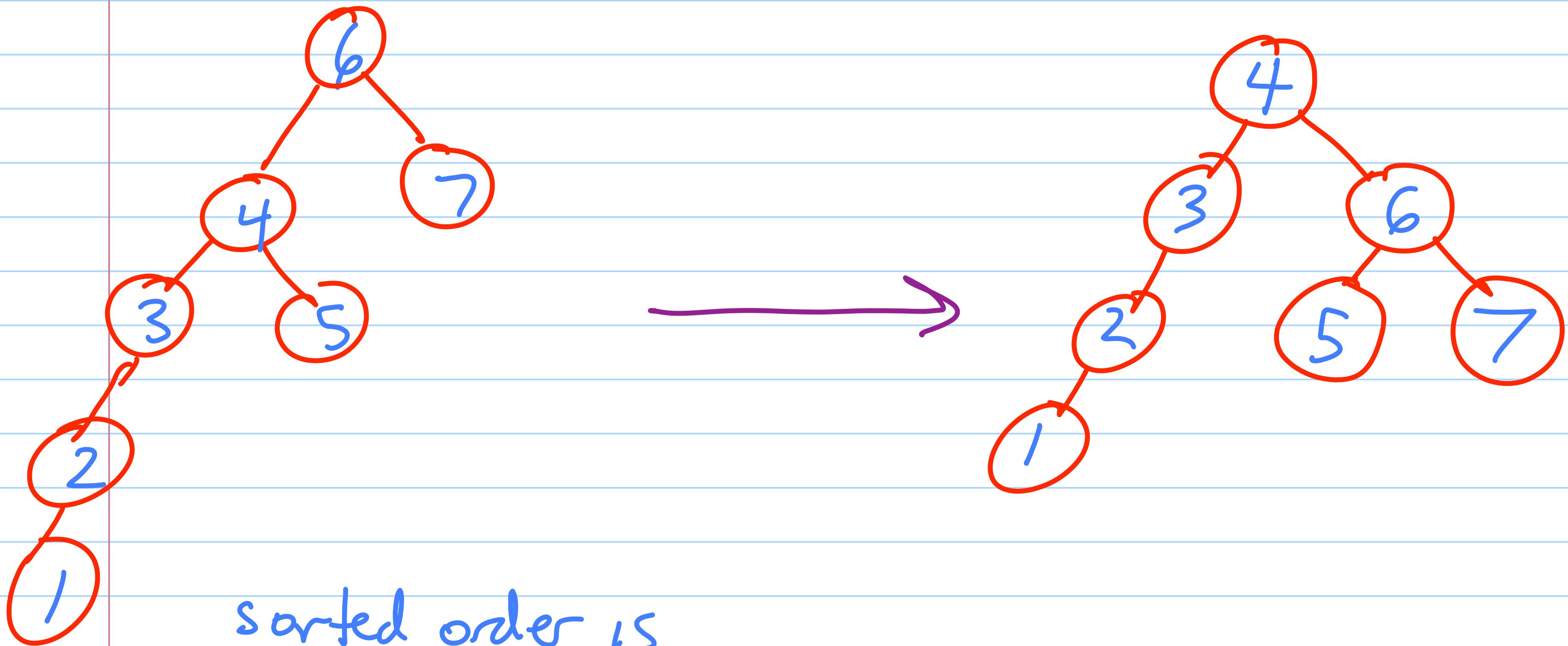
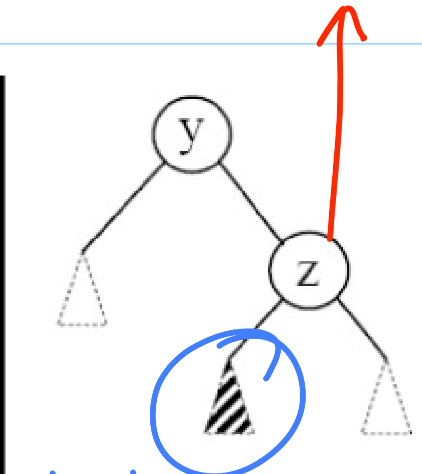
Balanced Binary Search Trees

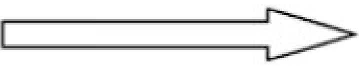
Note Title 10/12/2007



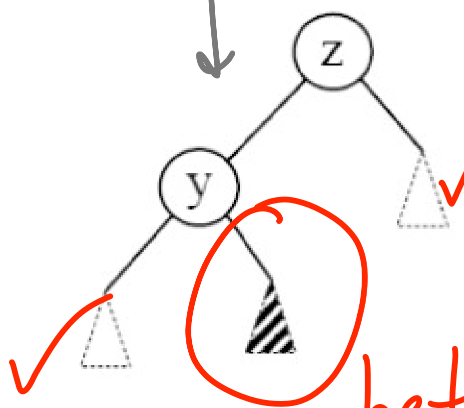
y & z change in their parent/child relation & one subtree reattached



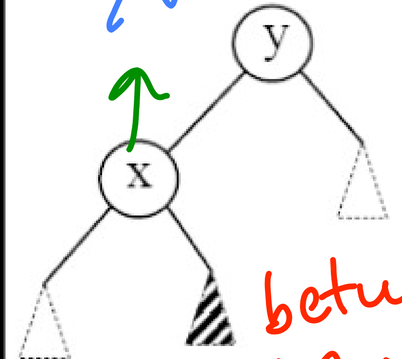
between y & z



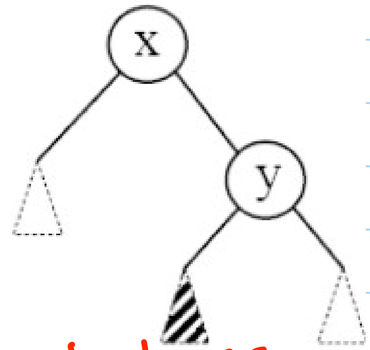
moves "weight" left



between y & z



between x & y

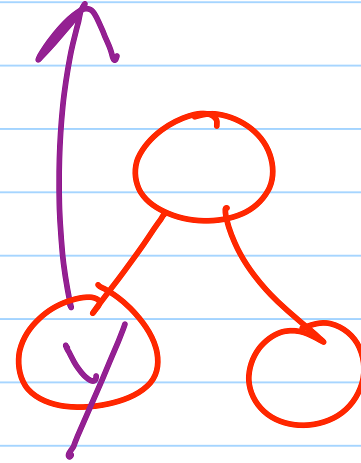


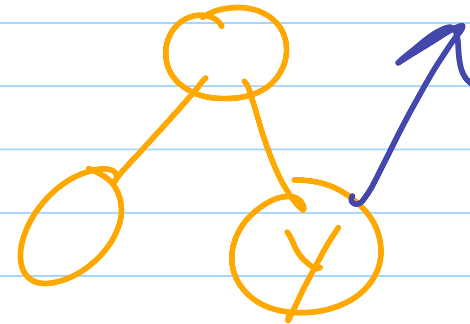
between x & y

Requires parent is not null

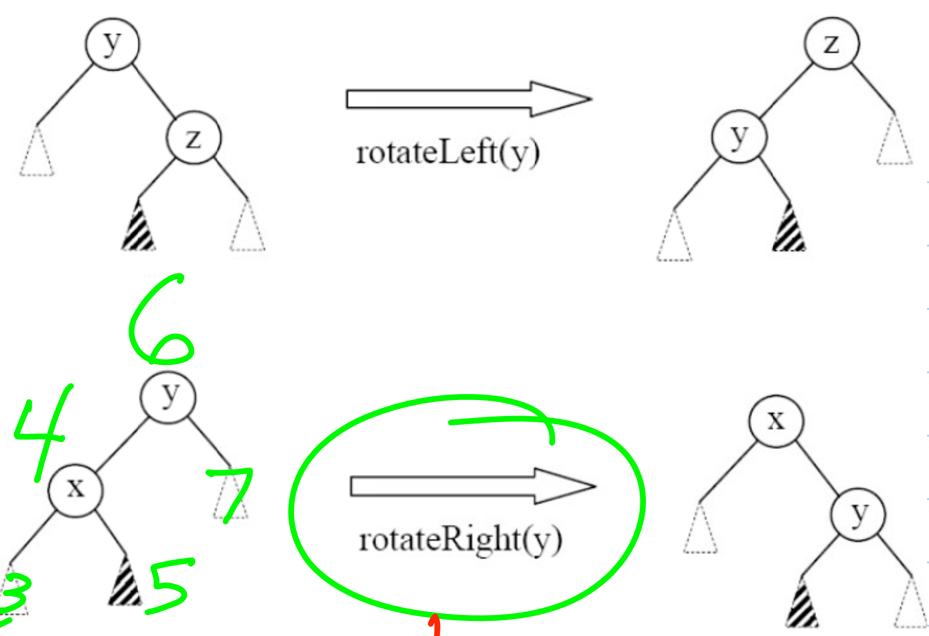
internal

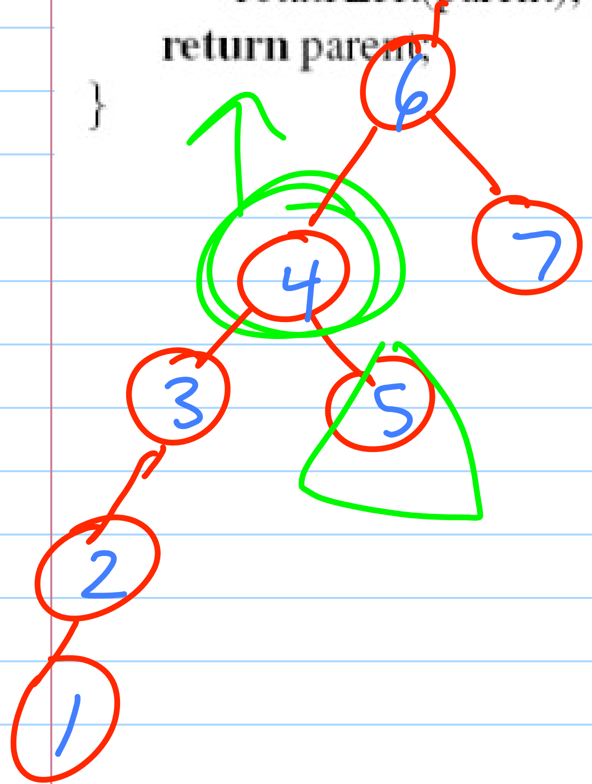
```
BSTNode liftUp(BSTNode y) {
    BSTNode parent = y.parent;
    if (y.isLeftChild())
        rotateRight(parent);
    else
        rotateLeft(parent);
    return parent;
}
```

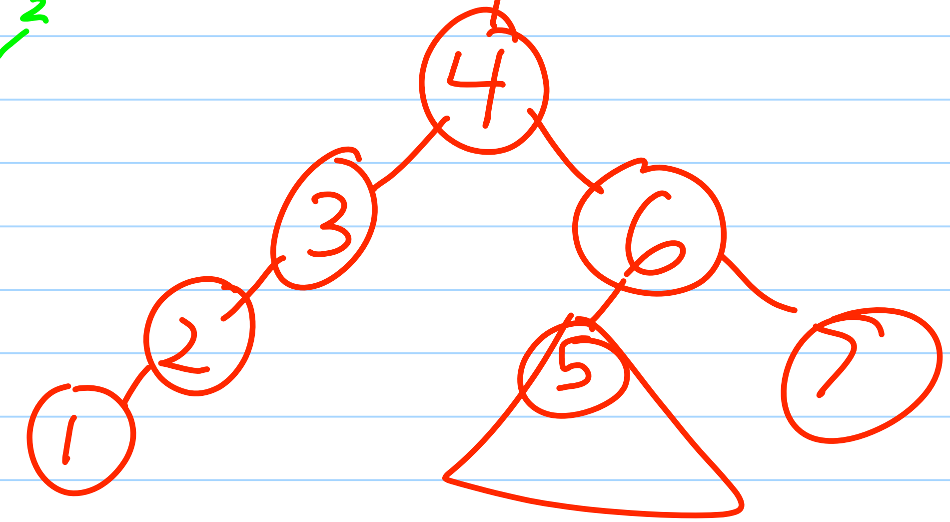




```
BSTNode liftUp(BSTNode y) {
    BSTNode parent = y.parent;
    if (y.isLeftChild())
        rotateRight(parent);
    else
        rotateLeft(parent);
    return parent;
}
```







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