CS180 - AVL Trees (part 2), Heaps

Announcements:

- Program coming today!
- Implement insert method in AVL tree code
Null

Height = 1

x
x
x

\text{Height of tree} \leq \log_2 n

\text{Height - Balance Property:}

\forall \text{ trees :
What was the running time?

$O(\log n)$ - Why?

**Insert:**

- Search where new element should go
  - $O(h)$
- Go back up and find lowest place where height balance is broken
  - $O(h)$
- Rotate & fix heights - $O(1)$
  (only 1 rotation is necessary!)
Remove (3a)
(but a bit more complicated)

Remove IS similar
What can this be to parent of this

remove (w)

weight = y

relevance

weight = z

weight = 3
Since fixing the subtree can reduce the height by 1.

(One at each level)

Helping branches (heap) - weight of O(log n)

Tree time of traverse.
Complicated

So faster, brute force and more

Pre: only O(1) rotations per operation

O(\log n) for search, insert, delete

Red Black: same worst case height

Yes, which is slower in practice

Downside: delete many nodes O(log n)

\[ \text{max height} = \frac{1}{2} \log n \]

- All: O(\log n) to search, insert, delete

Some search tree, all afterwards
A heap is a BST over the values and nodes will contain both values and priorities. A heap is a BST over the values and nodes will contain both values and priorities. A heap is a data structure: heaps.
Example:

BST

Min Heap

keep over numbers

BST over letters
Insert (s, 0)

Violate heap

Satisfy BST
Wrong: but priorities are x, y in BST order.

Ref: need a BST, not just swap values like in heap. Since
insert(S, 0)
Downside: heght can be 0cm
Expected height of random heap:

random events are expected in a series of

Expected value: number that is

Priority assigned to it,
each element will get a random
Randomized heap