CS203 Programming with Data Structures

Binary Search Tree

Chengyu Sun
California State University, Los Angeles

Three Important Operations of Collection Classes

- Insert
- Remove
- Search

Time Complexities of List Operations

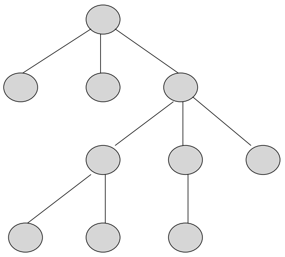
	ArrayList		LinkedList	
	Best-case	Worse-case	Best-case	Worst-case
Insert				
Remove				
Search				

Can we design a data structure with O(logN) *insert*, *remove*, and *search*?

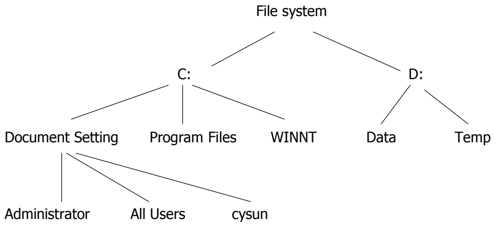
Tree

- A tree is a finite set of nodes
  - The set could be empty
  - When the set is not empty
    - There is a specially designated node called root
    - The remaining nodes are partitioned into zero or more disjoin sets, where each of these sets is also a tree

What a Tree Look Like



Tree Examples …



File system

C:

D:

Document Setting

Program Files

WINNT

Data

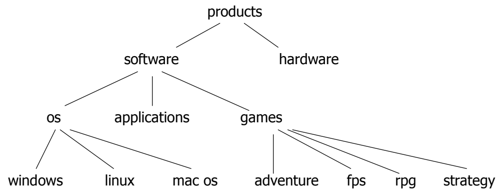
Temp

Administrator

All Users

cysun

… Tree Examples



Terminology

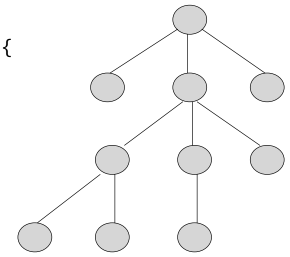
- node, edge
- root, leaf, subtree
- parent, child, sibling
- ancestor, descendant
- The degree of a node is the number of its subtrees
- The degree of a tree is the maximum degree of the nodes in the tree

More Terminology

- A path from node n_1 to n_k is a sequence of nodes $n_1, n_2, \ldots, n_k$ where n_i is the parent of n_{i+1} for $1 \le i < k$
- Length of a path is the number of edges on the path
- Height and depth of a node
- Height of the tree is the length of the longest path from root to a leaf

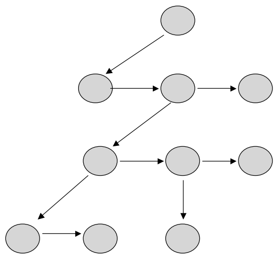
Representing a Tree …

```
class TreeNode {

    Object data;
    ??

}
```



… Representing a Tree

```
class TreeNode {

    Object data;
    ??

}
```

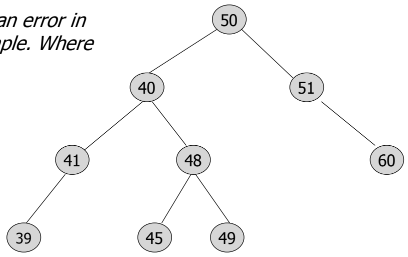


Binary Search Tree (BST)

- A binary tree – a tree with degree 2
- Each node has a value
- The left subtree of a node contains only values *less than* the node's value.
- The right subtree of a node contains only values *greater than* the node's value.

A BST Example

There is an error in the example. Where is it??



BSTreeNode class

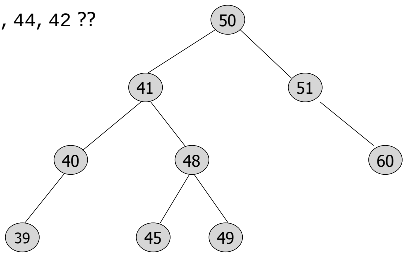
```
public class BSTreeNode {

   ??              value;
   BSTreeNode left, right;

}
```

BSTree Class

- void **insert**(Comparable o)
- Comparable **remove**(Comparable o)
- Comparable **find**(Comparable o)
- Comparable min()
- Comparable max()
- void **print**() // *traversal*
- void clear()
- boolean isEmpty()

Insert

Insert 37, 44, 42 ??

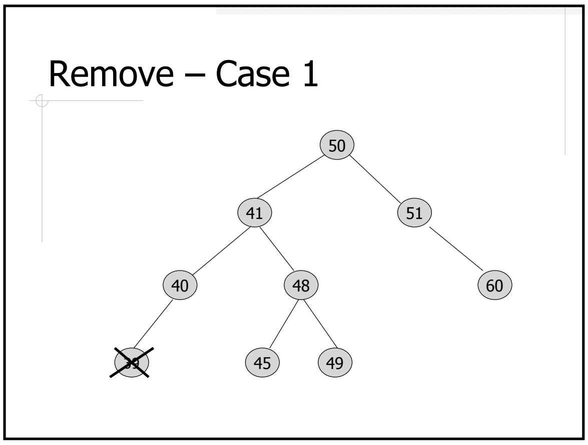


Handling Duplicates

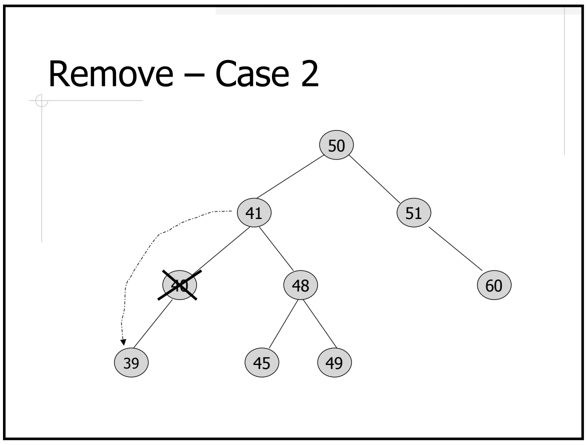
- Ignore them
- Keep frequency at node
- Keep a list of duplicates at node

Remove

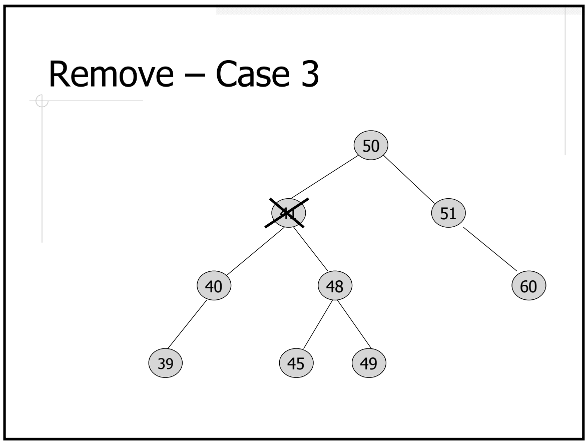
- A node with no child
- A node with one child
- A node with two children



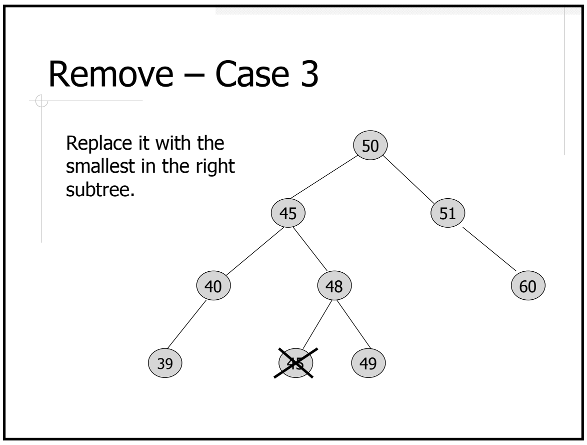
Remove – Case 1



Remove – Case 2

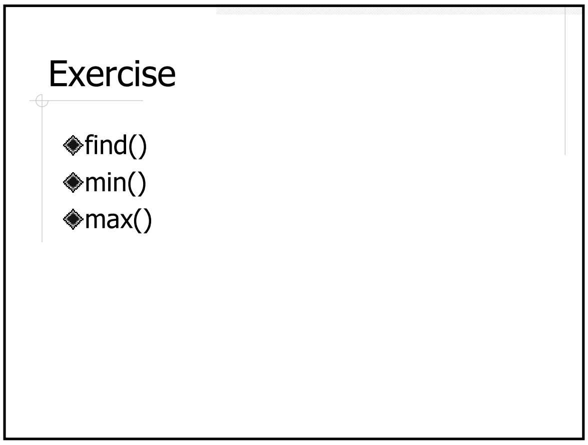


Remove – Case 3



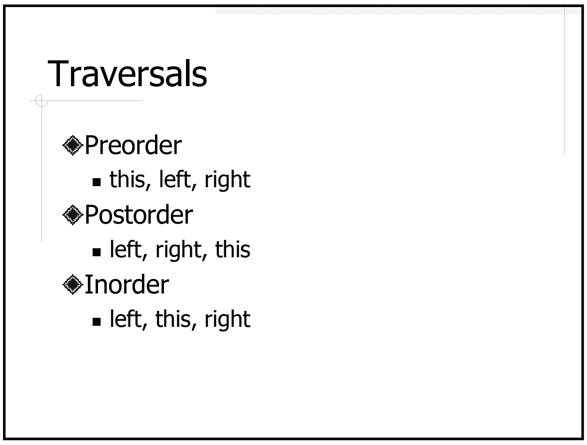
Remove – Case 3

Replace it with the smallest in the right subtree.



Exercise

- find()
- min()
- max()

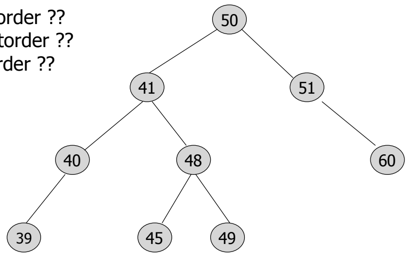


Traversals

- Preorder
  - this, left, right
- Postorder
  - left, right, this
- Inorder
  - left, this, right

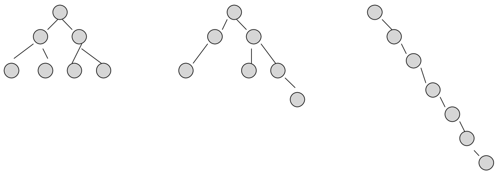
Traversal Examples

Preorder ??
Postorder ??
Inorder ??



Complexities of BST

- ◈Related to the height of the tree
- ◈What's the height of a tree with N nodes??



More About Trees in CS312

- ◈Balanced BST
  - AVL, Red-Black
  - Guarantees O(LogN) for insert, remove, and search
- ◈Trees with higher degrees
  - B-tree
  - Trie
- ◈...