CST 405
Algorithm Analysis & Design

Al Lake
Oregon Institute of Technology

Chapter 12

Binary Search Trees

Binary Search Trees

- A binary search tree is a binary tree that is searched.
- Basic operations on a binary search tree take time proportional to the height of the tree.
- For a complete binary tree with n nodes, operations run in $\Theta(\lg n)$ worst-case time.
- If the tree is a linear chain of n nodes, operations run in $\Theta(n)$ worst-case time.

Binary Search Tree Structure

- A binary search tree is organized in a binary tree.
- A binary tree can be represented by a linked data structure in which each node is an object.
- A binary search tree contains the following:
  - Left child node
  - Right child node
  - Parent node

Binary Tree

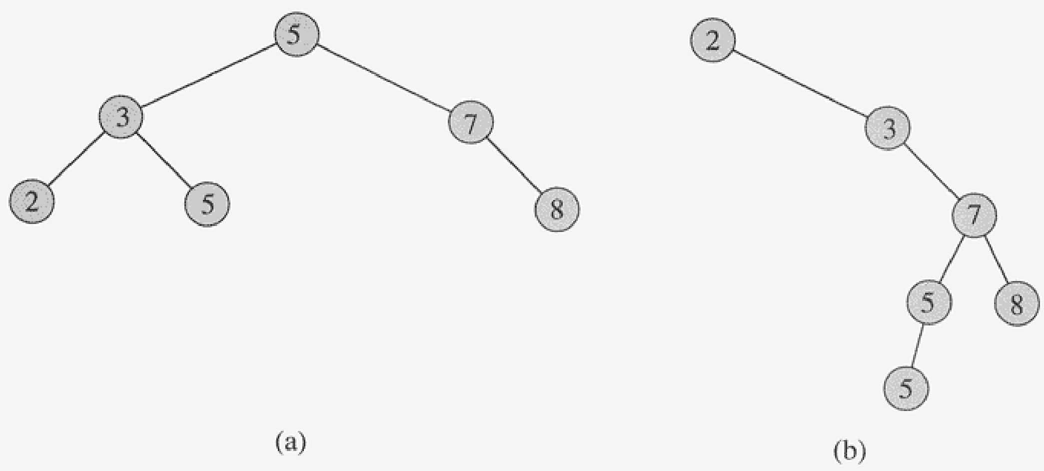

Figure 12.1 Binary search trees. For any node x, the keys in the left subtree of x are at most $key[x]$, and the keys in the right subtree of x are at least $key[x]$. Different binary search trees can represent the same set of values. The worst-case running time for most search-tree operations is proportional to the height of the tree. **(a)** A binary search tree on 6 nodes with height 2. **(b)** A less efficient binary search tree with height 4 that contains the same keys.

Inorder

```
INORDER-TREE-WALK(x)
if x ≠ NIL
    then INORDER-TREE-WALK(left[x])
         print key[x]
         INORDER-TREE-WALK(right[x])
```

Tree Search

- Given a node in a binary search tree, it is sometimes important to be able to find its successor in the sorted order determined by an inorder tree walk.
- If all keys are distinct, the successor of a node x is the node with the smallest key greater than key[x].

Tree Search

```
TREE-SEARCH(x, k)
if x = NIL or k = key[x]
    then return x
if k < key[x]
    then return TREE-SEARCH(left[x], k)
    else return TREE-SEARCH(right[x], k)
```

Tree Minimum

TREE-MINIMUM(x)

while $\mathit{left}[x] \neq$ NIL
  do $x \leftarrow \mathit{left}[x]$
return x

Tree Maximum

```
TREE-MAXIMUM(x)
while right[x] ≠ NIL
        do x ← right[x]
return x
```

Tree Insert

- The operations of insertion and deletion cause the dynamic set represented by a binary search tree to change.
- The data structure must be modified to reflect this change, in such a way that the binary-search-tree property continues to hold.
- The procedure TREE-INSERT run in $O(h)$ time on a tree of height h.

Tree Insert

```
TREE-INSERT(T, z)
y ← NIL
x ← root[T]
while x ≠ NIL
      do y ← x
         if key[z] < key[x]
            then x ← left[x]
            else x ← right[x]
p[z] ← y
if y = NIL
   then root[T] ← z                    ▷ Tree T was empty
   else if key[z] < key[y]
           then left[y] ← z
           else right[y] ← z
```

Deletion

```
TREE-DELETE(T, z)
if left[z] = NIL or right[z] = NIL
    then y ← z
    else y ← TREE-SUCCESSOR(z)
if left[y] ≠ NIL
    then x ← left[y]
    else x ← right[y]
if x ≠ NIL
    then p[x] ← p[y]
if p[y] = NIL
    then root[T] ← x
    else if y = left[p[y]]
            then left[p[y]] ← x
            else right[p[y]] ← x
if y ≠ z
    then key[z] ← key[y]
         copy y's satellite data into z
return y
```