Binary Search Trees

Peter Chapin

Vermont Technical College

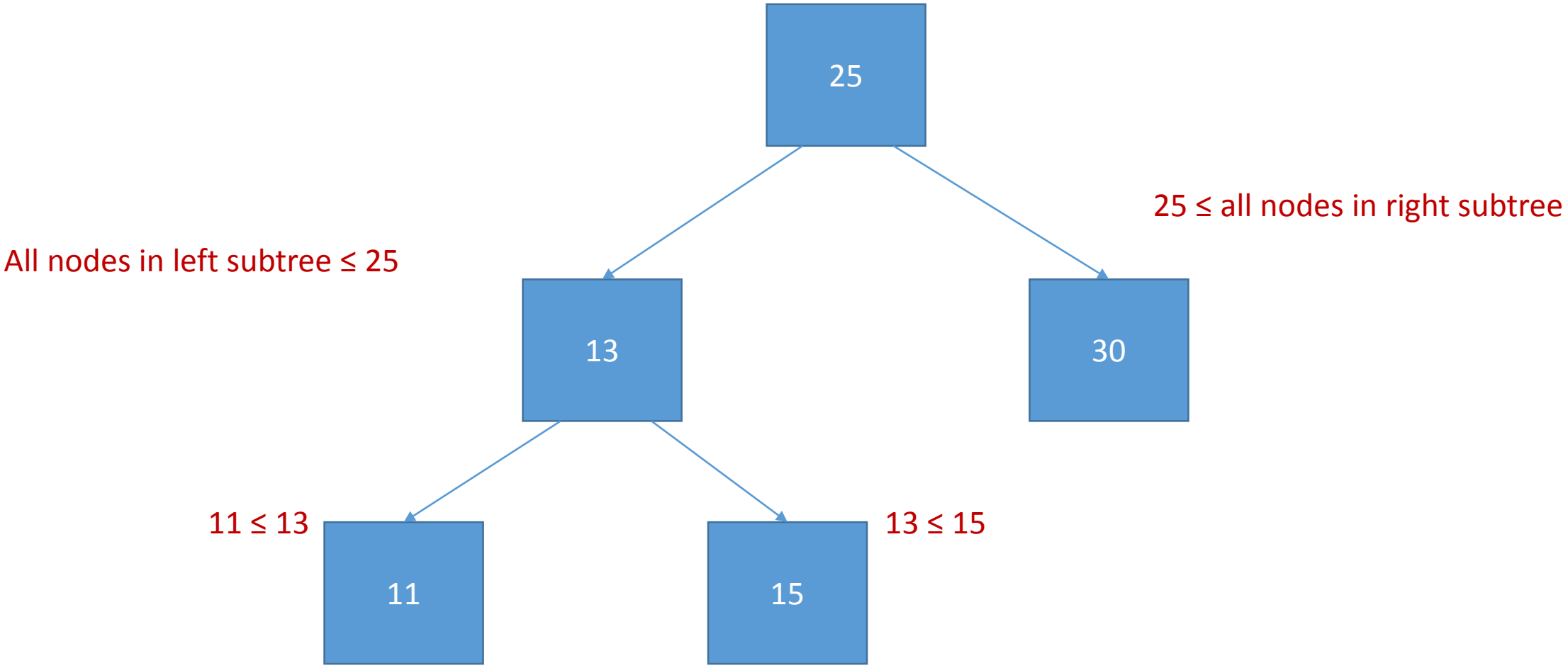
Binary Search Trees

- A **binary search tree** (BST) is a linked structure where each node has two *children*
- Each node also contains some additional data
  - That data type of the data must have a *total order* symbolized here by ≤
    - Transitive: if a ≤ b and b ≤ c, then a ≤ c
    - Antisymmetric: if a ≤ b and b ≤ a, then a = b
    - Total: for all a, b in the type, either a ≤ b or b ≤ a.
    - *Example: the type int with the "usual" meaning of ≤.*
- A BST is inherently recursive
  - The two children form the root of two subtrees

Properties & Terminology

- The ordered property states:
  - *Nodes in the left subtree hold data that is ≤ the parent node.*
  - *The parent node holds data that is ≤ the data in nodes of the right subtree*.
- This property holds recursively for all nodes.
- Nodes without any children are called *leaf nodes*
- Nodes with children are called *internal nodes*
- The node that is not a child of any other node is called the *root node*

Example


25
13
30
11
15
All nodes in left subtree ≤ 25
25 ≤ all nodes in right subtree
11 ≤ 13
13 ≤ 15

What Makes BSTs Great?

- They are *fast!*
  - **Lookup**: find a value in the tree: *O(log(n))*
  - **Insert**: add a value to the tree: *O(log(n))*
  - **Delete**: remove a value from the tree: *O(log(n))*
- This assumes the trees stay (approximately) balanced
  - Height of a node: number of links crossed in path from the node to the most distant leaf beneath that node
  - Height of the tree: height of the root node