

Binary Search Trees

$key[x]$ is the key of node x .

$p[x]$ is the parent of x .

$left[x]$ is the left child of x .

$right[x]$ is the right child of x .

Binary search tree property:

If y is in left subtree of x , then $key[y] < key[x]$.

If y is in the right subtree, then $key[x] < key[y]$.

Tree walking is $\Theta(n)$.

TREE-WALK(x)

if $x \neq \text{NIL}$

then **TREE-WALK**($left[x]$)

TREE-WALK($right[x]$)

Dynamic set ops. are $O(h)$, where h is the height.

Best-case: $O(\lg n)$. Worst-case: $O(n)$.

For n nodes, h can range from $\lfloor \lg n \rfloor$ to $n - 1$.

A binary tree of height h has $\leq 2^{h+1} - 1$ nodes.
 Note $\lfloor \lg(2^{h+1} - 1) \rfloor = h$ and $\lfloor \lg(2^{h+1}) \rfloor = h + 1$

Let $T(h) =$ maximum number of nodes in a binary tree of height h .

Basis: $T(0) = 1 = 2^1 - 1$

Assume: $T(h - 1) = 2^h - 1$

Show: $T(h) = 2^{h+1} - 1$

Induction: Subtrees have height $\leq h - 1$

$$T(h) = 2T(h-1) + 1 = 2(2^h - 1) + 1 = 2^{h+1} - 1$$

TREE-SEARCH(x, k)

if $x = \text{NIL}$ or $k = \text{key}[x]$

then return x

if $k < \text{key}[x]$

then return **TREE-SEARCH**($\text{left}[x], k$)

else return **TREE-SEARCH**($\text{right}[x], k$)

TREE-MAXIMUM(x, k)

while $\text{right}[x] \neq \text{NIL}$

do $x \leftarrow \text{right}[x]$

return x

TREE-MINIMUM(x, k)
while $left[x] \neq \text{NIL}$ **do** $x \leftarrow left[x]$
return x

TREE-SUCCESSOR(x)
if $right[x] \neq \text{NIL}$
 then return **TREE-MINIMUM**($right[x]$)
 $y \leftarrow p[x]$
while $y \neq \text{NIL}$ and $x = right[y]$
 do $x \leftarrow y$
 $y \leftarrow p[x]$
return y

TREE-INSERT(T, z)
 $y \leftarrow \text{NIL}$
 $x \leftarrow root[T]$
while $x \neq \text{NIL}$
 do $y \leftarrow x$
 if $key[z] < key[x]$
 then $x \leftarrow left[x]$
 else $x \leftarrow right[x]$
 $p[z] \leftarrow y$
if $y = \text{NIL}$
 then $root[T] \leftarrow z$
 else if $key[z] < key[y]$
 then $left[y] \leftarrow z$
 else $right[y] \leftarrow z$

TREE-DELETE

If a node has no children, it is easy to delete it. Otherwise, some adjustments are necessary.

In the following diagrams, node B is being deleted from the tree.

T_1 , T_2 , and T_3 are subtrees or NIL.

In the last diagram, T_4 is a subtree whose minimum element is node C .



