CS2100

Last day of BST
Recap

- HW due last night
- AVL tree HW (#9) due next Thursday
- Review next Friday
- MT2 following Monday
Last time,

Remove in an AVL tree.

AVL key: height-balance property

\[ h(v) = \max\{\text{height of child} + 1\} \]

these must be \( \leq 1 \) apart

(Nulls are \(-1\) height)
Insert:

→ Do BST insert

\[ v \text{ this gives a new leaf, } v \]

rebalance:

\[ v = v \rightarrow \text{parent of } v \]

while (not above root) \[ z \]

reset \( v \)'s height

if \( v \) is unbalanced

\[ z = v \]
\[ y = v \text{'s higher child} \]
\[ x = y \text{'s higher child} \]

pivot \((y)\) or pivot \((x)\) twice

else \[ v = v \text{, up} \]

(\text{could have done a break})
Remove

→ Do BST remove of parent of node removed

Need it = \[ \text{actual \ node \ removed} \]

(lower node)

reset its height

loop to travel up

at 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
Runtimes:

In each of insert/remove:

\[ O(\text{height}(t)) \rightarrow \text{for BST} \]

Then, starting at a leaf, travel up to root \( O(1) \) each time, reset height \( O(1) \) time.

loop that \( O(\text{height}) \) + do \( \leq 2 \) pivots

Repeat \( O(\text{height}) \) times.

Key - height?
Worst case:

Every node has children 1 apart.

Note: know $h(\text{tree w/n nodes}) \geq h(\text{tree w/n-1 nodes})$.
# nodes in entire tree =
1 + # nodes in height h-1 tree
+ # nodes in height h-2 tree

≥ 1 + 2^0 (# nodes in h-2 height tree)

T(h) ≥ 1 + 2T(h-2)

= 2T(h-2)

Know T(h-2) ≥ 2T(h-4)

So:

T(h) ≥ 2 (2T(h-4))

n ≥ 000 2^h/2

⇒ n > 2^h/2

⇒ \log_2 n ≥ \frac{h}{2} \Rightarrow h < 2 \log n
Red-black trees

Insert new node, color it red.

If red node has a red child, rebalance.

Color both nodes black.

$h \leq 1.4 \log_2 n$

Note: search for applets or demos.