Lecture Notes 05

Binary Search Tree

(Covered)

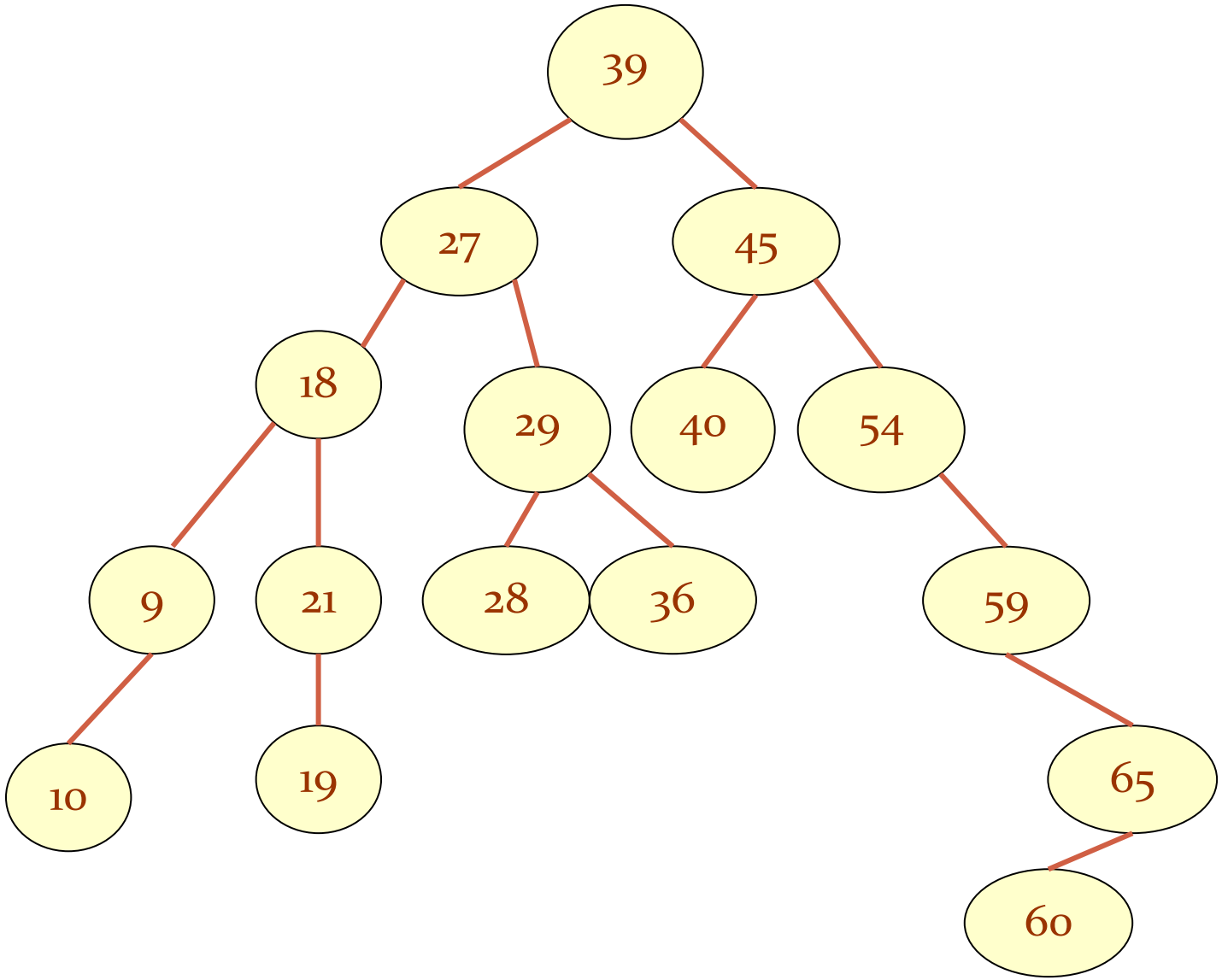
ECE217 Data Structure and Algorithms

Instructor: Dr. Shayan (Sean) Taheri

Binary Search Trees

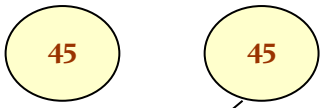
- A binary search tree (BST), also known as an ordered binary tree, is a variant of binary tree in which the nodes are arranged in order.
- In a BST, all nodes in the left sub-tree have a value less than that of the root node.
- Correspondingly, all nodes in the right sub-tree have a value either equal to or greater than the root node.
- The same rule is applicable to every sub-tree in the tree.
- Due to its efficiency in searching elements, BSTs are widely used in dictionary problems where the code always inserts and searches the elements that are indexed by some key value.

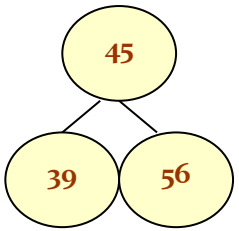
Binary Search Tree (Cont.)

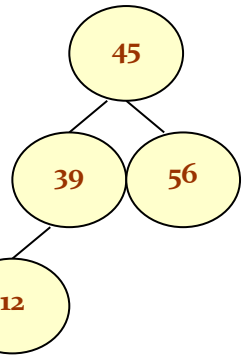


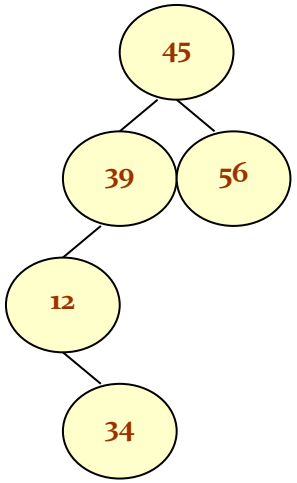
Creating a Binary Search from Given Values

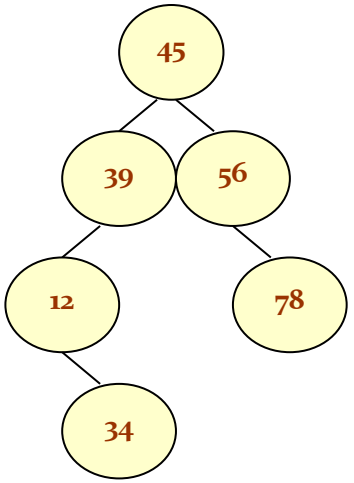
45, 39, 56, 12, 34, 78, 32, 10, 89, 54, 67

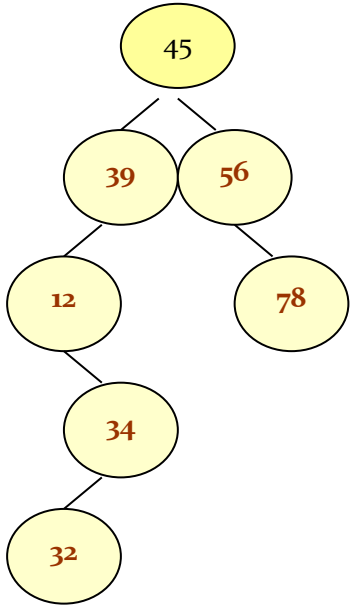


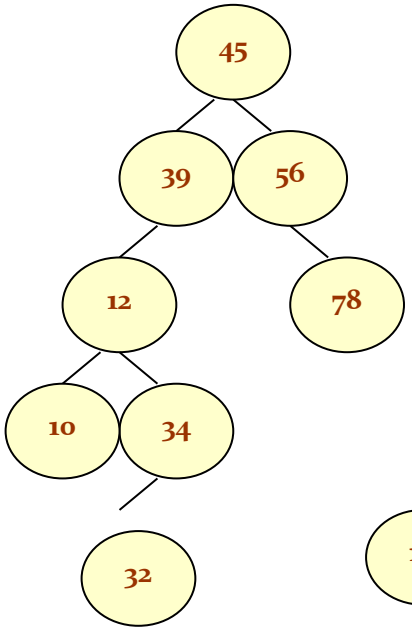


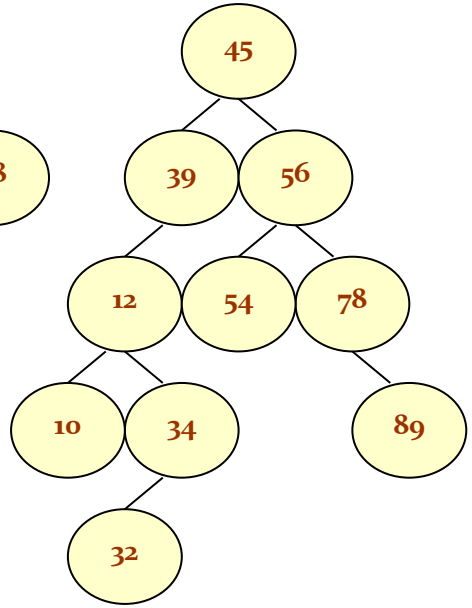


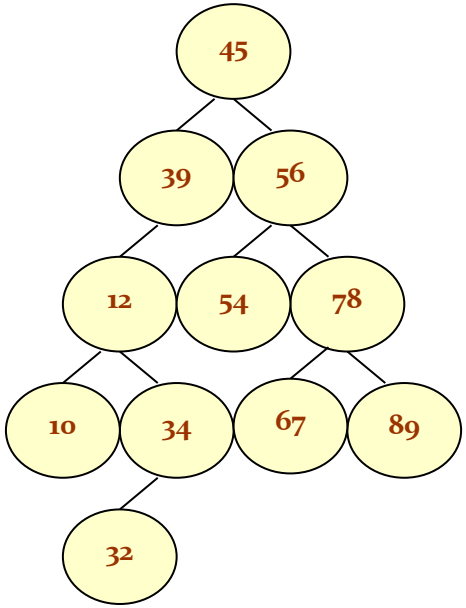












Algorithm to Insert a Value in a BST

```
Insert (TREE, VAL)

Step 1: IF TREE = NULL, then
          Allocate memory for TREE
          SET TREE->DATA = VAL
          SET TREE->LEFT = TREE ->RIGHT = NULL
        ELSE
          IF VAL < TREE->DATA
             Insert(TREE->LEFT, VAL)
          ELSE
             Insert(TREE->RIGHT, VAL)
          [END OF IF]
        [END OF IF]
Step 2: End
```

Searching for a Value in a BST

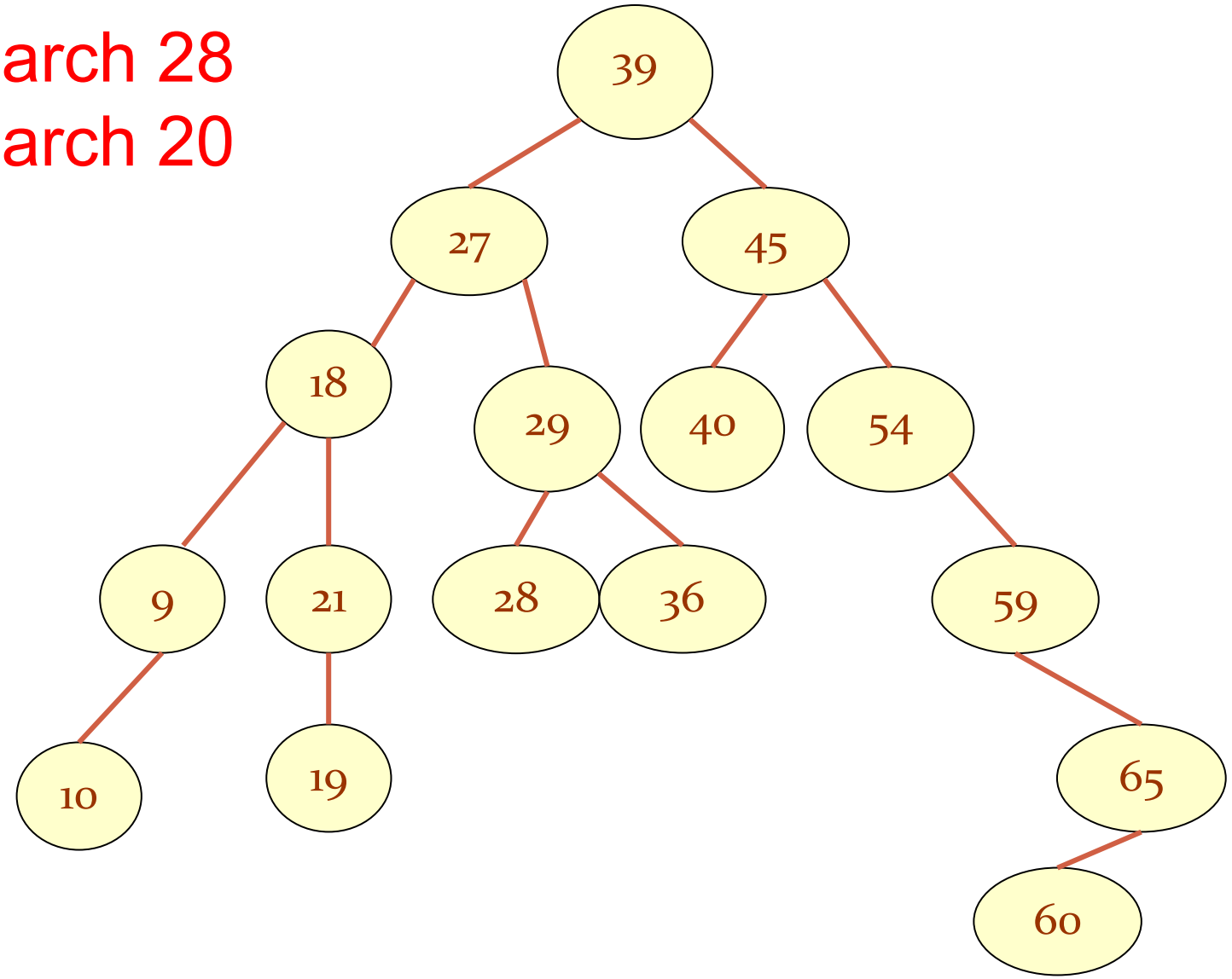
- The search function is used to find whether a given value is present in the tree or not.
- The function first checks if the BST is empty. If it is, then the value we are searching for is not present in the tree, and the search algorithm terminates by displaying an appropriate message.
- If there are nodes in the tree then the search function checks to see if the key value of the current node is equal to the value to be searched.
- If not, it checks if the value to be searched for is less than the value of the node, in which case it should be recursively called on the left child node.
- In case the value is greater than the value of the node, it should be recursively called on the right child node.

Algorithm to Search a Value in a BST

```
searchElement(TREE, VAL)
Step 1:
      IF TREE->DATA = VAL OR TREE = NULL, then
         Return TREE
      ELSE
         IF VAL < TREE->DATA
             Return searchElement(TREE->LEFT, VAL)
         ELSE
            Return searchElement(TREE->RIGHT, VAL)
         [END OF IF]
       [END OF IF]
Step 2: End
```

Searching for a Value in a BST (Cont.)

Search 28
Search 20

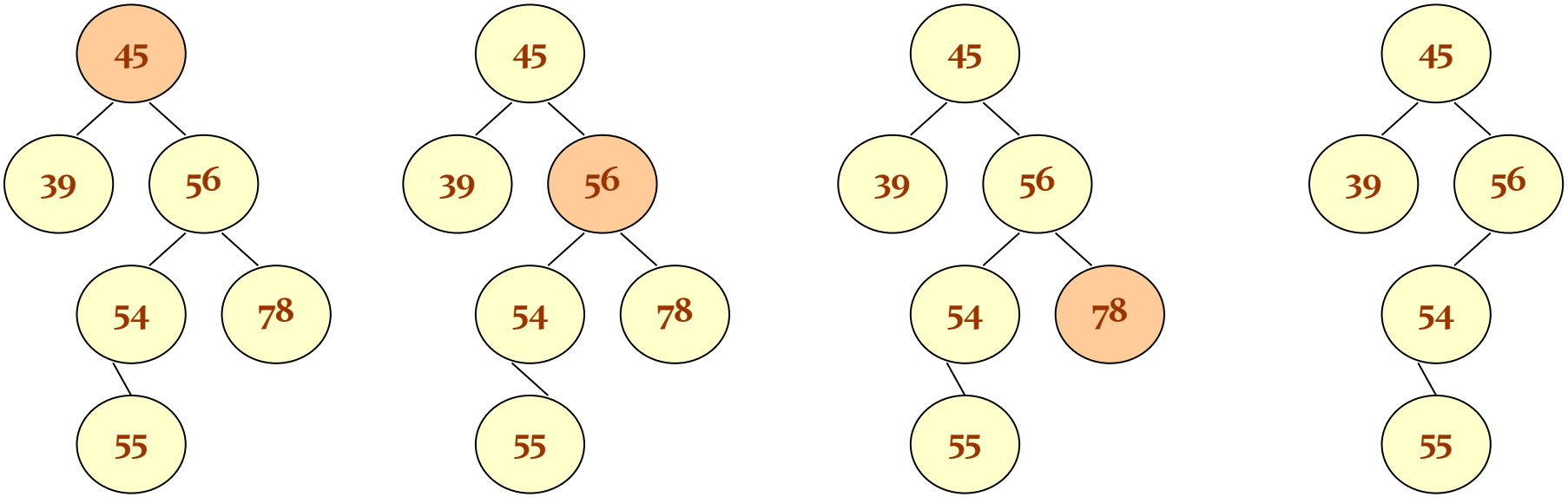


Deleting a Value from a BST

- The delete function deletes a node from the binary search tree.
- Deletion operation needs to keep the property of BSTs.
- The deletion of a node involves any of the three cases.

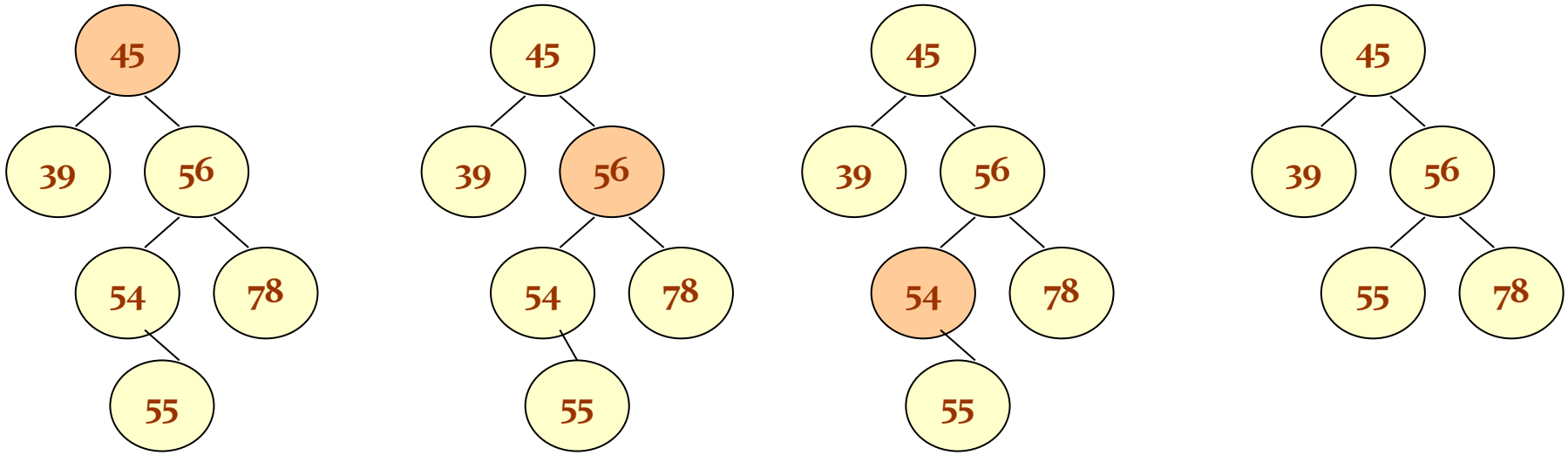
Case 1: Deleting a node that has no children.

For example, deleting node 78 in the tree below.



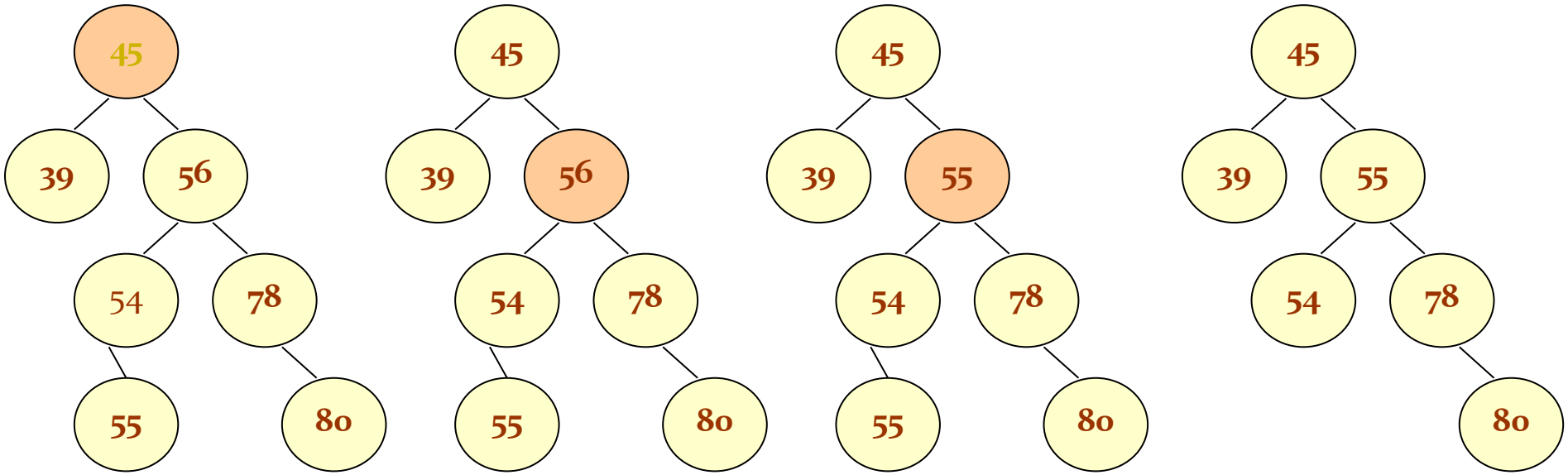
Deleting a Value from a BST (Cont.)

- *Case 2:* Deleting a node with one child (either left or right).
- To handle the deletion, the node's child is set to be the child of the node's parent.
- Now, if the node was the left child of its parent, the node's child becomes the left child of the node's parent.
- Correspondingly, if the node was the right child of its parent, the node's child becomes the right child of the node's parent.



Deleting a Value from a BST (Cont.)

- *Case 3:* Deleting a node with two children.
- To handle this case of deletion, replace the node's value with its in-order predecessor (largest value in the left sub-tree) or in-order successor (smallest value in the right sub-tree).
- The in-order predecessor or the successor can then be deleted using any of the above cases.

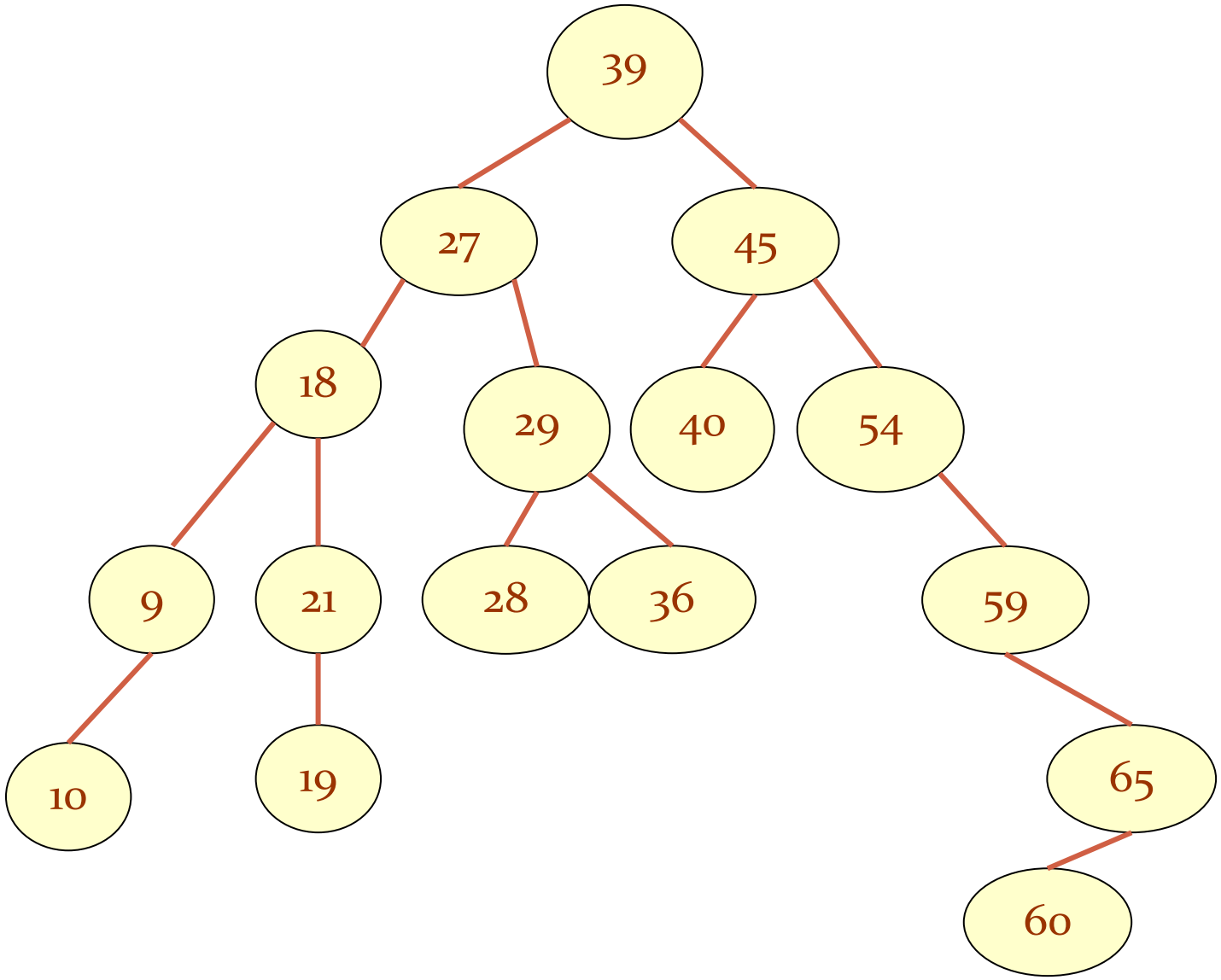


Algorithm to Delete from a BST

```
Delete (TREE, VAL)
Step 1: IF TREE = NULL, then
         Write "VAL not found in the tree"
        ELSE IF VAL < TREE->DATA
           Delete(TREE->LEFT, VAL)
        ELSE IF VAL > TREE->DATA
         Delete(TREE->RIGHT, VAL)
        ELSE IF TREE->LEFT AND TREE->RIGHT
           SET TEMP = findLargestNode(TREE->LEFT)
           SET TREE->DATA = TEMP->DATA
           Delete(TREE->LEFT, TEMP->DATA)
        ELSE
          SET TEMP = TREE
          IF TREE->LEFT = NULL AND TREE ->RIGHT = NULL
              SET TREE = NULL
          ELSE IF TREE->LEFT != NULL
              SET TREE = TREE->LEFT
          ELSE
              SET TREE = TREE->RIGHT
          [END OF IF]
             FREE TEMP
        [END OF IF]
Step 2: End
```

See class example

Find the Largest Value in a BST



Finding the Largest Node in a BST (Cont.)

- The basic property of a BST states that the larger value will occur in the right sub-tree.
- If the right sub-tree is NULL, then the value of root node will be largest as compared with nodes in the left sub-tree.
- So, to find the node with the largest value, we will find the value of the rightmost node of the right sub-tree.
- If the right sub-tree is empty then we will find the value of the root node.

```
findLargestElement (TREE)
Step 1: IF TREE = NULL OR TREE->RIGHT = NULL, then
           Return TREE
        ELSE
           Return findLargestElement(TREE->RIGHT)
      [END OF IF]
Step 2: End
```

Questions?