

# Exercises

**Cormen Problem 12-1.** Equal keys pose a problem for the implementation of binary search trees.

**a.** What is the asymptotic performance of TREE-INSERT when used to insert  $n$  items with identical keys into an initially empty binary search tree?

# Exercises

**Cormen Problem 12-1.** We propose to improve TREE-INSERT by testing before line 5 to determine whether  $z.key = x.key$  and by testing before line 11 to determine whether  $z.key = y.key$ .

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  $x = x.right$ 
8   $z.p = y$ 
9  if  $y == \text{NIL}$ 
10      $T.root = z$ 
11 elseif  $z.key < y.key$ 
12      $y.left = z$ 
13 else  $y.right = z$ 
```

If equality holds, we implement one of the following strategies. For each strategy, find the asymptotic performance of inserting  $n$  items with identical keys into an initially empty binary search tree. (The strategies are described for line 5, in which we compare the keys of  $z$  and  $x$ . Substitute  $y$  for  $x$  to arrive at the strategies for line 11.)

**b.** Keep a boolean flag  $x.b$  at node  $x$ , and set  $x$  to either  $x.left$  or  $x.right$  based on the value of  $x.b$ , which alternates between FALSE and TRUE each time we visit  $x$  while inserting a node with the same key as  $x$ .

# Exercises

**Cormen Problem 12-1.** We propose to improve TREE-INSERT by testing before line 5 to determine whether  $z.key = x.key$  and by testing before line 11 to determine whether  $z.key = y.key$ .

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  $x = x.right$ 
8   $z.p = y$ 
9  if  $y == \text{NIL}$ 
10      $T.root = z$ 
11 elseif  $z.key < y.key$ 
12      $y.left = z$ 
13 else  $y.right = z$ 
```

If equality holds, we implement one of the following strategies. For each strategy, find the asymptotic performance of inserting  $n$  items with identical keys into an initially empty binary search tree. (The strategies are described for line 5, in which we compare the keys of  $z$  and  $x$ . Substitute  $y$  for  $x$  to arrive at the strategies for line 11.)

**c.** Keep a list of nodes with equal keys at  $x$ , and insert  $z$  into the list.

# Exercises

**Cormen Problem 12-1.** We propose to improve TREE-INSERT by testing before line 5 to determine whether  $z.key = x.key$  and by testing before line 11 to determine whether  $z.key = y.key$ .

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  $x = x.right$ 
8   $z.p = y$ 
9  if  $y == \text{NIL}$ 
10      $T.root = z$ 
11 elseif  $z.key < y.key$ 
12      $y.left = z$ 
13 else  $y.right = z$ 
```

If equality holds, we implement one of the following strategies. For each strategy, find the asymptotic performance of inserting  $n$  items with identical keys into an initially empty binary search tree. (The strategies are described for line 5, in which we compare the keys of  $z$  and  $x$ . Substitute  $y$  for  $x$  to arrive at the strategies for line 11.)

**d.** Randomly set  $x$  to either  $x.left$  or  $x.right$ . (Give the worst-case performance and informally derive the expected running time.)

# Exercises

A **preorder** traversal of a tree is given by the following procedure:

- Visit (print) the root node
- Traverse the left sub-tree in pre-order
- Traverse the right sub-tree in pre-order

A **postorder** traversal of a tree is given by the following procedure:

- Traverse the left subtree by calling the postorder function recursively.
- Traverse the right subtree by calling the postorder function recursively.
- Visit (print) the current node.

**EX.** Given a BST in pre-order as {13,5,3,2,11,7,19,23}, draw this BST and determine if this BST is the same as one described in post-order as {2,3,5,7,11,23,19,13}.