COP 3502 - 10/31/23

(0) PS posted - BST

nodes by circles
customer smaller circles

(1) AVL Trees

- Insert
- Delete

} Cool today

R: Code

R: Heaps

Binary Search Trees → insert, delete, search O(h)

h = height tree

best height = O(lg n) n = # node

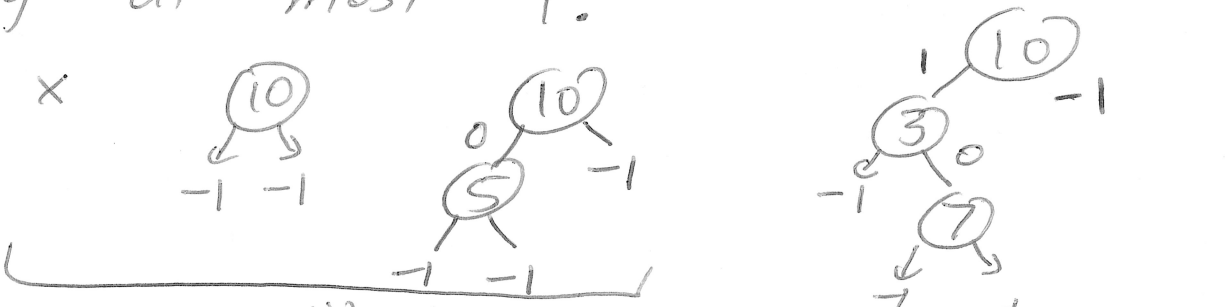
worst height = O(n)

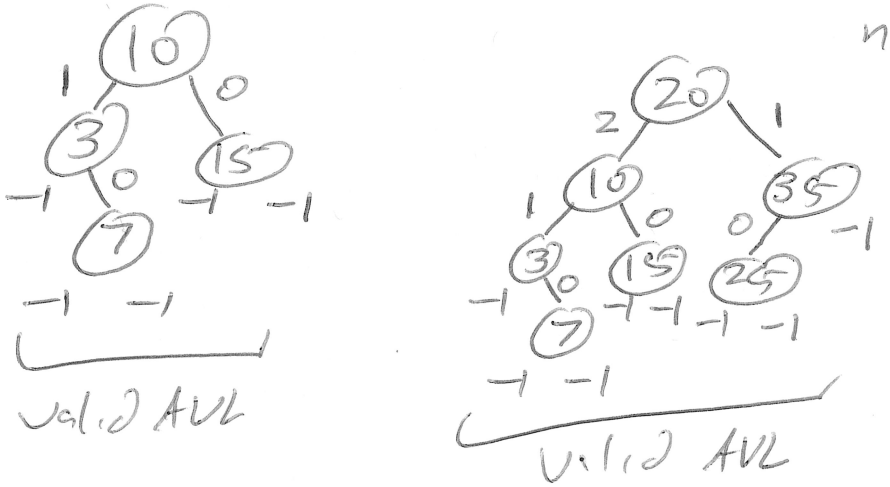
One Balanced Binary Search Tree

We'll do is the AVL tree

AVL Tree Node Property

height of the left + right subtrees differ by at most 1.





Prove that $h = O(\lg n)$, n = # nodes

Let T_h = min # nodes in an AVL Tree of height h.

$$T_h = F_{h+3} - 1$$

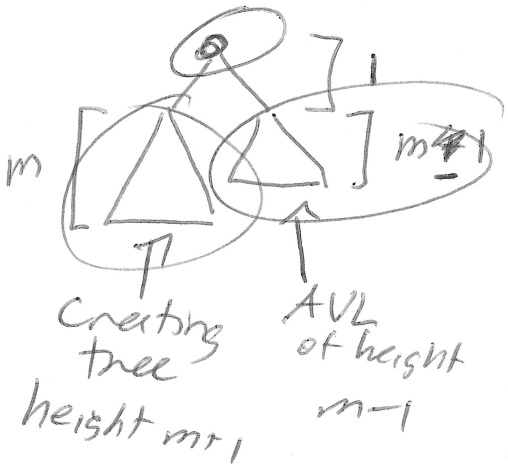
Base cases: $h = 0$, $h = 1$

$F_{1+3} - 1 = 3 - 1 = 2$

$F_{0+3} - 1 = 2 - 1 = 1$ ✓

Inductive hypothesis: Assume for all non-neg int. h, $h \leq m$, $m \geq 1$ that $T_h = F_{h+3} - 1$

Inductive step: Prove for $h = m+1$ that $T_{m+1} = F_{m+4} - 1$.



root ↓ bigger side ↓ smaller side ↓

$$T_{m+1} = 1 + T_m + T_{m-1}$$

$$= 1 + F_{m+3} - 1 + F_{m+2} - 1$$

$$= \boxed{F_{m+4} - 1}$$

$$n = T_h = F_{h+3} - 1$$

$$n \geq F_{h+3} - 1$$

$\phi \sim 1.6$

small → limit to 0

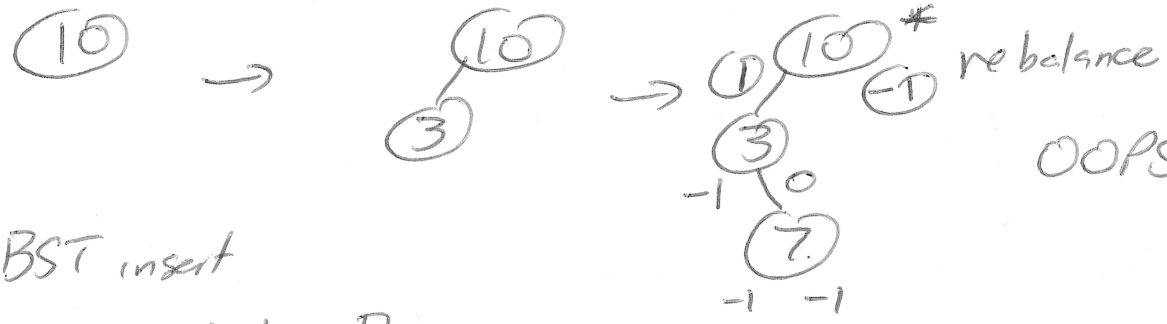
$$n \geq \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{h+3} - \left(\frac{1-\sqrt{5}}{2}\right)^{h+3}\right) - 1$$

$$\log_\phi n \sim \frac{1}{\sqrt{5}}\phi^{h+3}$$

$$\log_\phi n \sim (h+3) \times \frac{1}{\sqrt{5}} \times \boxed{\log_\phi \phi}^{1}$$

$$h = O(\lg n)$$

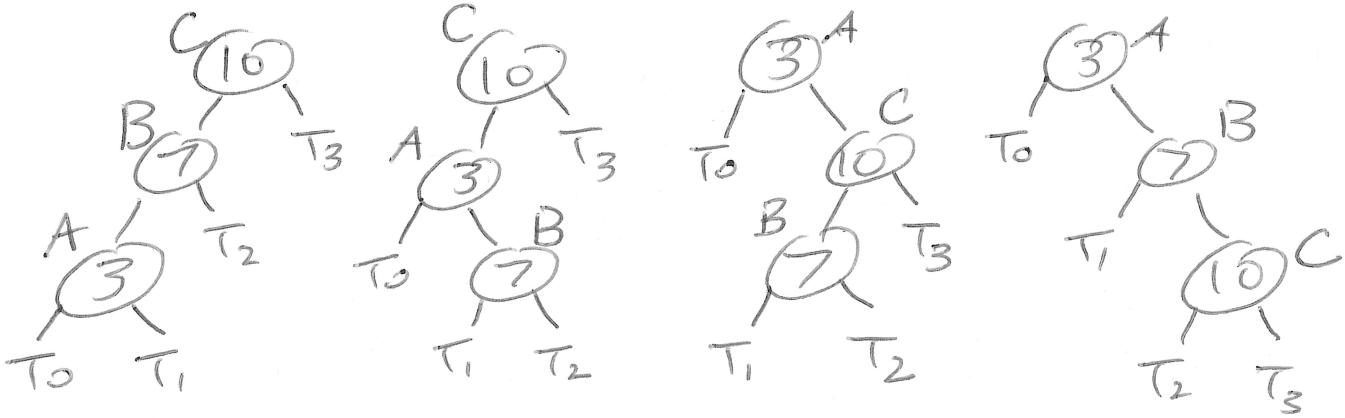
Difficulties of maintaining property



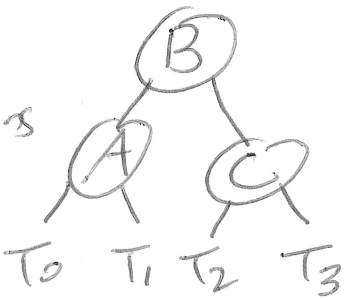
Insert

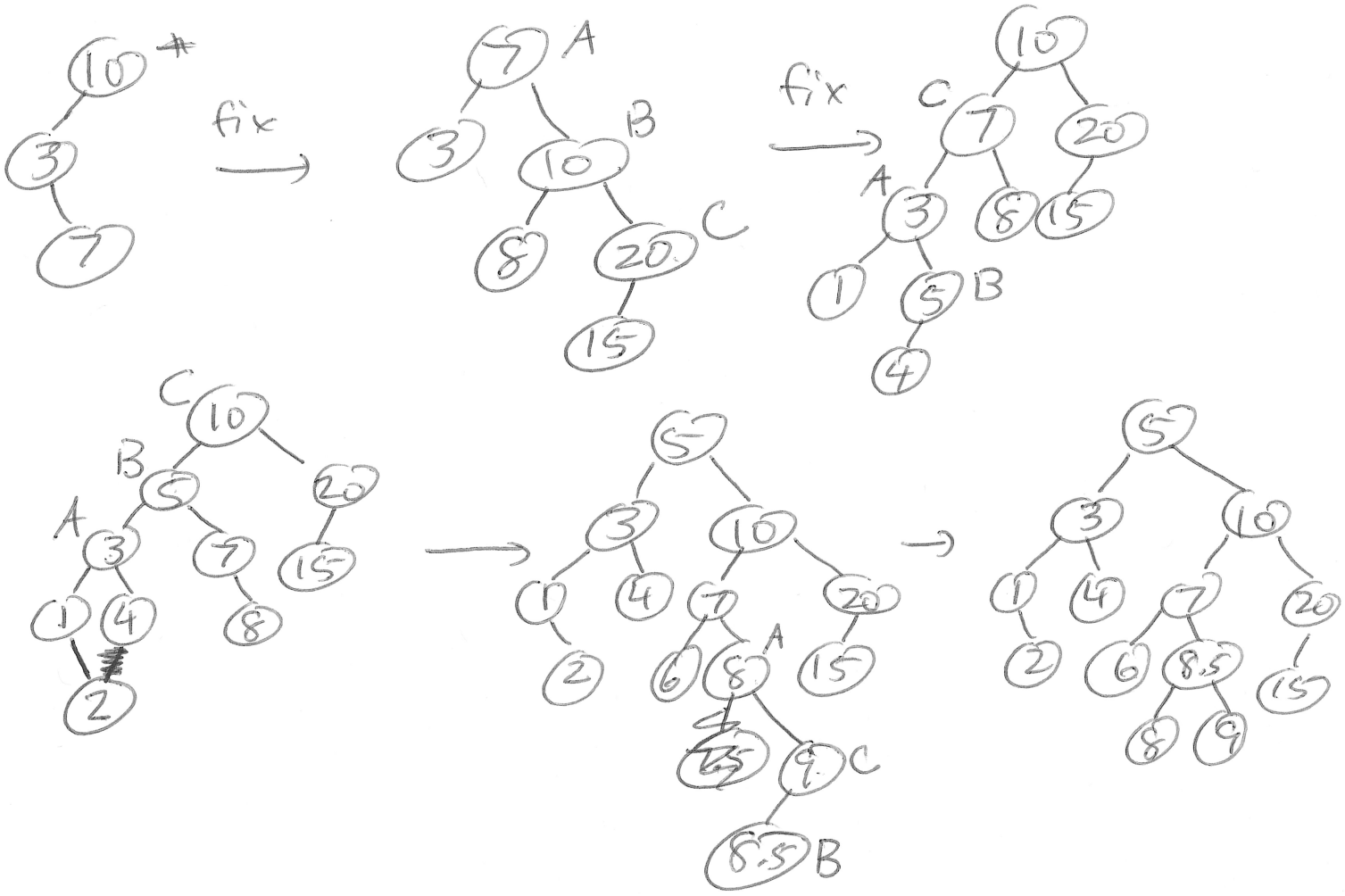
1. Do reg BST insert
2. Trace up ancestral path
   If you find an imbalanced node, rebalance it!

4 ways we could have imbalance



all 4 situations same fix





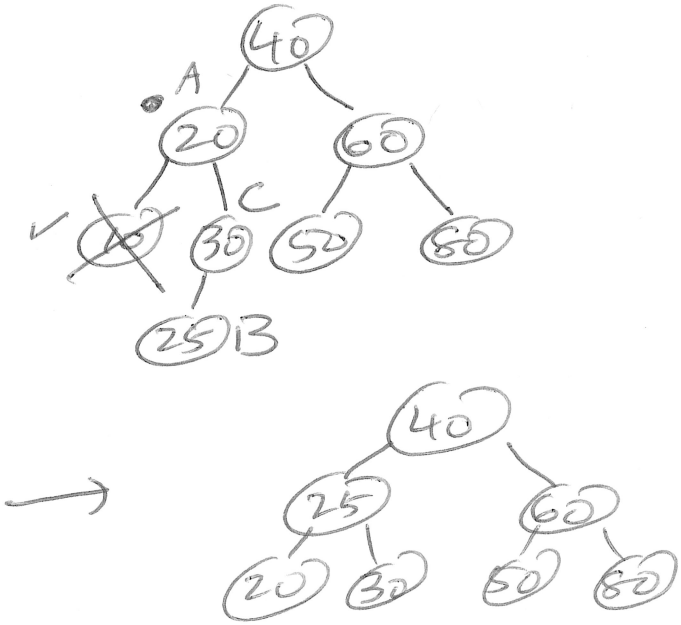
Delete! (Iterder)

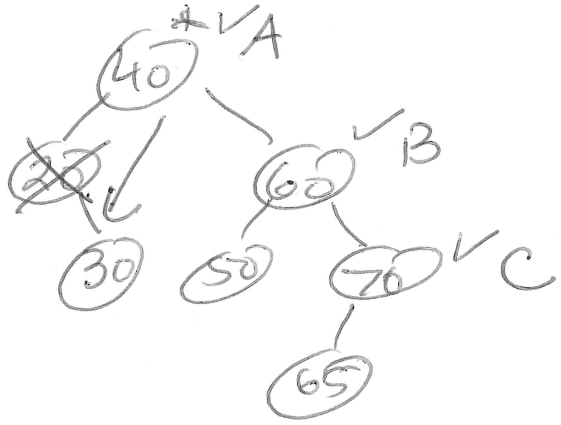
Delete 10

1) regular BST delete
2) trace up ancestral path

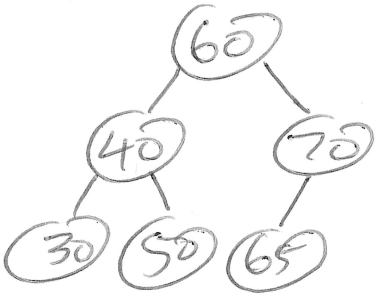
If a node is unbalanced balance it!

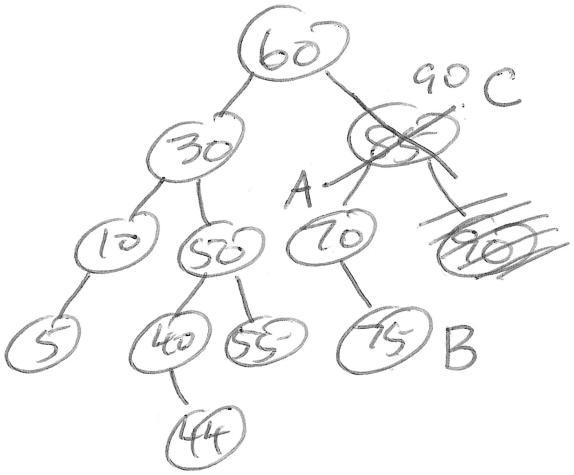
* more than 1 rebalance may be necessary!





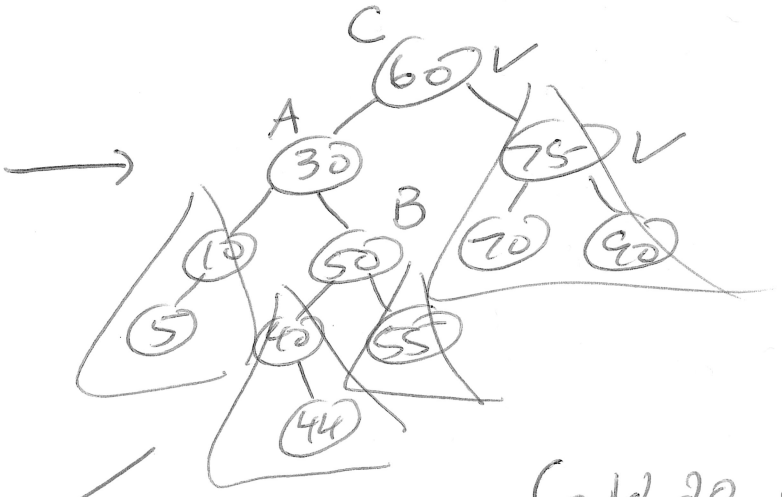
Delete 20

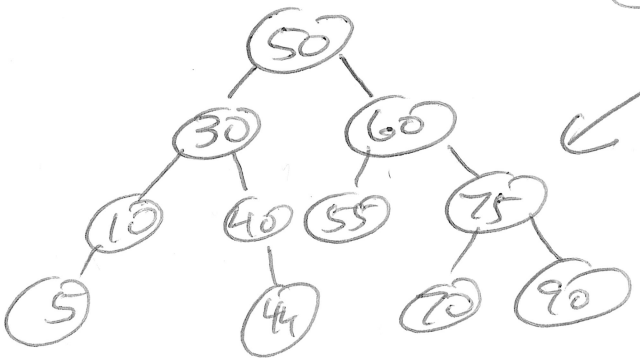




Delete 85

Replace w/ min right





Could do a rebalance at multiple nodes ancestral path but there are $O(\lg n)$ such nodes + each rebalance is $O(1)$ time.

Overall runtime $O(\lg n)$