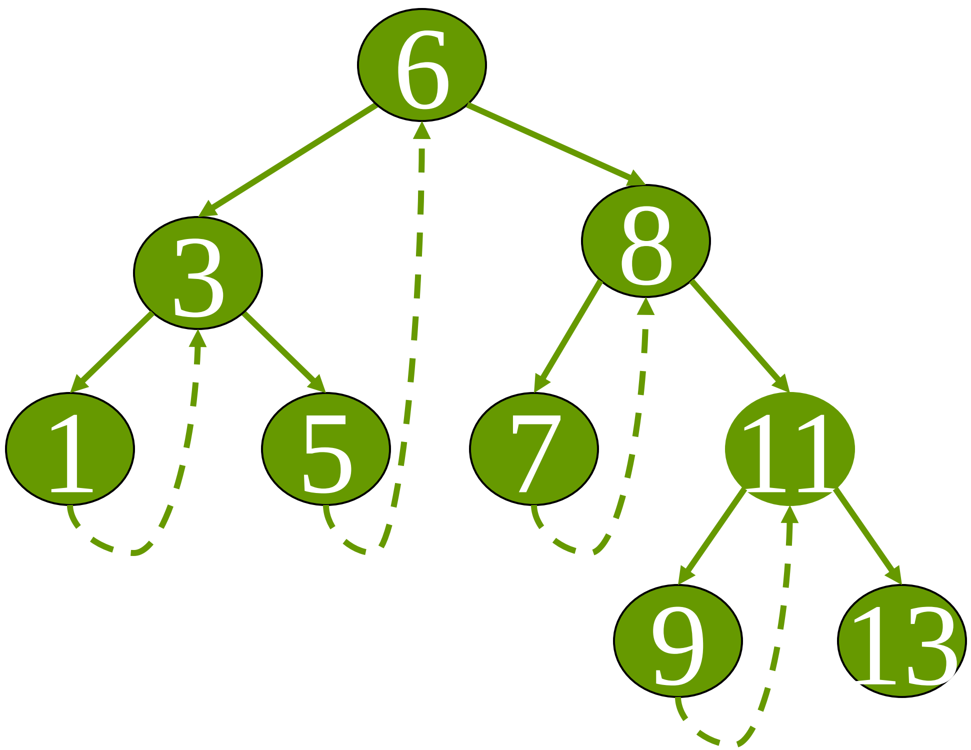
Lecture - 11
on
Data Structures

Threaded Trees

- Binary trees have a lot of wasted space: the leaf nodes each have 2 null pointers
- We can use these pointers to help us in inorder traversals
- We have the pointers reference the next node in an inorder traversal; called *threads*
- We need to know if a pointer is an actual link or a thread, so we keep a boolean for each pointer

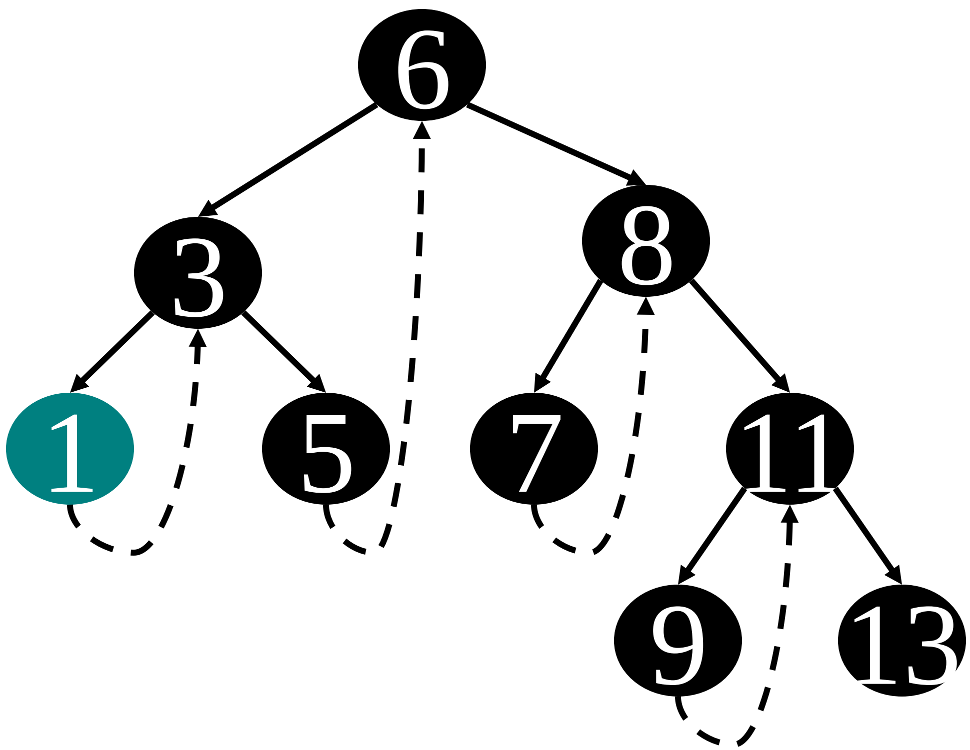
Threaded Tree Example



Threaded Tree Traversal

- We start at the leftmost node in the tree, print it, and follow its right thread
- If we follow a thread to the right, we output the node and continue to its right
- If we follow a link to the right, we go to the leftmost node, print it, and continue

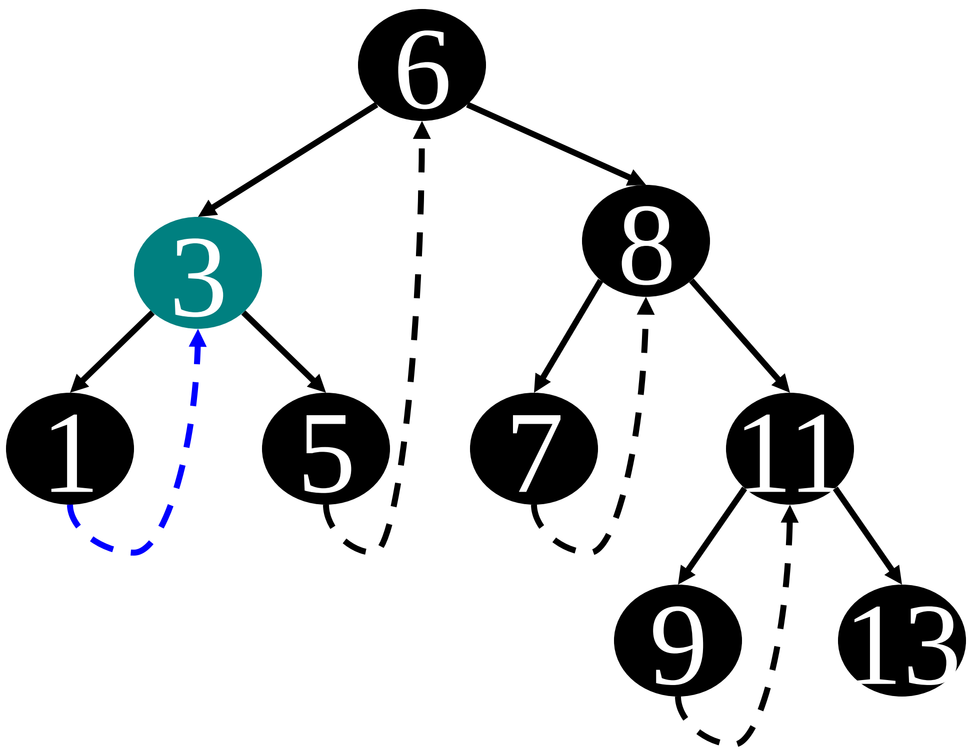
Threaded Tree Traversal



Output
1

Start at leftmost node, print it

Threaded Tree Traversal

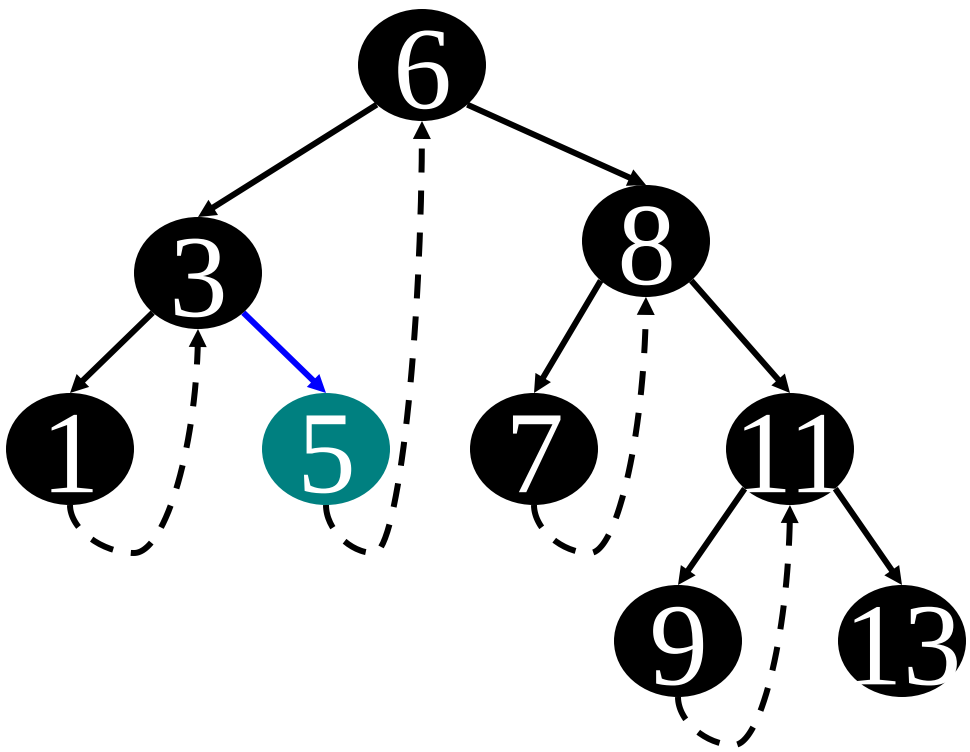


Output
1
3

Follow thread to right, print node

Prepared by, Jesmin Akhter,
Lecturer, IIT,JU

Threaded Tree Traversal

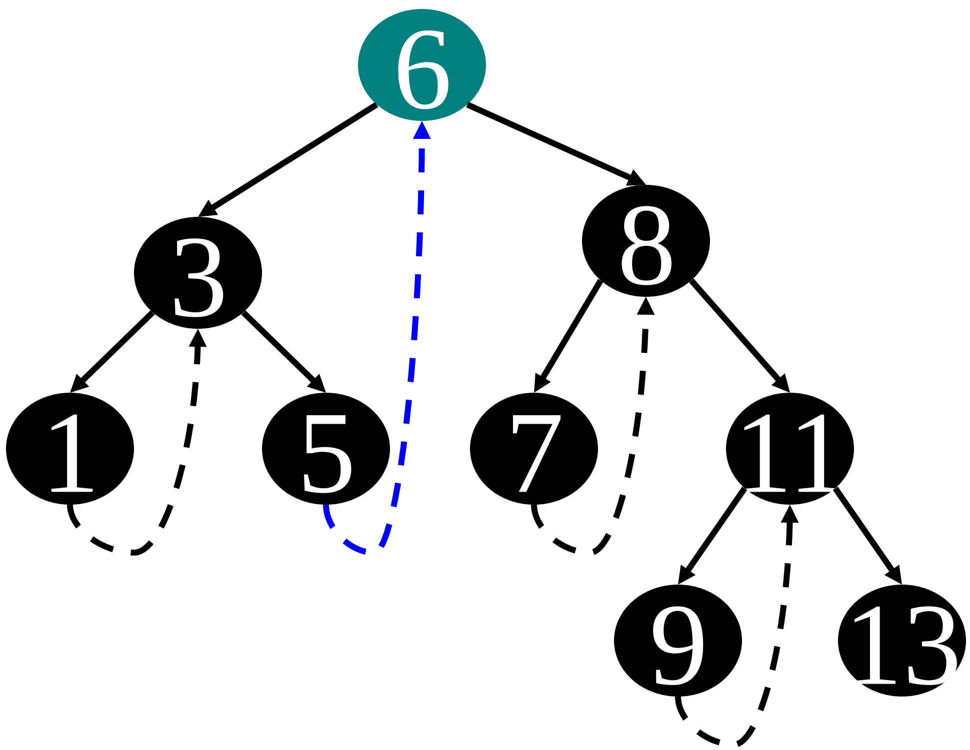


Output
1
3
5

Follow link to right, go to
leftmost node and print

Prepared by, Jesmin Akhter,
Lecturer, IIT,JU

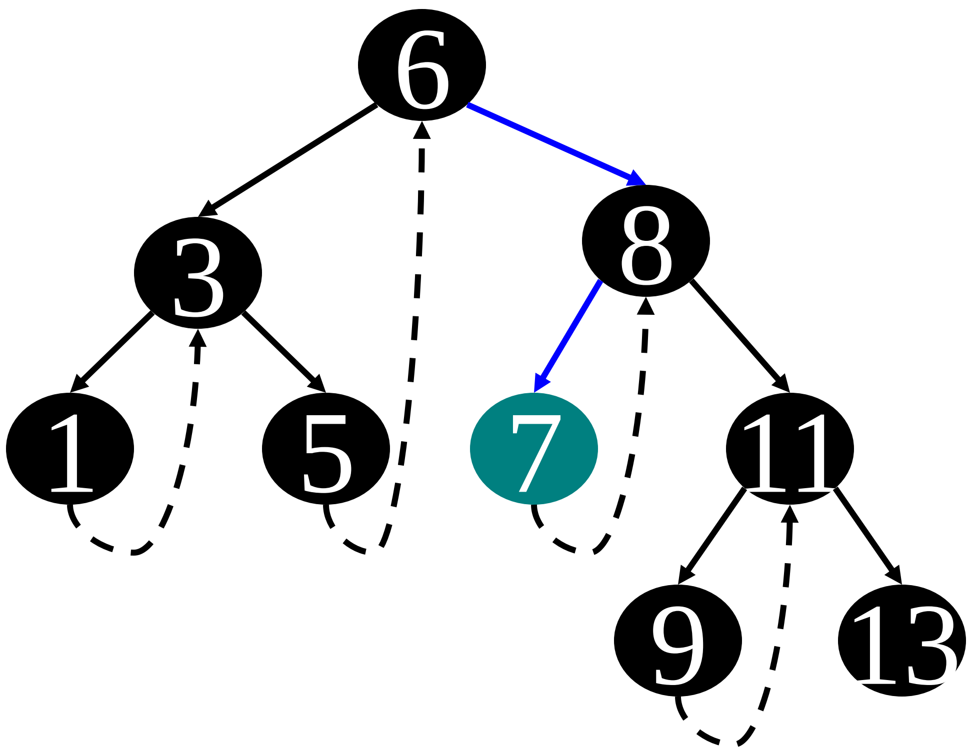
Threaded Tree Traversal



Output
1
3
5
6

Follow thread to right, print node

Threaded Tree Traversal

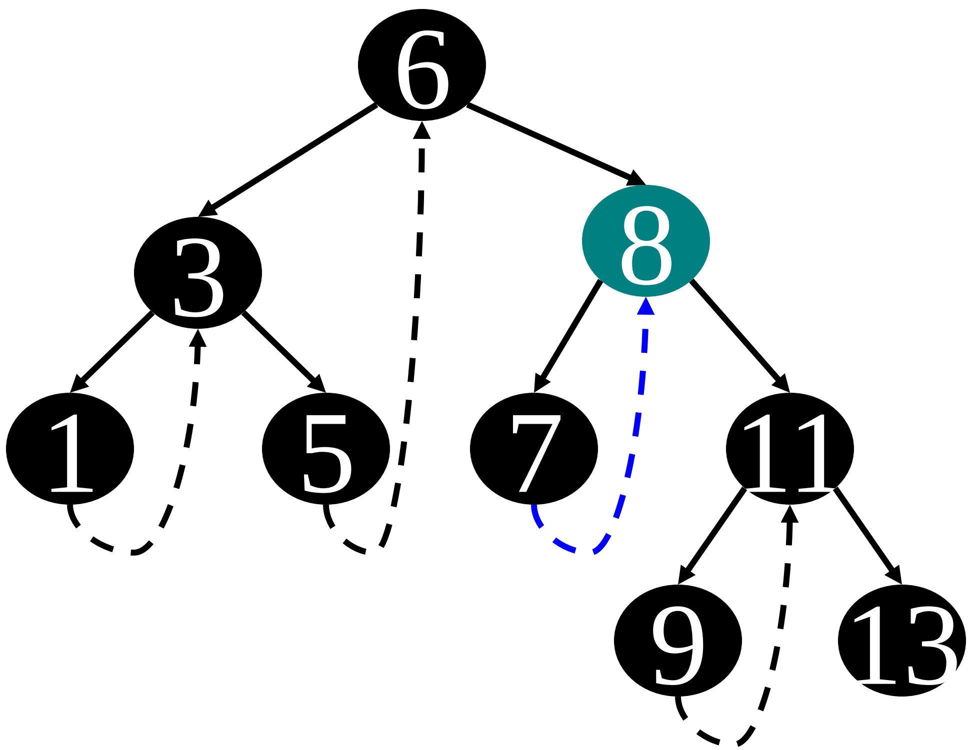


Output
1
3
5
6
7

Follow link to right, go to leftmost node and print

Prepared by, Jesmin Akhter,
Lecturer, IIT,JU

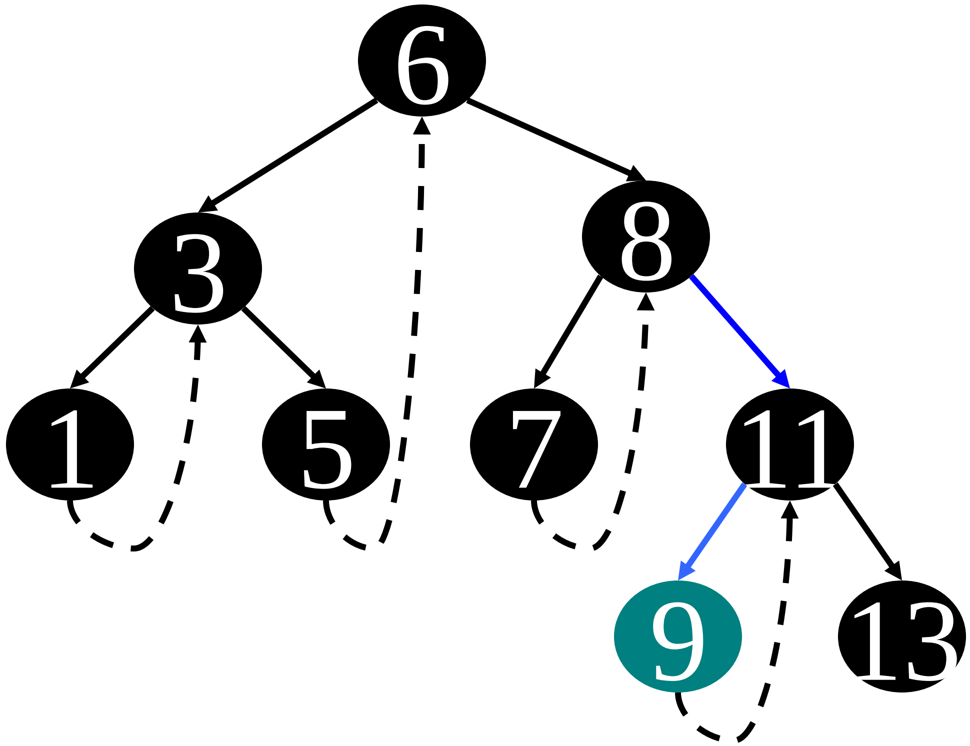
Threaded Tree Traversal



Output
1
3
5
6
7
8

Follow thread to right, print node

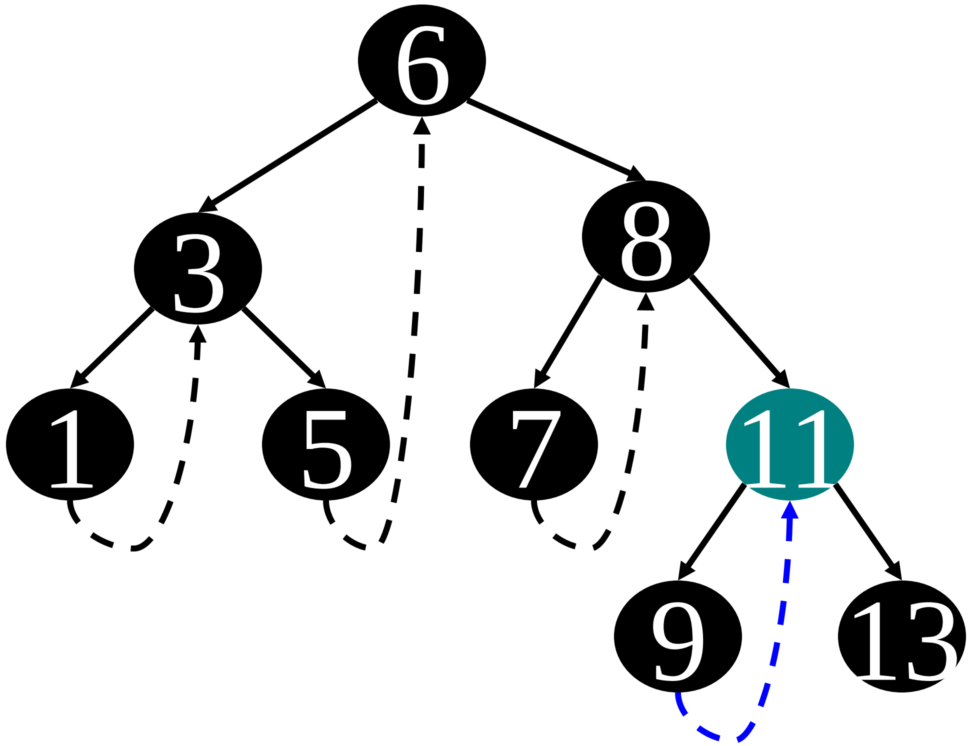
Threaded Tree Traversal



Output
1
3
5
6
7
8
9

Follow link to right, go to leftmost node and print

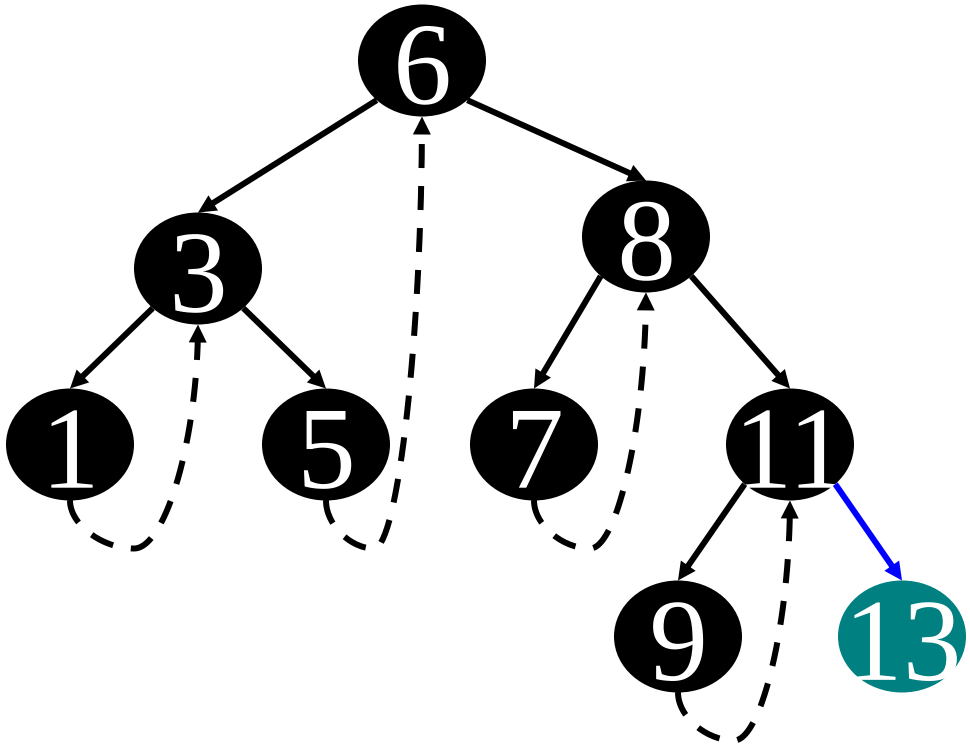
Threaded Tree Traversal



Output
1
3
5
6
7
8
9
11

Follow thread to right, print node

Threaded Tree Traversal



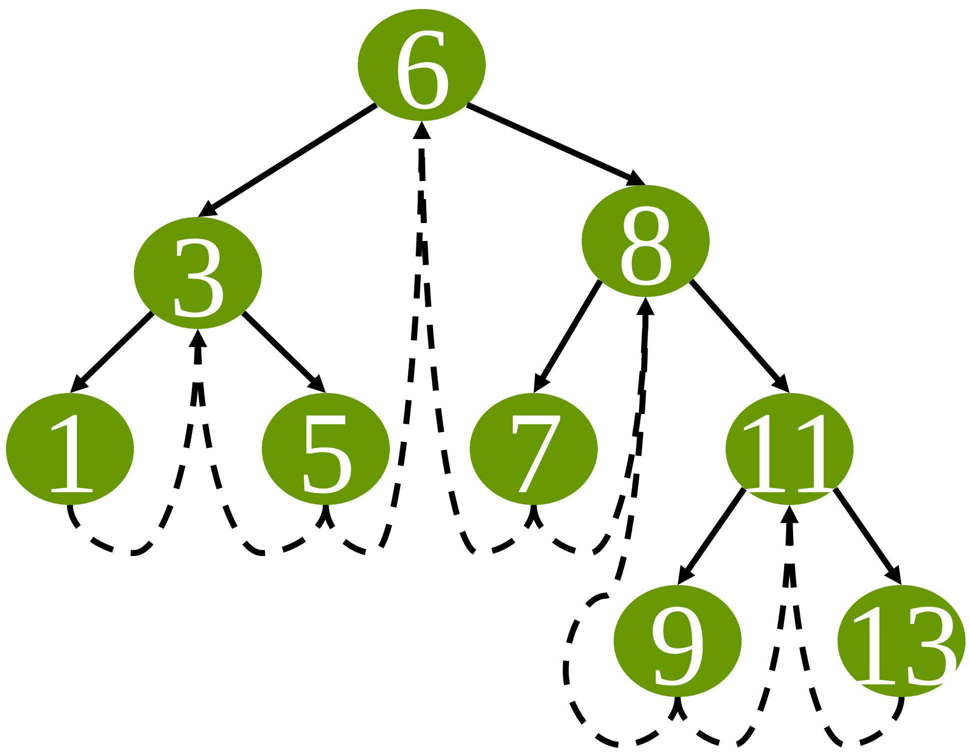
Output
1
3
5
6
7
8
9
11
13

Follow link to right, go to leftmost node and print

Threaded Tree Modification

- We’re still wasting pointers, since half of our leafs’ pointers are still null
- We can add threads to the previous node in an inorder traversal as well, which we can use to traverse the tree backwards or even to do postorder traversals

Threaded Tree Modification



Prepared by, Jesmin Akhter,
Lecturer, IIT,JU

Binary Search Trees

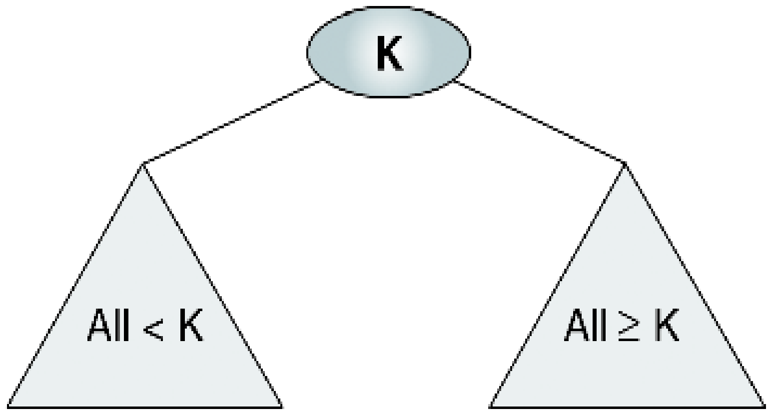
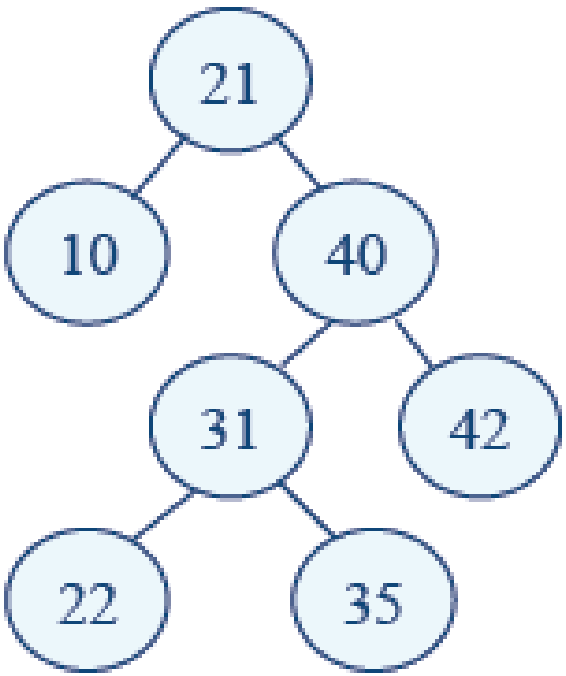


FIGURE 7-1 Binary Search Tree

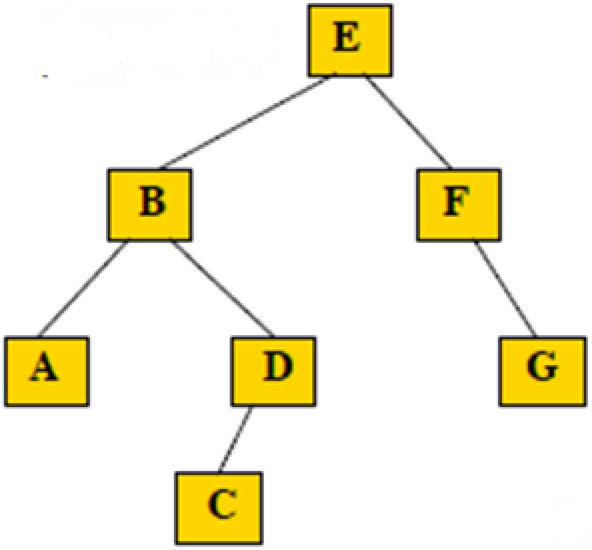
Binary search trees (Con..)



Binary Search Tree (Con..)

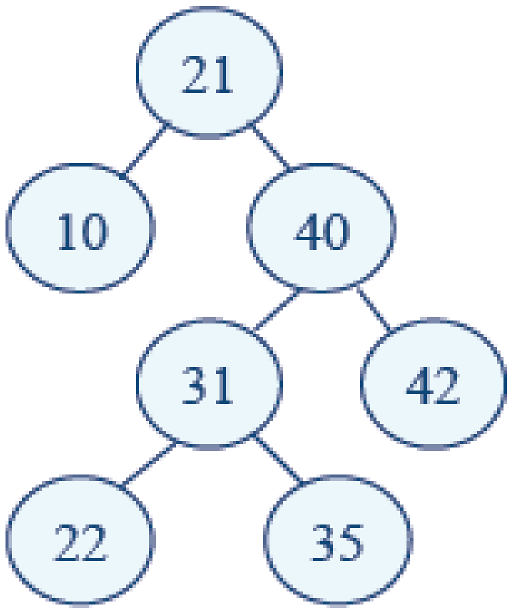
A **Binary Search Tree** is a binary tree with the following Basic properties:

- All items in the left subtree are less than the root.
- All items in the right subtree are greater or equal to the root.
- Each subtree is itself a binary search tree.



Binary search trees (Con..)

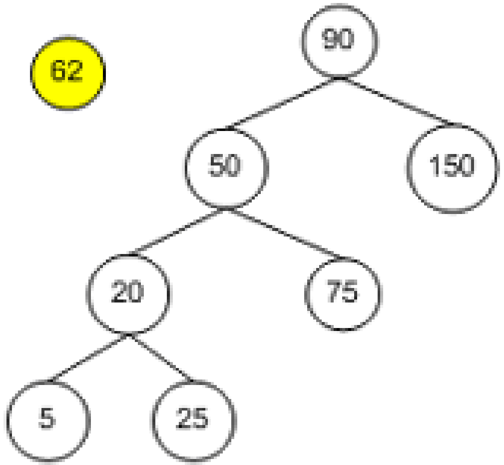
- If we want to find out whether a given object is present in a binary search tree:



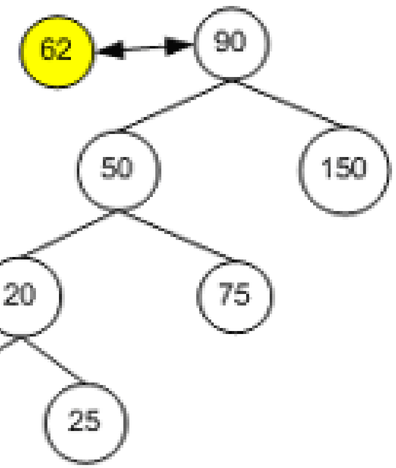
  - Suppose the item is the number 39. Beginning at the root of the tree, we see that 39 is greater than 21.
  - If 39 is in the tree, it must be in the right hand tree with respect to the root node.
  - We see that 40 is greater than 39, so we turn to the left hand tree with respect to 40.
  - We eventually establish that 40 is not in the tree.

Searching and inserting in Binary search trees

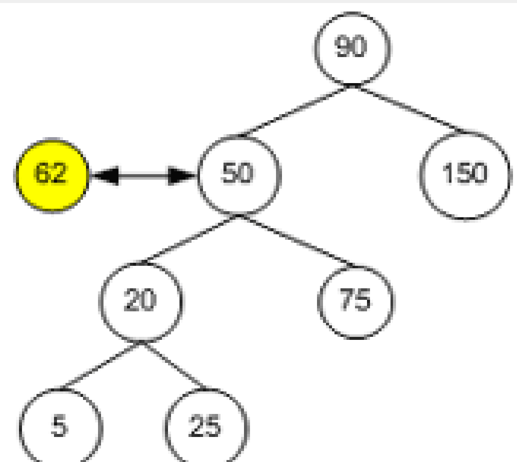
Given the following BST, we want to insert a node with the value 62...



We start by comparing the node to insert (62) with the root (90). We see that 62 is less than 90, so we know 62 must be added somewhere to the root's left subtree.



We next compare 62 to 50. Since 62 is greater than 50, 62 must belong somewhere in 50's right subtree.



Searching and inserting in Binary search trees

Searching and inserting in Binary search trees

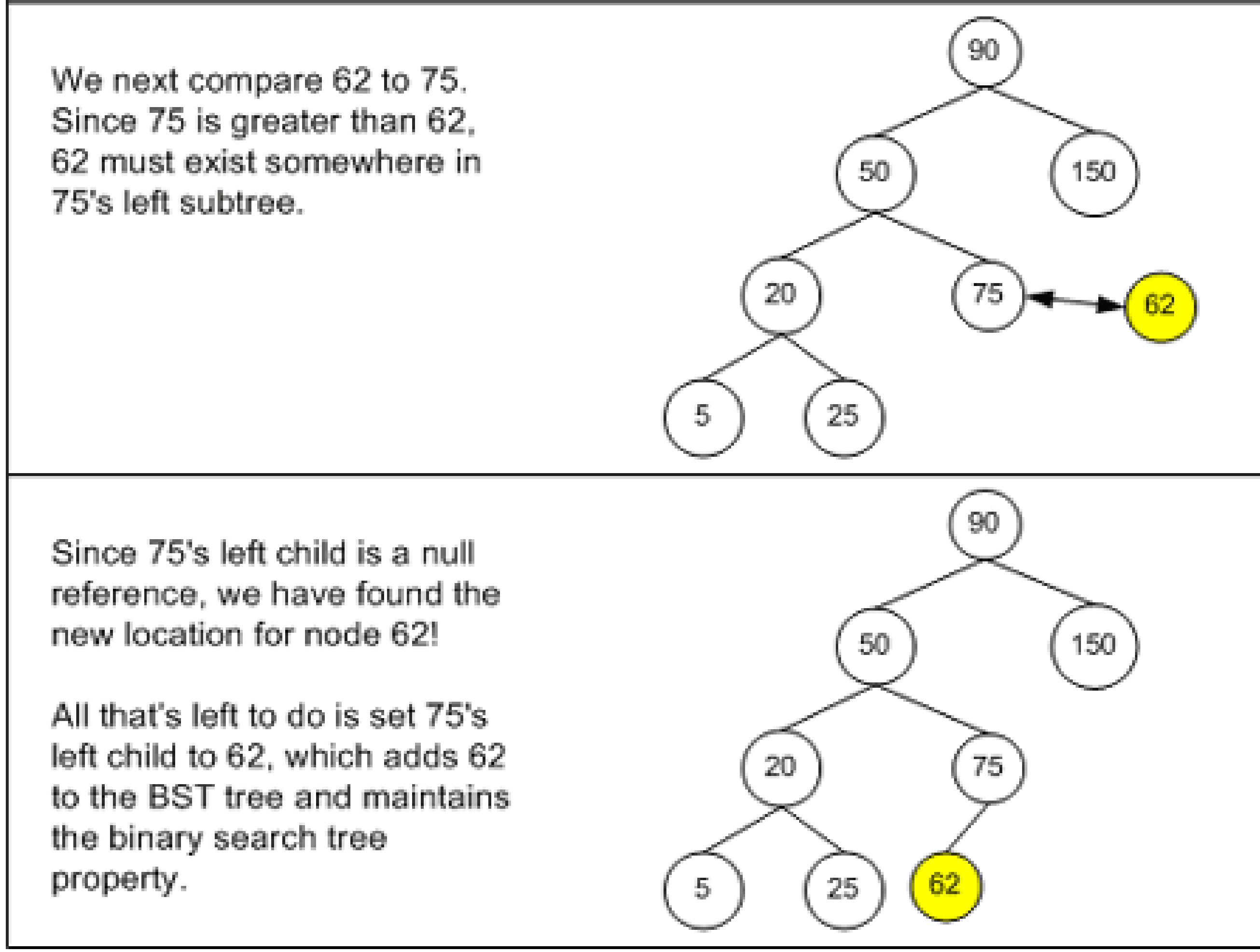
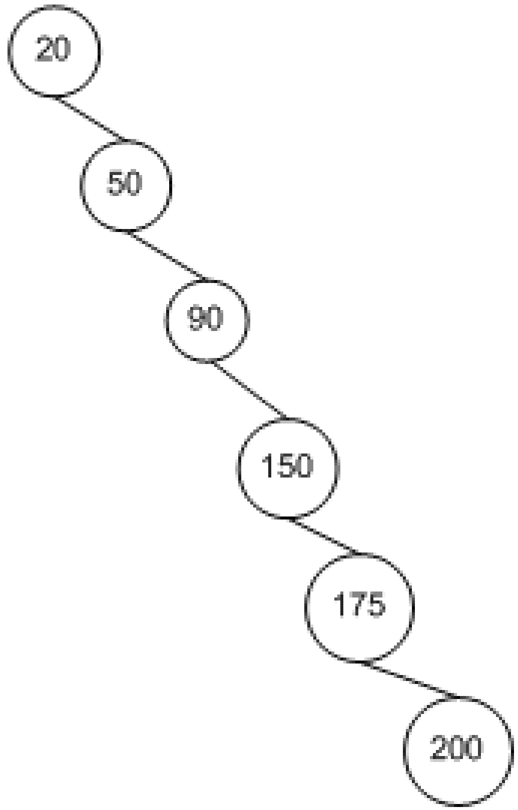


Figure 1. Inserting a new node into a BST

Searching and inserting in Binary search trees

Searching and inserting in Binary search trees



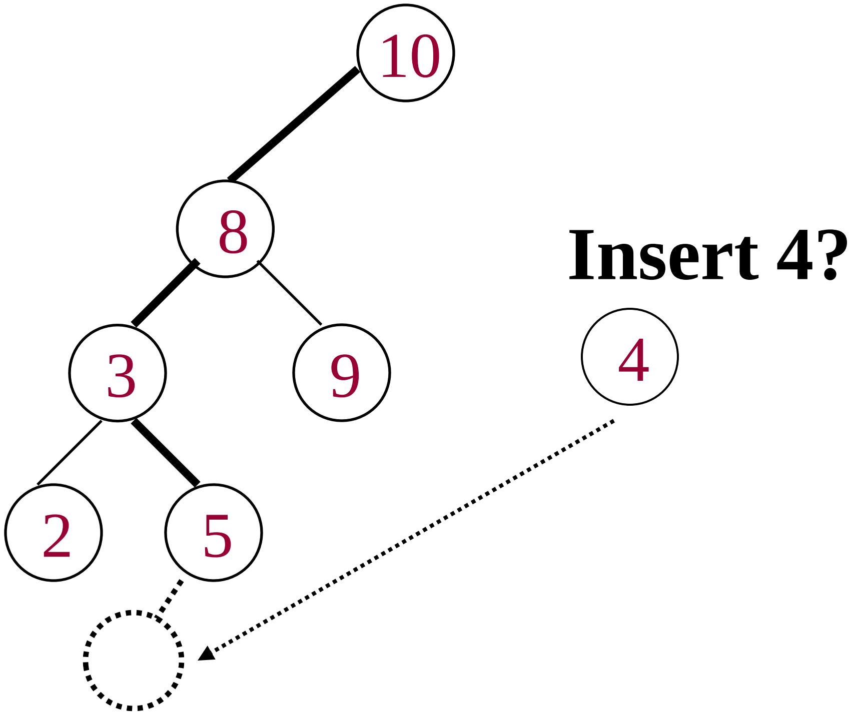
A BST after nodes with values of 20, 50, 90, 150, 175, and 200 have been added

Inserting a new key in a BST

How to insert a new key?

The same procedure used for search also applies: Determine the location by searching. Search will fail. Insert new key where the search failed.

Example:

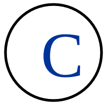


Building a BST

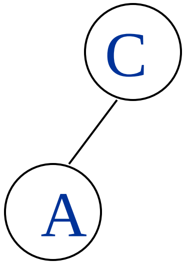
Build a BST from a sequence of nodes read one a time

Example: Inserting **C A B L M** **(in this order!)**

1) Insert C

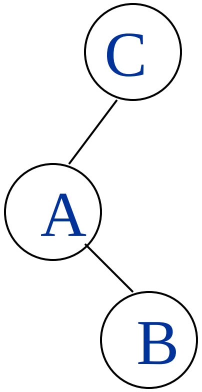


2) Insert A

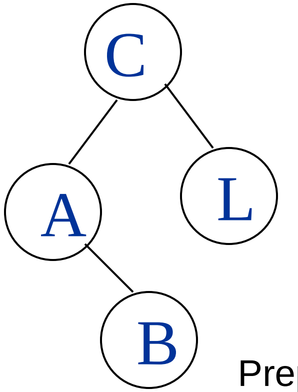


Building a BST

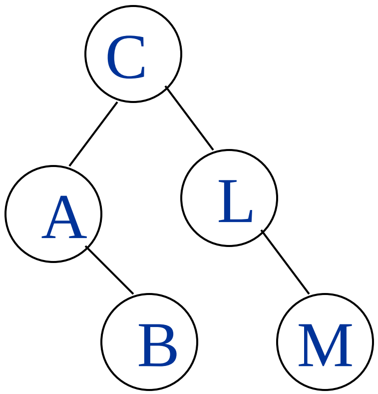
3) Insert B



4) Insert L



5) Insert M

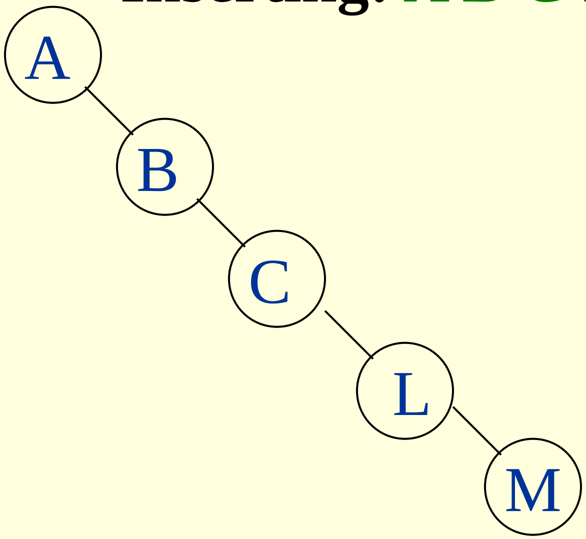


Building a BST

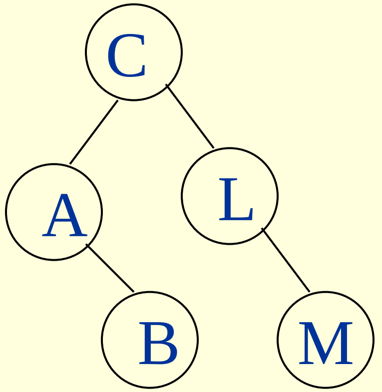
Is there a unique BST for letters A B C L M ?

NO! Different input sequences result in different trees

Inserting: A B C L M



Inserting: C A B L M



Sorting with a BST

Given a BST can you output its keys in sorted order?

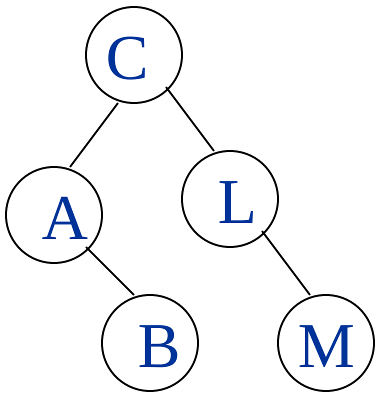
Visit keys with Inorder:

- visit left
- print root
- visit right

How can you find the minimum?

How can you find the maximum?

Example:



Inorder visit prints:

A B C L M

Preorder Traversal

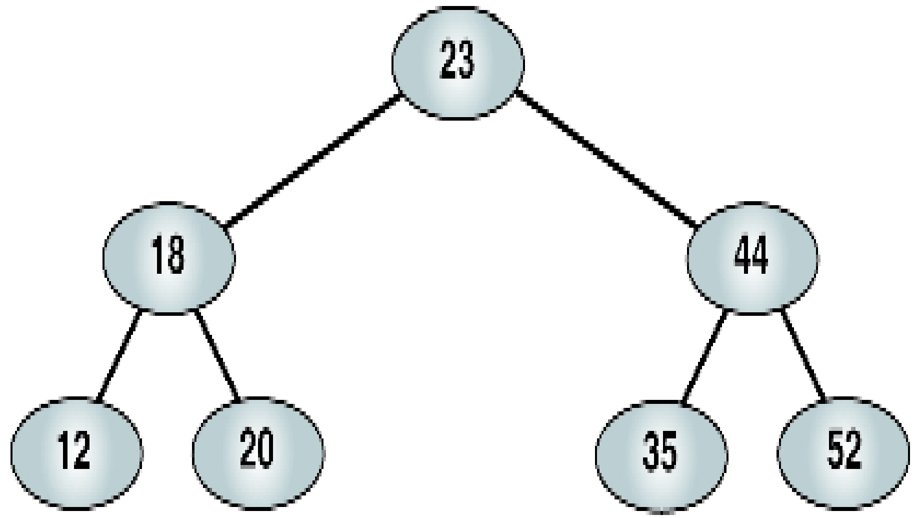


FIGURE 7-4 Example of a Binary Search Tree

23 18 12 20 44 35 52

Postorder Traversal

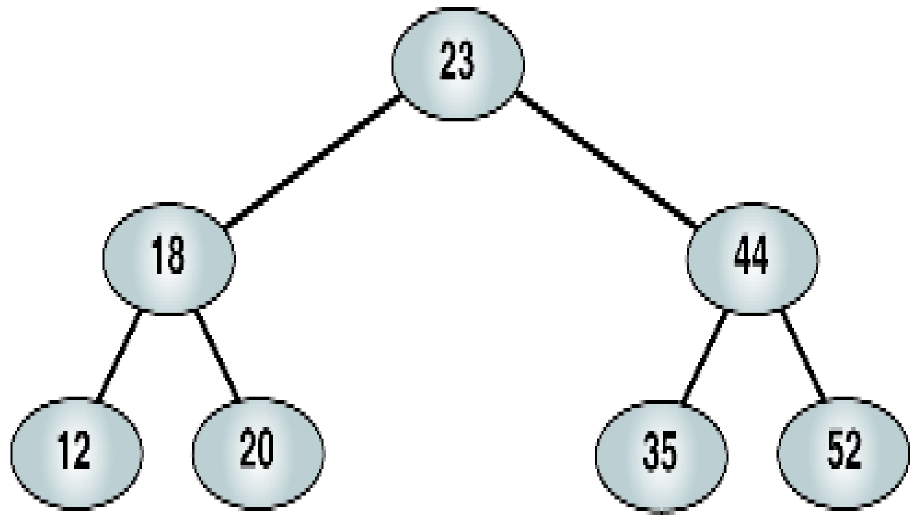


FIGURE 7-4 Example of a Binary Search Tree

12 20 18 35 52 44 23

Inorder Traversal

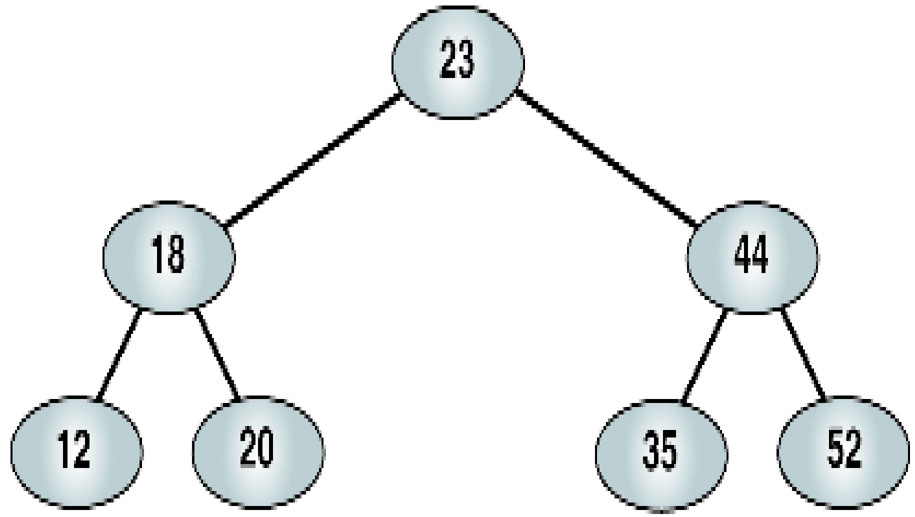


FIGURE 7-4 Example of a Binary Search Tree

12 18 20 23 35 44 52

Inorder traversal of a binary search tree produces a sequenced list

Right-Node-Left Traversal

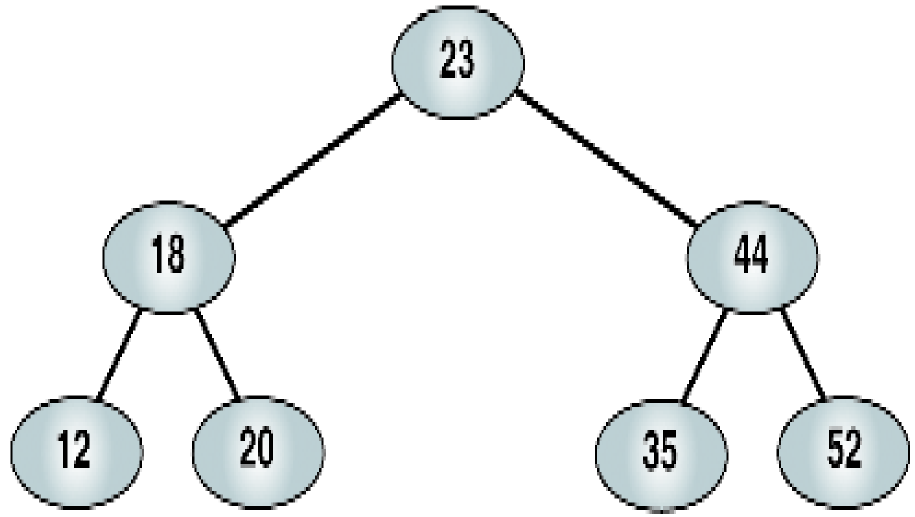


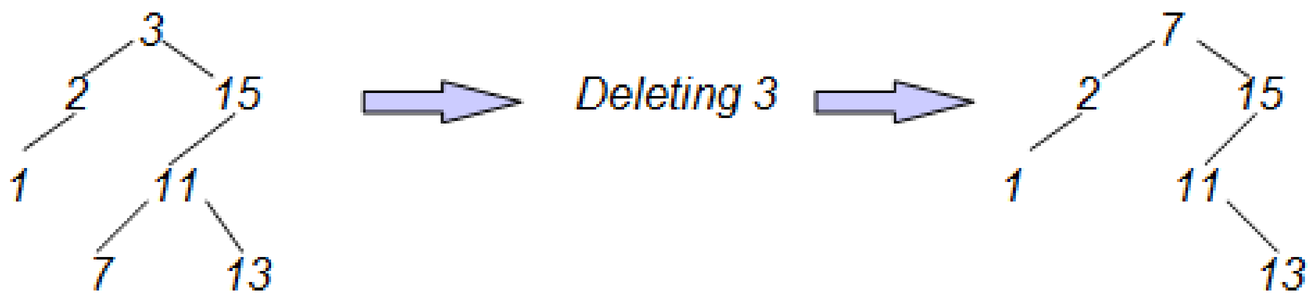
FIGURE 7-4 Example of a Binary Search Tree

52 44 35 23 20 18 12

Right-node-left traversal of a binary search tree produces a descending sequence

Deletion in binary search tree

Consider the tree:



The following cases of deletions are possible:

1. *Delete a note with no children, for example 1. This only requires the appropriate link in the parent node to be made null.*
2. *Delete a node which has only one child, for example 15. In this case, we must set the corresponding child link of the parent's parent to point to the only child of the node being deleted.*
3. *Delete a node with two children, for example 3. The delete method is based on the following consideration: in-order traversal of the resulting tree (after delete operation) must yield an ordered list. To ensure this, the following steps are carried out:*

   *<u>Step 1</u>: Replace **3** with the node with the next largest datum, i.e. **7**.*

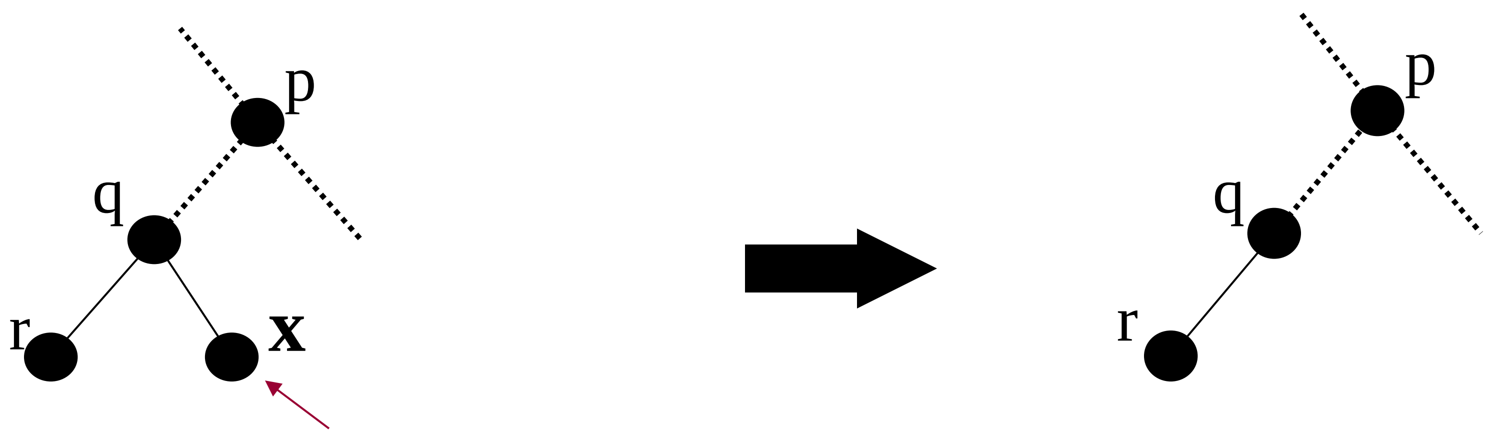
   *<u>Step 2</u>: Make the left link of **11** point to the right child of **7** (which is null here).*

   *<u>Step 3</u>: Copy the links from the node containing **3** to the node containing **7**, and make the parent node of **3** point to **7**.*

Deleting from a BST

To delete node with key x first you need to **search** for it. Once found, apply one of the following three cases

CASE A: x is a leaf

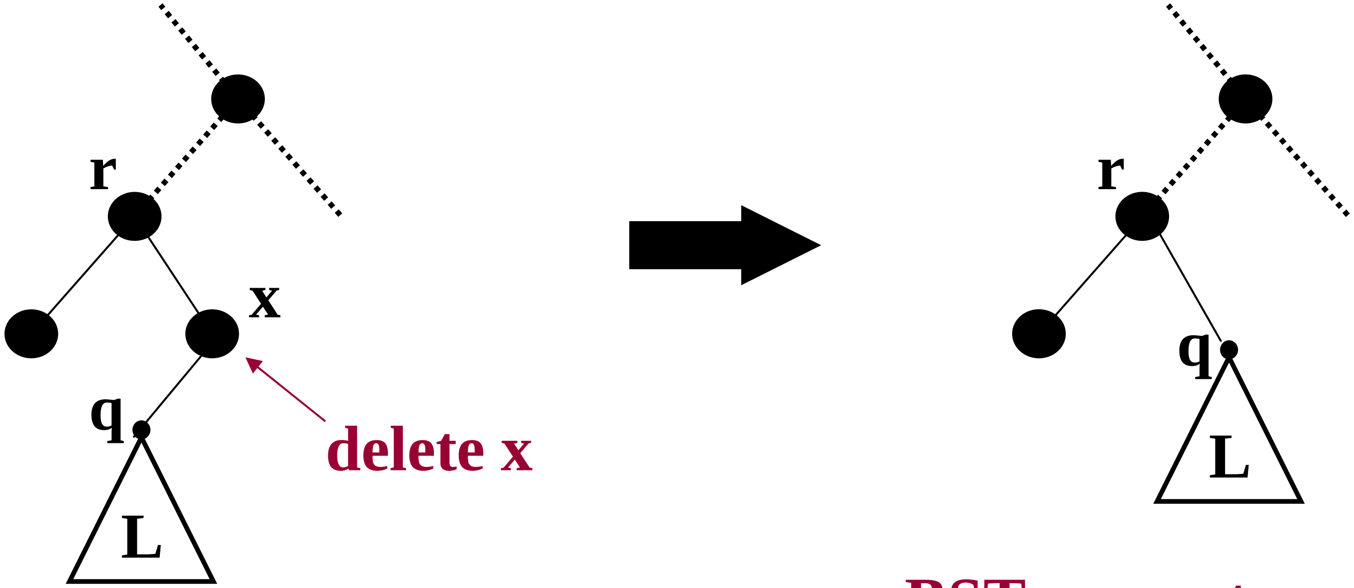


delete x

BST property maintained

Deleting from a BST cont.

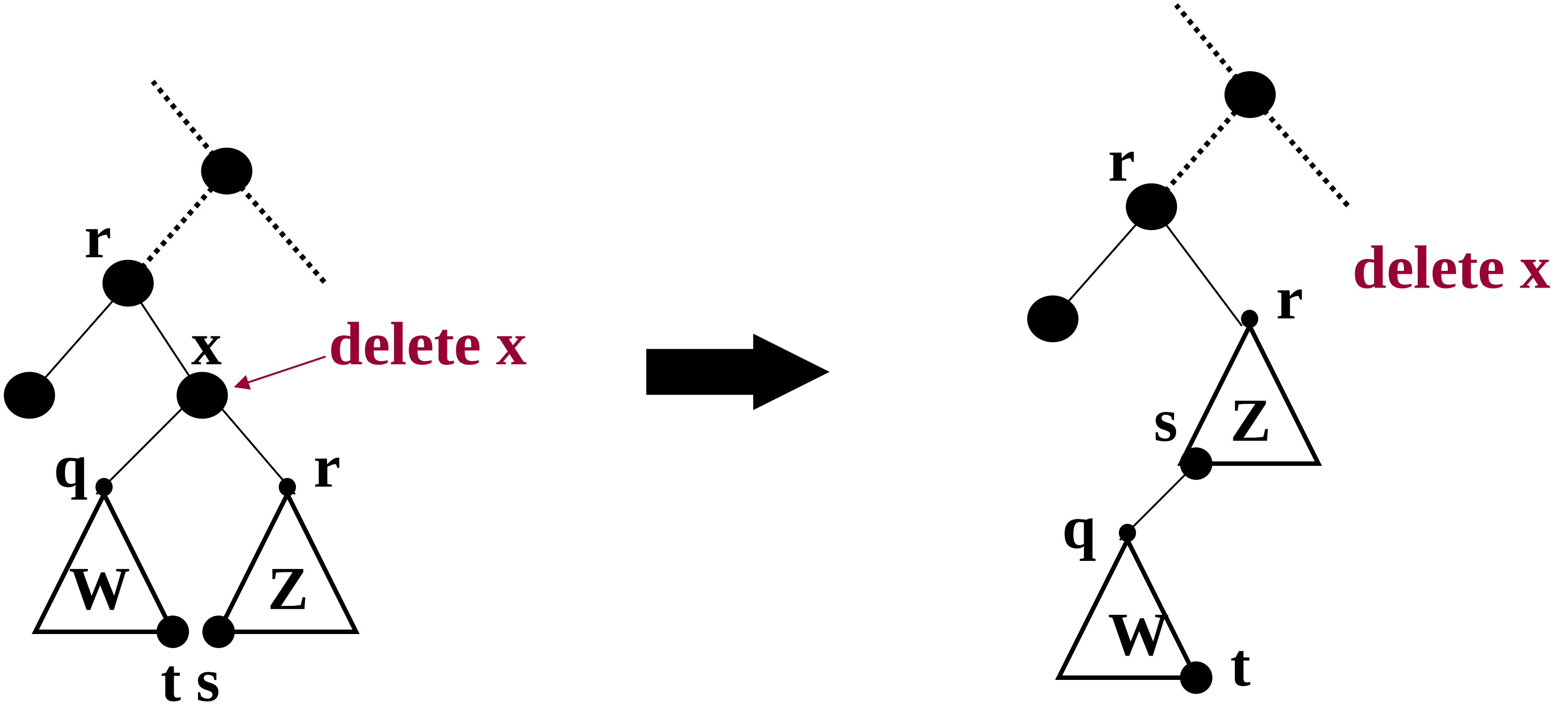
Case B: x is interior with only one subtree



BST property maintained

Deleting from a BST cont.

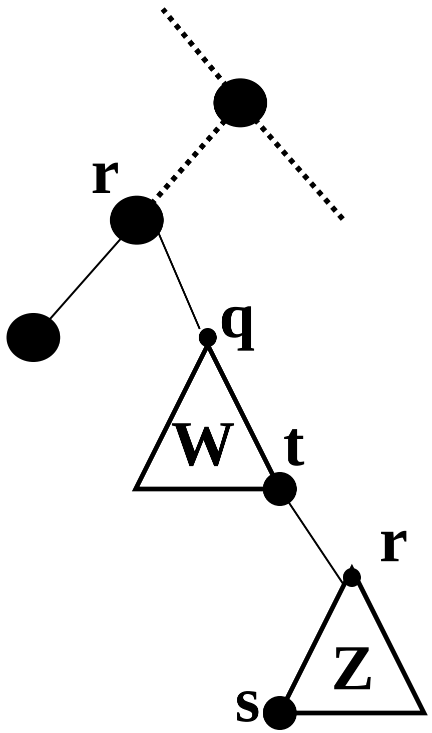
Case C: x is interior with two subtrees



BST property maintained

Deleting from a BST cont.

Case C cont: … or you can also do it like this



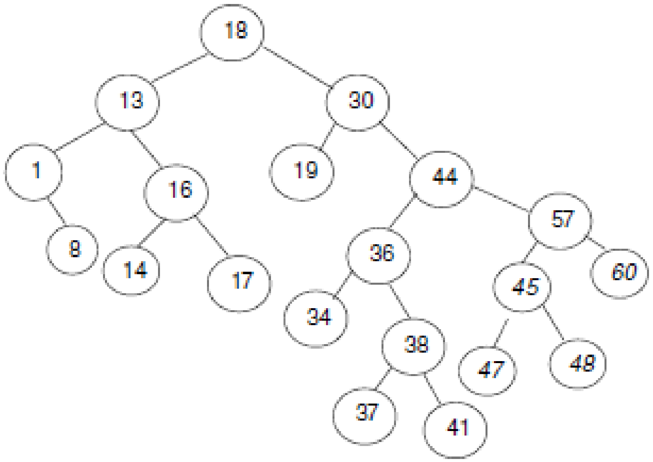
$q < x < r$

$\Rightarrow$ Q is smaller than the smaller element in Z

$\Rightarrow$ R is larger than the largest element in W

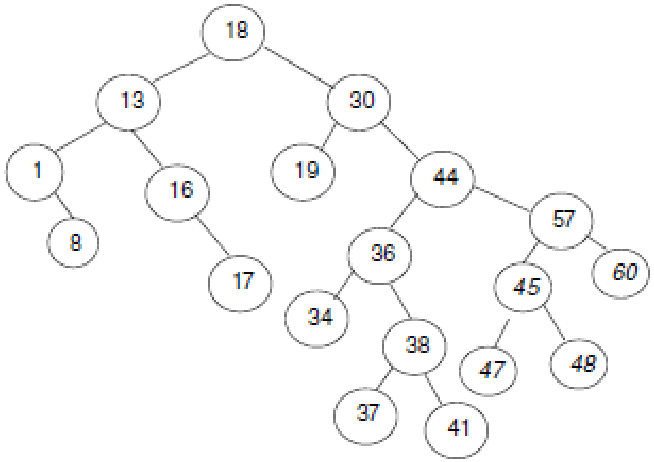
Deletion From Binary Search Trees

- There are three possible cases to consider:
  - **Deleting a leaf (node with no children):** Deleting a leaf is easy, as we can simply remove it from the tree.



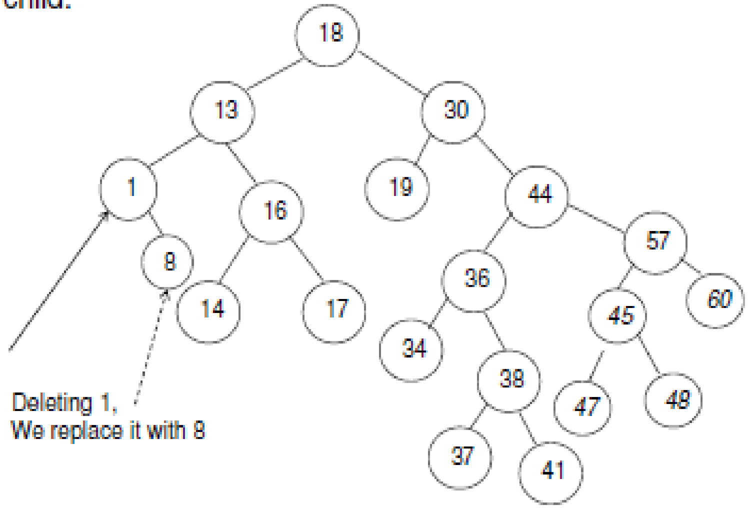
Deletion From Binary Search Trees

- Deleting 14 we obtain



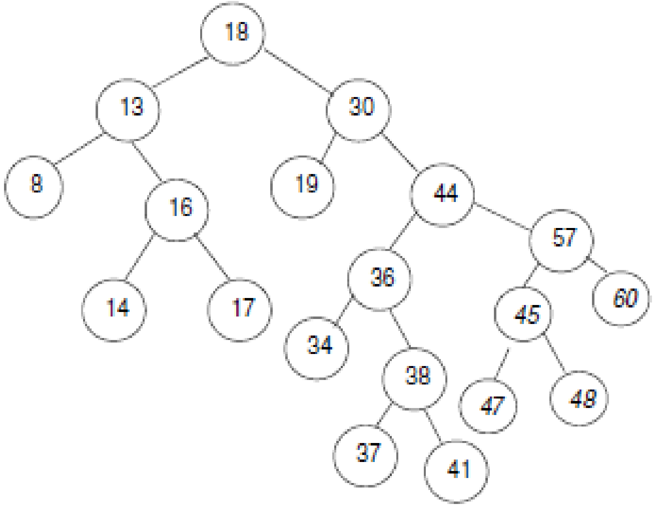
Deletion From Binary Search Trees

- **Deleting a node with one child:** Delete it and replace it with its child.



Deletion From Binary Search Trees

- We obtain

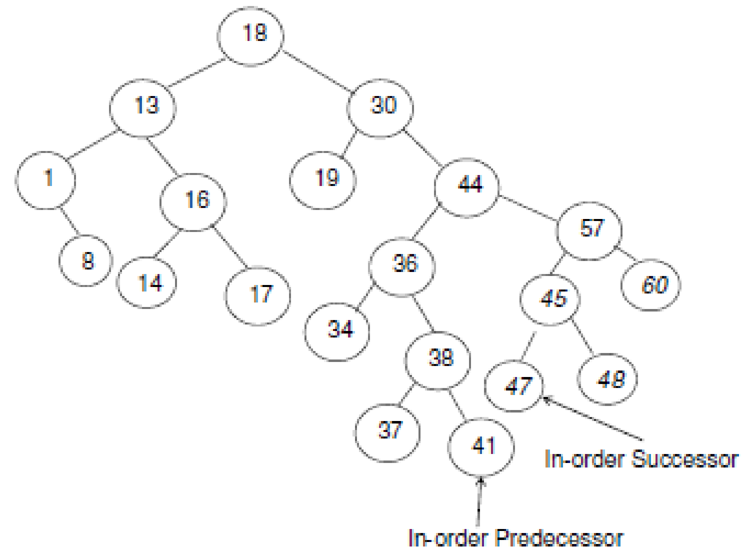


Deletion From Binary Search Trees

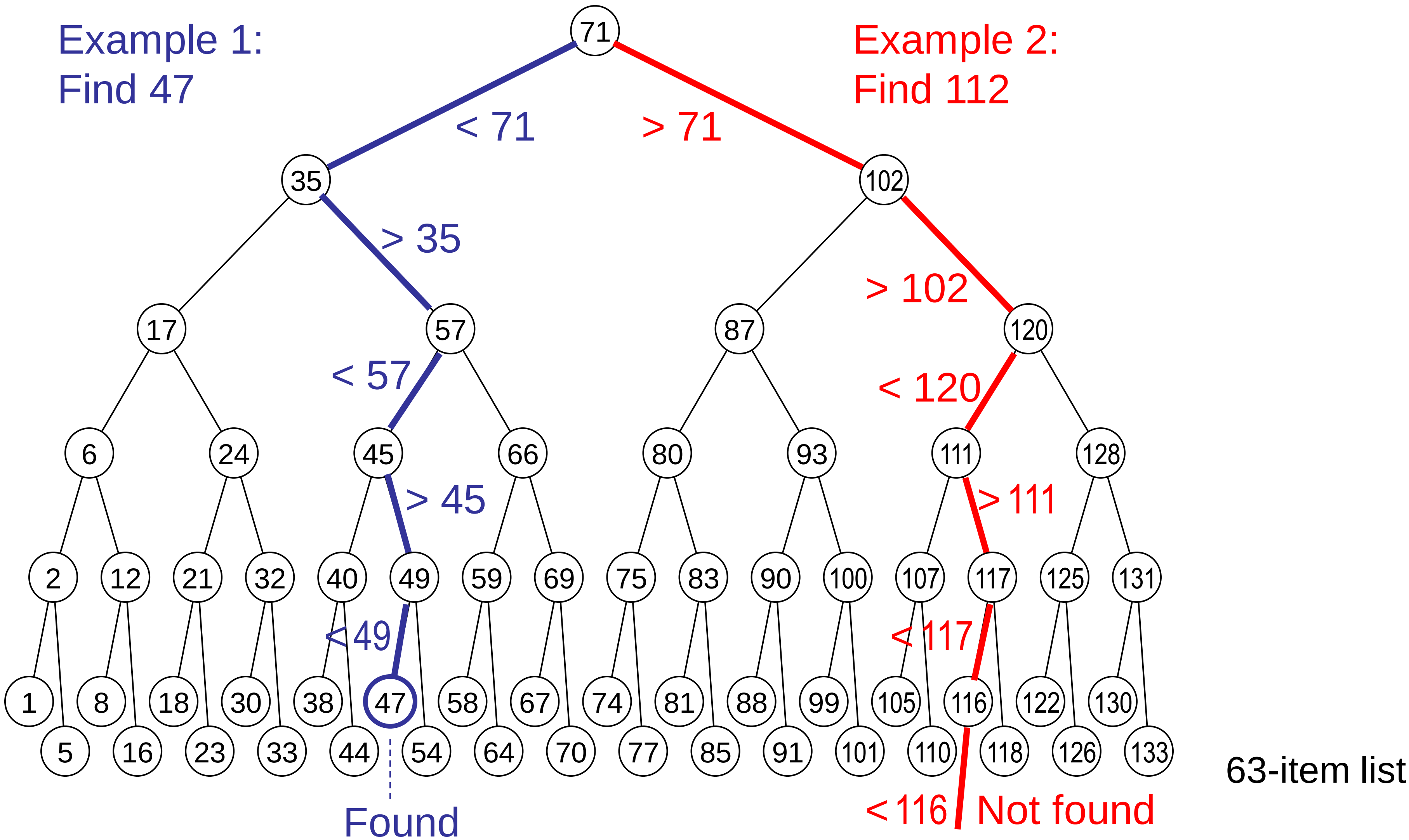
- **Deleting a node with two children:** Call the node to be deleted "N". Do not delete N. Instead, choose either its in-order successor node or its in-order predecessor node, "R". Replace the value of N with the value of R, then delete R. (Note: R itself has up to one child.)
- As with all binary trees, a node's in-order successor is the left-most child of its right subtree, and a node's in-order predecessor is the right-most child of its left subtree.
  - In either case, this node will have zero or one children.
  - Delete it according to one of the two simpler cases above.

Deletion From Binary Search Trees

- If we want to delete 44

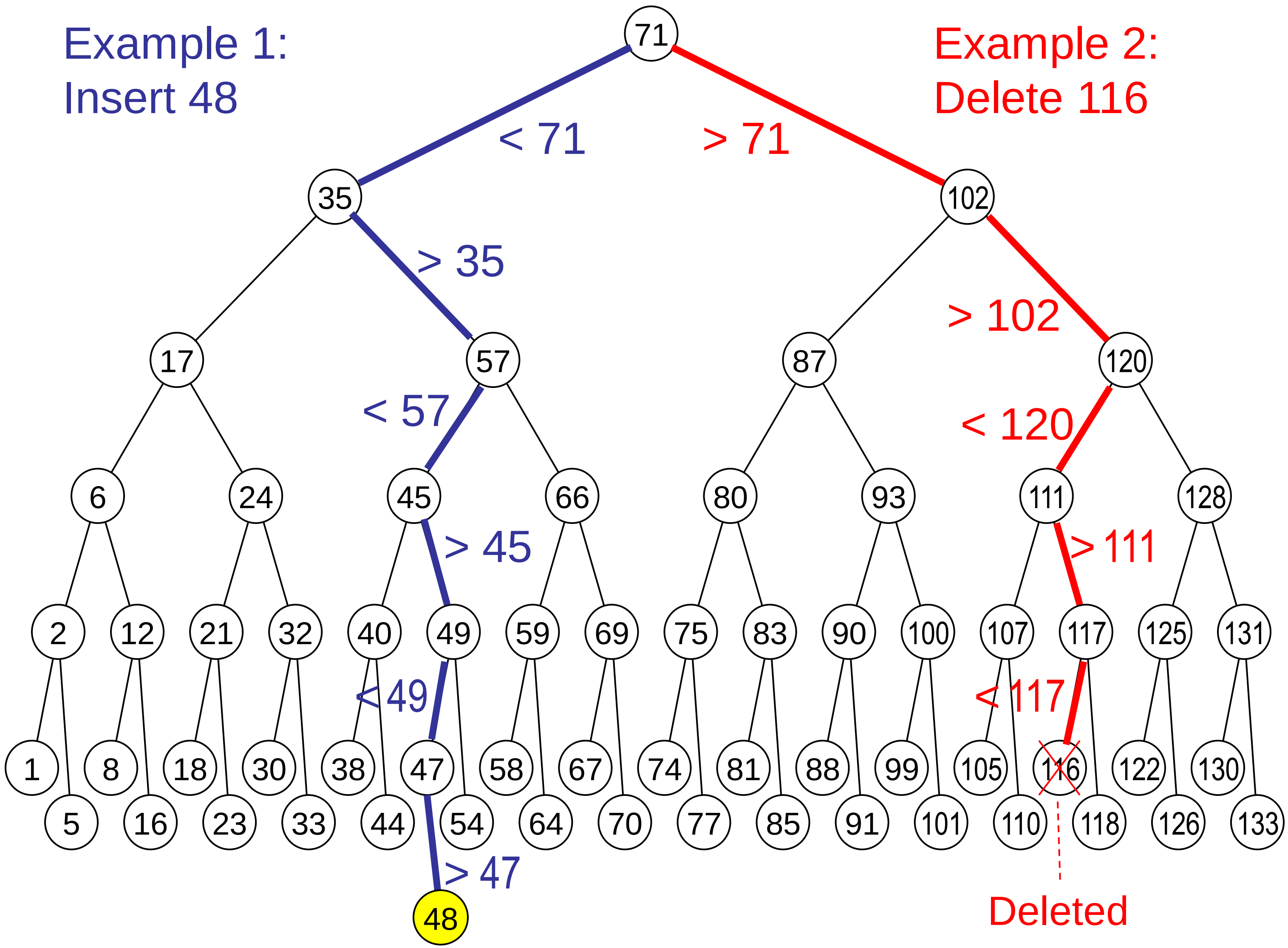


Binary Search Trees



Prepared by, Jesmin Akhter,
Lecturer, IIT,JU

Insertions and Deletions in Binary Search Trees



Prepared by, Jesmin Akhter,
Lecturer, IIT,JU