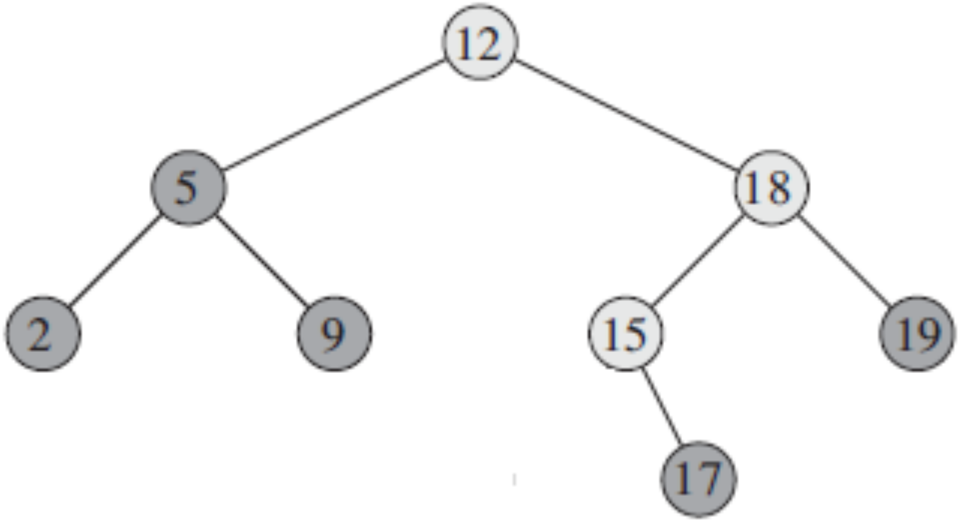
Binary Search Trees

BST Insertion

- Insertion & Deletion are modifying operations
  - BST changes as the result of these operations
  - BST property continues to hold

Insertion

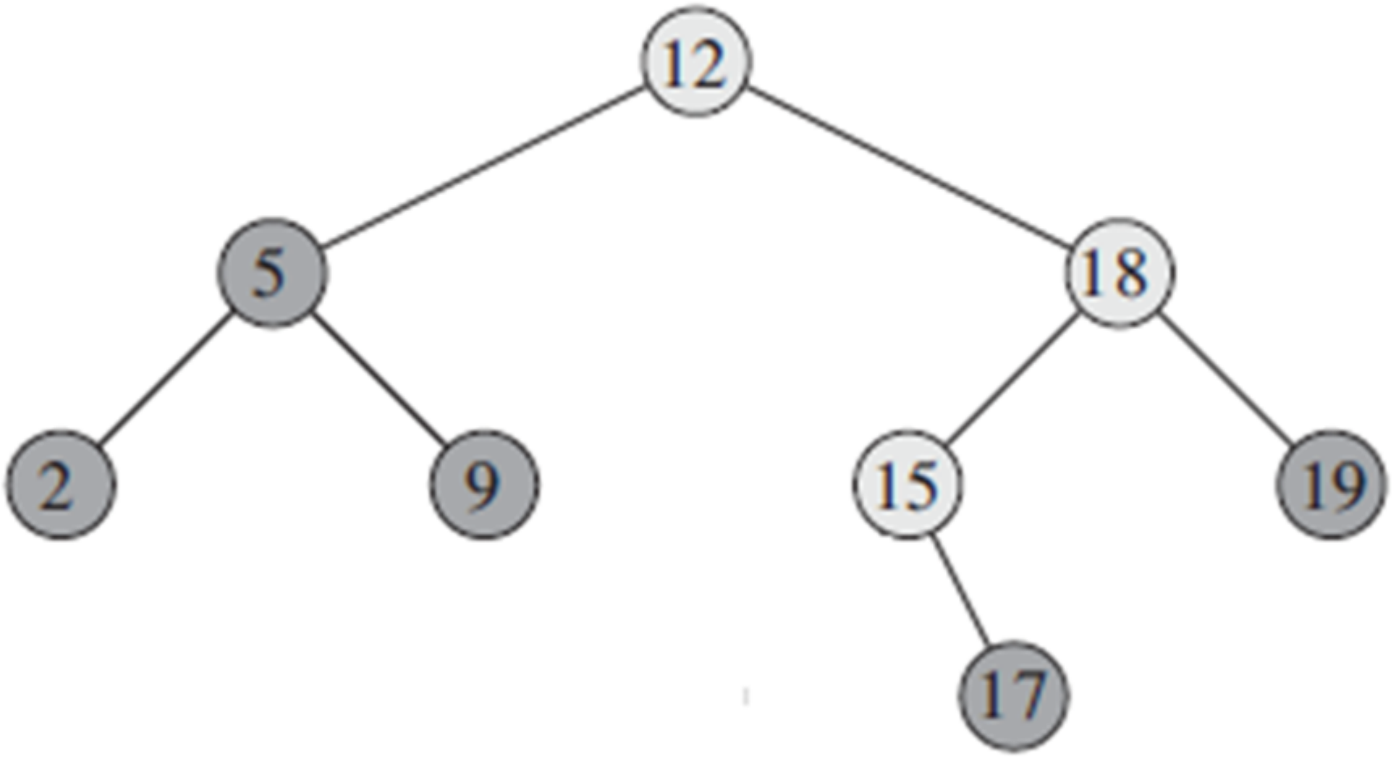
- How do we insert a node into a BST in such a way that the BST property continues to hold?
- Suppose we want to insert 13 to the BST below, how do we proceed?



Procedure to insert a value *v* into a BST T

- **Input**
  - BST *T*
  - A node z for which z.key = *v*, z.left = NIL & z.right =NIL
- **Tree-Insert** modifies
  - T
  - Some attributes of z
  - Inserts z into an appropriate position in T

Insert the value 13 to given T

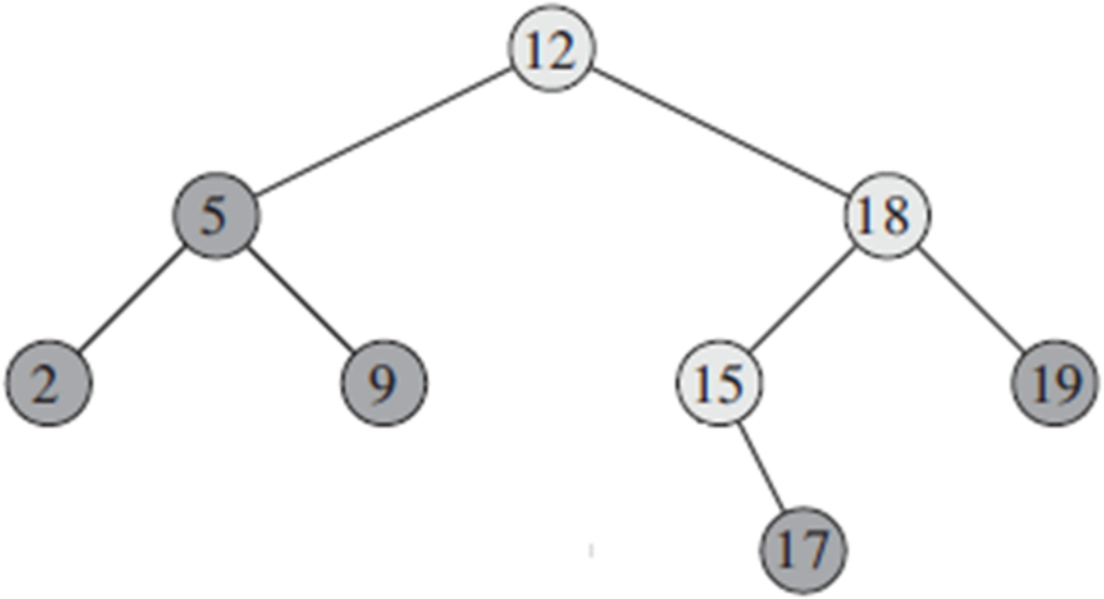


TREE-INSERT(T,z)

- Starts with the root of T
- Maintains two pointers x and y
- x traces a simple path downward looking for a NIL to replace with z
- y is the trailing pointer which is the parent of x
- Why do we need a trailing pointer y as the parent of x?

Why a trailing pointer y as the parent of x ?

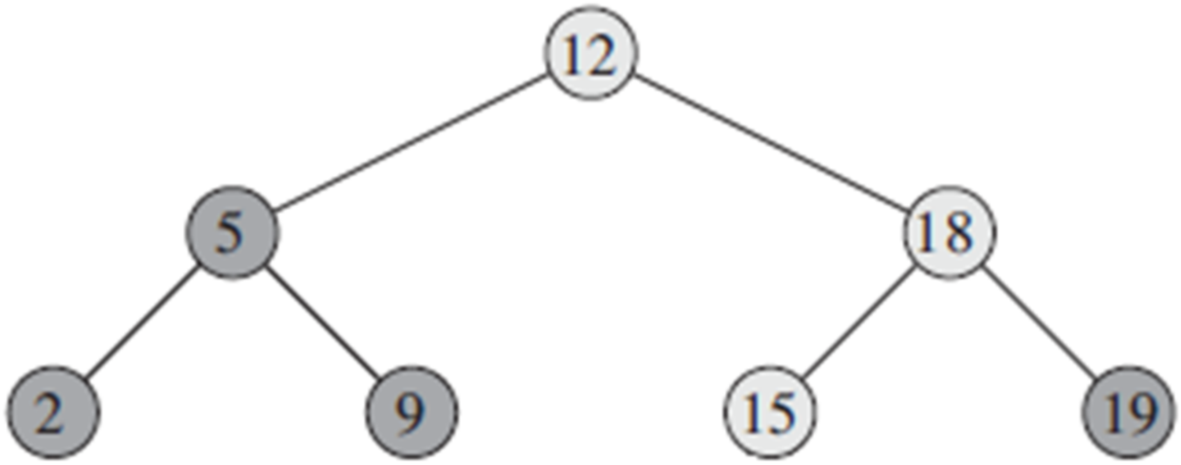
- Insert 13

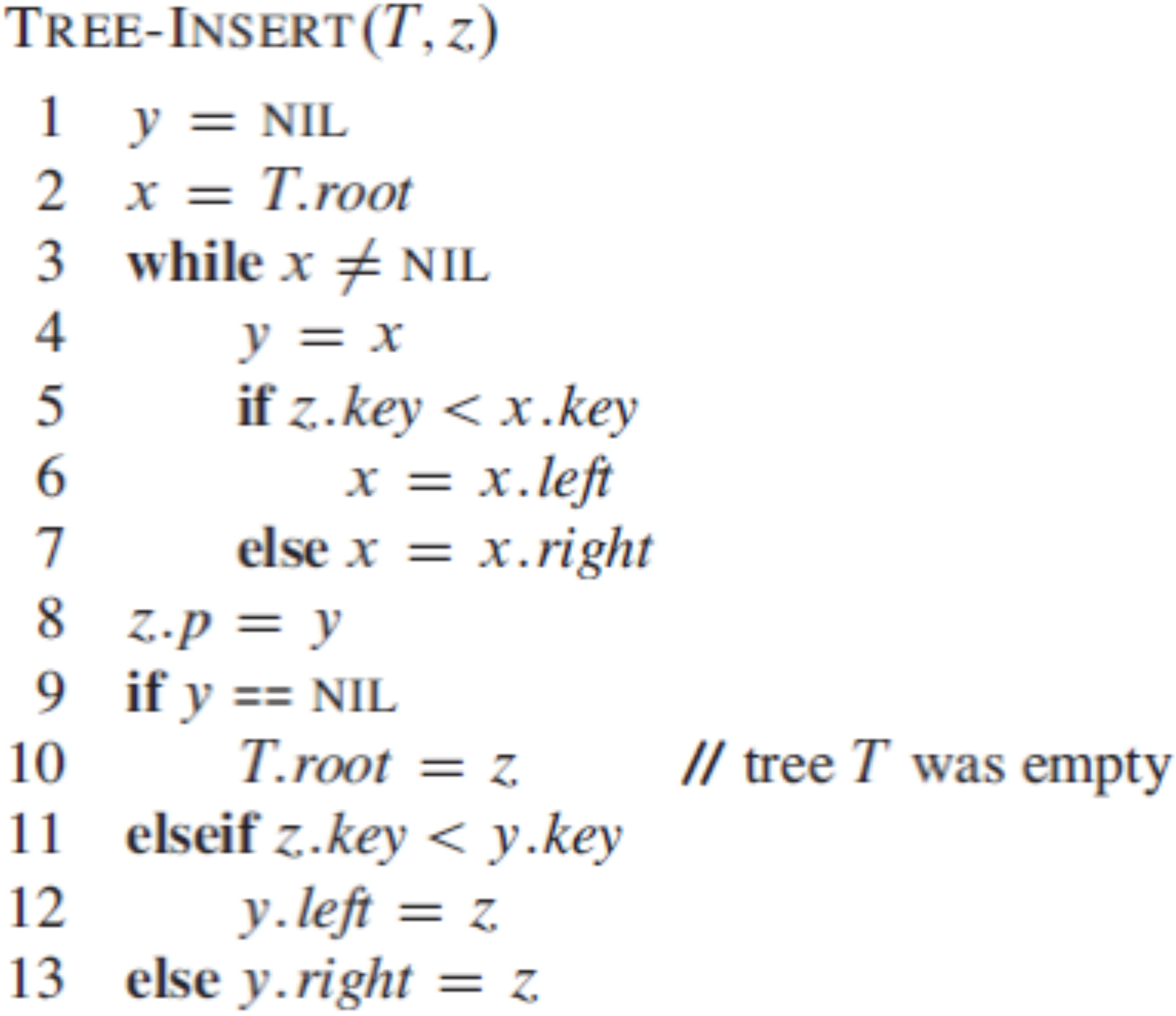


- Pointer x moves beyond where we have to insert
- Trailing pointer y keeps track of the exact position to insert z

Another T

- Insert 3,17, 25, 8 in sequence (one after another)

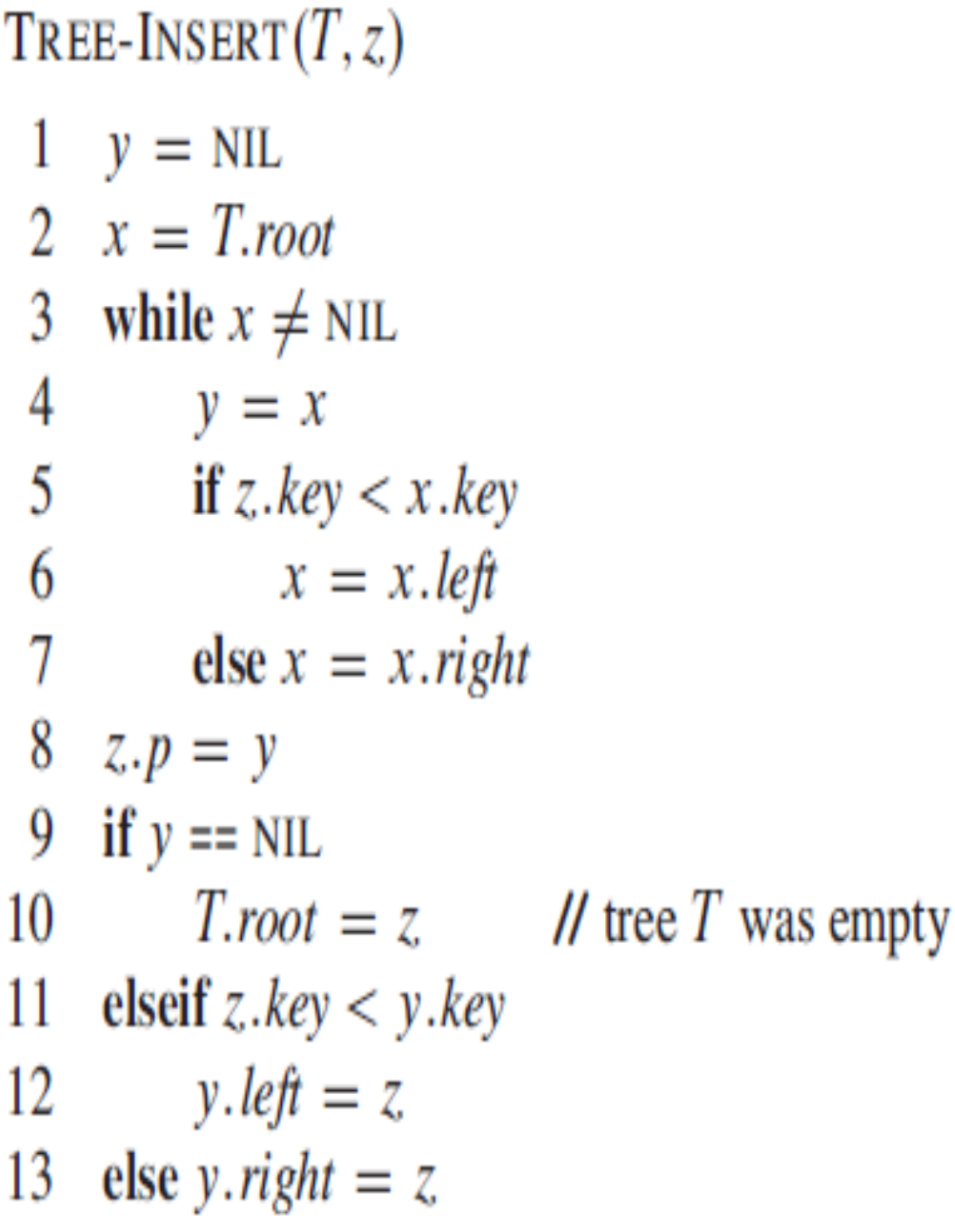




```
TREE-INSERT($T, z$)
$y = $ NIL
$x = T.root$
while $x \neq$ NIL
    $y = x$
    if $z.key < x.key$
        $x = x.left$
    else $x = x.right$
$z.p = y$
if $y ==$ NIL
    $T.root = z$        // tree $T$ was empty
elseif $z.key < y.key$
    $y.left = z$
else $y.right = z$
```

TREE-INSERT(T,z)

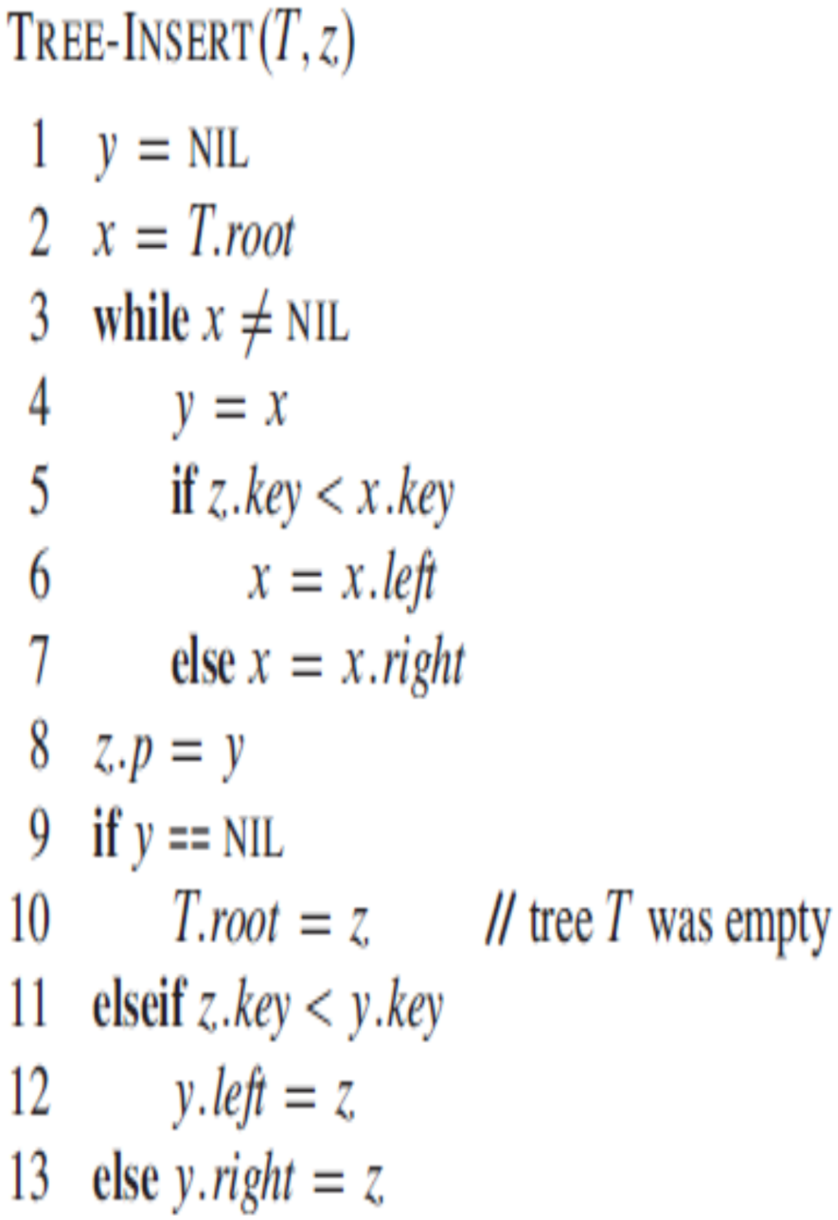
- Starts with the root of T
- Maintains two pointers x and y
- x traces a simple path downward looking for a NIL to replace with z
- y is the trailing pointer which is the parent of x



```
TREE-INSERT(T, z)
y = NIL
x = T.root
while x ≠ NIL
    y = x
    if z.key < x.key
        x = x.left
    else x = x.right
z.p = y
if y == NIL
    T.root = z          // tree T was empty
elseif z.key < y.key
    y.left = z
else y.right = z
```

TREE-INSERT(T,z) – contd.

- While loop (lines 3-7) causes x & y to move down T until x becomes NIL
- NIL occupies the position where we need to insert z
- Lines 8-13 sets the pointers that cause z to be inserted

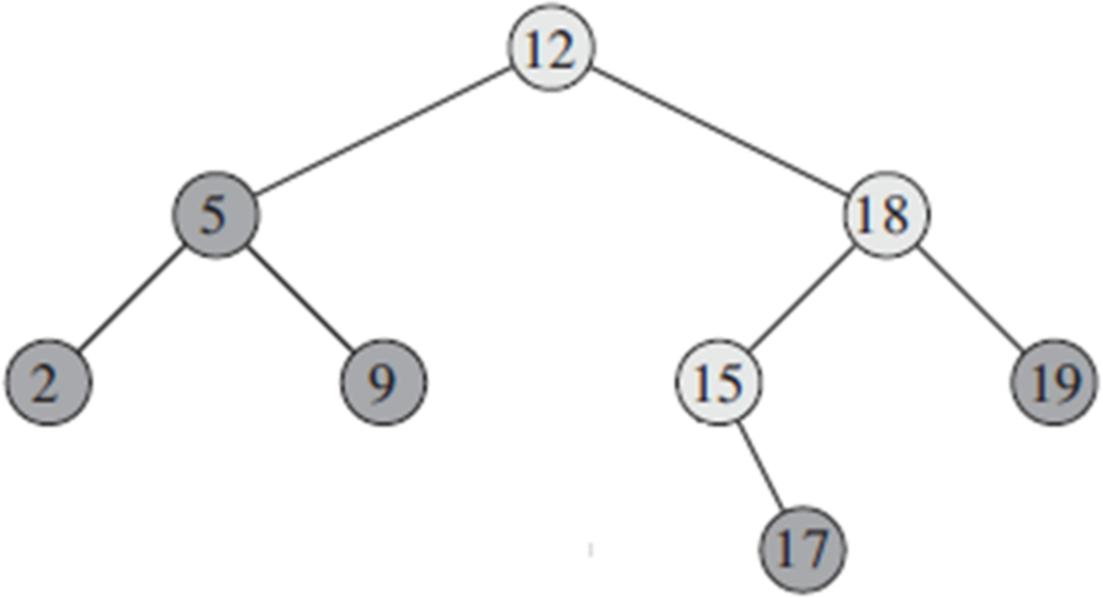


TREE-INSERT(T, z)

```
1  $y$ = NIL
2  $x = T.root$
3  while $x \neq$ 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 $y$ == NIL
10     $T.root = z$        // tree $T$ was empty
11 elseif $z.key < y.key$
12     $y.left = z$
13 else $y.right = z$
```

Running time of TREE-INSERT(T,z)

- Traverses from root to the appropriate position where z (eg: 13) has to be inserted



- Running time : O(h) on a tree of height h

Exercise

Write the recursive version of TREE-INSERT

Reference

1. CLRS Book, Third Edition.