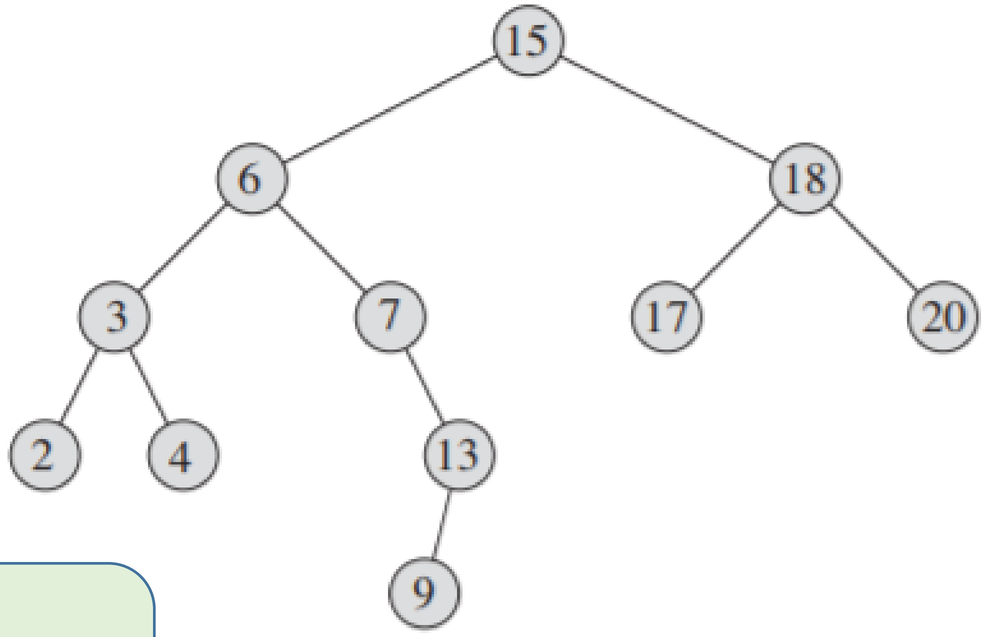
COMP 2611, Data Structures

LECTURE 8: BINARY TREES AND BINARY SEARCH TREES

Binary Trees: Inorder Traversal

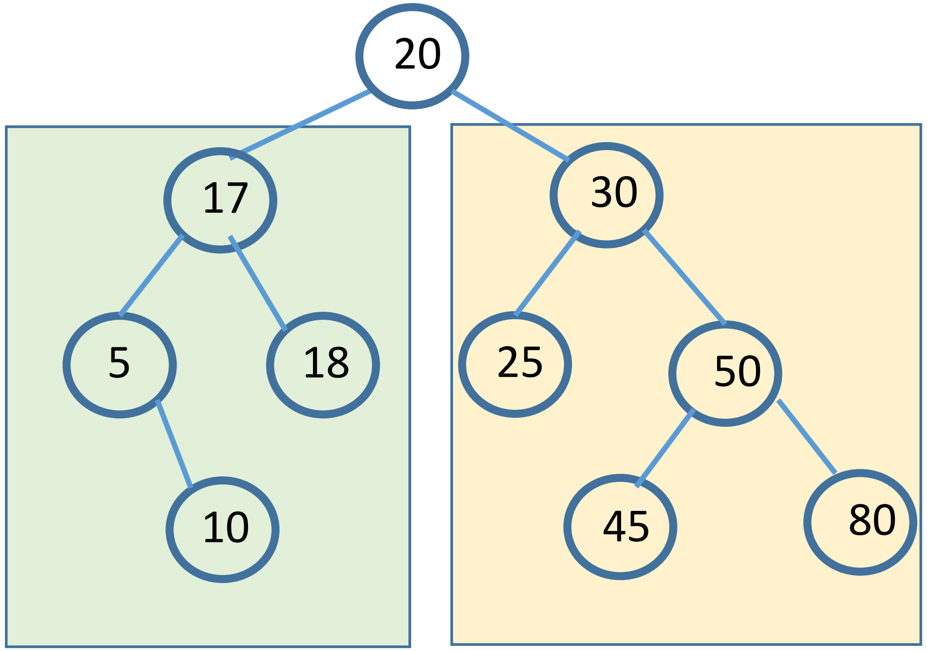
- Give the inorder traversal of this binary tree:



2, 3, 4, 6, 7, 9, 13, 15, 17, 18, 20

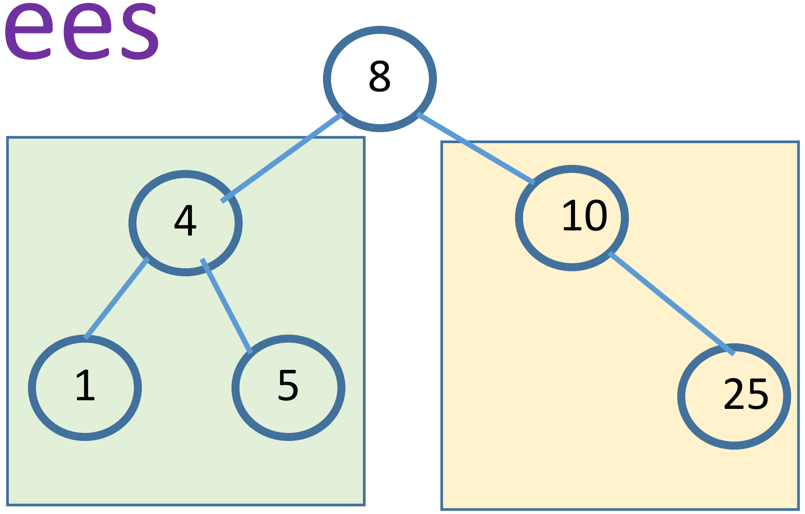
Binary Trees: Inorder Traversal

- Give the inorder traversal of this binary tree:



5, 10, 17, 18, 20, 25, 30, 45, 50, 80

Binary Search Trees



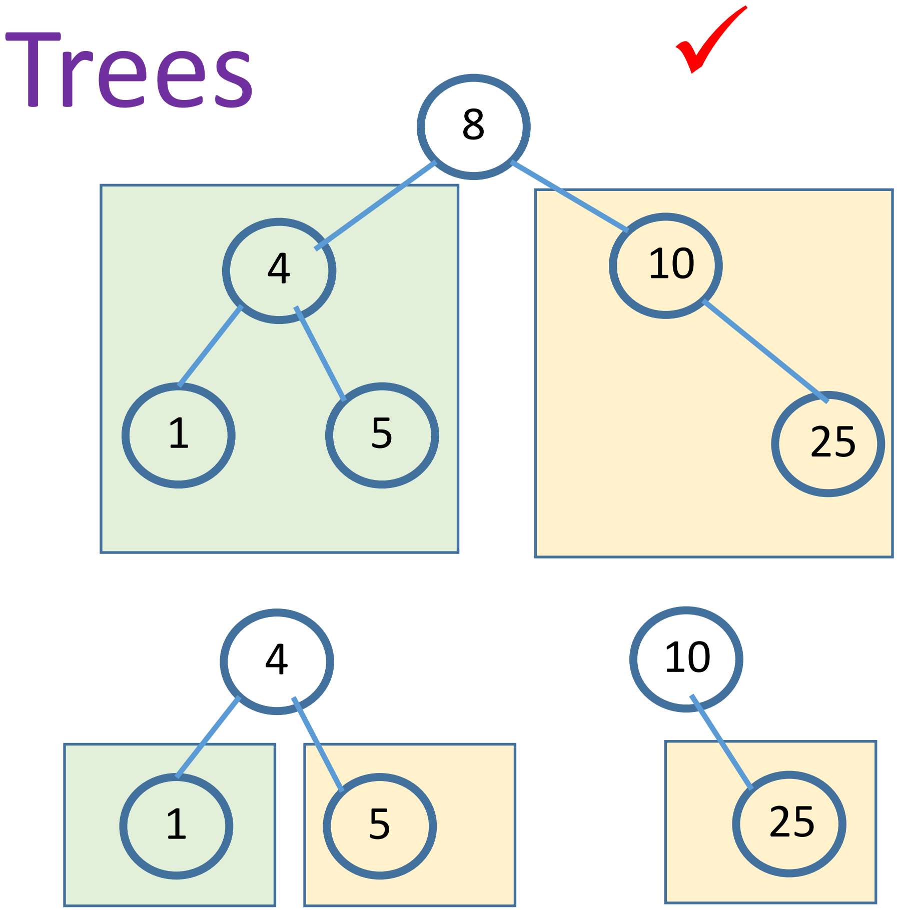
- A binary search tree (BST) is a binary tree where the keys stored at each node satisfy the *binary-search-tree property*:

  - Let *x* be a node in a BST.
  - If *y* is a node in the left subtree of *x*, then y.key ≤ x.key.
  - If *y* is a node in the right subtree of *x*, then y.key ≥ x.key.

Binary Search Trees

- A binary search tree (BST) is a binary tree where the keys stored at each node satisfy the *binary-search-tree property*:

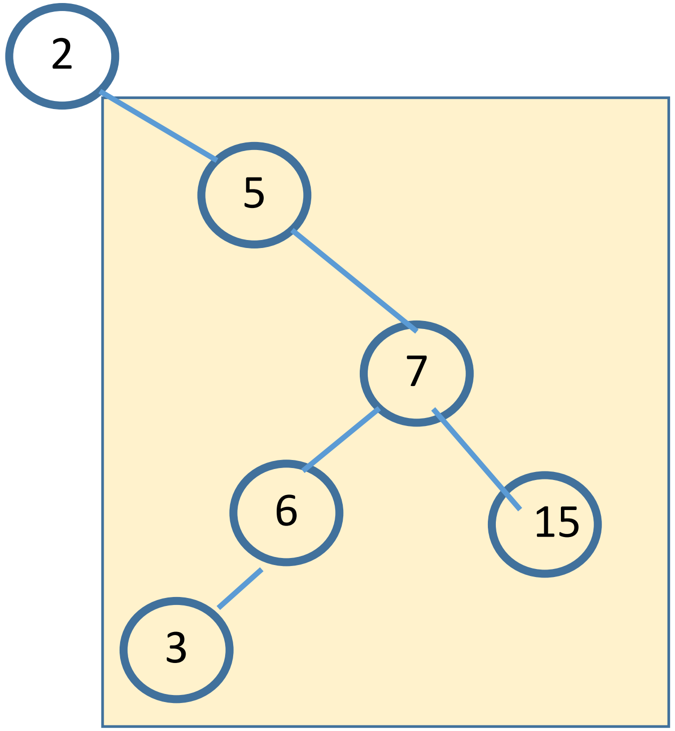
  - Let *x* be a node in a BST.
  - If *y* is a node in the left subtree of *x*, then y->data ≤ x->data.
  - If *y* is a node in the right subtree of *x*, then y->data ≥ x->data.



Binary Search Trees

- A binary search tree (BST) is a binary tree where the keys stored at each node satisfy the *binary-search-tree property*:

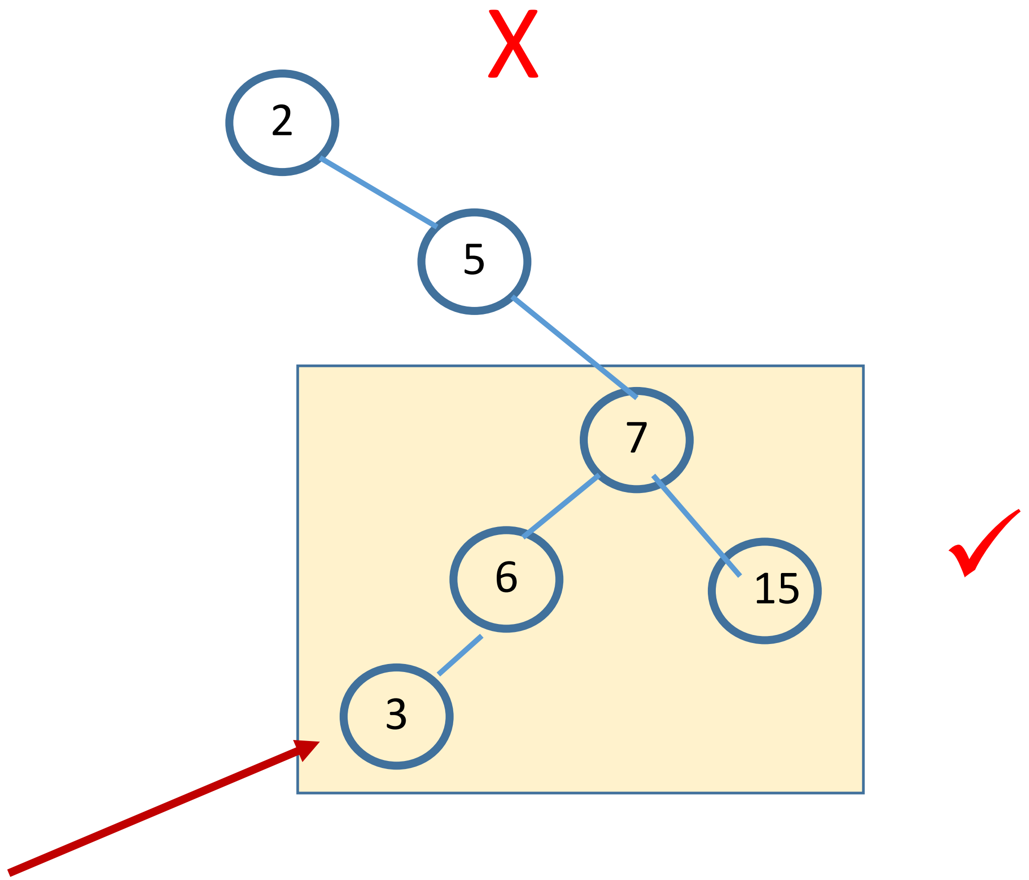
  - Let *x* be a node in a BST.
  - If *y* is a node in the left subtree of *x*, then y->data ≤ x->data.
  - If *y* is a node in the right subtree of *x*, then y->data ≥ x->data.



Binary Search Trees

- A binary search tree (BST) is a binary tree where the keys stored at each node satisfy the *binary-search-tree property*:

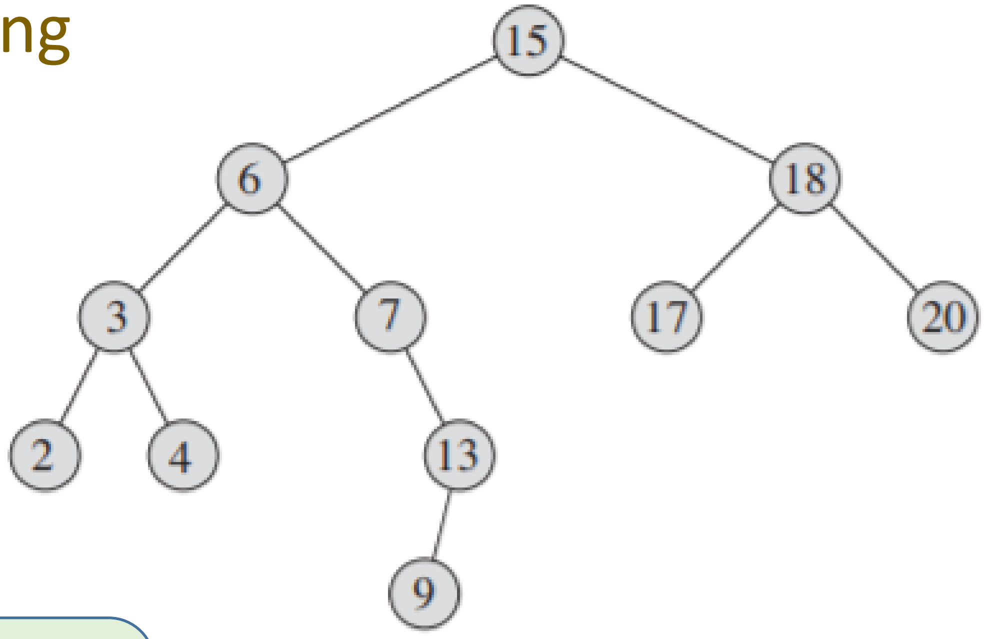
  - Let *x* be a node in a BST.
  - If *y* is a node in the left subtree of *x*, then y->data ≤ x->data.
  - If *y* is a node in the right subtree of *x*, then y->data ≥ x->data.



3 is not greater than 5

Binary Search Trees: Inorder Traversal

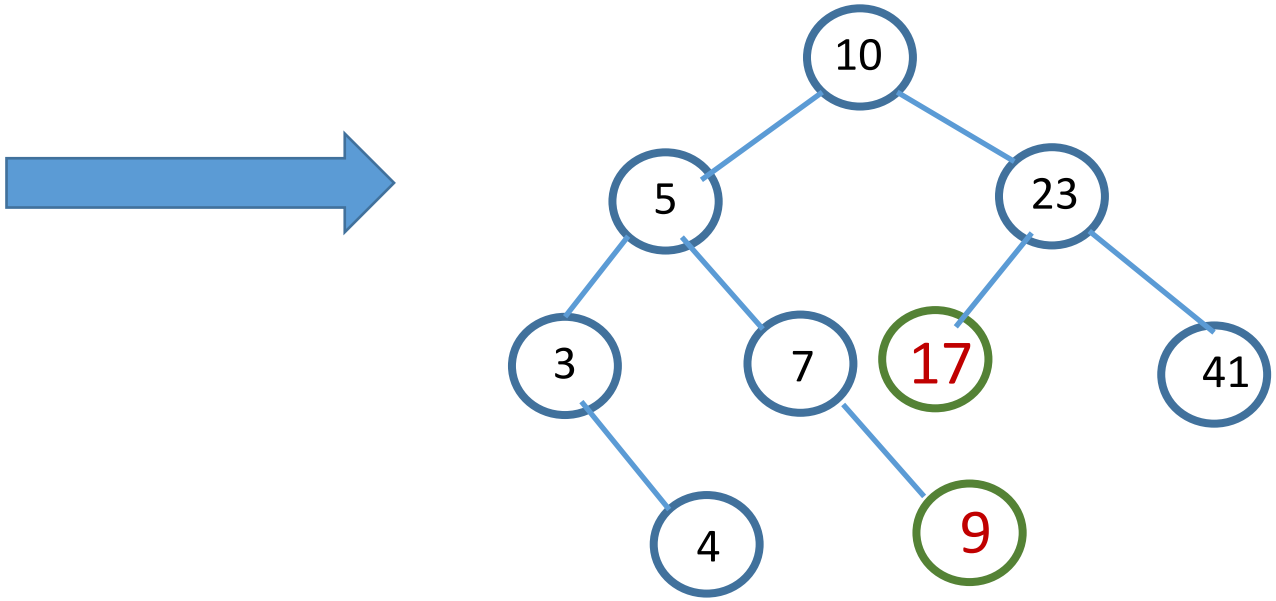
- An inorder traversal of a binary search tree always results in the nodes being visited in ascending order:



2, 3, 4, 6, 7, 9, 13, 15, 17, 18, 20

Binary Search Trees: Insertion

- Where to insert 9 and 17?

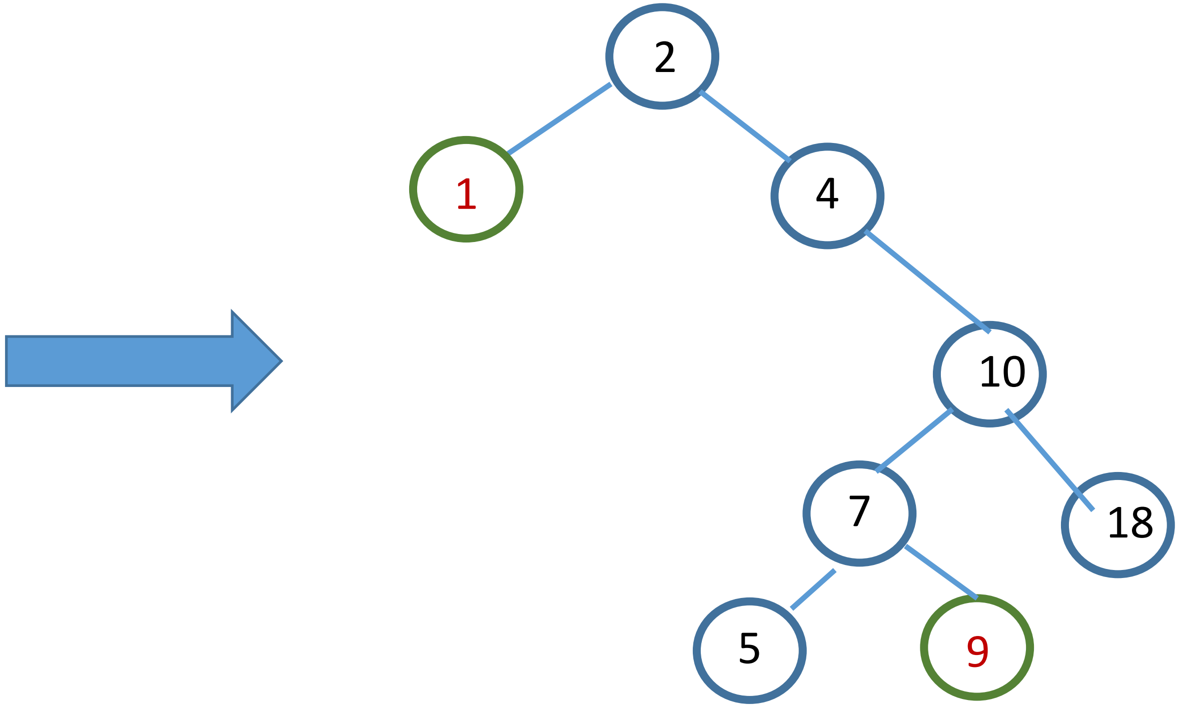


- Insert 9 as the right child of 7.

- Insert 17 as the left child of 23.

Binary Search Trees: Insertion

- Where to insert 1 and 9?

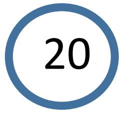


Binary Search Trees: Insertion

- Draw the BST obtained by inserting the following keys in the order given:

20, 30, 25, 17, 5, 10, 50, 80, 18, 45

Insert 20:

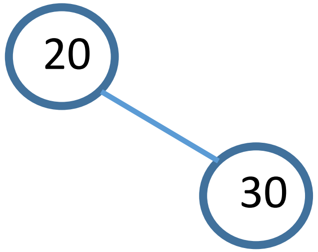


Binary Search Trees: Insertion

- Draw the BST obtained by inserting the following keys in the order given:

20, 30, 25, 17, 5, 10, 50, 80, 18, 45

Insert 30:

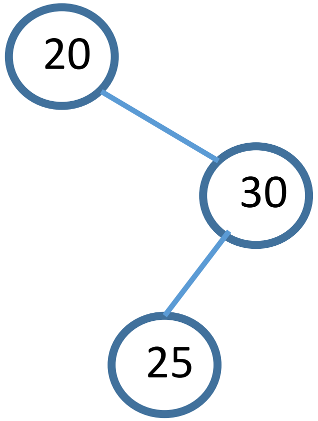


Binary Search Trees: Insertion

- Draw the BST obtained by inserting the following keys in the order given:

20, 30, 25, 17, 5, 10, 50, 80, 18, 45

Insert 25:

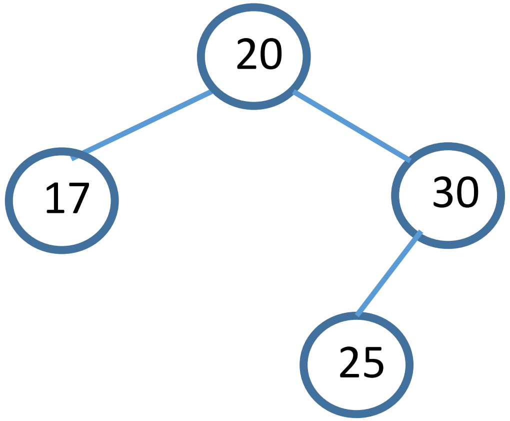


Binary Search Trees: Insertion

- Draw the BST obtained by inserting the following keys in the order given:

20, 30, 25, 17, 5, 10, 50, 80, 18, 45

Insert 17:

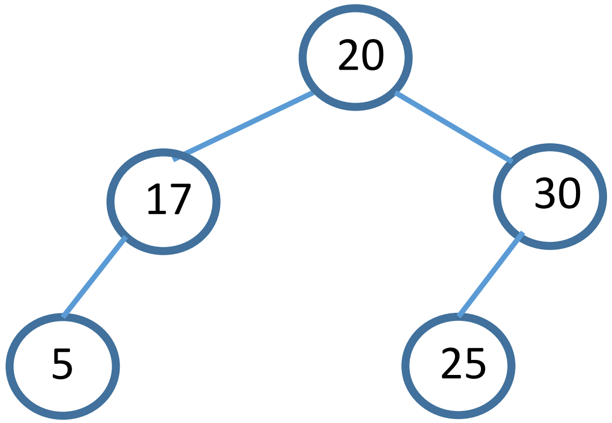


Binary Search Trees: Insertion

- Draw the BST obtained by inserting the following keys in the order given:

20, 30, 25, 17, 5, 10, 50, 80, 18, 45

Insert 5:

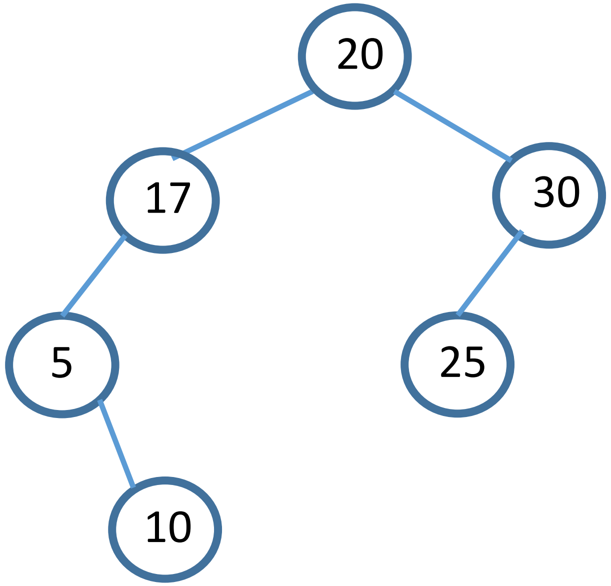


Binary Search Trees: Insertion

- Draw the BST obtained by inserting the following keys in the order given:

20, 30, 25, 17, 5, 10, 50, 80, 18, 45

Insert 10:

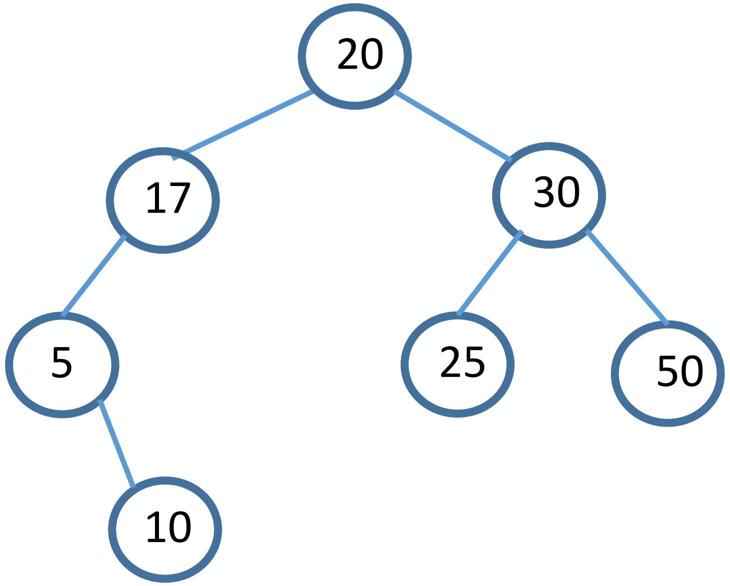


Binary Search Trees: Insertion

- Draw the BST obtained by inserting the following keys in the order given:

20, 30, 25, 17, 5, 10, 50, 80, 18, 45

Insert 50:

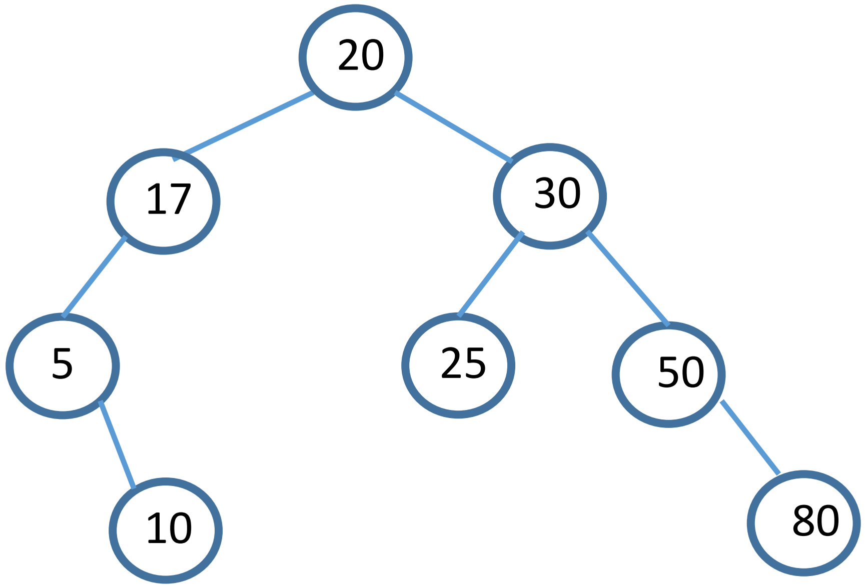


Binary Search Trees: Insertion

- Draw the BST obtained by inserting the following keys in the order given:

20, 30, 25, 17, 5, 10, 50, 80, 18, 45

Insert 80:

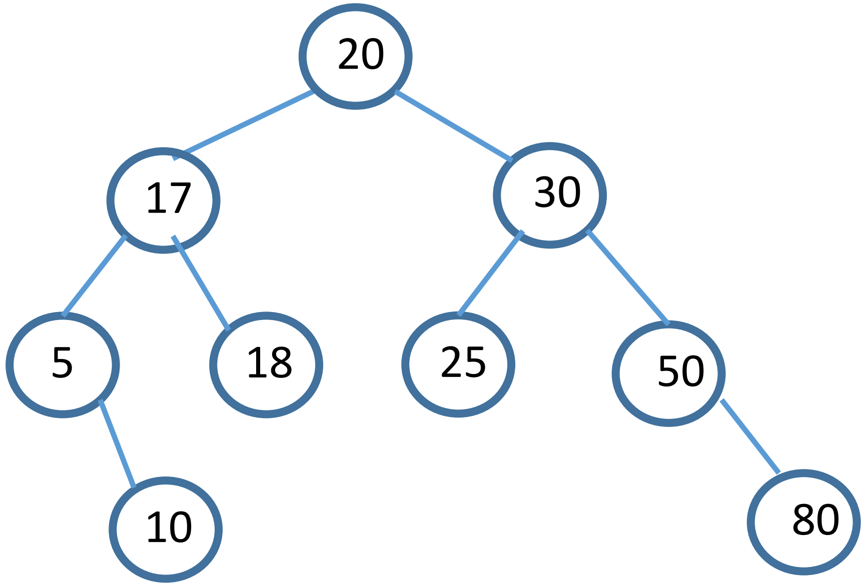


Binary Search Trees: Insertion

- Draw the BST obtained by inserting the following keys in the order given:

20, 30, 25, 17, 5, 10, 50, 80, 18, 45

Insert 18:



Binary Search Trees: Insertion

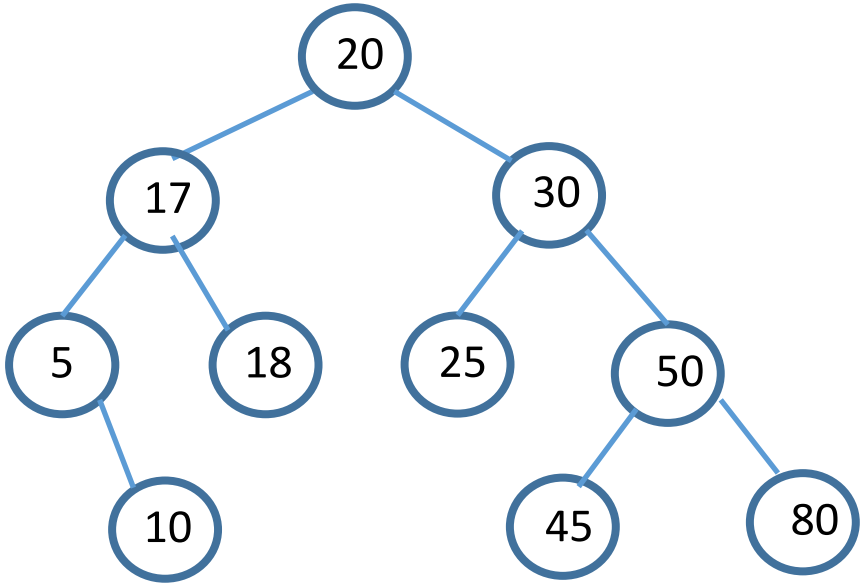
- Draw the BST obtained by inserting the following keys in the order given:

20, 30, 25, 17, 5, 10, 50, 80, 18, 45

Insert 45:

Inorder Traversal:

5, 10, 17, 18, 20, 25, 30, 45, 50, 80



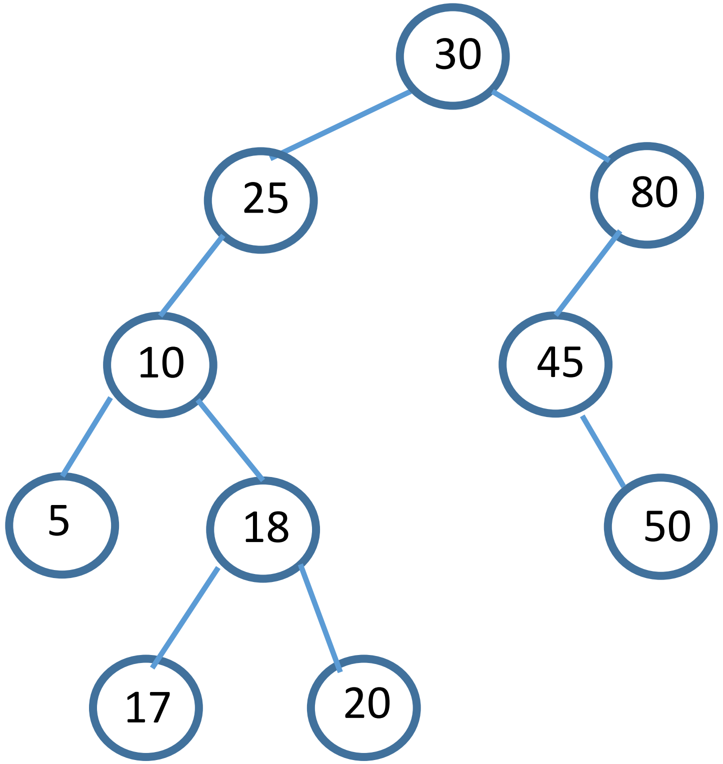
Binary Search Trees: Insertion

- Draw the BST obtained by inserting the following keys in the order given:

30, 25, 80, 45, 10, 18, 20, 5, 50, 17

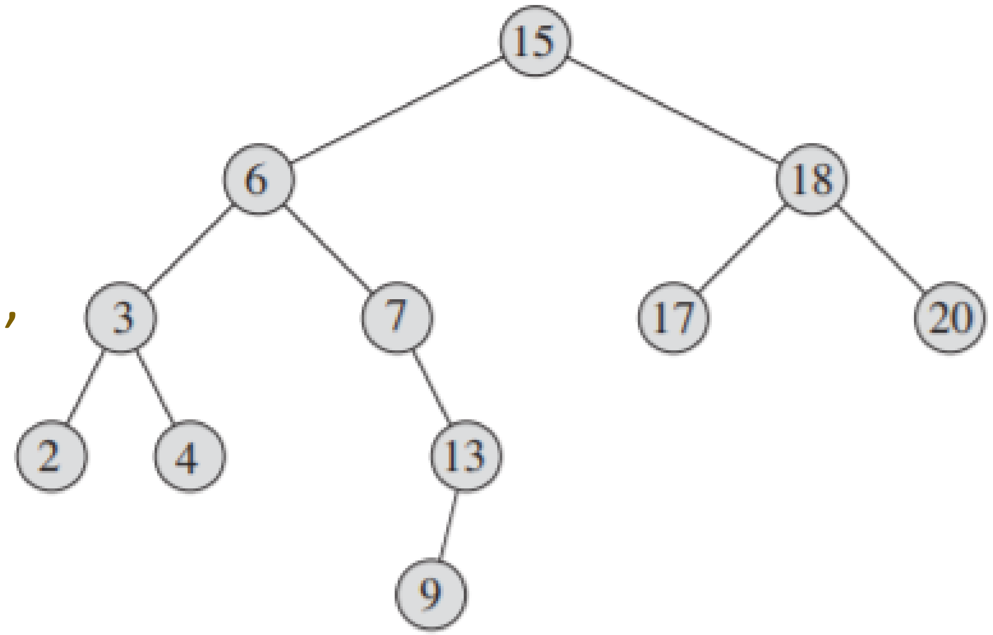
Inorder Traversal:

5, 10, 17, 18, 20, 25, 30, 45, 50, 80



Binary Search Trees: Insertion

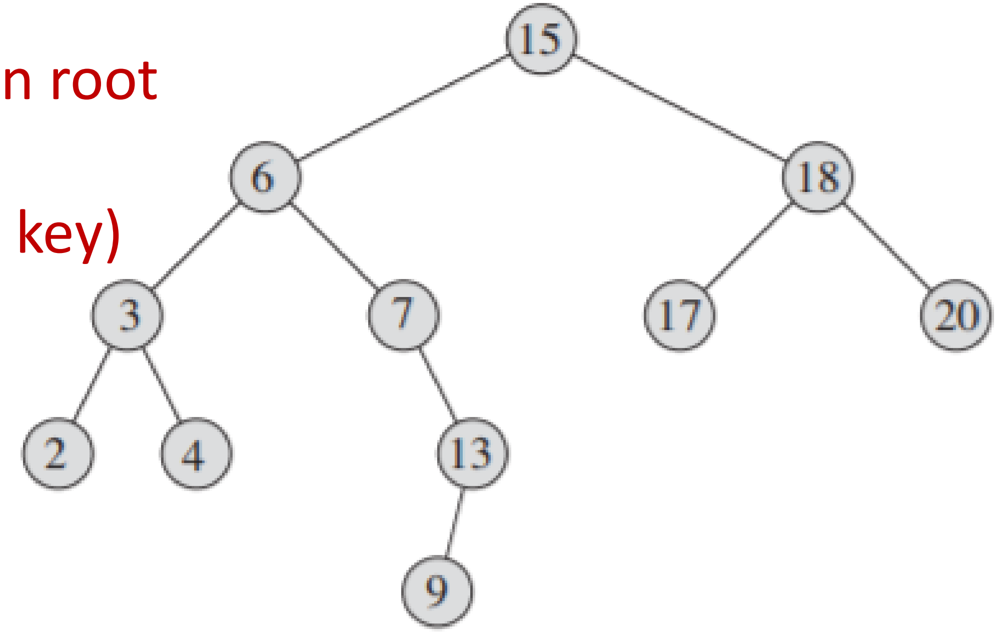
- Case 0: Tree is empty – insert node in empty tree (root is the address of this node)
- Case 1: If data < root->data – go left
- Case 2: If data > root->data – go right
- Repeat (1) and (2) until position found (i.e., parent with empty left or right subtree
- Create BTNode and connect to parent.



Binary Search Trees: Search (Return Node Where Found)

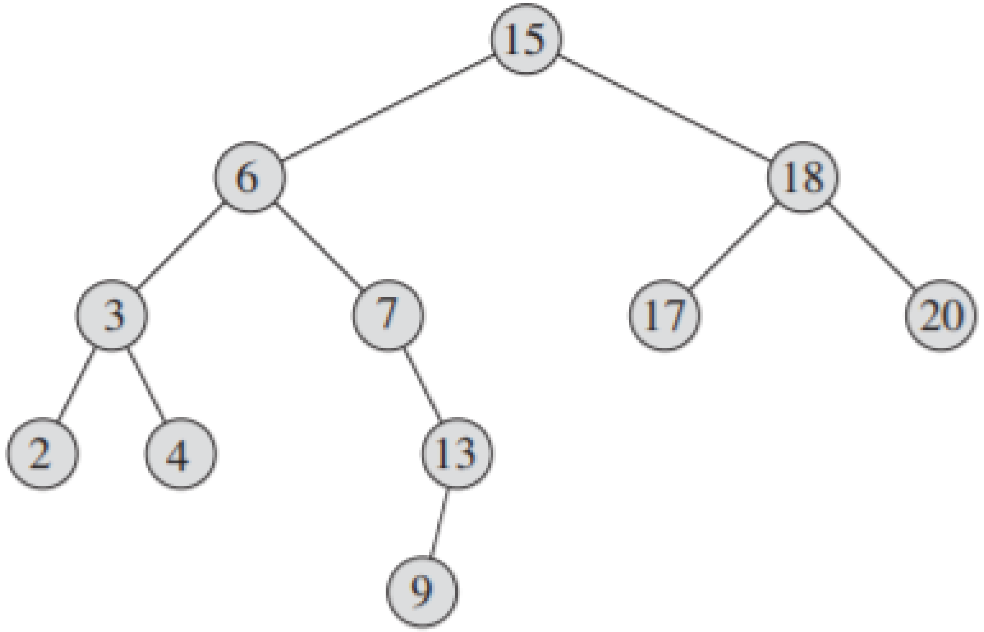
Cases for contains (BTNode * root, int key):

- The root of binary tree is empty: return NULL (or root)
- The root of the binary tree contains key: return root
- If key < root->data: return contains(root->left, key)
- Otherwise: return contains(root->right, key)



Binary Search Trees: Finding the Nodes with the Minimum and Maximum Keys

- `BTNode * minimum (BTNode * root)`
- `BTNode * maximum (BTNode * root)`


15
6
18
3
7
17
20
2
4
13
9