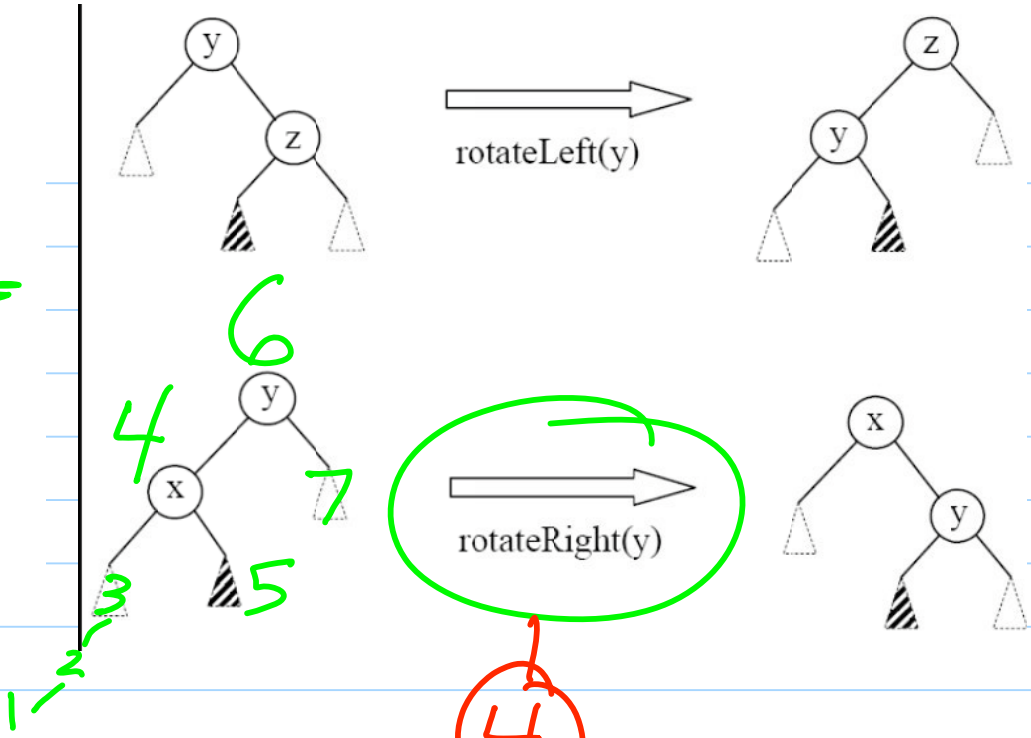
internal

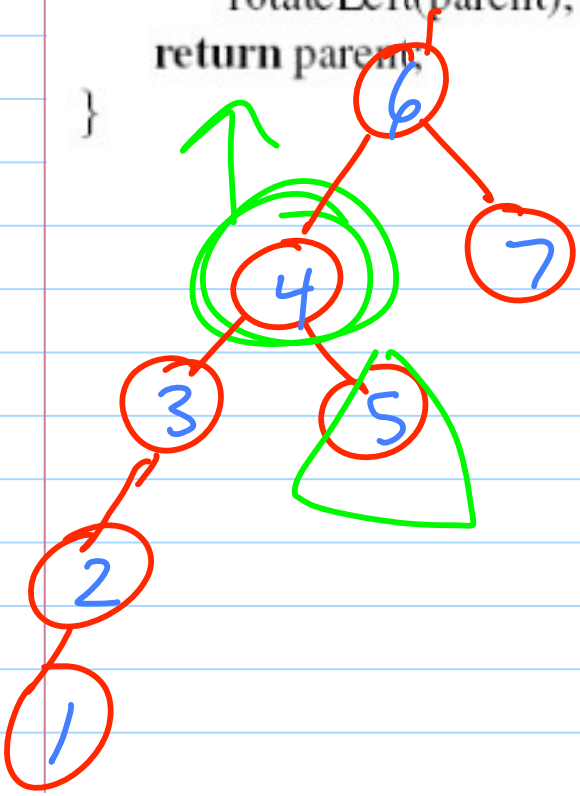
```
BSTNode liftUp(BSTNode y) {
    BSTNode parent = y.parent;
    if (y.isLeftChild())
        rotateRight(parent);
    else
        rotateLeft(parent);
    return parent;
}
```

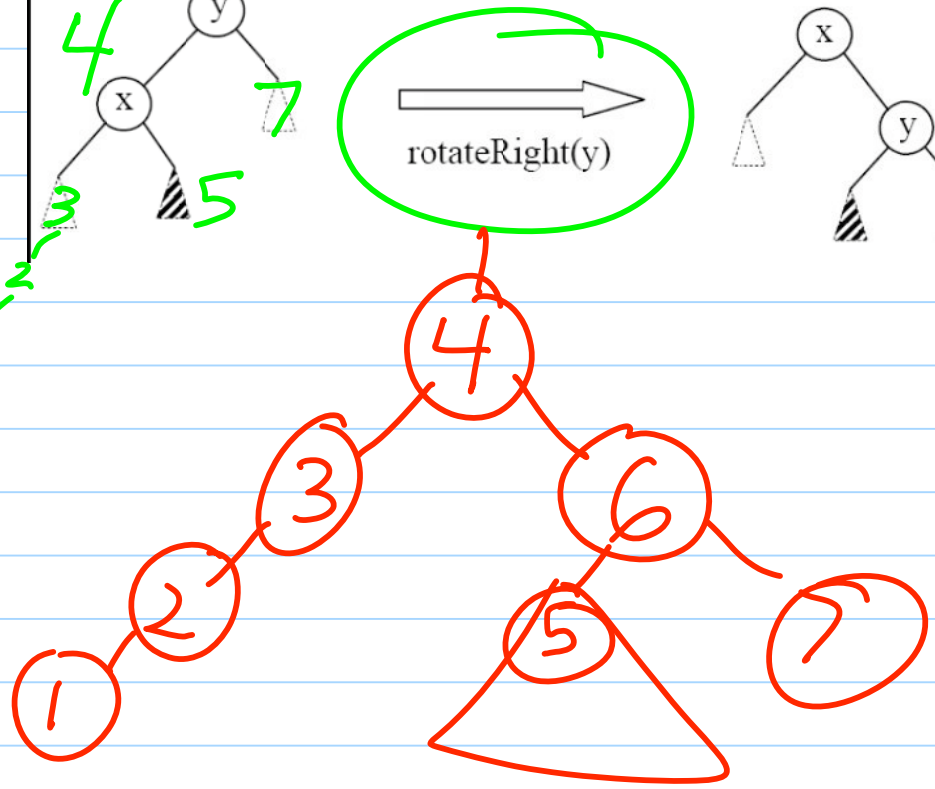




```
BSTNode liftUp(BSTNode y) {
    BSTNode parent = y.parent;
    if (y.isLeftChild())
        rotateRight(parent);
    else
        rotateLeft(parent);
    return parent;
}
```







We can use rotations to help keep a binary search tree balanced while maintaining INORDER property.

Completely structural (no new comparisons)

Difference between different data structures that are balanced binary search trees is how you decide when & where to rotate.

Red-black trees (after fall break)
Guarantees height $\leq 2 \log_2(n+1)$

Book includes

Splay tree - amortized
rotates each element
accessed/insert to root

AVL trees - red-black trees
are a better choice

Midterm Topics

100 pt exam

Note Title 10/12/2007

Review Hws & practice problems.

Bring one 8½×11 crib sheet

Divide-and-Conquer Algorithms

15-20

Should be able to design with good hints

Should be able to analyze (use master method)