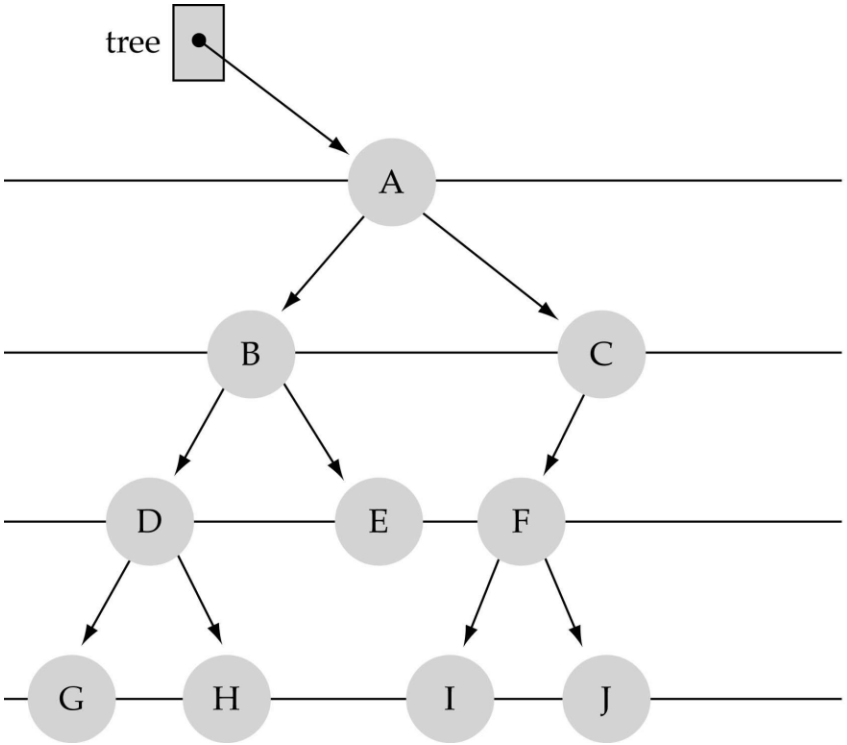
Subject: Data structures using c
Topic: Binary tree
Name of the teacher: Lisna Thomas
Academic year: 2020-2021

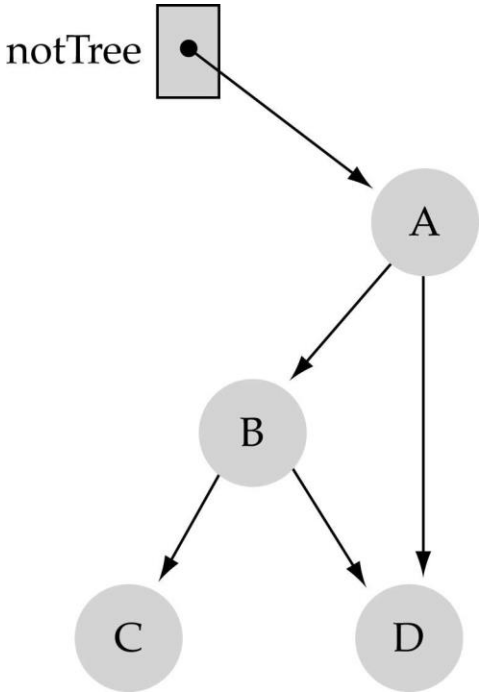
What is a binary tree?

- *Property 1*:



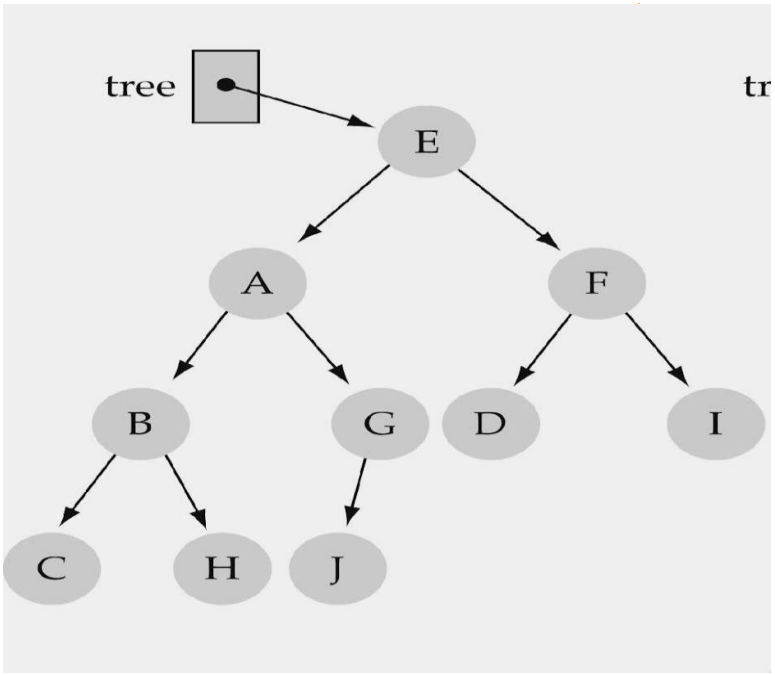
What is a binary tree? (cont.)

- *Property 2*:



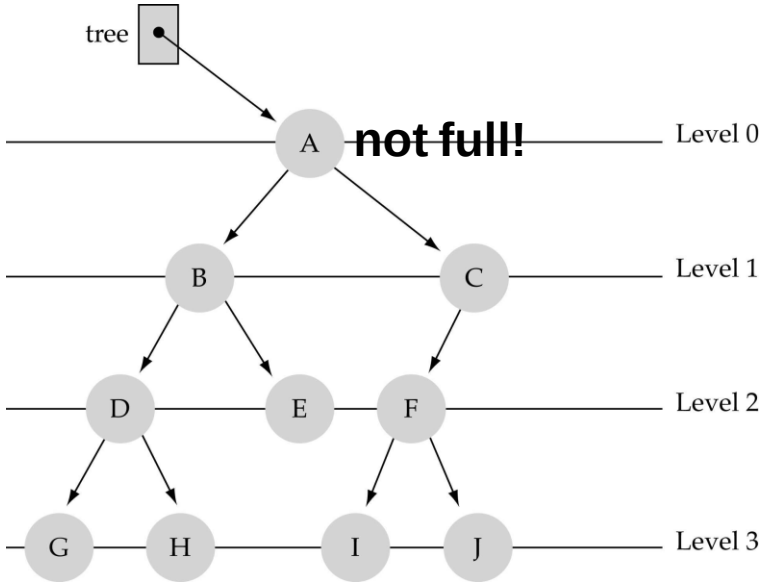
Some terminology

- *children*
- *parent*
-
-



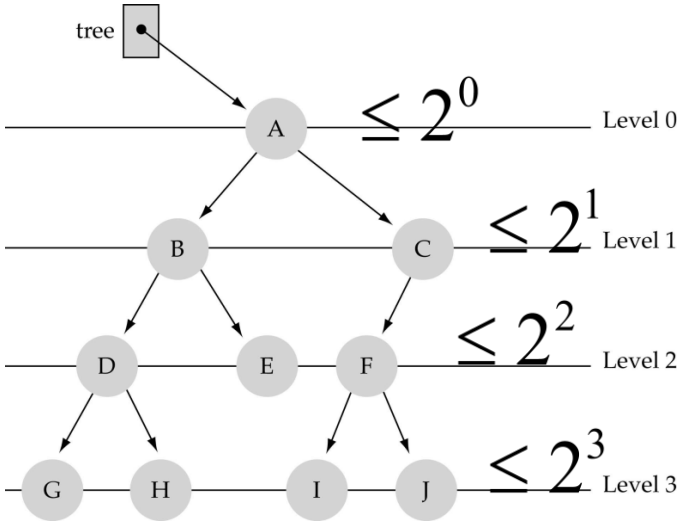
Some terminology (cont'd)

-
-
-
  Warning:
-



What is the max #nodes at some level *l*?

l $\quad$ 2^l



What is the total #nodes **N** of a full tree with height **h**?

$$N = \underset{l=0}{2^0} + \underset{l=1}{2^1} + \ldots + \underset{l=h-1}{2^{h-1}} = 2^h - 1$$

$$x^0 + x^1 + \ldots + x^{n-1} = \sum_{i=0}^{n-1} x^i = \frac{x^n - 1}{x - 1}$$

What is the height **h** of a full tree with **N** nodes?

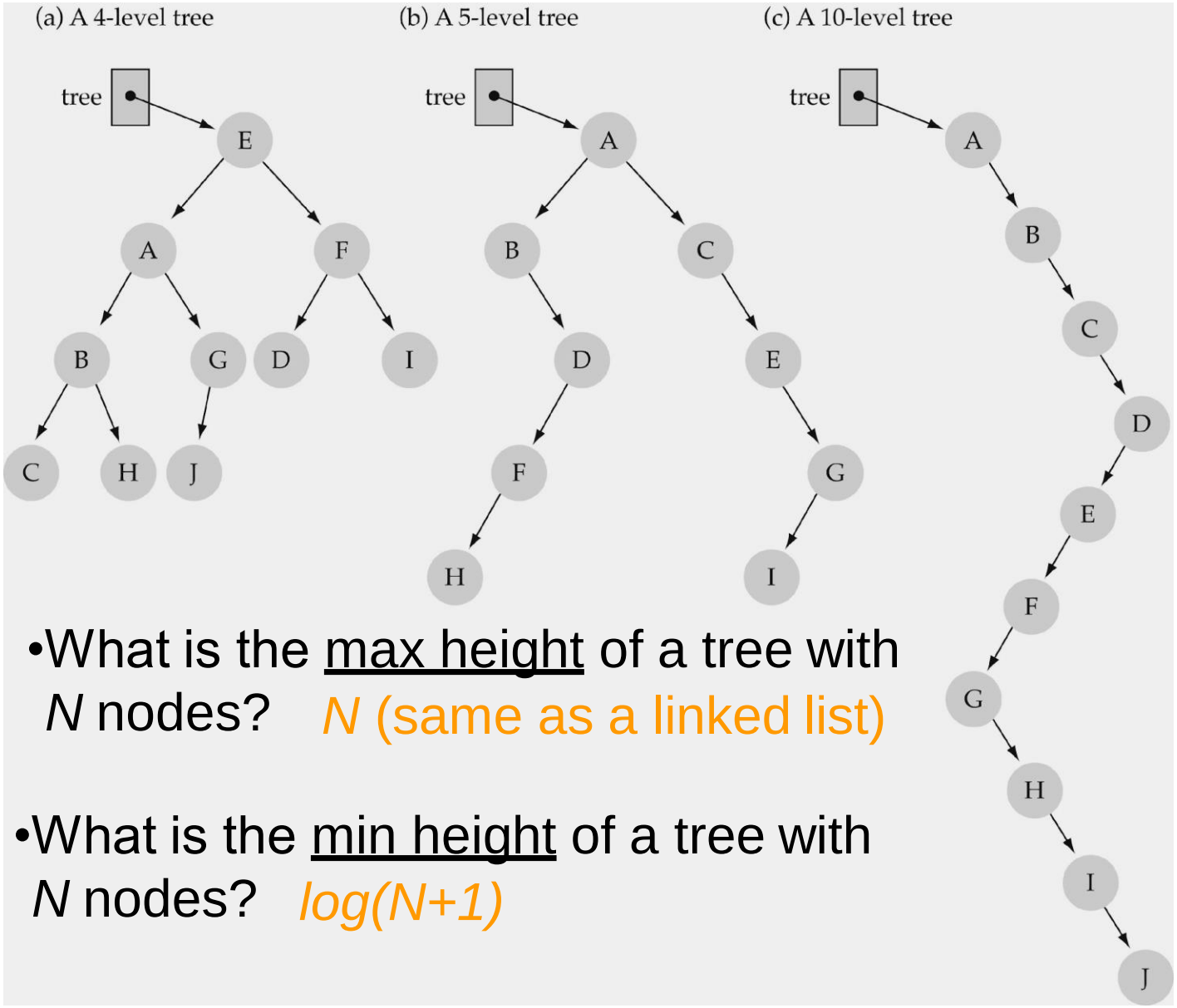
$$2^h - 1 = N$$

$$\Rightarrow 2^h = N + 1$$

$$\Rightarrow h = \log(N+1) \rightarrow O(\log N)$$

Why is h important?

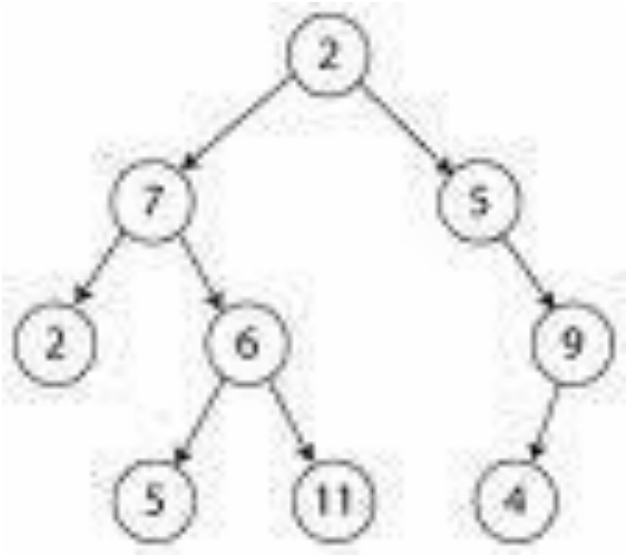
-
-



- What is the <u>max height</u> of a tree with *N* nodes? *N* (same as a linked list)

- What is the <u>min height</u> of a tree with *N* nodes? *log(N+1)*

How to search a binary tree?

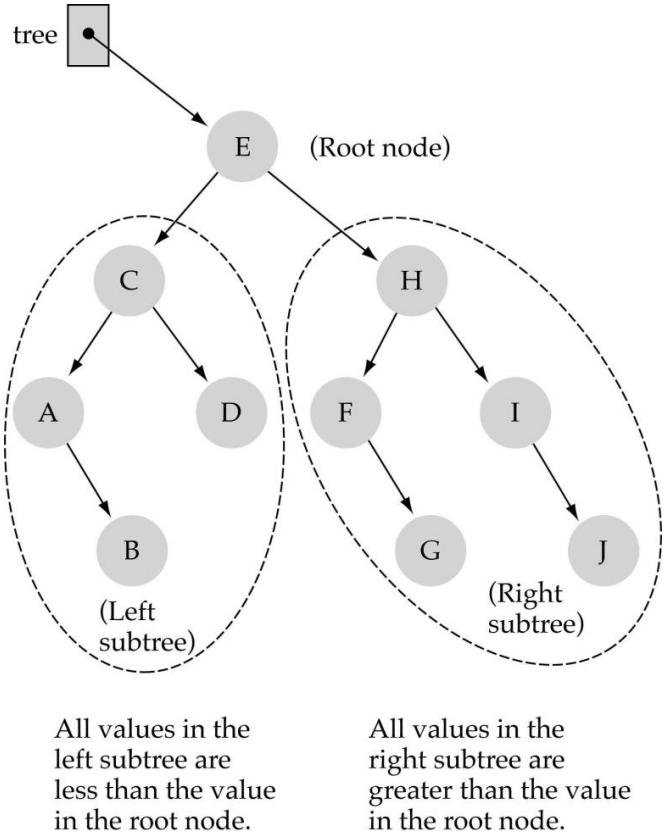


No O(N)

Binary **Search** Trees (BSTs)

- *greater*

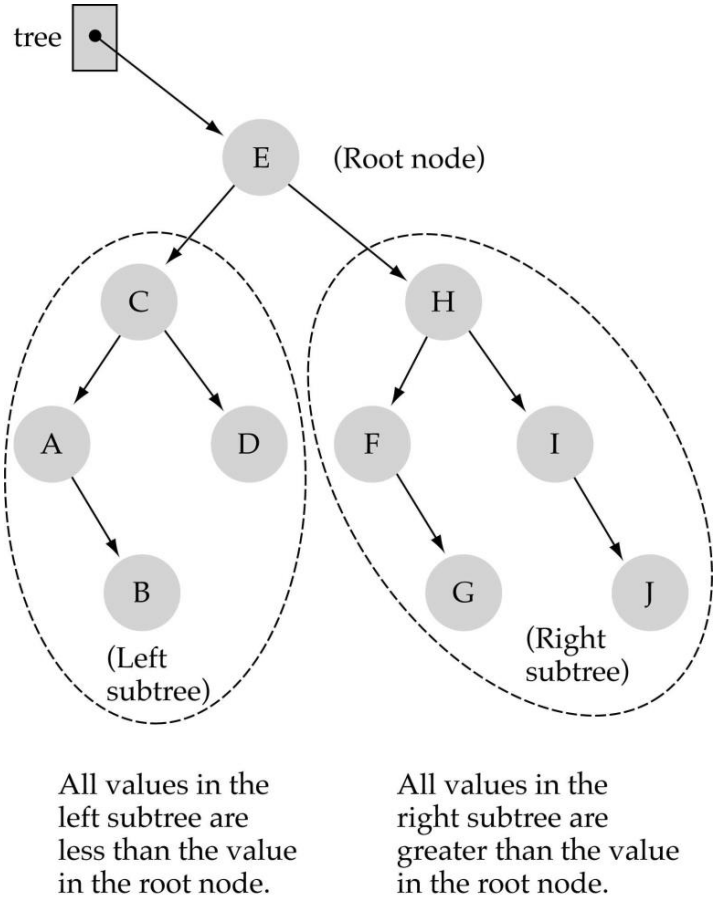
- *less*



Binary **Search** Trees (BSTs)

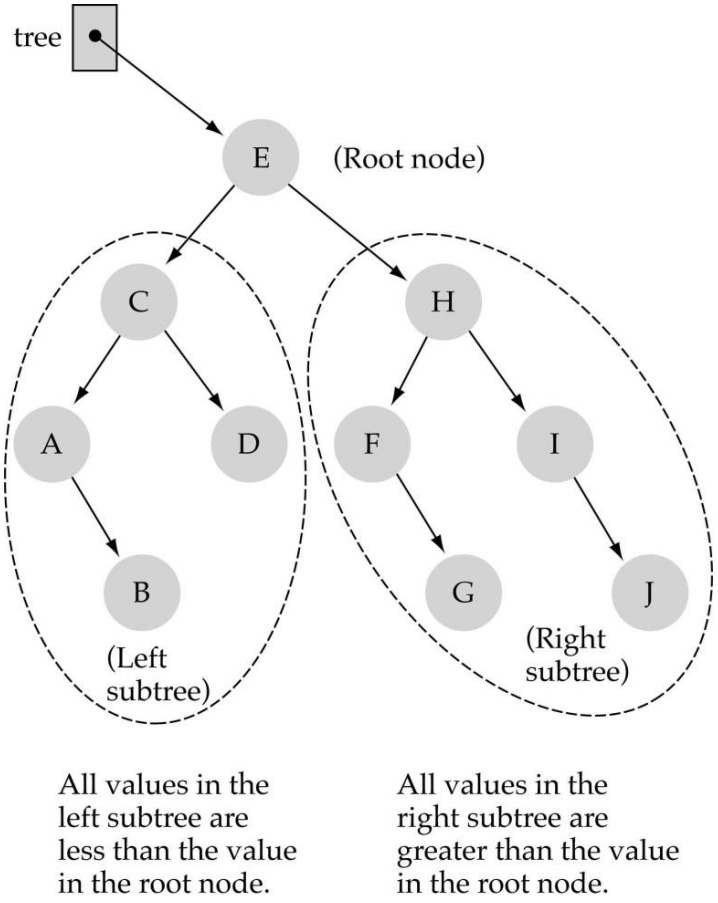
greater

less



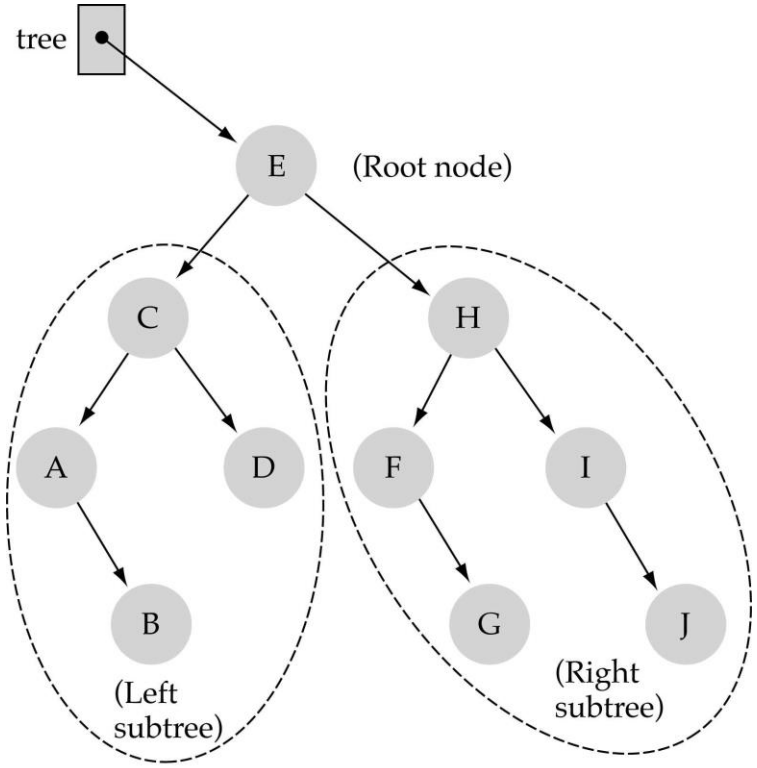
Binary **Search** Trees (BSTs)

Ans:



Ans:

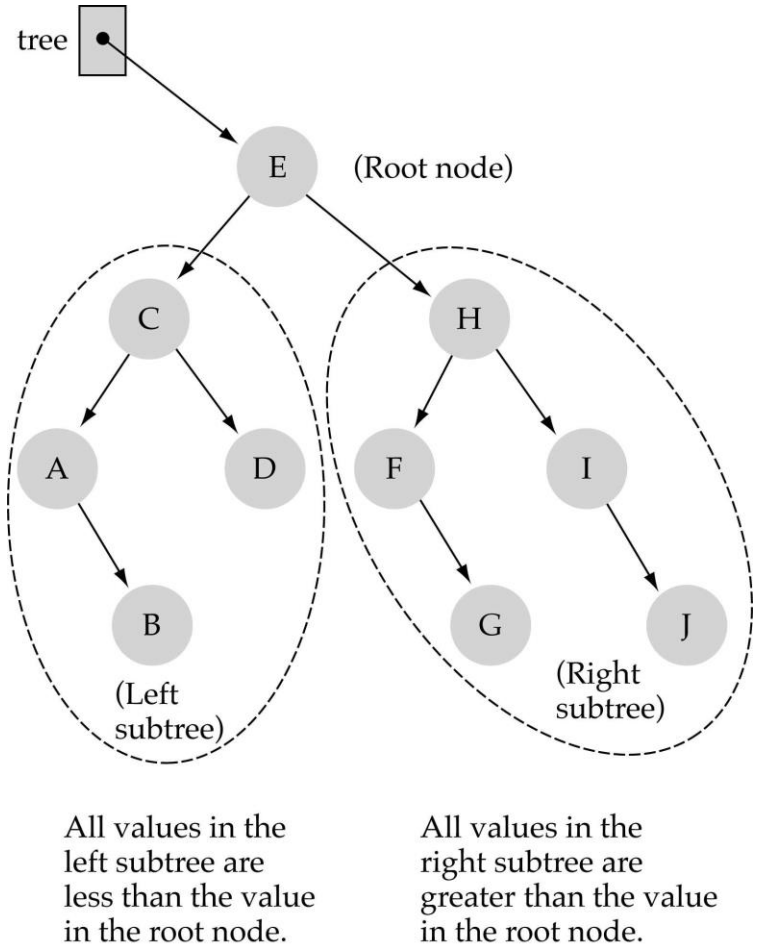
How to search a binary search tree?



All values in the left subtree are less than the value in the root node.

All values in the right subtree are greater than the value in the root node.

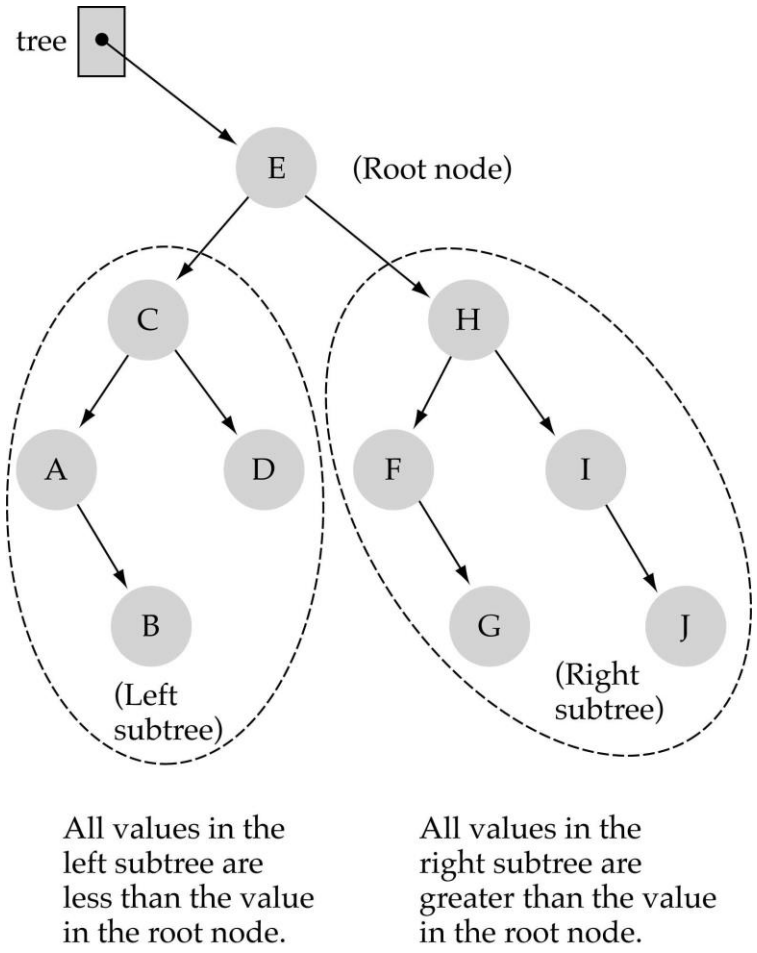
How to search a binary search tree?



left subtree

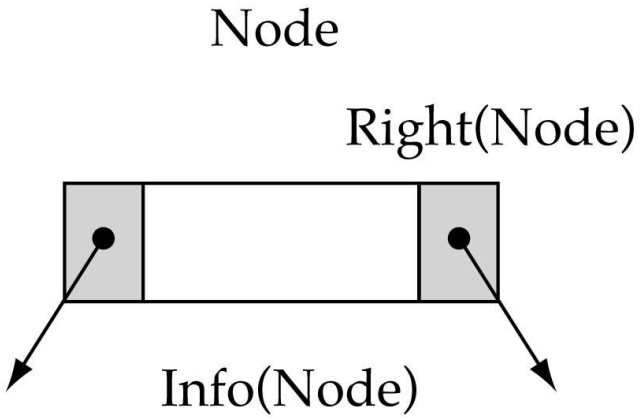
right subtree

How to search a binary search tree?



Yes !! ---> O(logN)

Tree node structure



Binary Search Tree Specification

Binary Search Tree Specification

(cont.)

Function NumberOfNodes

-
-
-
-

Function NumberOfNodes (cont.)

NumberOfNodes

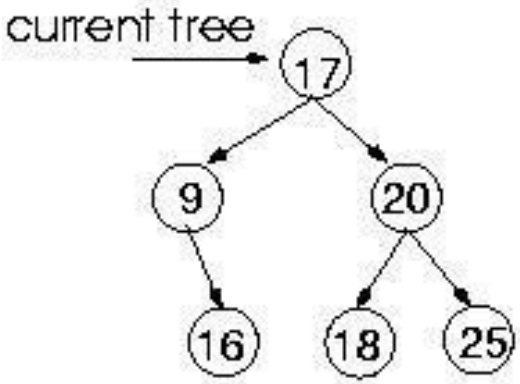
CountNodes

Function RetrieveItem

Retrieve: 18

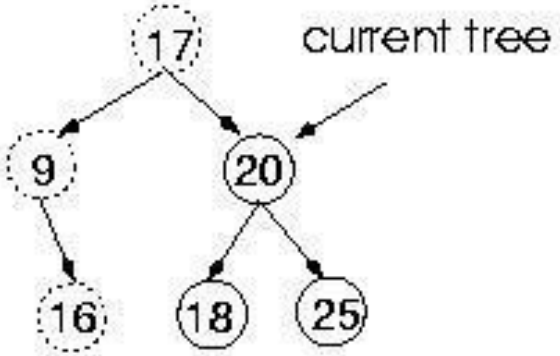
Compare 18 with 17:

Choose right subtree



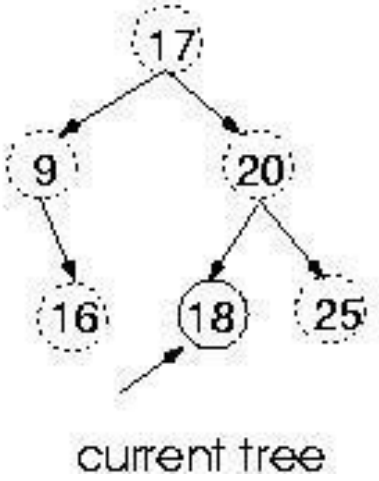
Compare 18 with 20:

Choose left subtree



Compare 18 with 18:

Found !!



Function RetrieveItem

-
-
-

Function RetrieveItem (cont.)

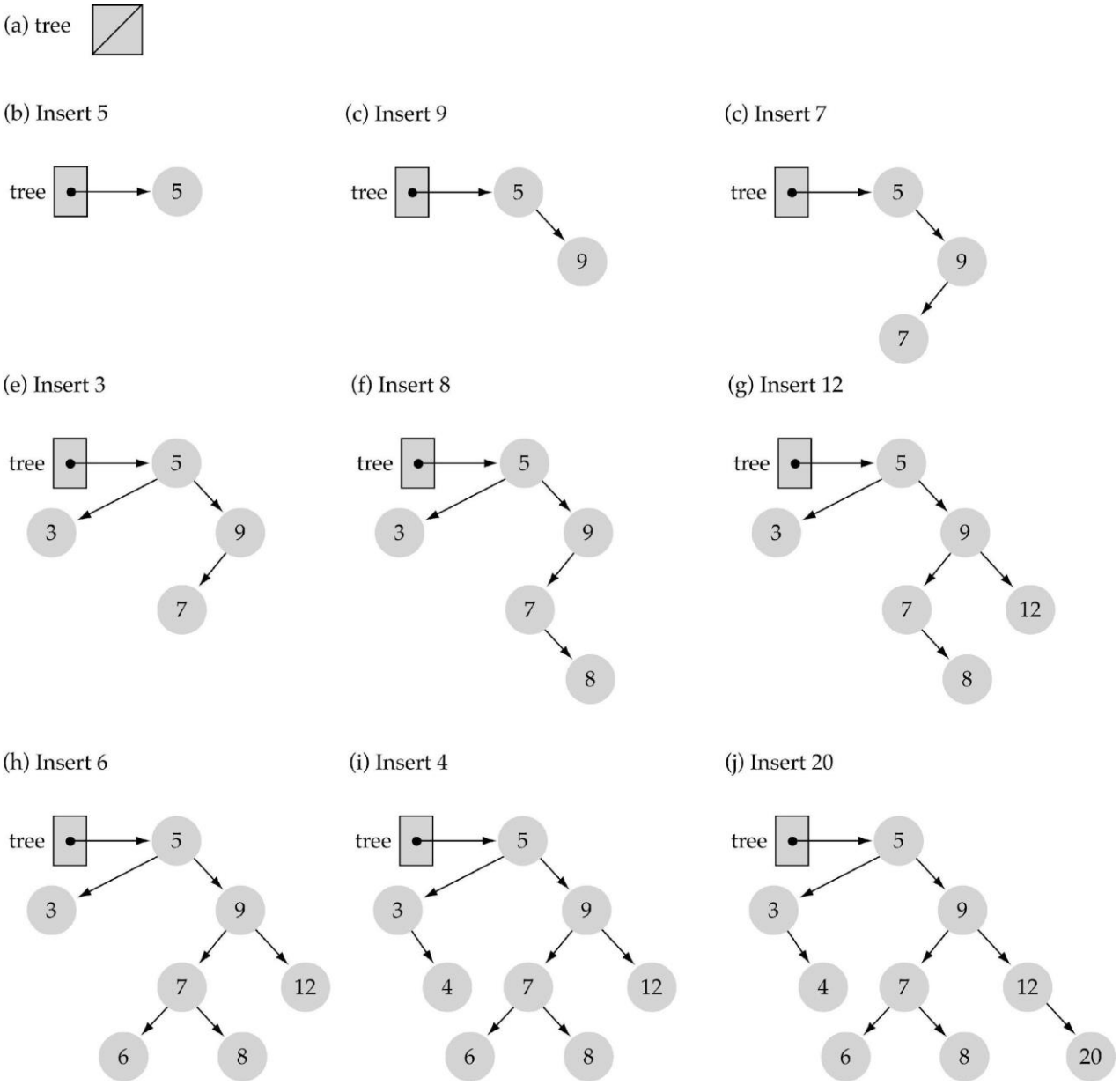
RetrieveItem

Retrieve

// base case 2

// base case 1

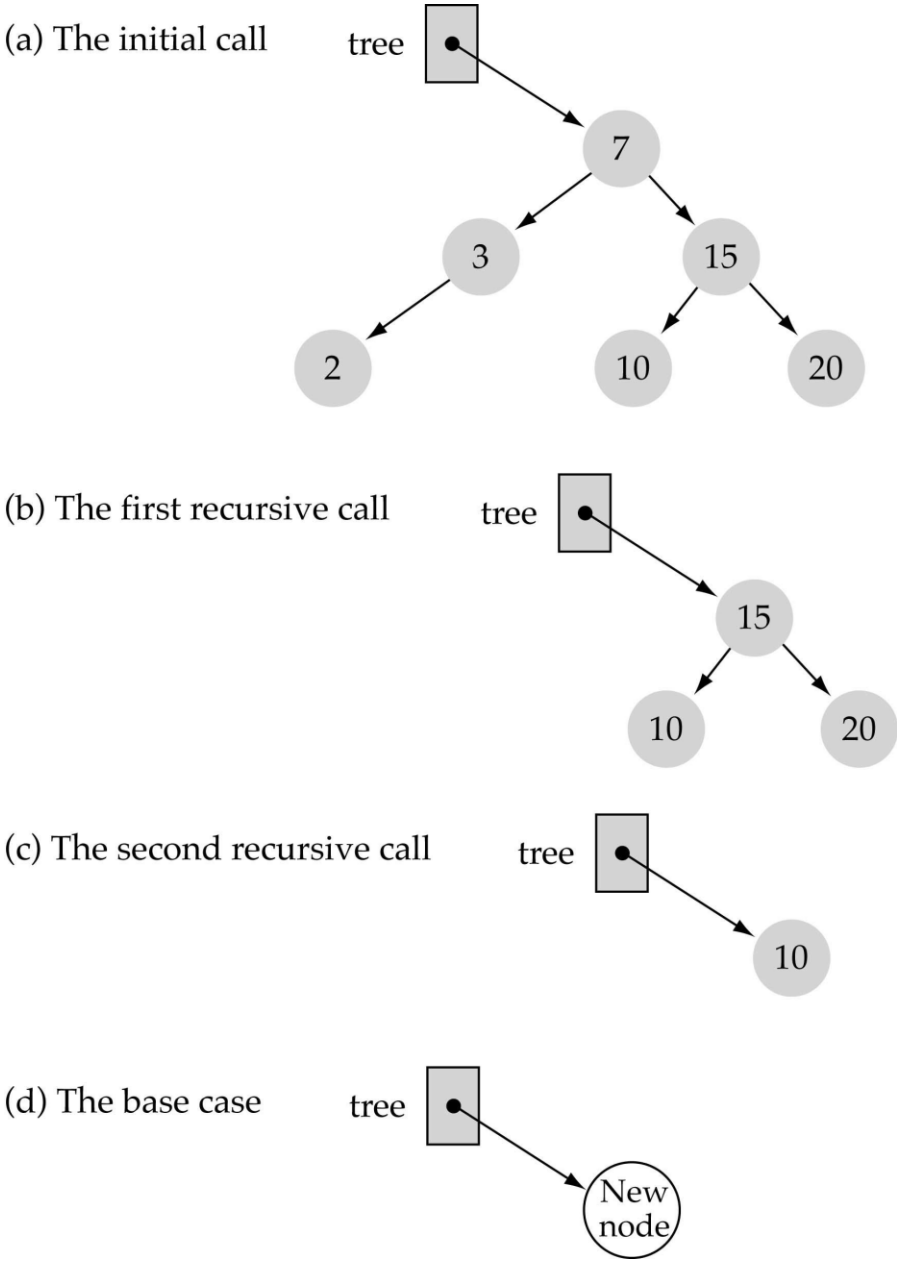
Function InsertItem



Function InsertItem (cont.)

-

e.g., insert 11



Function InsertItem (cont.)

- size
- base case(s)
- general case

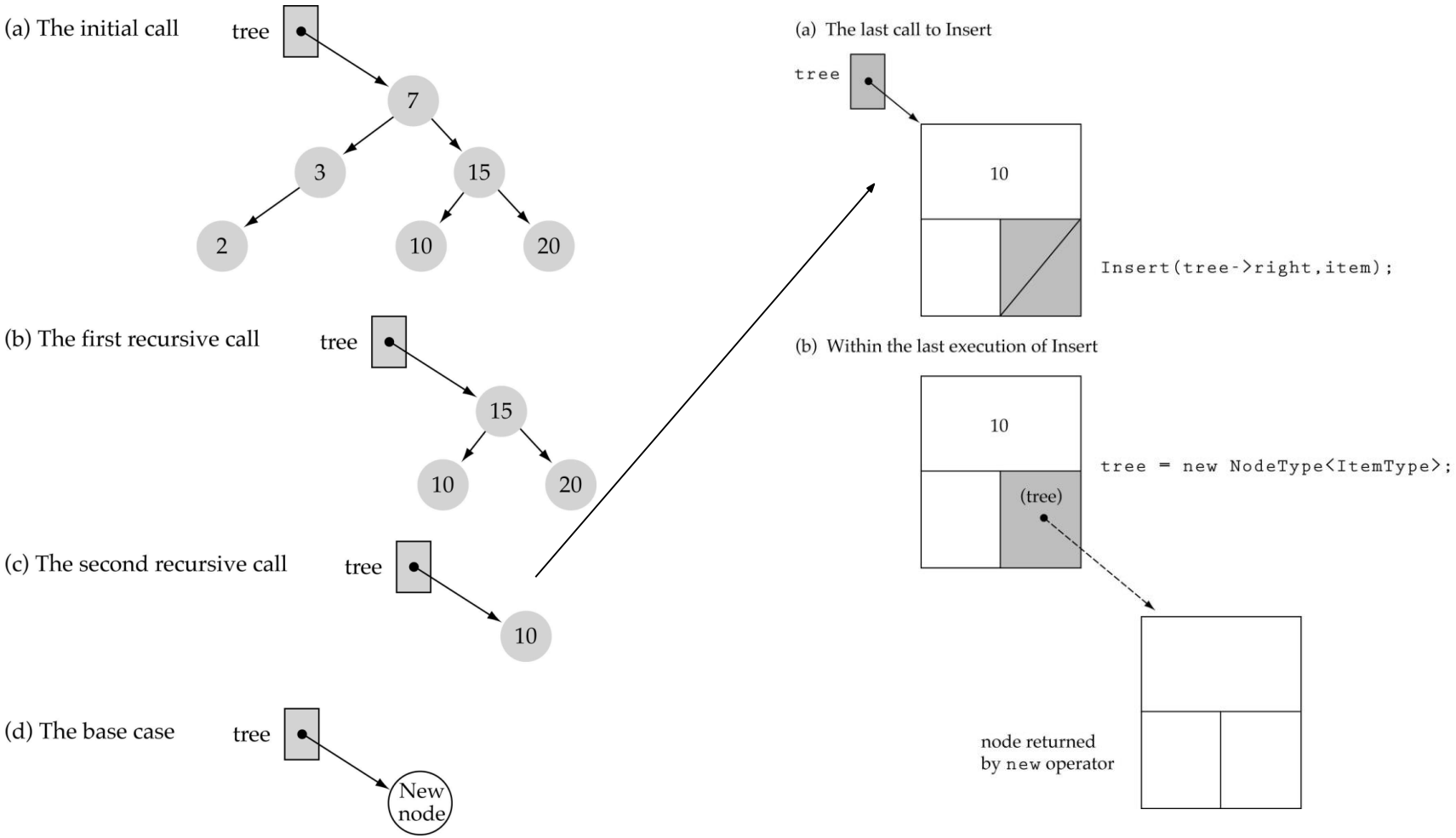
Function InsertItem (cont.)

InsertItem

Insert

// base case

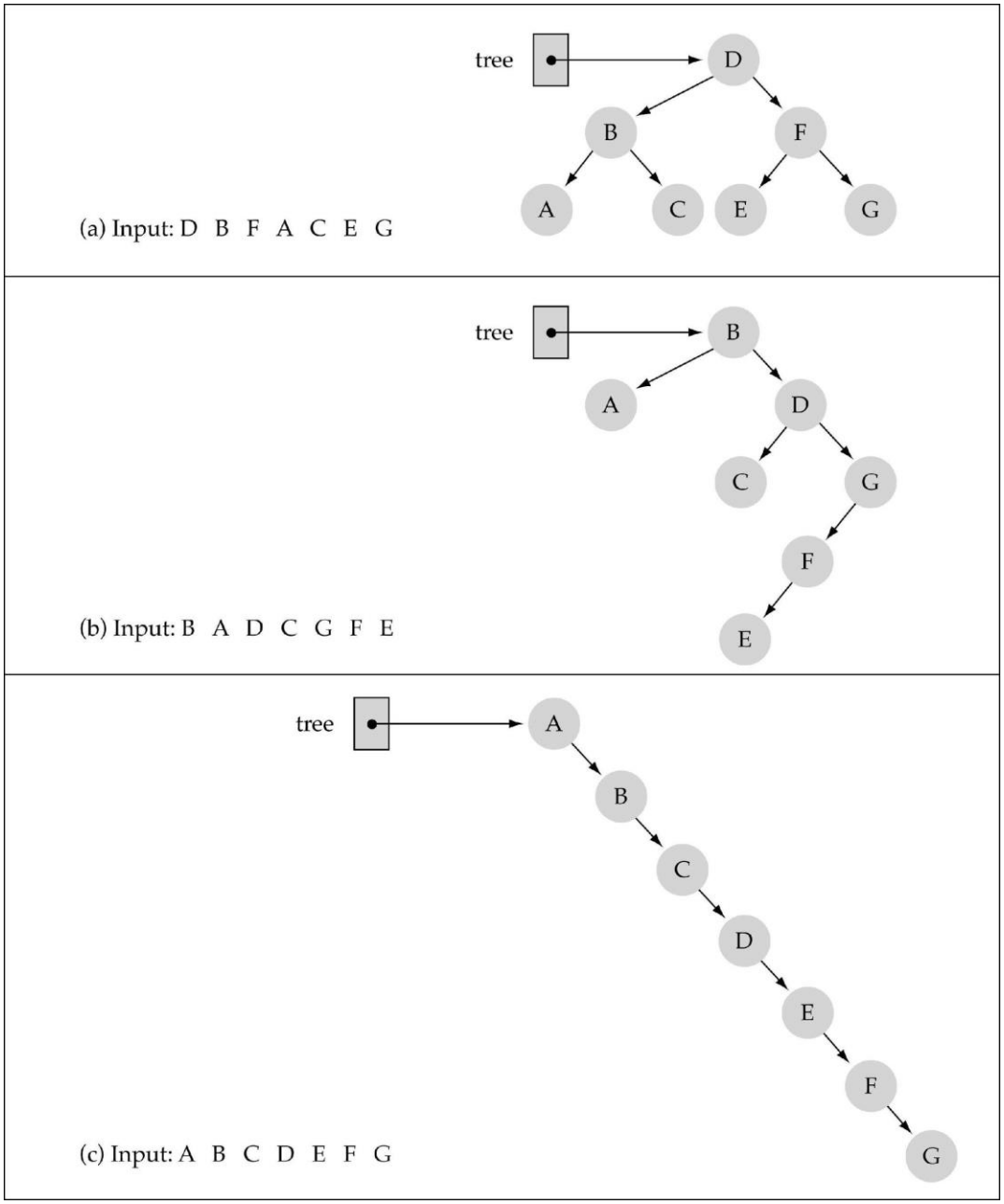
Function InsertItem (cont.)



Does the order of inserting elements into a tree matter?

-

Does the order of inserting elements into a tree matter? (cont.)



Does the order of inserting elements into a tree matter? (cont'd)

-
- red-black trees

Function DeleteItem

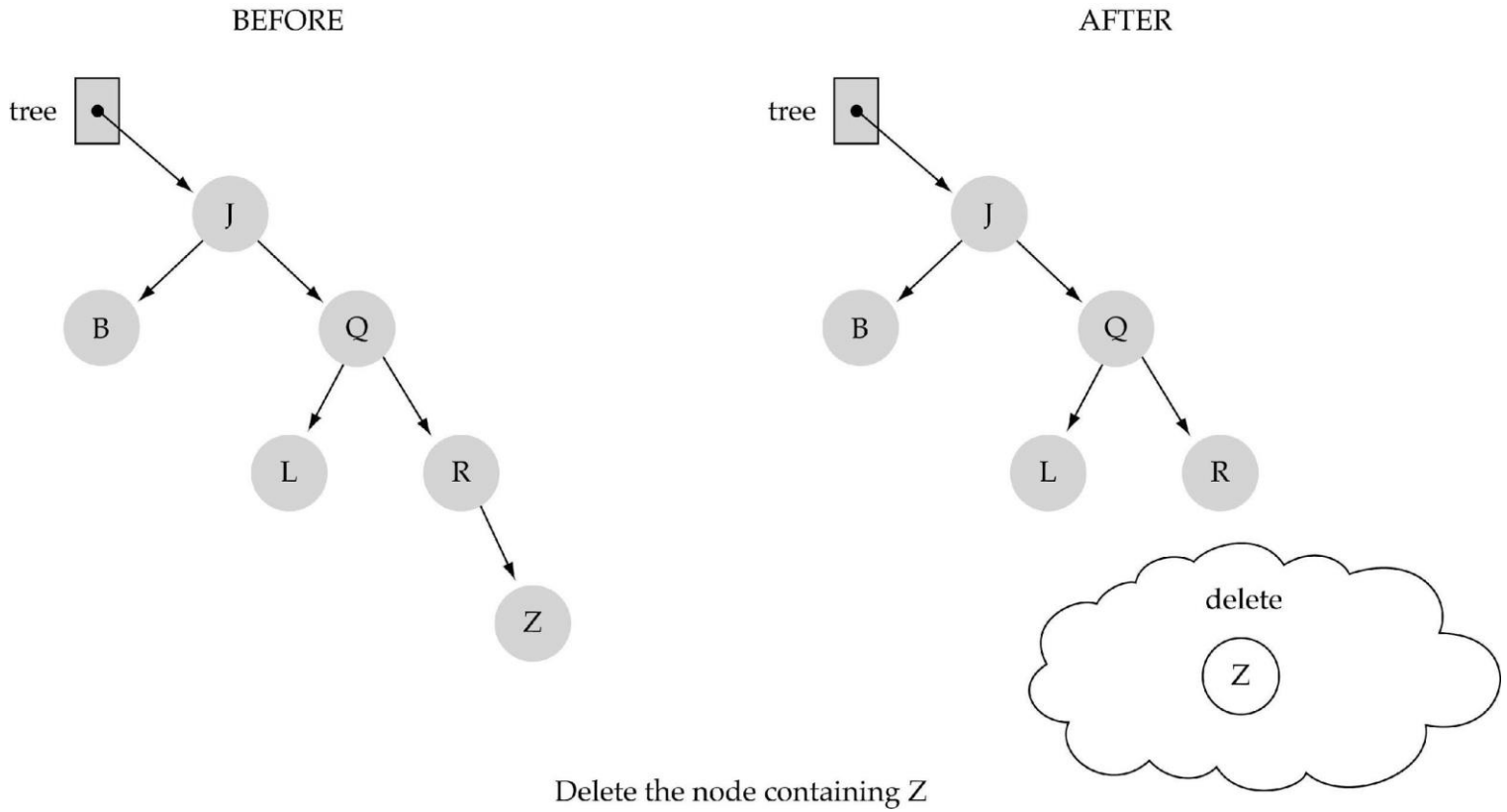
-
-
-

(1)

(2)

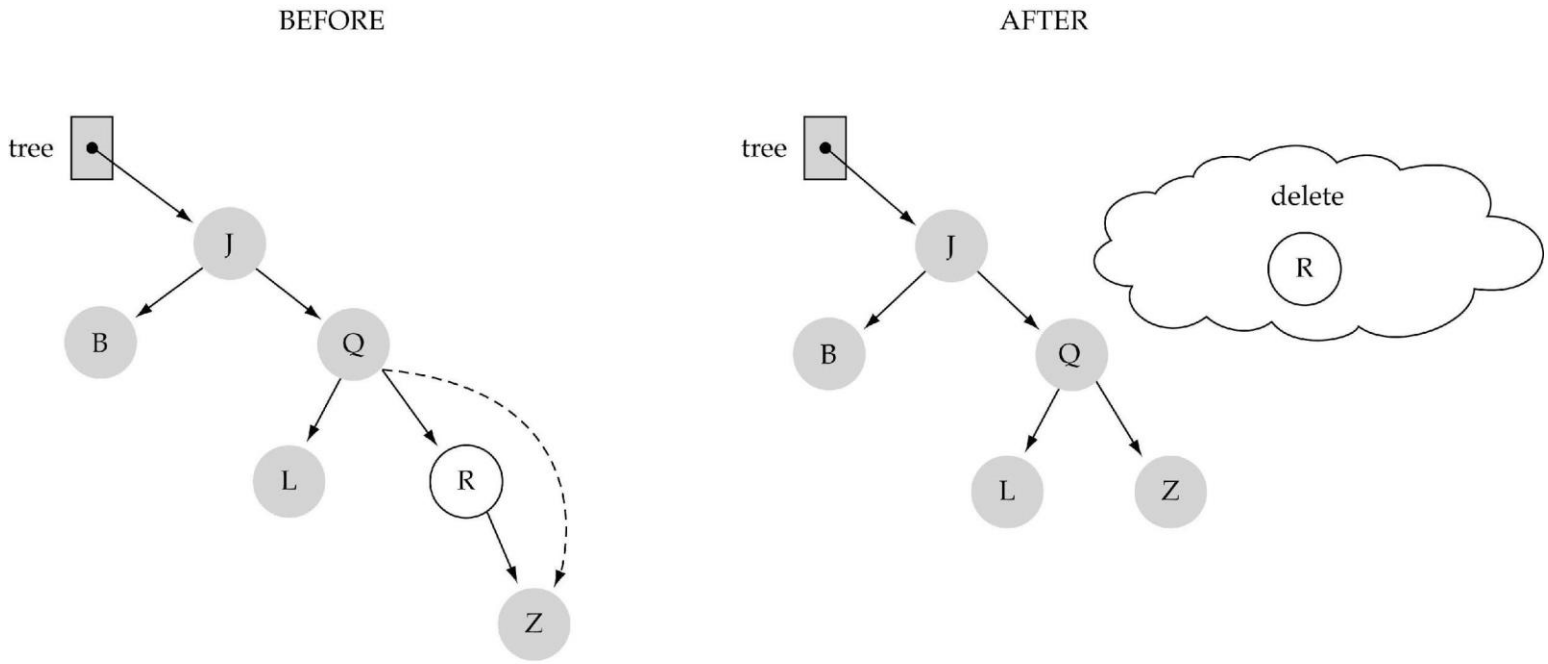
(3)

(1) Deleting a leaf



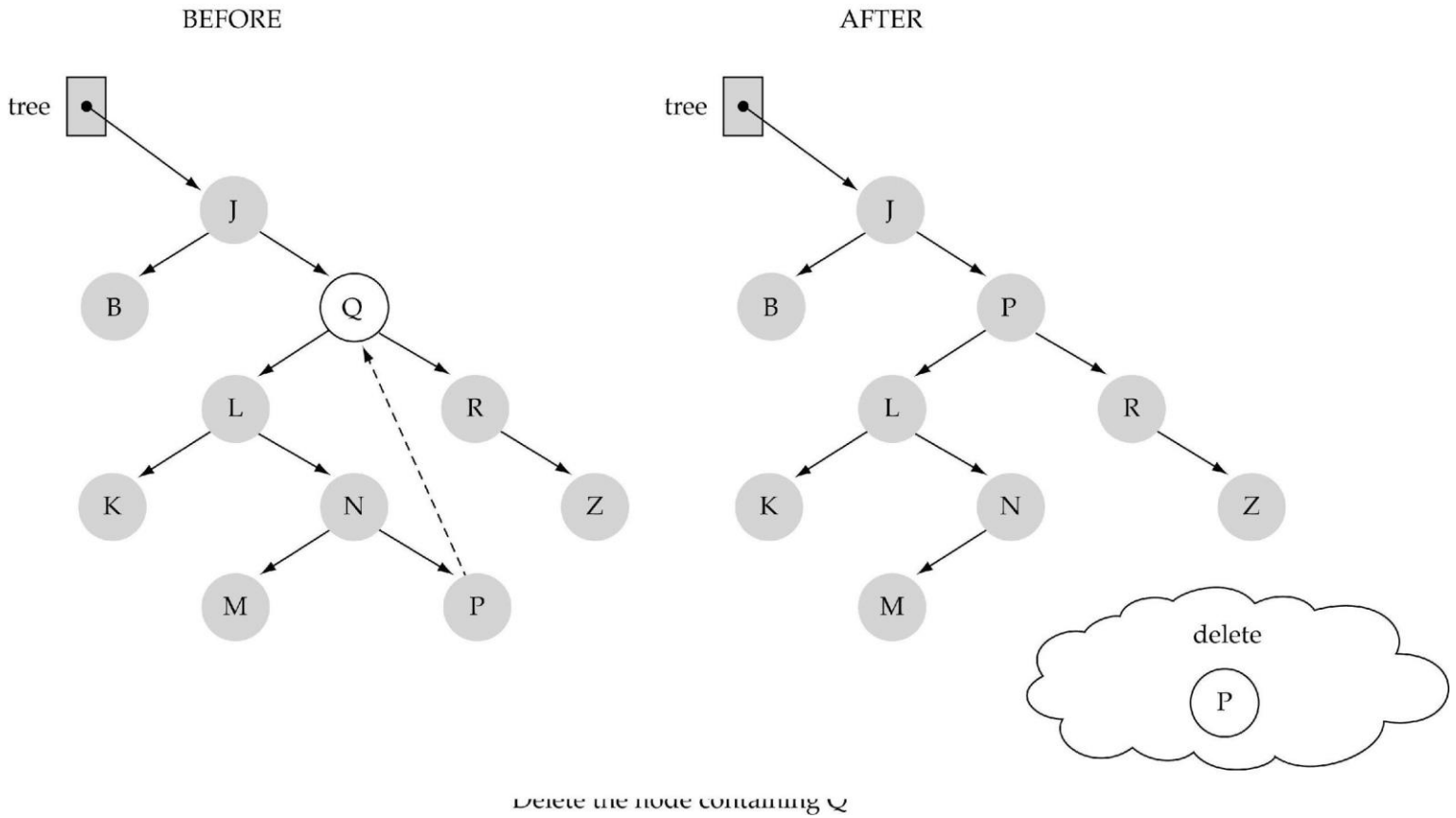
Delete the node containing Z

(2) Deleting a node with only one child



Delete the node containing R

(3) Deleting a node with two children



Delete the node containing Q

(3) Deleting a node with two children (cont.)

- predecessor
-
-

Function DeleteItem (cont.)

- size
- base case(s)
- general case

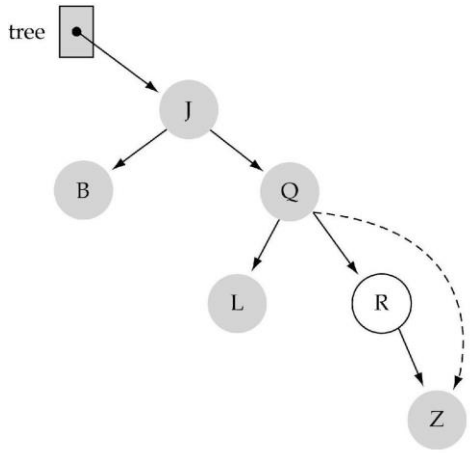
Function DeleteItem (cont.)

DeleteItem

Delete

Function DeleteItem (cont.)

DeleteNode

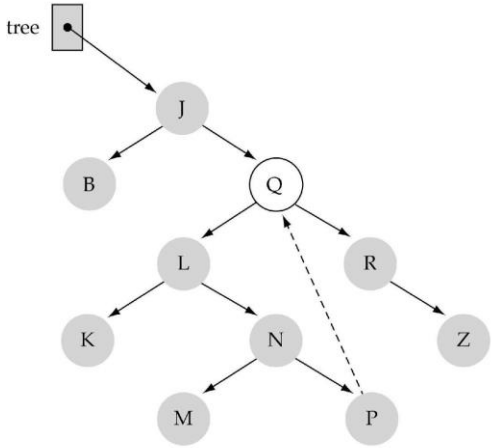


// right child

0 children or
1 child

// left child

0 children or
1 child



2 children

Function DeleteItem (cont.)

GetPredecessor

Function DeleteItem (cont.)

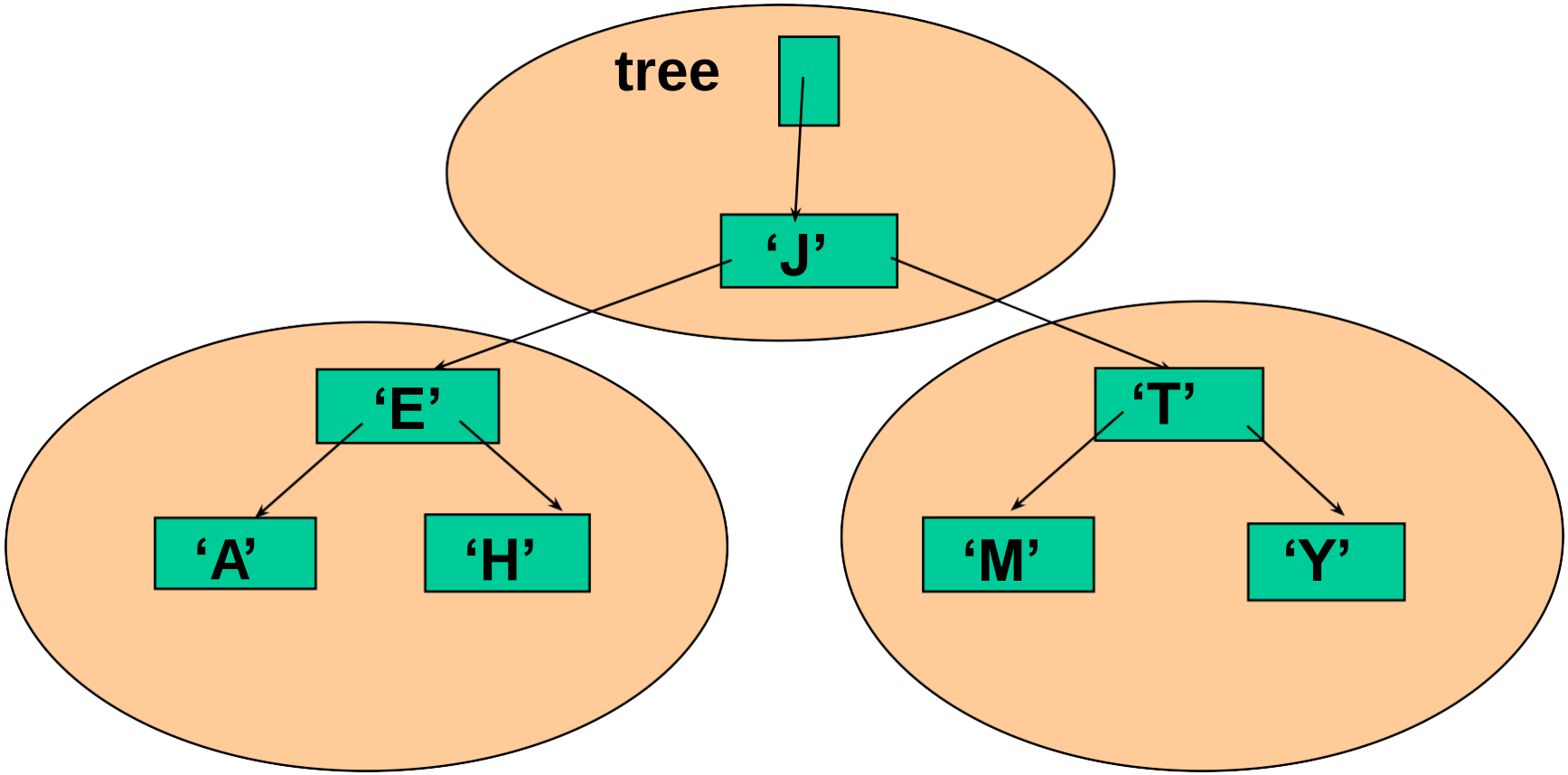
DeleteItem

Delete

Function DeleteItem (cont.)

Tree Traversals

Inorder Traversal:



Visit left subtree first

Visit right subtree last

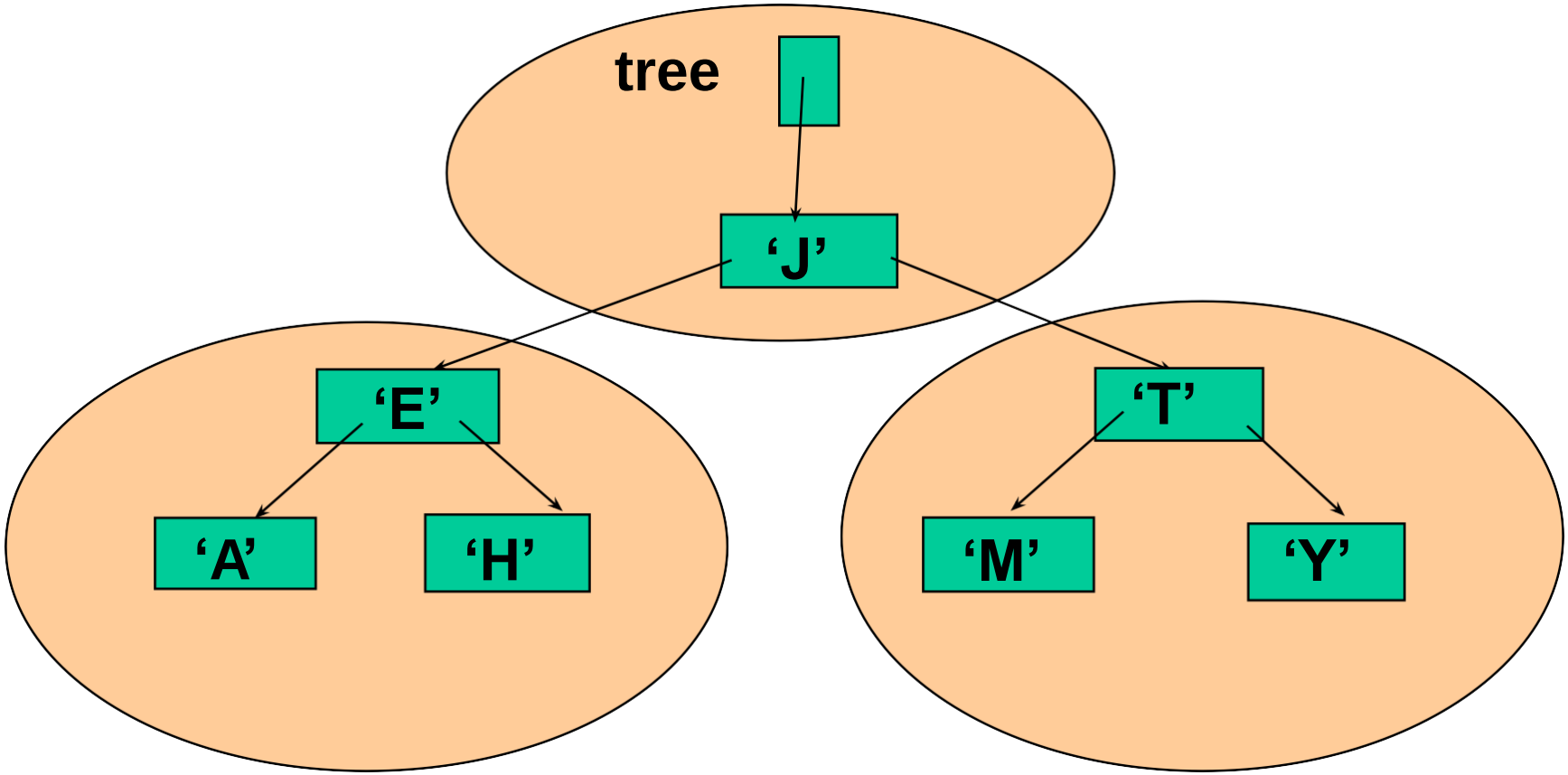
Inorder Traversal

-

Inorder

Warning

Preorder Traversal:



Visit left subtree second

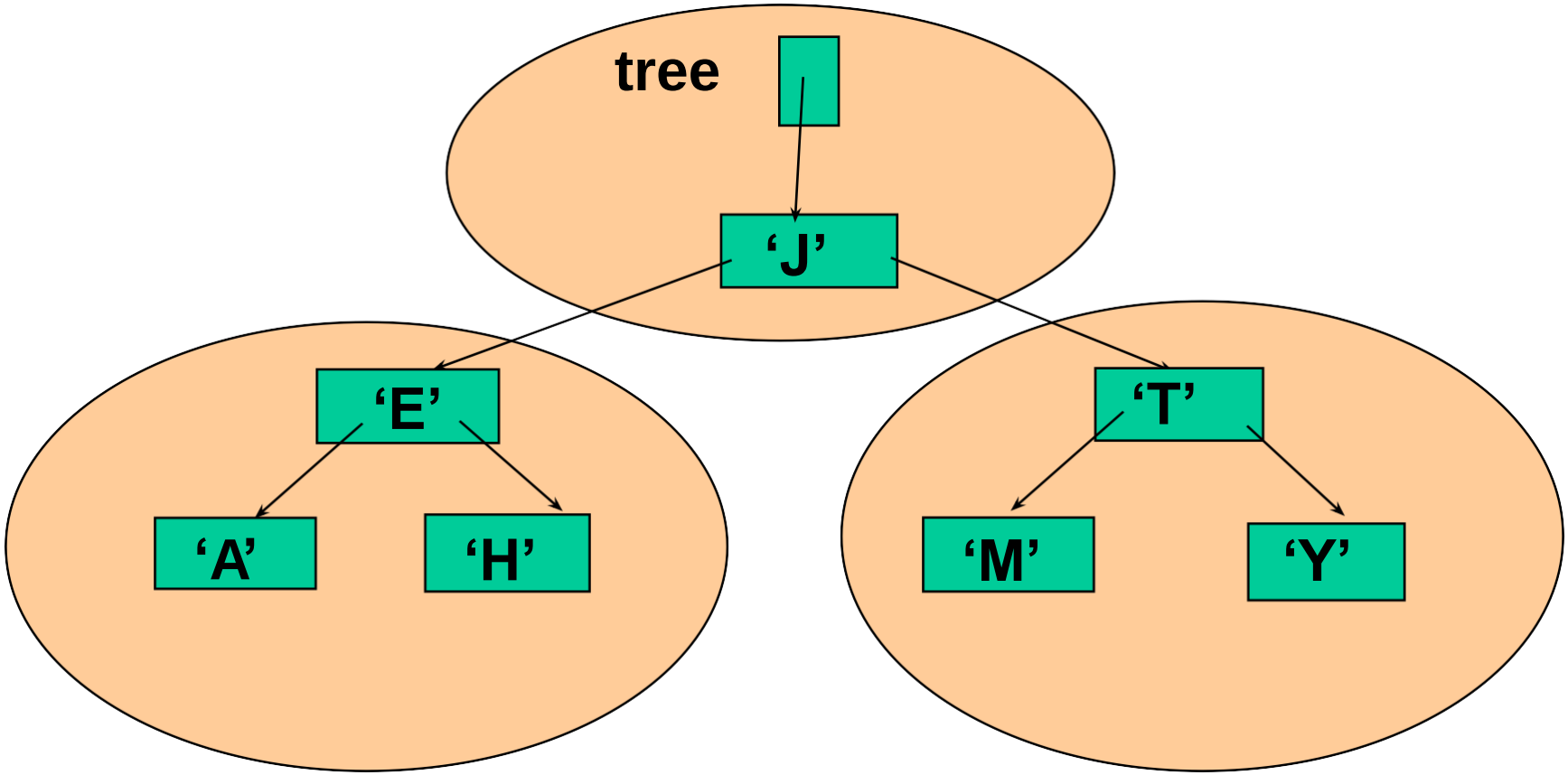
Visit right subtree last

Preorder Traversal

-

Preorder

Postorder



Visit left subtree first

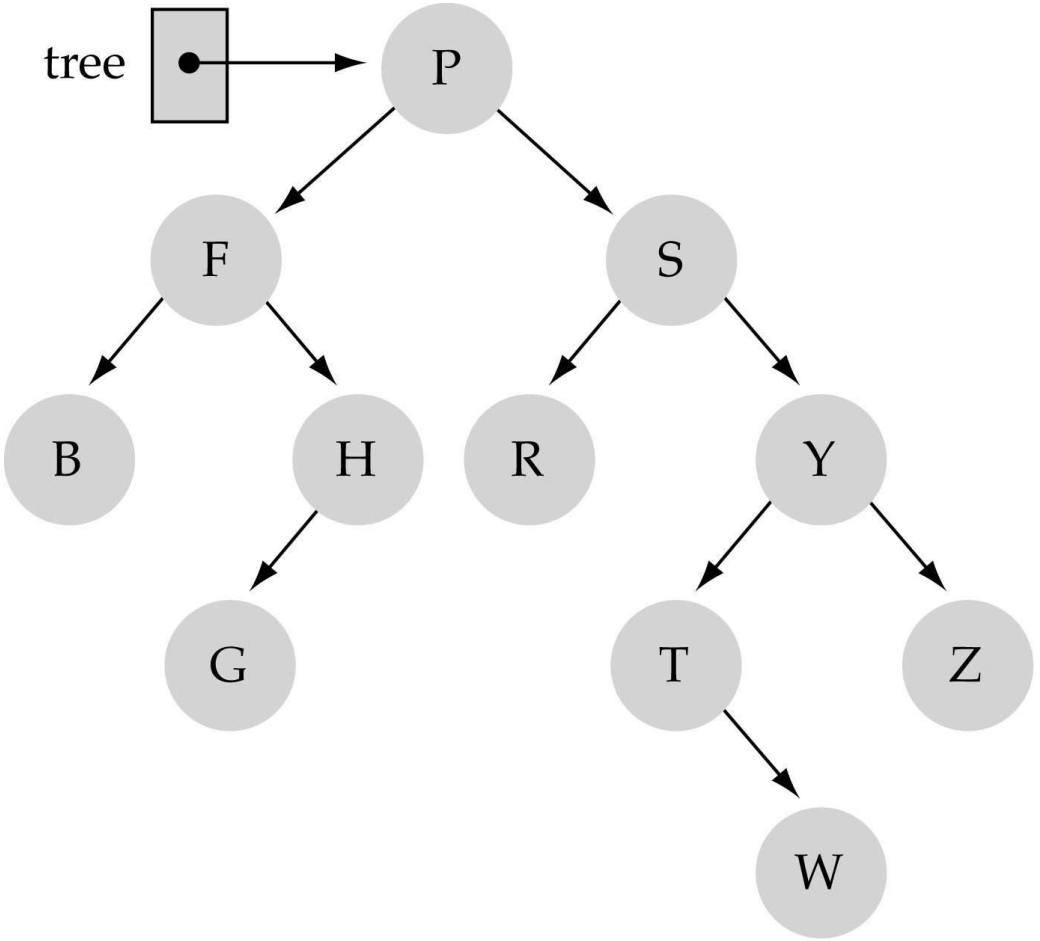
Visit right subtree second

Postorder Traversal

-

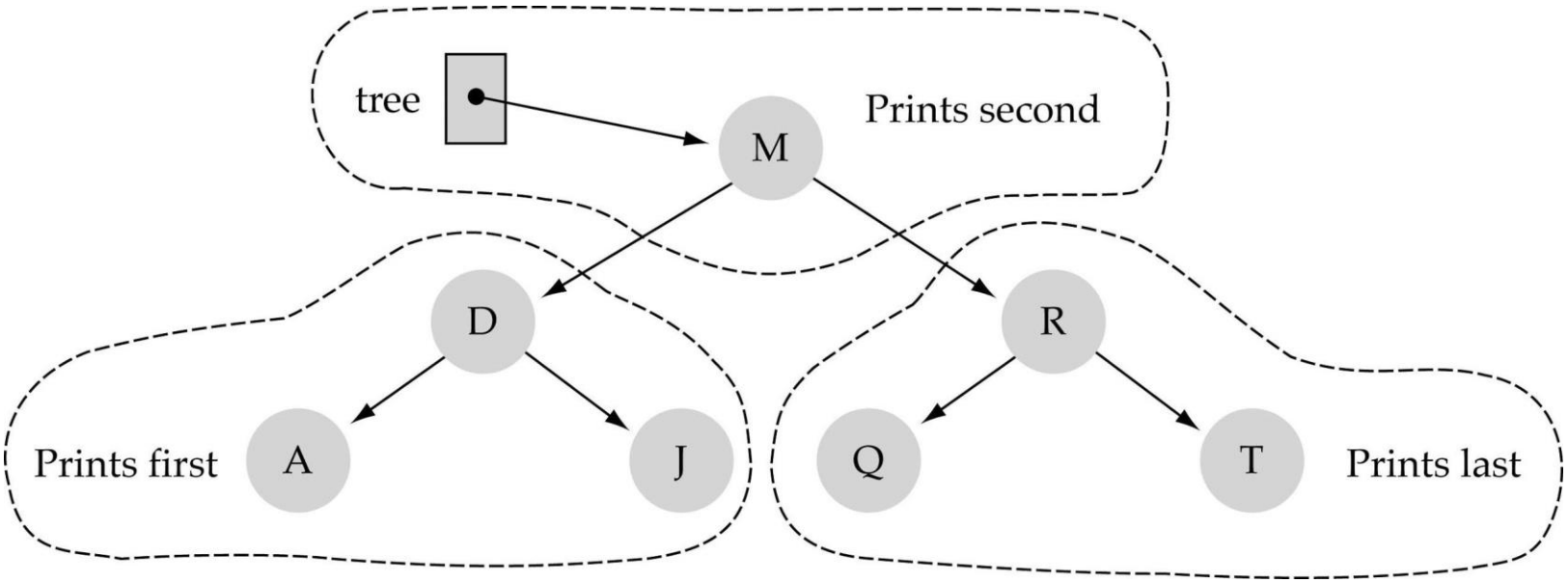
Postorder

Tree Traversals: another example



Inorder:	B	F	G	H	P	R	S	T	W	Y	Z
Preorder:	P	F	B	H	G	S	R	Y	T	W	Z
Postorder:	B	G	H	F	R	W	T	Z	Y	S	P

Function PrintTree



Function PrintTree (cont.)

PrintTree

Print

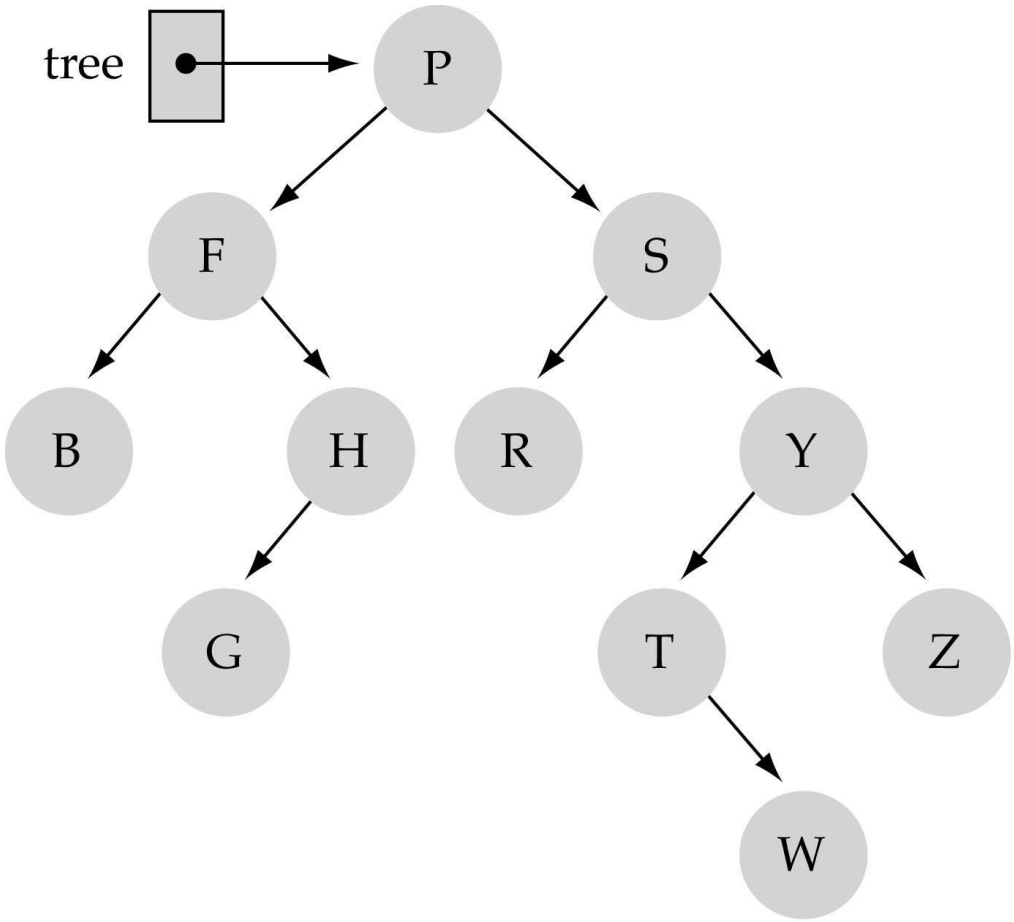
// “visit”

overload

Class Constructor

TreeType

Class Destructor



Use postorder!

Class Destructor (cont'd)

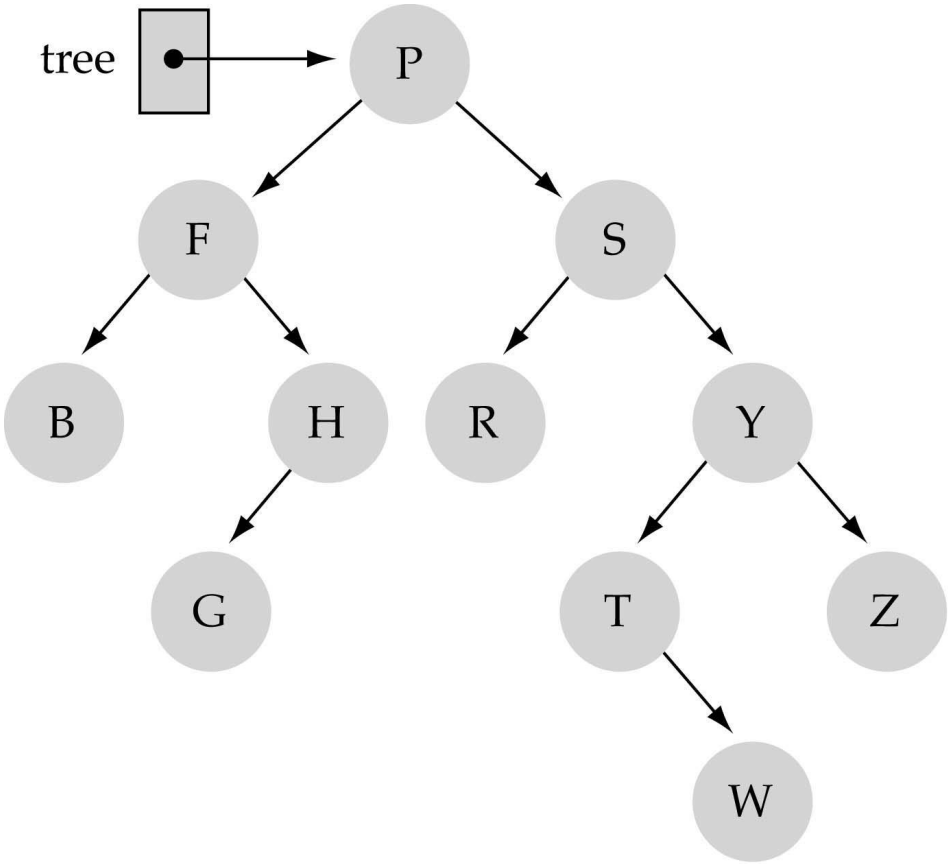
~TreeType

Destroy

postorder

// "visit"

Copy Constructor



Use preorder!

Copy Constructor (cont'd)

TreeType

CopyTree

preorder

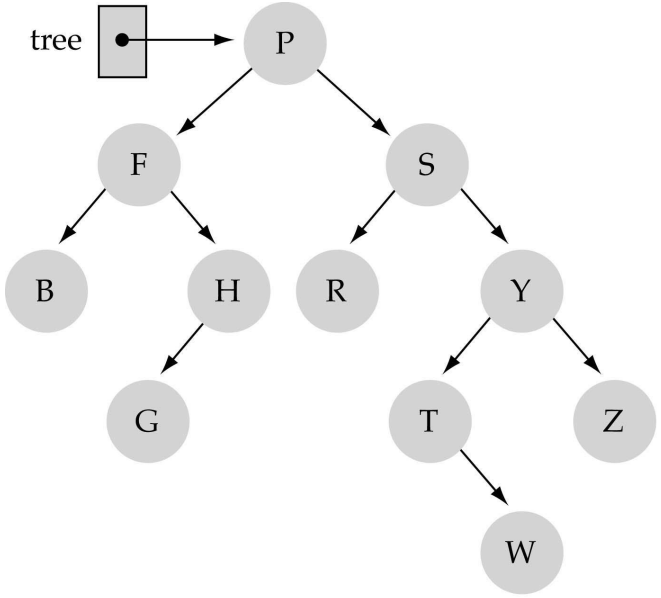
// "visit"

ResetTree and GetNextItem

- Then, *GetNextItem*, dequeues node values from the queue.

```
void ResetTree(OrderType);

void GetNextItem(ItemType&
```



Inorder: B F G H P R S T W Y Z
Preorder: P F B H G S R Y T W Z
Postorder: B G H F R W T Z Y S P

Revise Tree Class Specification

new member functions

// previous member functions
PreOrder
InOrder
PostOrder

preQue;
inQue;
postQue;

new private data

ResetTree and GetNextItem (cont.)

PreOrder

// “visit”

ResetTree and GetNextItem (cont.)

InOrder

// “visit”

ResetTree and GetNextItem (cont.)

PostOrder

// “visit”

ResetTree

ResetTree

GetNextItem

GetNextItem

Iterative Insertion and Deletion

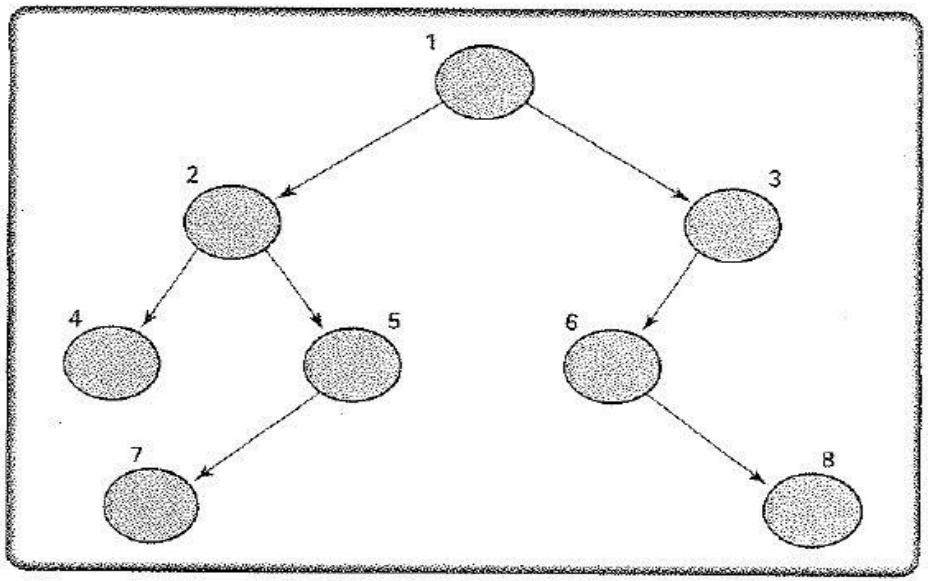
-

Comparing Binary Search Trees to Linear Lists

	O(N)		O(N)
	O(logN)*	O(logN)	O(N)
	O(logN)*	O(N)	O(N)
	O(logN)*	O(N)	O(N)

Exercises 37-41 (p. 539)

Examine the following binary search tree and answer the questions in Exercises 37–40. The numbers on the nodes are *labels* so that we can talk about the nodes; they are not key values within the nodes.



37. If an item is to be inserted whose key value is less than the key value in node 1 but greater than the key value in node 5, where would it be inserted?
38. If node 1 is to be deleted, the value in which node could be used to replace it?
39. 4 2 7 5 1 6 8 3 is a traversal of the tree in which order?
40. 1 2 4 5 7 3 6 8 is a traversal of the tree in which order?

Exercise 17 (p. 537)

17. True or false?

    a. Invoking the delete function in this chapter might create a tree with more levels than the original tree had.

    b. A preorder traversal processes the nodes in a tree in the exact reverse order that a postorder traversal processes them.

    c. An inorder traversal always processes the elements of a tree in the same order, regardless of the order in which the elements were inserted.

    d. A preorder traversal always processes the elements of a tree in the same order, regardless of the order in which the elements were inserted.

Exercise 18 (p. 537)

18. If you wanted to traverse a tree, writing all the elements to a file, and later (the next time you ran the program) rebuild the tree by reading and inserting, would an inorder traversal be appropriate? Why or why not?