BST Operations

15-111

Data Structures

Ananda Gunawardena

Traversal Algorithms

- Inorder traversal
  - Left Root Right
- Preorder traversal
  - Root Left Right
- Postorder traversal
  - Left Right Root
- Level order traversal

Expression Trees

- Draw the expression tree of

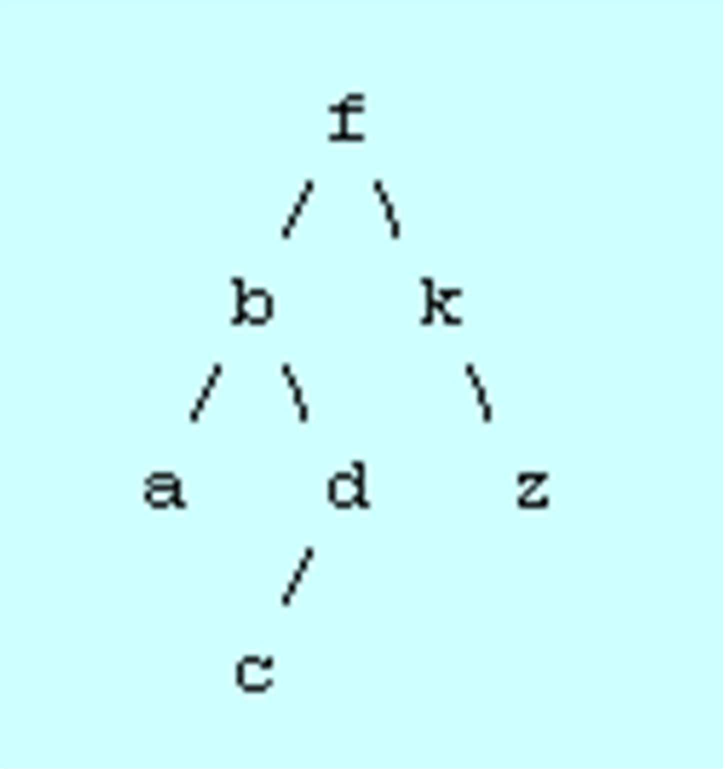
  (1 + 2) * 3 - (4 ^ (5 - 6)) (1 + 2) * 3 - (4 ^ (5 - 6))

- Perform preorder traversal

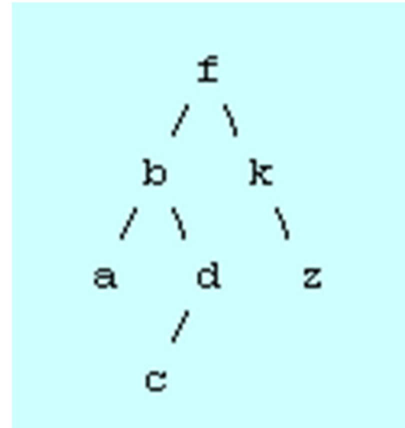
- Perform postorder traversal

Level order or Breadth-first traversal

- Visit nodes by levels
- Root is at level zero
- At each level visit nodes from left to right
- Called "Breadth-First-Traversal(BFS)"



Level order or Breadth-first traversal



BFS Algorithm

enqueue the root
while (the queue is not empty)
{
  dequeue the front element
  print it
  enqueue its left child (if present)
  enqueue its right child (if present)
}

Tree Operations

- Insert Operation (recursive)

Insert(Node, T) = Node if T is empty
= insert(Node, T.left) if Node < T
= insert(Node, T.right) if Node > T

- Homework: Write an iterative version of insert

Insert code

```
public void insert(Comparable key, Object item) {
    int result = key.compareTo(this.key);
    if (result < 0) { // to the left
        if (left == null)
            left = new BinaryNode(key, item);
        else left.insert(key,item);
   } else { // to the right
        if (right == null)
            right = new BinaryNode(key, item);
        else right.insert(key,item);
        }
}
```


Note: Assume left and right references are public

Tree Operations

- Search Operation

  Search(Node, T) = false if T is empty

  = true if T = Node

  = search(Node, T.left) if Node < T

  = search(Node, T.right) if Node > T

- Homework: Write an iterative version of search

Insertions

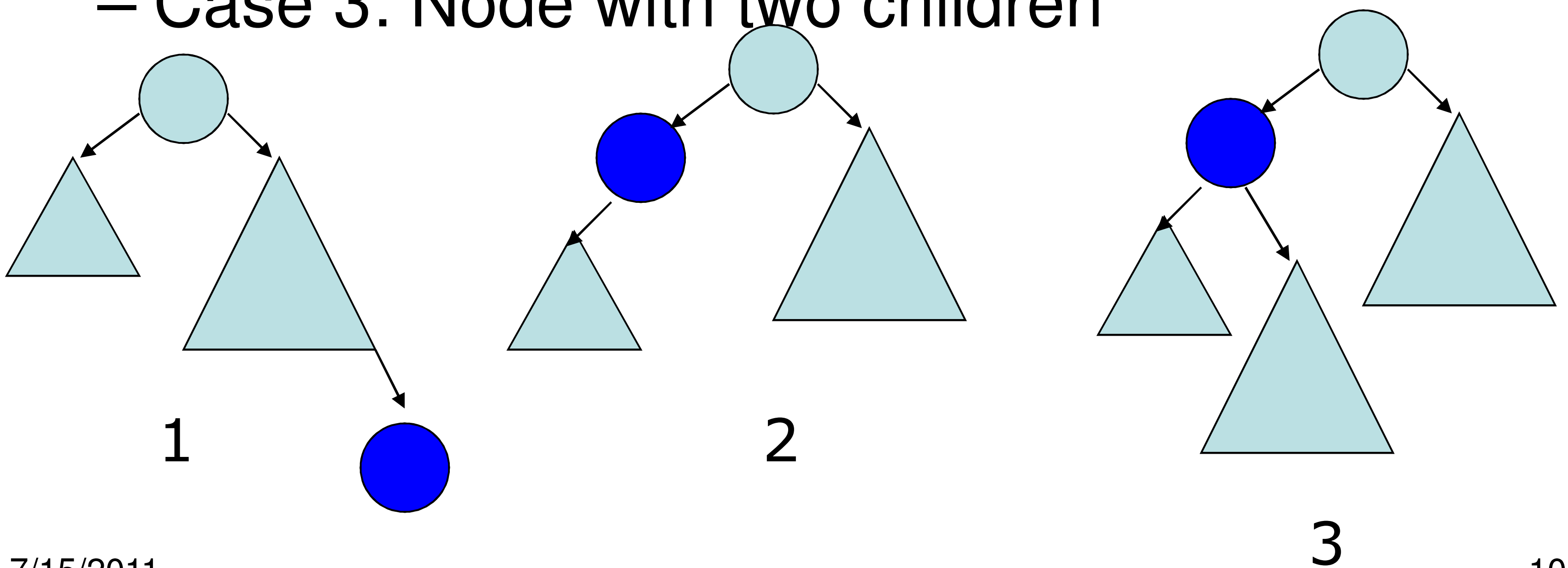
• Insertions in a BST are very similar to searching: find the right spot, and then put down the new element as a new leaf.

•We will not allow multiple insertions of the same element, so there is always exactly one place for the new entry.

•How do we handle duplicate elements in a tree? What is the complexity of an algorithm to determine if there are duplicate elements in a tree?

Delete Node

- 3 cases
  - Case 1: Leaf node
  - Case 2: Node with one child
  - Case 3: Node with two children



Delete Node

- Case 1 – Node is a leaf node
  - Just delete the leaf node
  - No changes to any subtree as a result
- Case 2 – Node has one child
  - If child is a left child, make the parent pointer go to left child

Delete Node

- Case 3 – Node has 2 children
  - This is a complicated case
  - Best strategy is to find the
    - Largest node in the left subtree OR
    - Smallest node in the right subtree
  - Swap the data of the node to be deleted with one of the nodes as above
  - Delete the leaf node

Delete code

```
public BinaryNode delete(Comparable key) {
    int result = key.compareTo(this.key);
    if (result != 0) { // not there yet
        if (result < 0 && left != null) left = left.delete(key);
        if (result > 0 && right != null) right = right.delete(key);
        return this;
    }
    if (left == null && right == null) return null;
            // case 1 (not actually needed)
    if (left == null) return right; // case 2
    if (right == null) return left; // case 2
    BinaryNode next; // case 3
    for (next = right; next.left != null; next = next.left);
    this.key = next.key; this.item = next.item;
    right = right.delete(this.key);
    return this;
}
```

Other Operations

- Counting nodes

- Height of a tree

Other Operations

- Max node

- Min Node

Good Tree

- A good tree has the minimum search depth for any node
- But in a "good" BST we have

depth of T = O(log # nodes)

Theorem: If the tree is constructed from *n* inputs given in random order, then we can expect the depth of the tree to be $\log_2 n$.

But if the input is already (nearly, reverse,…) sorted we are in trouble.

Forcing good behavior

- We can show that for any n inputs, there always is a BST containing these elements of logarithmic depth.
- But if we just insert the standard way, we may build a very unbalanced, deep tree.
- Can we somehow force the tree to remain shallow?
    - At low cost?
- Next we will discuss balanced trees