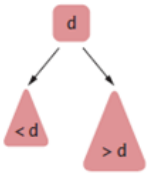
Characteristic of Binary Tree

Binary Search Trees

- All nodes in a binary search tree fulfill this property:
  - The descendants to the left have smaller data values than the node data value,
  - The descendants to the right have larger data values.



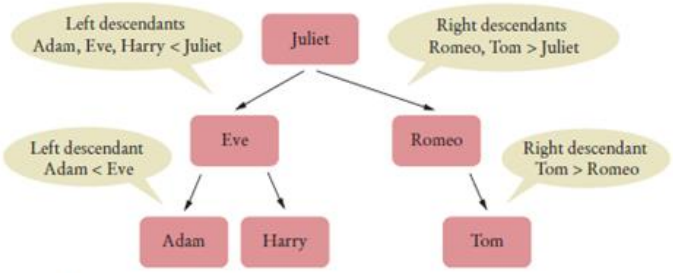


Figure 9 A Binary Search Tree

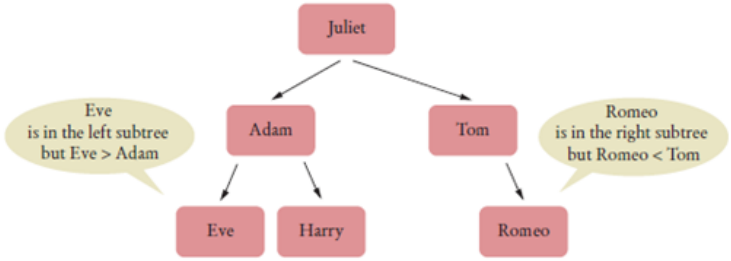


Figure 10 A Binary Tree That Is Not a Binary Search Tree

- Algorithm to insert data:
  - If you encounter a non-null node reference, look at its data value.
  - If the data value of that node is larger than the value you want to insert,
    - Continue the process with the left child.
  - If the node's data value is smaller than the one you want to insert,
    - Continue the process with the right child.
  - If the node's data value is the same as the one you want to insert,
    - You are done. A set does not store duplicate values.
  - If you encounter a null node reference, replace it with the new node.

Here is an example of insert data on standford.edu

```
/**
 Inserts the given data into the binary tree.
 Uses a recursive helper.
*/
public void insert(int data) {
  root = insert(root, data);
}


/**
 Recursive insert -- given a node pointer, recur down and
 insert the given data into the tree. Returns the new
 node pointer (the standard way to communicate
 a changed pointer back to the caller).
*/
private Node insert(Node node, int data) {
  if (node==null) {
    node = new Node(data);
  }
  else {
    if (data <= node.data) {
      node.left = insert(node.left, data);
    }
    else {
      node.right = insert(node.right, data);
    }
  }

  return(node); // in any case, return the new pointer to the caller
}
```

Binary Search Trees - Removal

- First, find the node.
- Case 1: The node has no children
  - Set the link in the parent to `null`

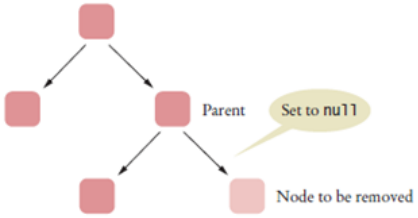


Figure 13 Removing a Node with No Children

- Case 2: The node has 1 child
  - Modify the parent link to the node to point to the child node

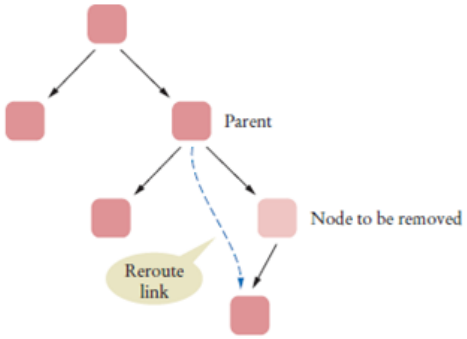


Figure 14 Removing a Node with One Child

- Case 3: The node has 2 children
  - Replace it with the smallest node of the right subtree

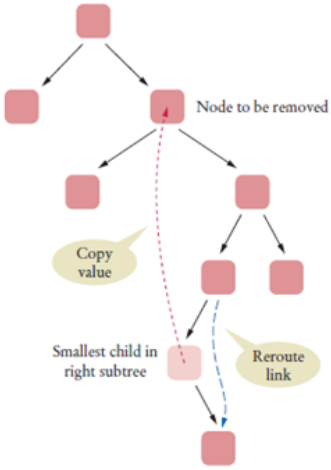


Figure 15
Removing a Node with Two Children

Binary Search Trees - Efficiency of the Operations

- If a binary search tree is balanced, then adding, locating, or removing an element takes $O(\log(n))$ time.

Table 2 Efficiency of Binary Search Tree Operations

Operation	Balanced Binary Search Tree	Unbalanced Binary Search Tree
Find an element.	$O(\log(n))$	$O(n)$
Add an element.	$O(\log(n))$	$O(n)$
Remove an element.	$O(\log(n))$	$O(n)$