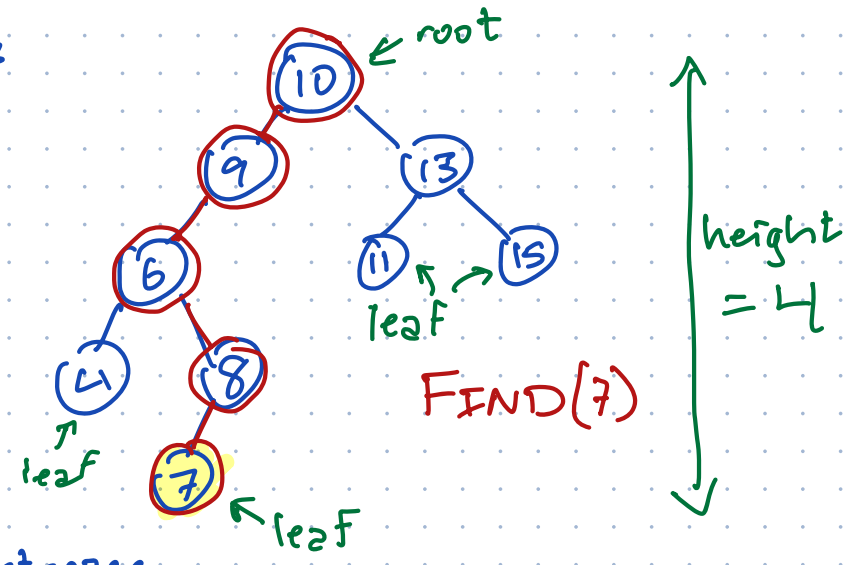
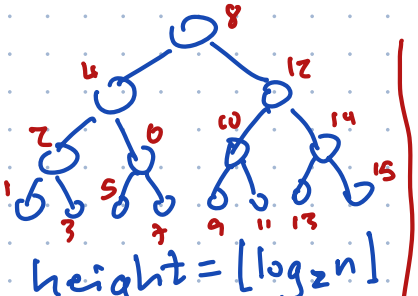
Admin: Anakin is traveling this Saturday
↳ HW4 due Tuesday 9pm

Binary search trees



Perfectly balanced



height = $\lfloor \log_2 n \rfloor$

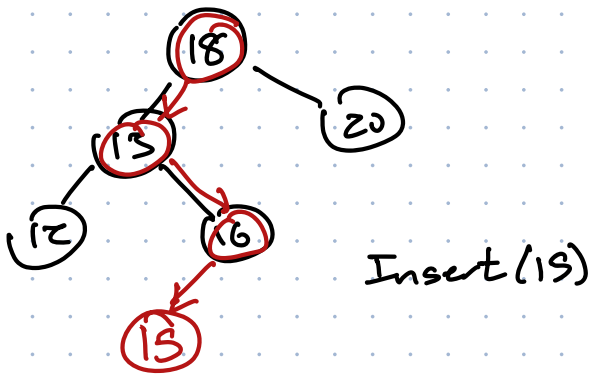
FIND = binary search

Worst-case Time for FIND = Θ(height of tree)

DICTIONARY / ASSOCIATIVE ARRAY / MAP

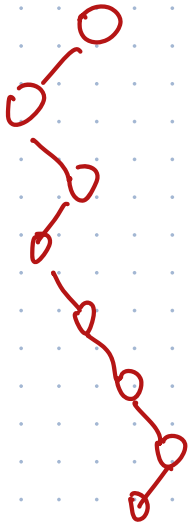
Store (key, value) pairs

- Find (key) returns value if (key, value) is in the set
- Insert (key, val)
- Delete (key)



Binary search tree:

All operations take $\Theta(n)$ time in worst case



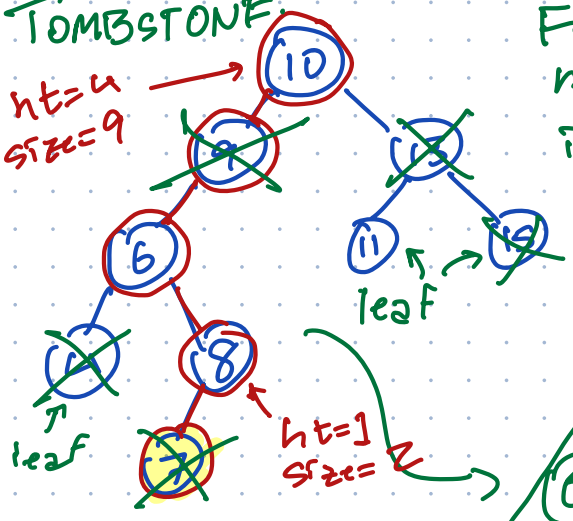
How do we keep BSTs balanced?

In class: AVL trees
⇒ $\Theta(\log n)$ worst case

Here: Scapegoat tree.
⇒ $\Theta(\log n)$ amortized time

Delete

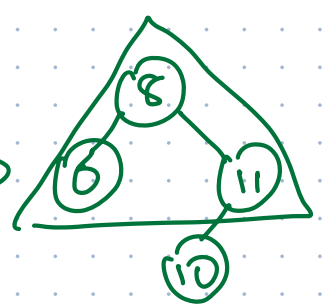
TOMBSTONE!

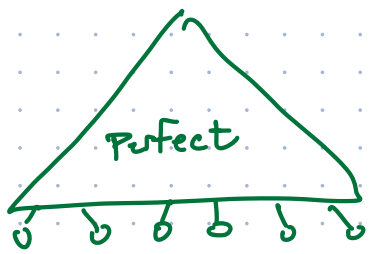


Delete(x):

Find(x)
mark x "dead"
if half nodes are "dead"
rebuild the tree

GLOBAL REBUILDING
$\Rightarrow +O(1)$ am. time





Rebuild:
Transcribe T into array A via inorder traversal
~~median~~ $A[\lceil n/2 \rceil]$ → root
Rebuild($A[1..\lceil n/2 \rceil - 1]$) → left
Rebuild($A[\lceil n/2 \rceil + 1..n]$) → right

$\Theta(n)$

$\Theta(n)$ extra space (in addition to T)
Hw: $O(\log n)$ extra space *$O(1)$

Insert: LOCAL REBUILDING

Declare a node v to be unbalanced
iff $\text{height}(v) \geq \log_\beta(\text{size}(v))$

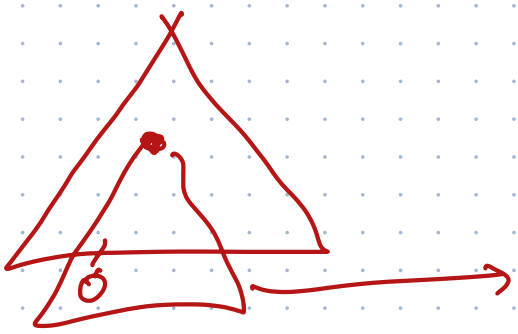
dist to deepest leaf descendant

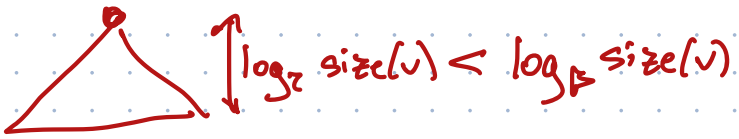
descendants of v

Constant $1 < \beta < 2$

Insert strategy:

After Inserting x, follow search path upward
If we find unbalanced node v, rebuild v's subtree





Rebuilding at v costs $\Theta(\text{size}(v))$ time

Charge that to earlier insertions into v's subtree.

$$\text{Imbalance}(v) = \left|\text{size}(\text{left}(v)) - \text{size}(\text{right}(v))\right|$$

Lemma: Just before rebalancing at v,

$$\text{Imbalance}(v) = \Omega(\text{size}(v))$$

⇓

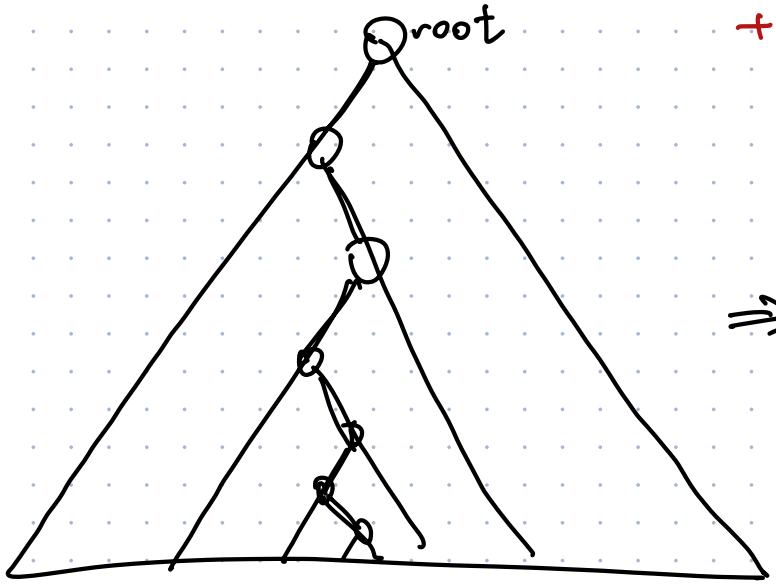
Charge time to rebuild v's subtree

to earlier insertions into v's subtree

Am. cost of Insert = $O(\log n)$

+ $O(1)$ per subtree

$O(\log n)$



⇒ Insert takes

$O(\log n)$ am. time