CS 2100 - Trees

Announcements

- HW 4 was emailed back to one partner (if you had a partner)
  yesterday
  (Email me w/ any issues)
Treaps: a new binary tree data structure

- Nodes will contain both values and priorities

- A treap is a BST over the values and a heap over the priorities.
Example:

Priorities form min heap: priority at a node is ≤ priority at both children.

Values (alphabetical order as my comparison for BST):
Insert

insert: (S, 0)

(6) problem meeting heap requirement
In a heap, we "bubbled up". Will that work here?
Try AVL operation: pivot
Rotations

$x$ & $y$ are in correct BST order, with $x < y$, but priorities are wrong.

$\xrightarrow{\text{Fix:}}$
So: insert procedure:

Insert as in BST using value
while (my priority < parent's priority)
pivot
Downside: What can height be?
In: Can we force them to be (an)balanced?
In fact: Treaps are unique!
Order of insertion does not matter:

- If you try to change height of a node, will violate heap property.
- If you try to move position, will violate BST property.

(like having 2 traversals)
Ex: Draw treap with \((A, 4)\), \((C, 2)\), \((X, 11)\), \((M, 3)\), \((Q, 1)\), \((Z, 5)\).
Randomized treaps:

Alternative to AVL trees.

Each element will get a random priority.

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

Insertion: (Q, A, L, M, Z, R, ...)

random priorities: 13, 82, 50, 46, ...
Code: How do we implement?

Inherit from `BinaryTree.h` (or `BinarySearchTree.h`)

- `aux` to be `float` (?)

- `find` - same as BST
- `insert` - 5 slides back
- `remove`: