Adv. of List is that you can splice things into the middle with low cost.

Problem: Finding the right spot is slow.

What if? — had a way of quickly getting to the middle of the list?
Binary Tree:

- Middle
- Middle of left half
- Middle of right half
Representation

Invariant:
For all nodes,
- everything in the left subtree is smaller
- everything in the right subtree is larger

5

1, 2, 3, 4, 5

Want:
Nice bushy tree

5, 7, 2, 3, 6, 8
class TreeNode {
    int value;
    TreeNode left;
    TreeNode right;

    public TreeNode(int value) {
        this.value = value;
        left = right = null;
    }
}
add(x):
  let ptr = root
  item = new TreeNode(x);
  while (ptr.value != x) {
    if (root == null) if (x < ptr.value) // go left
      root = item;
      ptr = ptr.left;
    else if (ptr.right == null) ptr.right = item
      ptr = ptr.right;
  }
  return;
3 return false;
3 else ptr = ptr.right
        if ptr(leaf) == x return true;
        if ptr(root) == x return false;
        while (ptr.contains(x))
            if (ptr(right) == x) return true;
            if (ptr(left) == x) return true;
            ptr = ptr(left);
        return false;
Class TreeSet {
    TreeNode root = null;
    int size = 0;
    add
    contains
}
What about remove?

Idea: Take the smallest node value larger than the one to be removed, and replace the removed node with that one.