Successor and Predecessor

Eugene Weinstein
Data Structures Fall 2008
NYU Computer Science
Successor, Predecessor

- We are given a binary search tree t
- `t.successor(x)` is the smallest item in t that is strictly greater than x
  - null if there is no such node
- `t.predecessor(x)` is the largest item in t that is strictly smaller than x
  - null if there is no such node
Successor Implementation

- Recursive implementation
- This is a traversal: compare the root of current subtree with x, and recurse to children as necessary
- There are many cases - this is an exercise in if-else!
- In code, this will be implemented with Comparable, but the pseudocode assumes the items are integers for simplicity
- The successor pseudocode is on the next slide. The predecessor algorithm is the same, but reversing all comparisons and traversal directions.
Successor Pseudocode

- successor(x, node)
  - if(node.element <= x)
    - // current <=x, so successor must be in right subtree
    - if (node.right = null)
      - return null
    - else
      - return successor(x, node.right)
  - else
    - // current >x, so successor is either in left subtree or current itself
    - if (node.left = null)
      - // if there is nothing on the left, it's certainly the current element
      - return node.element
    - else
      - // there is a left subtree, so the successor is the minimum of the successor of the
      - // left subtree, if any and the current node
      - recurse ← successor(x, node.left)
      - if (recurse = null)
        - return node.element
      - else
        - return min(recurse, node.element)