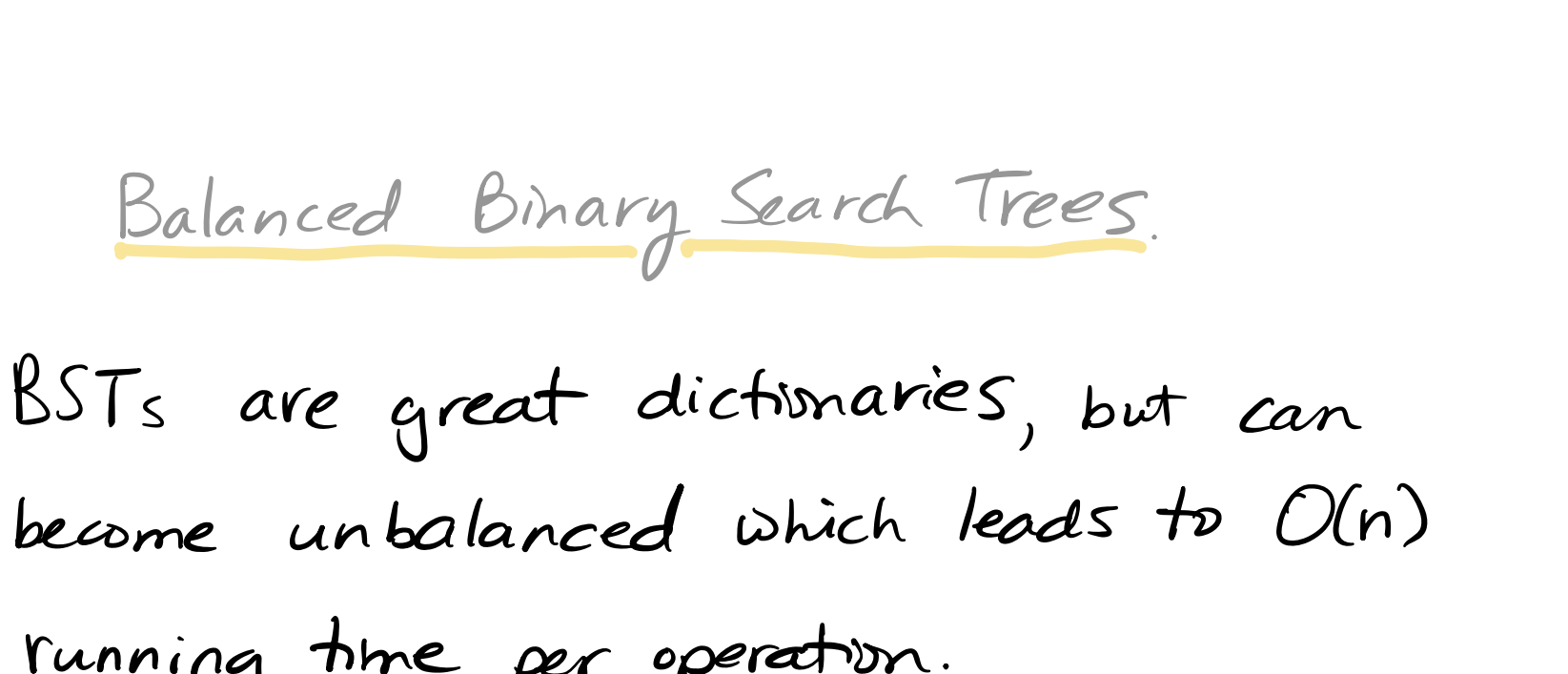
Balanced Binary Search Trees

- BSTs are great dictionaries, but can become unbalanced which leads to O(n) running time per operation.

∴ we seek to do a little "housekeeping" as we go along to ensure the tree does not get out of balance...

Keeping it perfectly balanced....



if you have 11 items, the least-height tree has $h = 3$

If we need to store height h at a node x, the following should always be maintained:

$$x \to h = \max(x \to left \to h, x \to right \to h) + 1$$

Keeping the tree balanced, but not perfectly

Defn: The balance factor at a node x is denoted $x \to BF$ and is given by

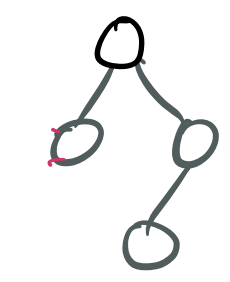
$$x \to right \to h \; - \; x \to left \to h$$

i.e., it is the height of right subtree minus the height of left subtree of x

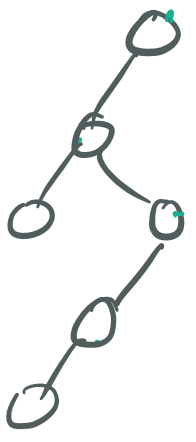
Defn: An AVL tree is a BST where $\forall$ nodes have balance factor $\in \{-1, 0, 1\}$

A note about tree height, $h(T)$.

height is # edges on longest leaf-root path.



$h = 2$



$h = 4$



$h = 1$



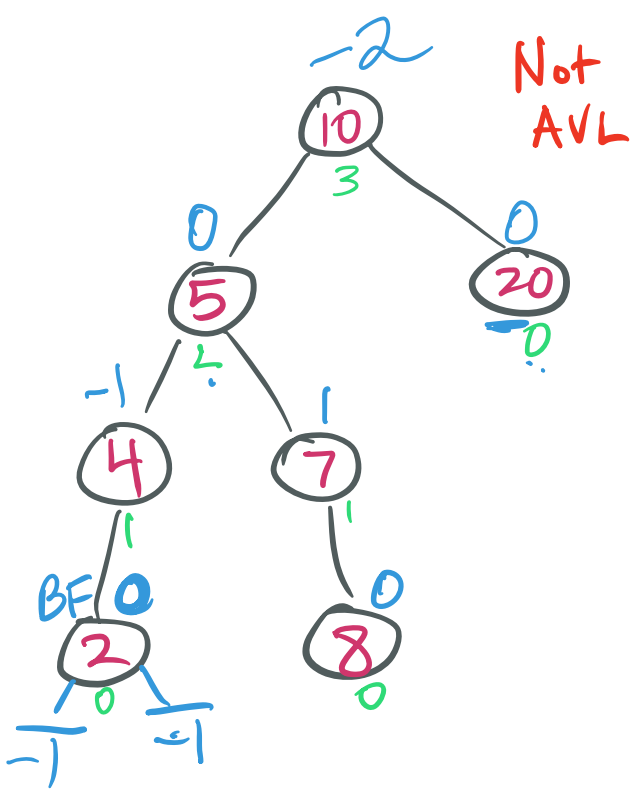
$h = 0$

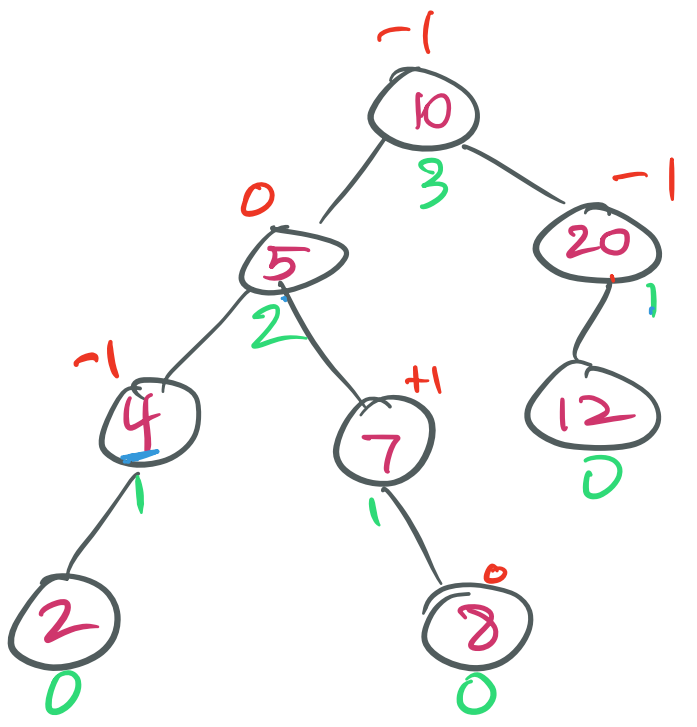
empty tree



$h = -1$

Examples of AVL tree, and not AVL tree





Two things that would make AVL trees interesting:

① Are they "balanced enough" that heights stay $O(\log n)$?

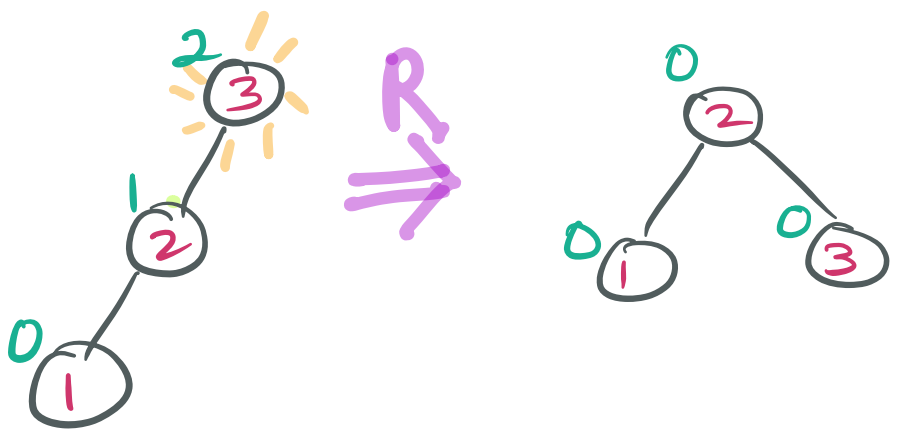
② Can we maintain "balance" (AVL-style) in time $O(\log n)$ per operation?

(Insert, Delete, Search, Max, Min, Succ, Pred)

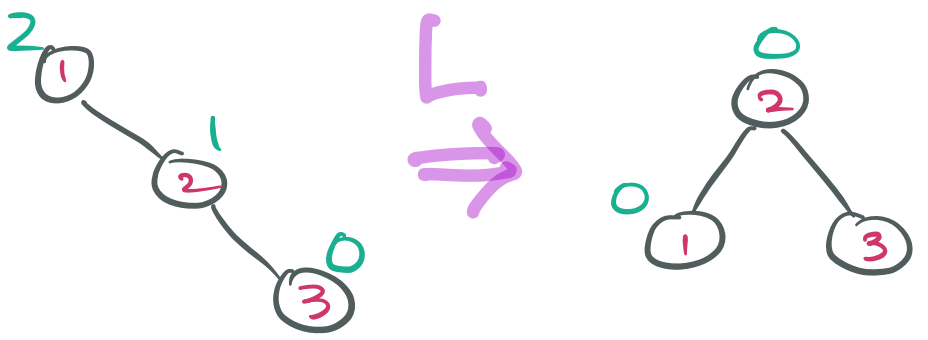
may require rebalancing during operation.

Tools for maintaining balance - AVL style:

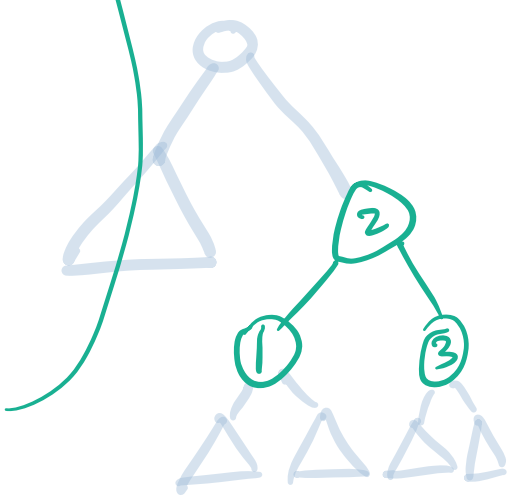
R - rotation



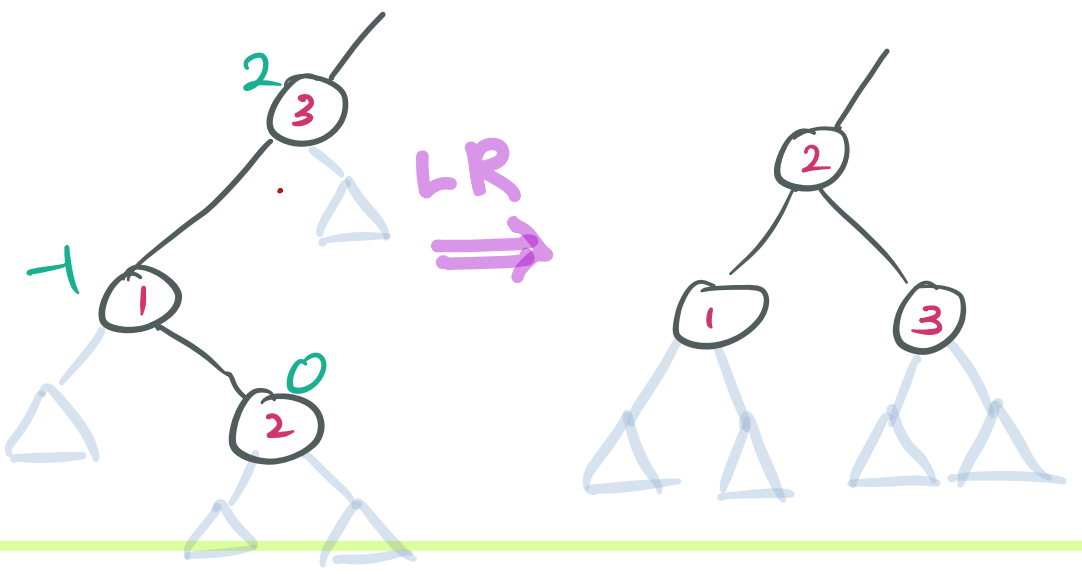
L - rotation



Single-rotations are local transformations.

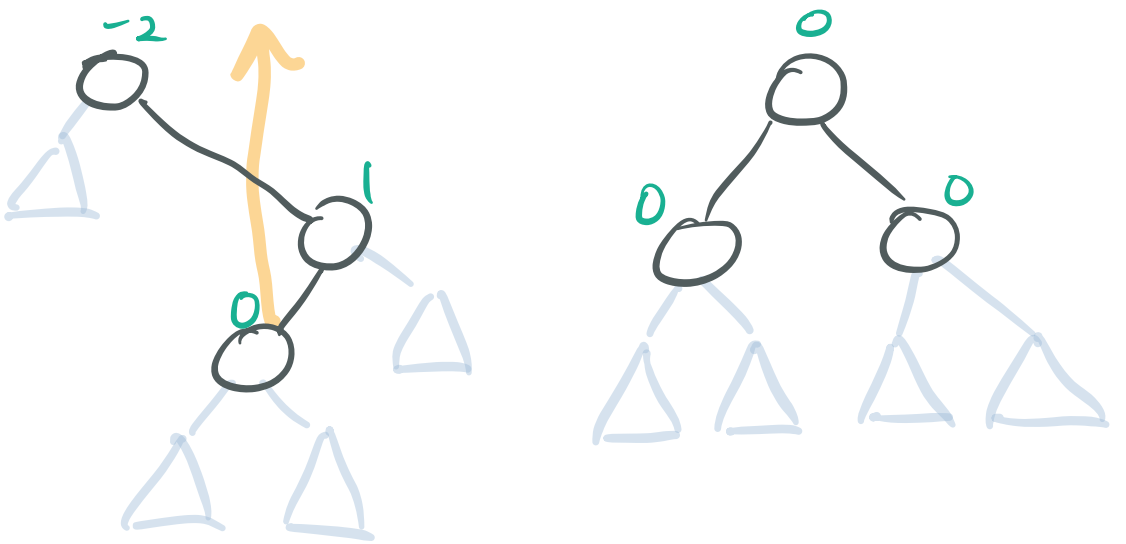


LR - rotation

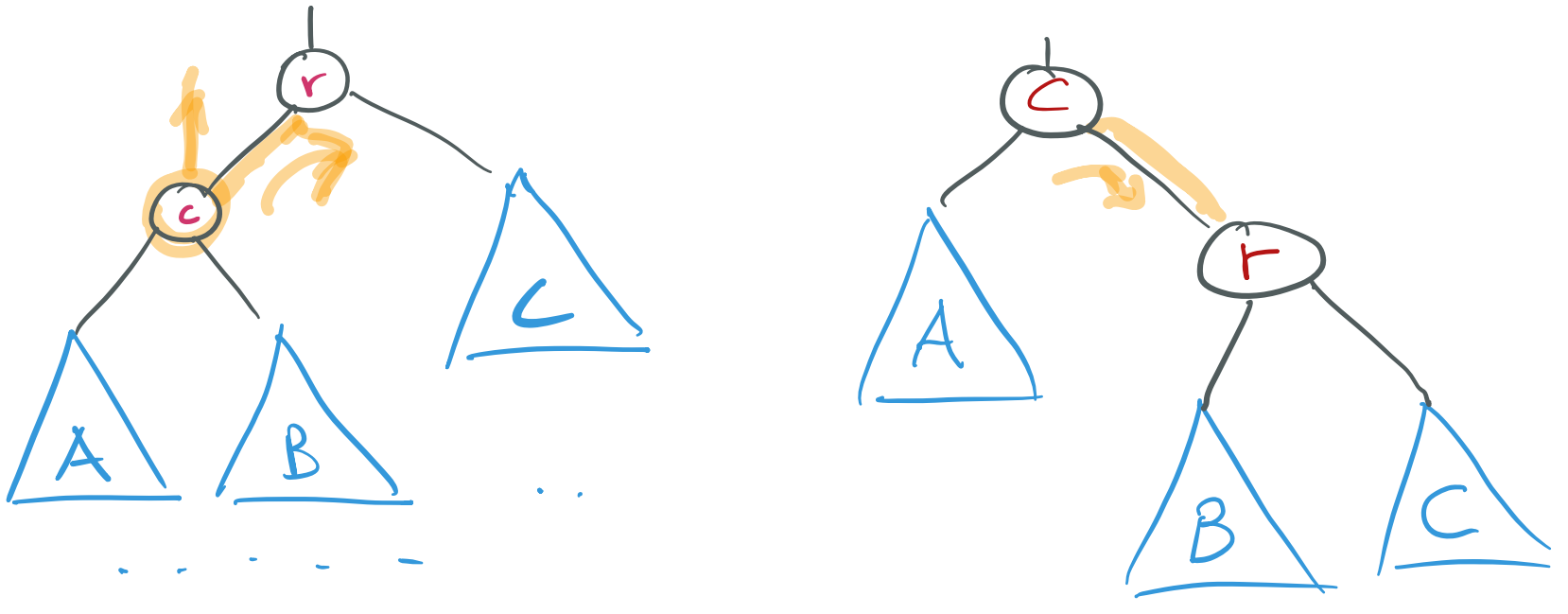


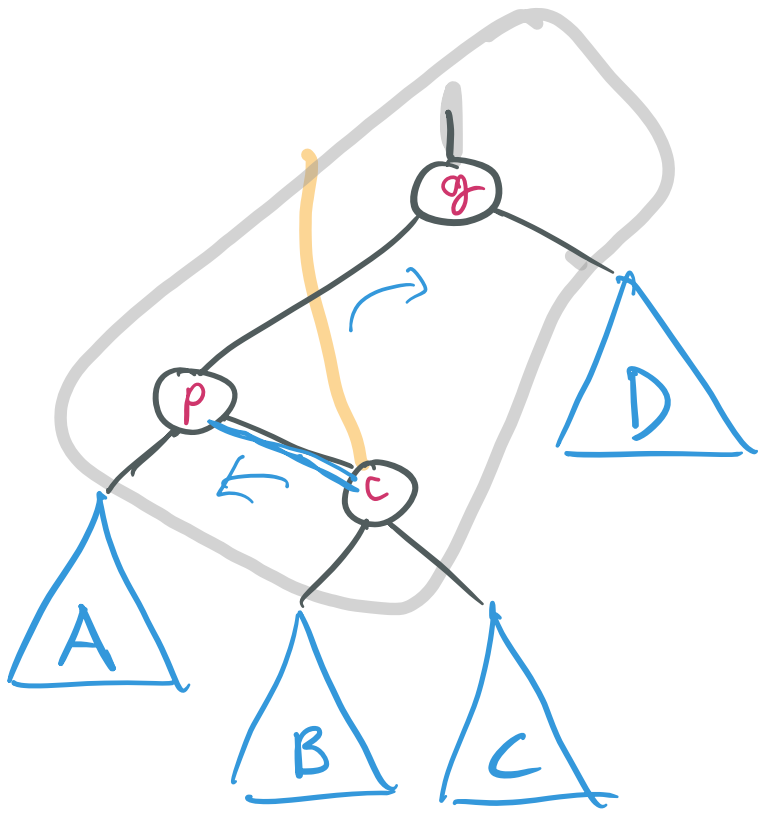
double rotations are also local

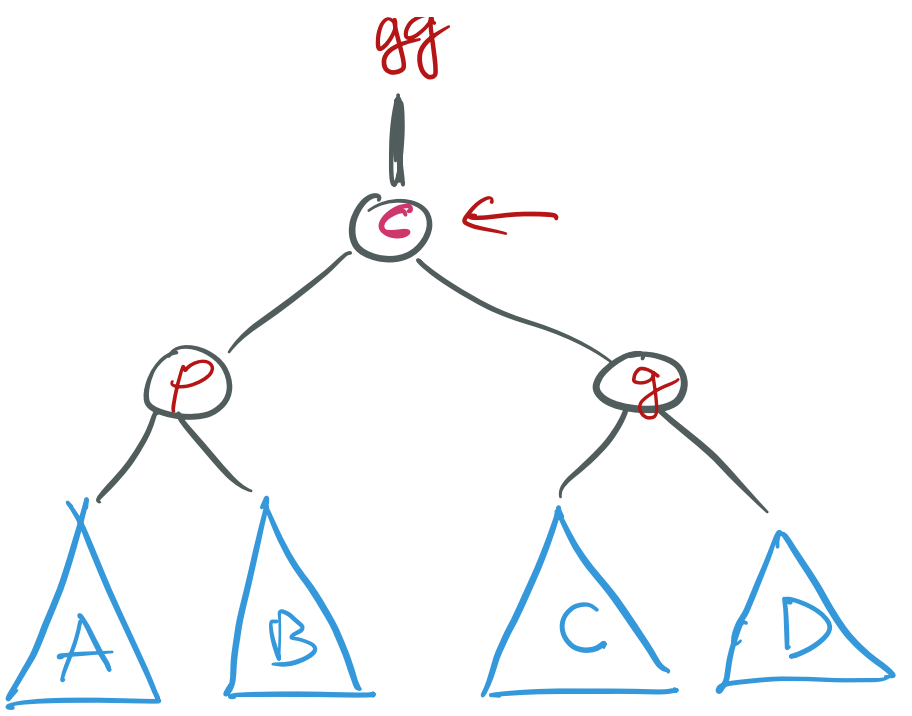
RL - rotation



Single Rotation R







Each can be done in $O(1)$ time, but may propogate up ... total is $O(h(T))$. *

where $h(T)$ is height of the AVL tree.

Is it an effective enough rebalancing that the trees height is kept small ... in $O(\lg n)$?

Theorem: $\forall$ AVL tree T,

$$\lfloor \lg n \rfloor \leq h(T) < 1.4405 \lg(n+2) - 1.3277$$

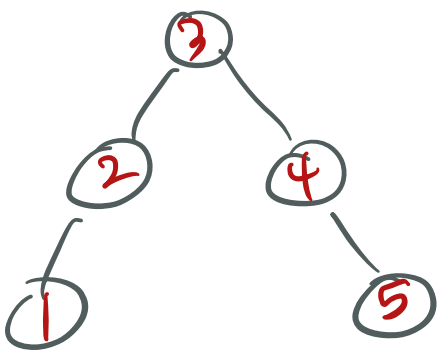
For $n = 1{,}000{,}000$, $h \leq 29$ versus ideal of $h \leq 20$

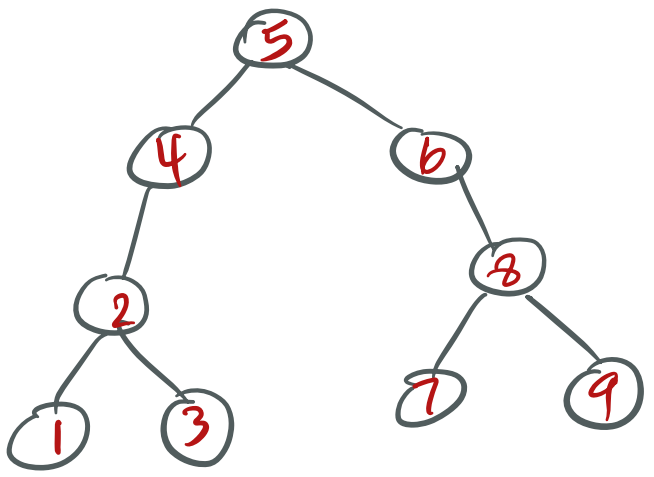
Average case for large n: experiments say

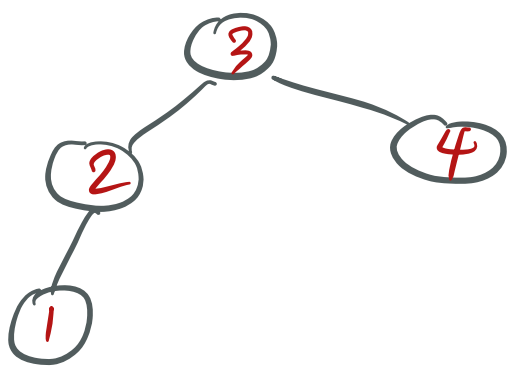
$$1.01 \cdot \lg n + .1$$

Questions

1. Which of these BSTs are AVL?

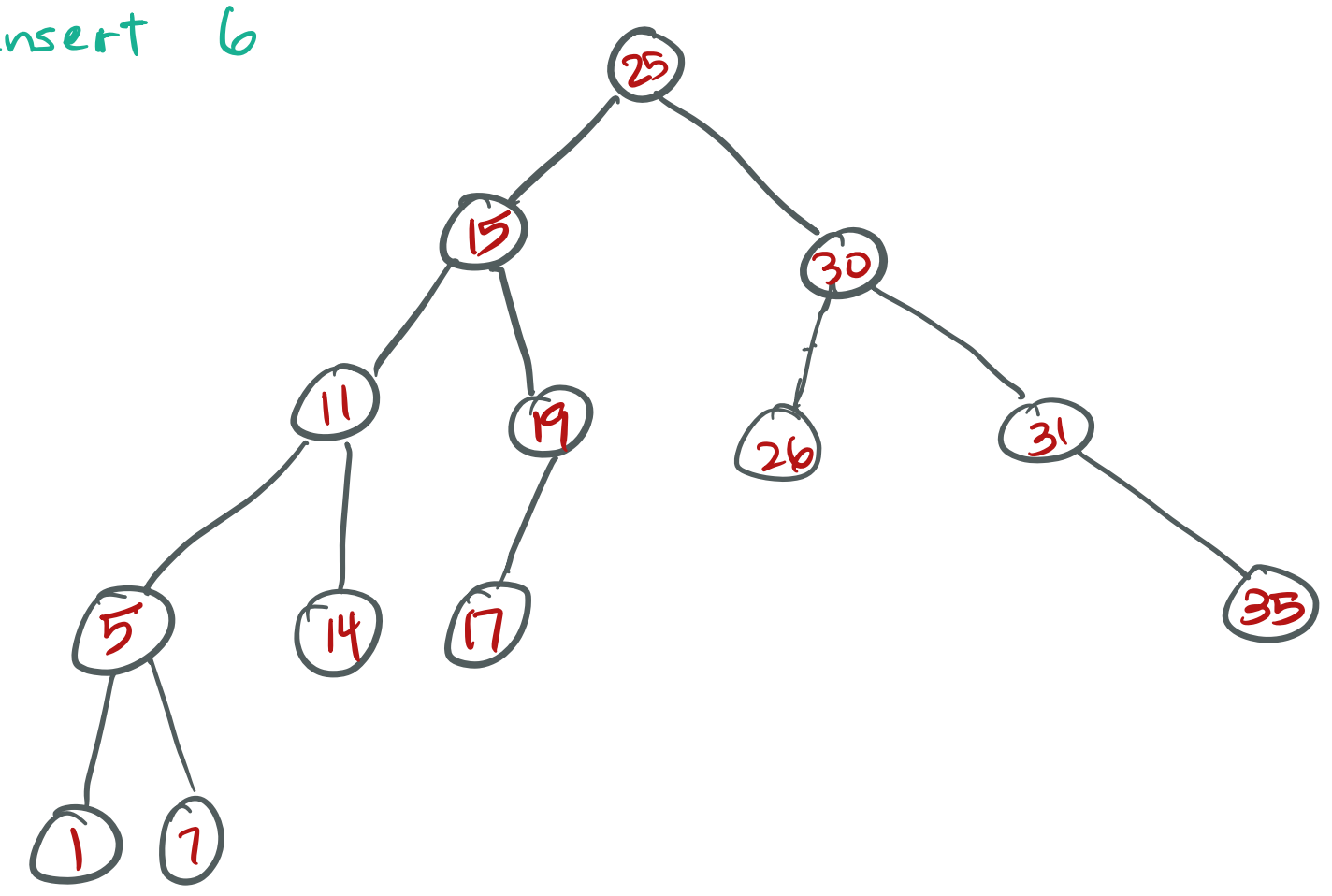






* How might the rotations propogate up?

Insert 6



Wikipedia AVL Tree entry

```
// After insertion at z. which is a child of p, the height of
// subtree rooted at z. has increased as a result. This triggers a
// rebalancing if necessary at z->parent (p)
// z is AVL compliant
for ( p = z->parent; p != NULL; p = z->parent )
    if ( z is a right child )
        if p->BF > 0    // p is right-heavy
            g = p->parent
            if z->BF < 0
                w = rotateRightLeft(p, z)
            else
                w = rotateLeft(p, z)
        else
            if p->BF < 0
                p->BF = 0
                break
            p->BF = 1
            z = p
            continue
```

```
else    // z is a left child
        if  p->BF < 0
            g = p->parent
            if  z->BF > 0
                w = rotateLeftRight(p, z)
            else
                w = rotateRight(p, z)
        else
            if  p->BF > 0
                p->BF = 0
                break
            p->BF = -1
            z = p
            continue
w->parent = g
if  g != NULL
        if  (p is a left child)
            g->left = w
        else
            g->right = w
else
        w is made root of the tree
break
```

// Note: $BF = h_r - h_\ell$

Test Review: Unit Task Scheduling.

Job	1	2	3	4	5	6	7
deadline	3	4	3	3	5	3	6
penalty	15	20	10	2	8	6	4

Job	1	2	3	4	5	6	7
deadline	5	1	5	2	3	2	6
penalty	15	20	2	10	8	6	4

Job	1	2	3	4	5	6	7
deadline							

Prim's, Kruskal's

- How do they work?
- Other variants?

Dijkstra's

- Know how to run it "by hand"