

Cpts 122

ADT


BST

Cpts 122 Jack R. Hagemeister 1 WASHINGTON STATE UNIVERSITY

- ## Objectives
1. Describe and explain the form and operations of an ADT binary search tree.
 2. Implement a BST ADT using dynamically linked nodes in C
 3. Describe an algorithm for deleting nodes from a BST
 4. Describe and implement recursive tree traversals.
 5. Create a print tree function using a queue.
- Cpts 122 Jack R. Hagemeister 2 WASHINGTON STATE UNIVERSITY

Review: The Basics of Trees [1/2]

By a **tree**, mathematicians mean a structure like this:



Each dot is called a **vertex** (note: Latin plural is “**vertices**”) or a **node**. Each line is called an **edge**.

Each edge joins two distinct vertices.

A tree is connected (all one piece) and there are no cycles.

Reused from Glenn G. Chappell, UAFairbanks
Cpts 122 Jack R. Hagemeister 3 WASHINGTON STATE UNIVERSITY

Review: The Basics of Trees [2/2]

One use of trees is to represent hierarchical structures.

We place one vertex at the top: the **root**.
Each other vertex of the tree hangs from some vertex. The result is a **rooted tree**.

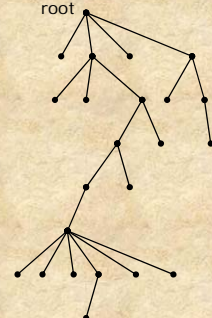
From now on in this class, “tree” always means “rooted tree”.

Some of the terminology for rooted trees comes from the plant world; other terminology comes from family trees.

“Root” is an obvious plant example.
Another: A vertex with nothing hanging off of it is a **leaf**.

Also know:

- Parent, child, sibling.**
- Ancestor, descendant.**
- Level, height.**
- Subtree.**



Reused from Glenn G. Chappell, UAFairbanks
Cpts 122 Jack R. Hagemeister 4 WASHINGTON STATE UNIVERSITY

Binary Trees: Overview

ADT Binary Tree.

What a Binary Tree is.

Special kinds of Binary Trees.

The ADT Binary Tree interface.

Traversals of Binary Trees.

we can implement **any** tree as a binary tree.

So, in some sense, this is the only tree ADT we need.

After Binary Tree, ADT Binary Search Tree.



Reused from Glenn G. Chappell, UAFairbanks
CptS 122 Jack R. Hagemeister

5



Review: Binary Trees —What a Binary Tree Is [1/2]

A **Binary Tree** consists of a set T of nodes so that either:

T is empty (no nodes), or

T consists of a node r , the root, and two subtrees of r , each of which is a binary tree:

the *left subtree*, and

the *right subtree*.

An **empty** Binary Tree is a Binary Tree with no nodes.



Reused from Glenn G. Chappell, UAFairbanks
CptS 122 Jack R. Hagemeister

6



Review: Binary Trees —What a Binary Tree Is [1/2]

Here is a drawing of a larger Binary Tree.

Note: A Binary Tree is **not** a special kind of general tree.

Left and right children make a difference in a Binary Tree. In a general tree all children are just children.



Reused from Glenn G. Chappell, UAFairbanks
CptS 122 Jack R. Hagemeister

7



Review: Binary Trees — Special Kinds

A typical **full** Binary Tree:



A typical **complete** Binary Tree:



A typical **balanced** Binary Tree:



Note that every full Binary Tree is complete and balanced, and every complete Binary Tree is balanced.

Full → Complete → Balanced



Reused from Glenn G. Chappell, UAFairbanks
CptS 122 Jack R. Hagemeister

8



Some More Terminologies

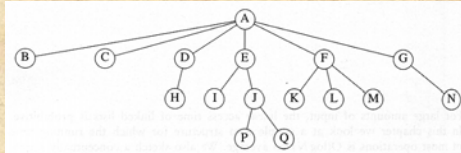


Figure 4.2 A tree

Child and Parent

- Every node except the root has one parent
- A node can have an zero or more children

Leaves

- Leaves are nodes with no children

Sibling

- nodes with same parent



More Terminologies

Path

A sequence of edges

Length of a path

number of edges on the path

Depth of a node

length of the unique path from the root to that node

Height of a node

length of the longest path from that node to a leaf
all leaves are at height 0

The height of a tree = the height of the root

= the depth of the deepest leaf

Ancestor and descendant

If there is a path from $n1$ to $n2$

$n1$ is an ancestor of $n2$, $n2$ is a descendant of $n1$

Proper ancestor and *proper descendant*



Example: UNIX Directory

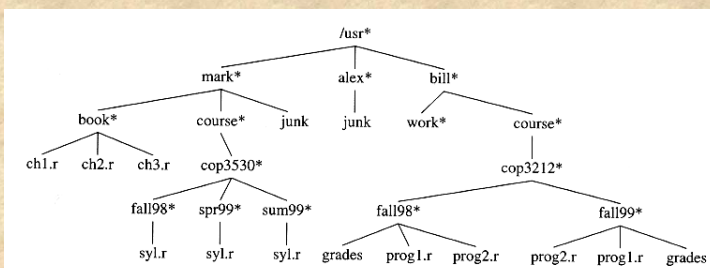


Figure 4.5 UNIX directory



Example: Expression Trees

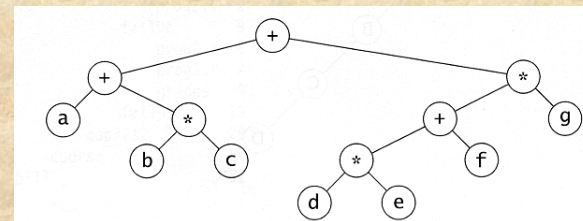


Figure 4.14 Expression tree for $(a + b * c) + ((d * e + f) * g)$

Leaves are operands (constants or variables)

The internal nodes contain operators

Will not be a binary tree if some operators are not binary



Review: Binary Trees — ADT

Data

A set of nodes.

Operations

Create (empty).

Create, given a root and two subtrees.

Destroy.

isEmpty.

getRootData & **setRootData**. Access to data in root node.

attachLeft & **attachRight**. Attach a child to the root.

attachLeftSubtree & **attachRightSubtree**. Attach a subtree to the root.

detachLeftSubtree & **detachRightSubtree**. Detach a subtree from the root.

leftSubtree & **rightSubtree**. Returns a subtree.

preorderTraverse, **inorderTraverse**, & **postorderTraverse**.

Visit all nodes in the appropriate order.



Binary Search Tree

BST Property

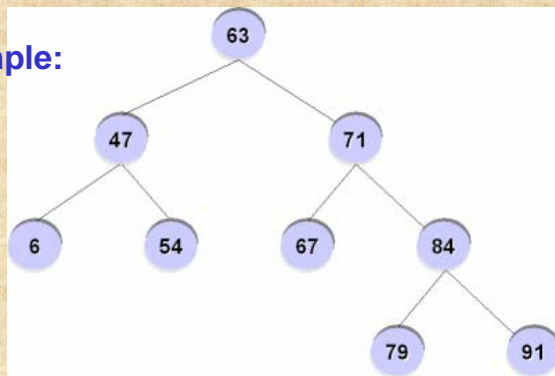
All elements stored in the left subtree of a node whose value is K have values less than K . All elements stored in the right subtree of a node whose value is K have values greater than or equal to K .

That is, a node's left child must have a key less than its parent, and a node's right child must have a key greater than its parent



Binary Search Tree

Example:



Binary Search Tree

Operations on BST ADT

Create a BST (create a node)

Insert an element (Node)

Remove an element (Node)

Find an element (Node)

Clear (remove all elements)

Display all elements in a sorted order

Traversals.

Print the Tree in a "tree view" using a queue.



Binary Search Tree

Inorder traversal

Visit the left subtree, then the node, then the right subtree.

Algorithm:

If there is a left child visit it
Visit the node itself
If there is a right child visit it

What does it mean to "visit" a node????



Binary Search Tree

Postorder traversal

Visit each node after visiting its children.

Algorithm:

If there is a left child visit it
If there is a right child visit it
Visit the node itself



Binary Search Tree

Preorder traversal

Visit each node then visit its children.

Algorithm:

Visit the node itself
If there is a left child visit it
If there is a right child visit it



Binary Search Tree

Insert Algorithm

If value we want to insert < key of current node, we have to go to the left subtree

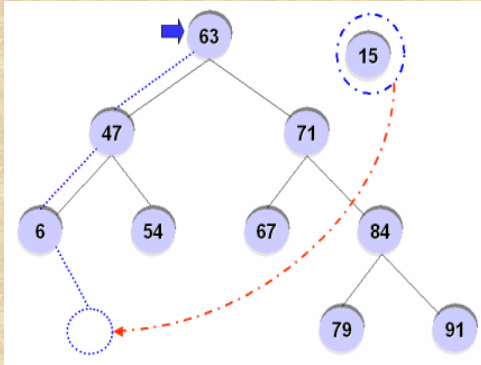
Otherwise we have to go to the right subtree

If the current node is empty (not existing) create a node with the value we are inserting and place it here.



Binary Search Tree

For example, inserting '15' into the BST?



Binary Search Tree

Delete Algorithm

How do we delete a node from BST?

Similar to the insert function, after deletion of a node, the property of the BST must be maintained.



Binary Search Tree

There are 3 possible cases

Node to be deleted has no children

→ We just delete the node.

Node to be deleted has only one child

→ Replace the node with its child
and make the parent of the
deleted node be a parent of the
child of the deleted node

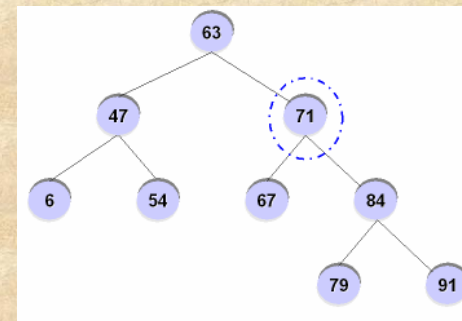
Node to be deleted has two children

→ We just need a trick. Heh heh heh 😊



Binary Search Tree

Node to be deleted has two children



Binary Search Tree

Node to be deleted has two children

Steps:

Find minimum value of right subtree

Replace the value of the node to be deleted by the minimum value

Delete minimum node of right subtree



Binary Search Tree

