CS 2100

Finishing AVL trees

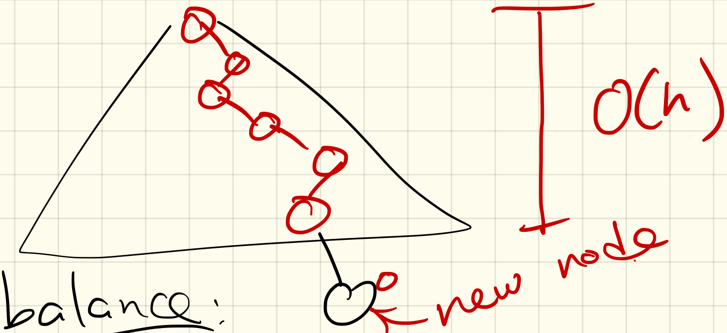
Recap

- Zybook reading: Friday & next Monday
- HW due Saturday
- Next HW (written) is posted: due Monday April 8 at 2pm
  NO EXTENSIONS
- Review session: Monday April 8
- 2^{nd} midterm: April 10
  Covering up through heaps (which we'll do tomorrow)
- Practice midterm coming Monday
- Lab next Thursday: grafting (part 2)

Recall: AVL insert

→ Do BST insert

↳ this gives a new leaf, v



rebalance:

```
v = v -> parent v
while (not above root) {
    reset v's height
    if v is unbalanced
        z = v
        y = v's higher child
        x = y's higher child
        pivot(y) or pivot(x) twice
    else
        v = v.up
}
```

could break

AVL Remove:

Do BST remove

Need it = parent of *actual* node removed
(lower node)

reset it's height

loop to travel up
& rebalance (*don't* do a break)

Note one difference:

Each insert will trigger
at most 1 set of pivots

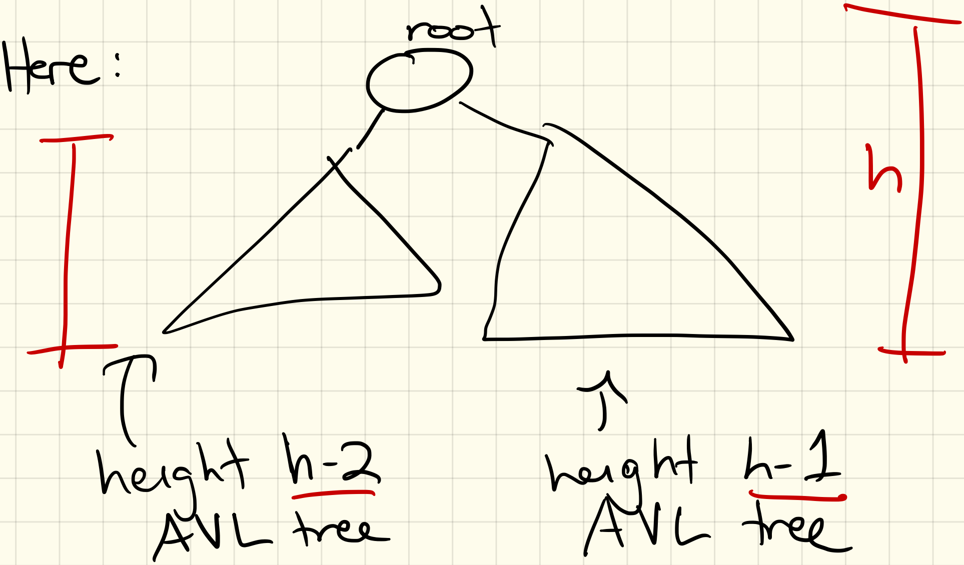
In remove, may have to
pivot at *every* level

Runtime:

For find, insert + remove, worst case is that we traverse a root-to-leaf path (maybe twice)

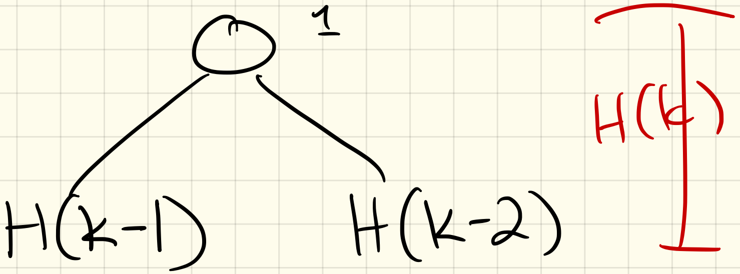
So: $O(h) = O(\log_2 n)$

Here:



(Assume worst case each time, so $H(k)$ is # of nodes in worst unbalanced tree)

Result: a recurrence!



$n =$ #nodes in AVL tree of height $\underline{k}$

$$= H(k) \geq 1 + H(k-1) + H(k-2)$$

$$\geq 1 + \underline{2}\,\underline{H(k-2)}$$

$$\geq 2H(k-2)$$

$$= 2(2H(k-4))$$

$$= 2(2(2H(k-6)))$$

$$\boxed{n \geq 2^{k/2}}$$

$$\log_2 n \geq \frac{k}{2} \Rightarrow \overset{\overset{\text{height}}{\shortparallel}}{k} \leq \boxed{2}\log_2 n$$

Vectors

insert: $O(n)$

access (find): $O(1)$

Lists

Insert: $O(1)$

access: $O(n)$

Balanced BST:

$O(\log n)$

only: find
remove
insert

Tomorrow: Heaps