

Binary Search Tree Pseudocode

Here are the procedures for binary search trees discussed in the book.

INORDER-TREE-WALK(x)

```
1  if  $x \neq \text{NIL}$ 
2      INORDER-TREE-WALK( $x.\text{left}$ )
3      print  $x.\text{key}$ 
4      INORDER-TREE-WALK( $x.\text{right}$ )
```

PREORDER-TREE-WALK(x)

```
1  if  $x \neq \text{NIL}$ 
2      print  $x.\text{key}$ 
3      INORDER-TREE-WALK( $x.\text{left}$ )
4      INORDER-TREE-WALK( $x.\text{right}$ )
```

POSTORDER-TREE-WALK(x)

```
1  if  $x \neq \text{NIL}$ 
2      INORDER-TREE-WALK( $x.\text{left}$ )
3      INORDER-TREE-WALK( $x.\text{right}$ )
4      print  $x.\text{key}$ 
```

TREE-SEARCH(x, k)

```
1  if  $x == \text{NIL}$  or  $k == x.\text{key}$ 
2      return  $x$ 
3  if  $k < x.\text{key}$ 
4      return TREE-SEARCH( $x.\text{left}, k$ )
5  else
6      return TREE-SEARCH( $x.\text{right}, k$ )
```

ITERATIVE-TREE-SEARCH(x, k)

```
1  while  $x \neq \text{NIL}$  and  $k \neq x.\text{key}$ 
2      if  $k < x.\text{key}$ 
3           $x = x.\text{left}$ 
4      else
5           $x = x.\text{right}$ 
6  return  $x$ 
```

TREE-MINIMUM(x)

```
1  while  $x.left \neq \text{NIL}$ 
2       $x = x.left$ 
3  return  $x$ 
```

TREE-MAXIMUM(x)

```
1  while  $x.right \neq \text{NIL}$ 
2       $x = x.right$ 
3  return  $x$ 
```

TREE-PREDECESSOR(x)

```
1  if  $x.left \neq \text{NIL}$ 
2      return TREE-MAXIMUM( $x.left$ )
3   $y = x.p$ 
4  while  $y \neq \text{NIL}$  and  $x == y.left$ 
5       $x = y$ 
6       $y = y.p$ 
7  return  $y$ 
```

TREE-SUCCESSOR(x)

```
1  if  $x.right \neq \text{NIL}$ 
2      return TREE-MINIMUM( $x.right$ )
3   $y = x.p$ 
4  while  $y \neq \text{NIL}$  and  $x == y.right$ 
5       $x = y$ 
6       $y = y.p$ 
7  return  $y$ 
```

TREE-INSERT(T, z)

```
1   $y = \text{NIL}$ 
2   $x = T.root$ 
3  while  $x \neq \text{NIL}$ 
4       $y = x$ 
5      if  $z.key < x.key$ 
6           $x = x.left$ 
7      else
8           $x = x.right$ 
9   $z.p = y$ 
10 if  $y == \text{NIL}$ 
11      $T.root = z$  // tree  $T$  was empty
12 elseif  $z.key < y.key$ 
13      $y.left = z$ 
14 else
15      $y.right = z$ 
```

```

TRANSPLANT(T, u, v)
1  if u.p == NIL
2      T.root == v
3  elseif u == u.p.left
4      u.p.left == v
5  else
6      u.p.right == v
7  if v ≠ NIL
8      v.p = u.p

TREE-DELETE(T, z)
1  if z.left == NIL
2      TRANSPLANT(T, z, z.right)
3  elseif z.right == NIL
4      TRANSPLANT(T, z, z.left)
5  else
6      y = TREE-MINIMUM(z.right)
7      if y.p ≠ z
8          TRANSPLANT(T, y, y.right)
9          y.right = z.right
10         y.right.p = y
11     TRANSPLANT(T, z, y)
12     y.left = z.left
13     y.left.p = y

```