Basic Concepts

CS2123 Data Structures
Binary Search Tree

Dr. Weining Zhang
Department of Computer Science
University of Texas at San Antonio
August 30, 2009

Outline

- Basic Concepts
  - Binary Search Tree
  - Implementation

Tasks of the Week

- Discuss basic tree terminology and representations.
- Case study: Develop a linked implementation of BinarySearchTreeADT.
- Case study: Develop a tree implementation of an ordered list: BinarySearchTreeOrderedList
- Case study: Using the Java TreeSet and TreeMap

Binary Search Tree

Definition

A binary search tree is a binary tree in which the element in each node is larger than all elements the left subtree and less than all elements in the right subtree (assume elements are distinct)

Traversals:
- Inorder: 2, 7, 9, 11, 14, 19
- Preorder: 9, 2, 7, 14, 11, 19
- Postorder: 7, 2, 11, 19, 14, 9
Search a BST

Example

Search for 8 and 11 in following BST. Time complexity is $O(\log_2 n)$ on average for BST of $n$ elements

```
9
/  \
2   14
/ \
7  11
    19
```

Binary Search Tree ADT

```java
public interface BinarySearchTreeADT<T extends Comparable> extends BinaryTreeADT<T>{
    public void addElement(T element);
    public T removeElement(T targetElement);
    public void removeAllOccurrences(T targetElement);
    public T removeMin();
    public T removeMax();
    public T findMin();
    public T findMax();
}
```

Add Element

- If the tree is empty, make the element the root
- Otherwise set current to the root.
- Traverse down the tree and add element according to following rules.
  - If (element < current) and current has no left child: add element as the left child of current.
  - If (element < current) and current has a left child: move the current to the left child of current
  - If (element >= current) and current has no right child: add element as right child of current.
  - If (element >= current) and current has a right child: move current to the right child of current.
**Remove Element**

- Search for the node containing the element, delete the element.
- Find a node to replace current node:
  - If current node has no children, the replacement is null
  - If current node has only one child: the replacement is that child.
  - If current node has both children: the replacement is the **inorder successor**

**Example**

- Remove 5

- Add 12 followed by add 5

- Remove 5

- Remove 11
An Application of BST

- An ordered list can be implemented using a binary search tree
- Elements of the list are stored in the BST
- Methods of the OrderedList class are wrappers of corresponding methods of BinarySearchTree
  - Iterator of OrderedList calls inorderIterator of BinarySearchTree
  - Add() of OrderedList calls addElement of BinarySearchTree
  - ...

Array Implementation of BST

- Store tree nodes in elements of an array
- The parent-child relationships among nodes are determined as follows.
- The root is element 0
- For node at element $i$, the left child is at element $2i + 1$, and the right child is at element $2i + 2$