CS2100

Traps (cont)
Recap:
- HW due Thursday
- Lab tomorrow
  (also one next week)
- Review Friday, test in 1 week

Goal: Each node will contain a value (like a BST) and a priority (like a heap).

- BST over values
- Heap over priorities

Ex: Suppose values are names and priorities are integers.

Both can be "sorted":

- Values/names have alphabetical order
- Integers (obviously)
Example

Insert: \((Q, 101)\)
Removing:
- Do BST remove
- Fix priorities

Note: pivot up or down

remove (H,50)
remove (H, 50)

pivot (E)

pivot F
Implementation:

- Inherit from binary search tree
  - data: values (letters)
  - aux: priorities (ints)

- use BST's insert/remove to fix

Avoid: AVL's get/set height
Note: Treaps are unique!

Given a set of values/keys, order of insertion is irrelevant.

pf: Consider one valid treap w/ set of values + keys.
Consider x, a node.

If we change x's height: 
means child/parent swaps - violate priority

If we change x's order: 
violates BST
Example: Draw a heap with:

- (A, 2), (C, 4), (Q, 11),
- (X, 2), (Z, 5), (M, 3)

Worst case height,
Randomized treaps: Balanced BS1

Alternative to AVL trees.

Given a value to insert, give it a random priority.

Thus: Expected height of the treap will be $O(\log n)$.

Why? Remember quicksort:

\[ \frac{1}{3} \]

root's priority to be middle of range

w/ prob. \(\frac{1}{3}\), get "good enough" root
From here:
These will be on written HW, due towards end of semester.