Announcements

- HW4 graded
- Midterm grades submitted
- HW6 due tonight
- HW7 is up - due Monday
Today:
- Finish code for BinaryTree.h
- Start BST.h
**Find in a BST - find(x)**

Start at root:
1. **it = root**
2. **if x == (* it)**
   - **return it**
3. **else**
   - **if x > * root**
     - **it = it. right**
   - **else**
     - **it = it. left**

repeat
Insert in a BST

Find
when hit a "leaf"
put new value
as a child
Remove in a BST

Several cases: let \( v \) be our target node to delete

When is it easy?

leaf - easy
Case 1: $v$ is a leaf or $v$ has only 1 child.

Ex:

14

16

14

16

15

18
Case 2: \( v \) has two children

One of the neighbors in an inorder traversal.

What can go in \( v \)'s spot?
Ex:
Key: Next node in an inorder traversal can have at most one child.

Why? It can't have a left child.

(Why?)

If it had a left child, that comes first in an inorder traversal.
Delete:

Find node v:

if only 0 or 1 child
  delete and promote (x) child

If 2:
  find next node w in inorder traversal
  copy it to v
  delete w (promote child)
Operator ++

Inorder traversal:
  go left
  self
  right

in tree:
Runtimes

Worst case: $O(n)$

In fact, worse than list or vector implementation \( \Rightarrow O(n) \).
Consider this tree:

1
2 3 4 5 6
7 8 9 10

Take out a piece of paper.

Redraw it and make this as good as possible.

(balance the tree)
Possible answers

$\log_2 n$ 7

Balanced tree: many possibilities

Goal: $\log_2 n$
Balanced binary tree

- AVL trees
- Red-Black trees
- Splay tree

- keep $O(\log n)$ time
AVL Trees

Height - Balance Property:

For every node of the tree, the heights of the children differ by at most 1.

\[ \Rightarrow \text{max height} \leq 2^{\left\lfloor \log_2 n \right\rfloor} \]

(How do we calculate height again?)
Ex.
Now: How can we mess this up?

(In other words, how can the height change?)
Insert:
insert(54)

Fix:
So, consider the lowest node which does not satisfy the height-balanced property. Call this \( z \).

Let \( y \) be \( z \)'s child with larger height.

Let \( x \) be \( y \)'s child with larger height.

Now - fix it!
What did you do?
tomorrow:

Another - insert (4,9)
So: consider the lowest node which does not satisfy height-balance property U - call this

Let be t's child with larger height.

Let be y's child with larger height.

Now - fix it!
What did you do?