CSCI 241

Scott Wehrwein

AVL Trees: Definition, Insertion

Goals

Know the definition and properties of an AVL tree.

Know how AVL insertion rebalances the tree to correct violations of the AVL property.

AVL Trees

An AVL tree is a Binary Search Tree in which:

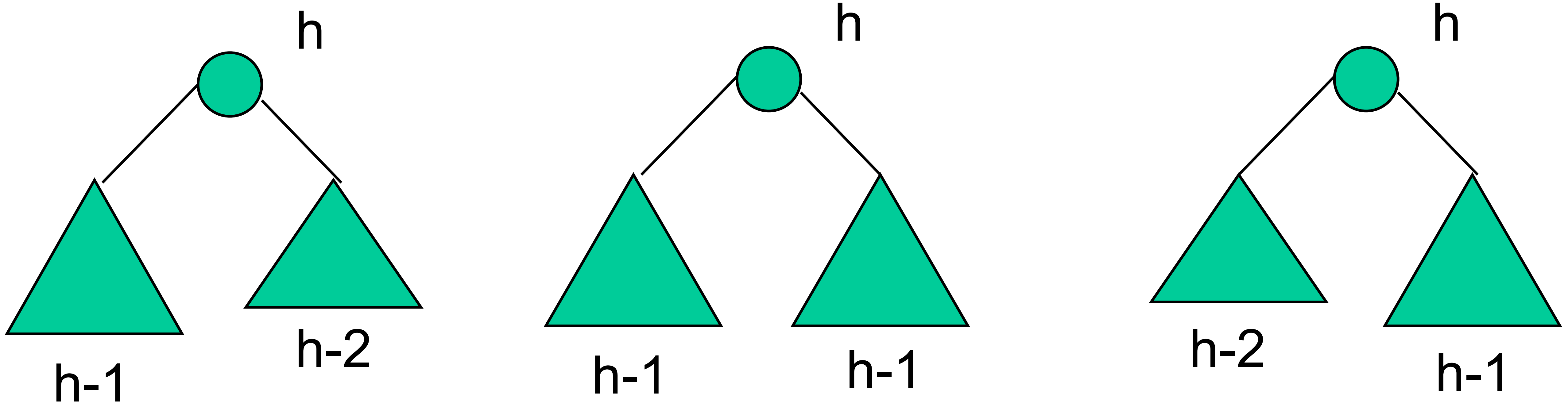
$$\boxed{-1 <= \text{balance}(n) <= 1 \text{ for all nodes } n.}$$

Balance(n): height(n.right) - height(n.left)

Balance Factor in AVL Trees

AVL property: -1 <= balance(n) <= 1 for all nodes n.

Every subtree in an AVL tree looks like one of these three trees:



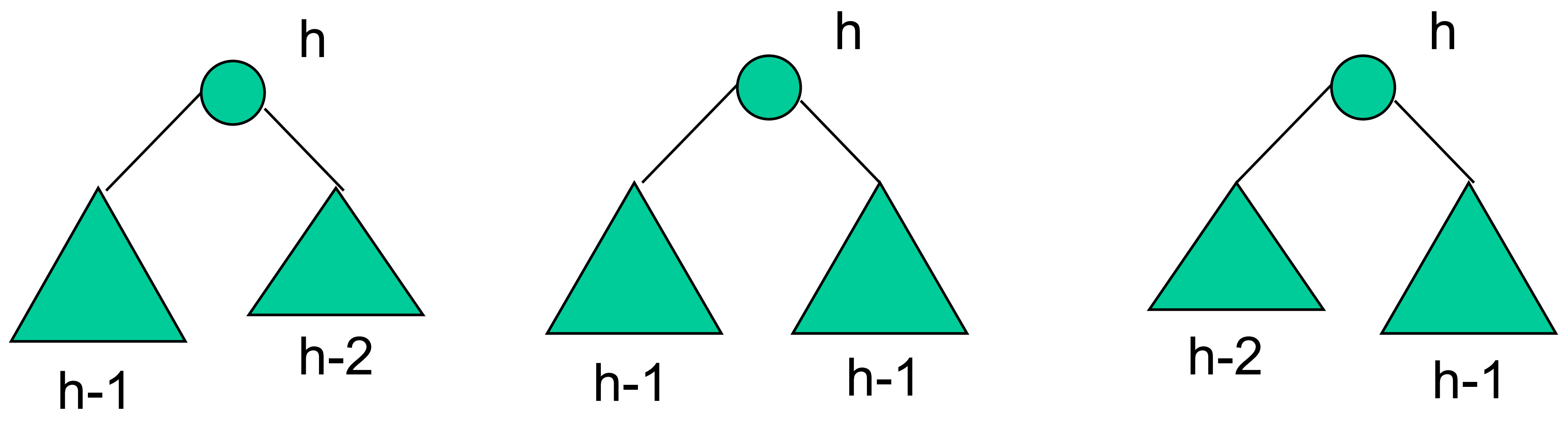
(a) Balance factor:-1 (b) Balance factor: 0 (c) Balance factor: +1

AVL Trees: Insertion

AVL property: -1 <= b(n) <= 1 for all nodes n.

To insert into an AVL tree:

1. Do a normal BST insertion
2. Fix any violations of the AVL property using rotations.



(a) Balance factor:-1 (b) Balance factor: 0 (c) Balance factor: +1

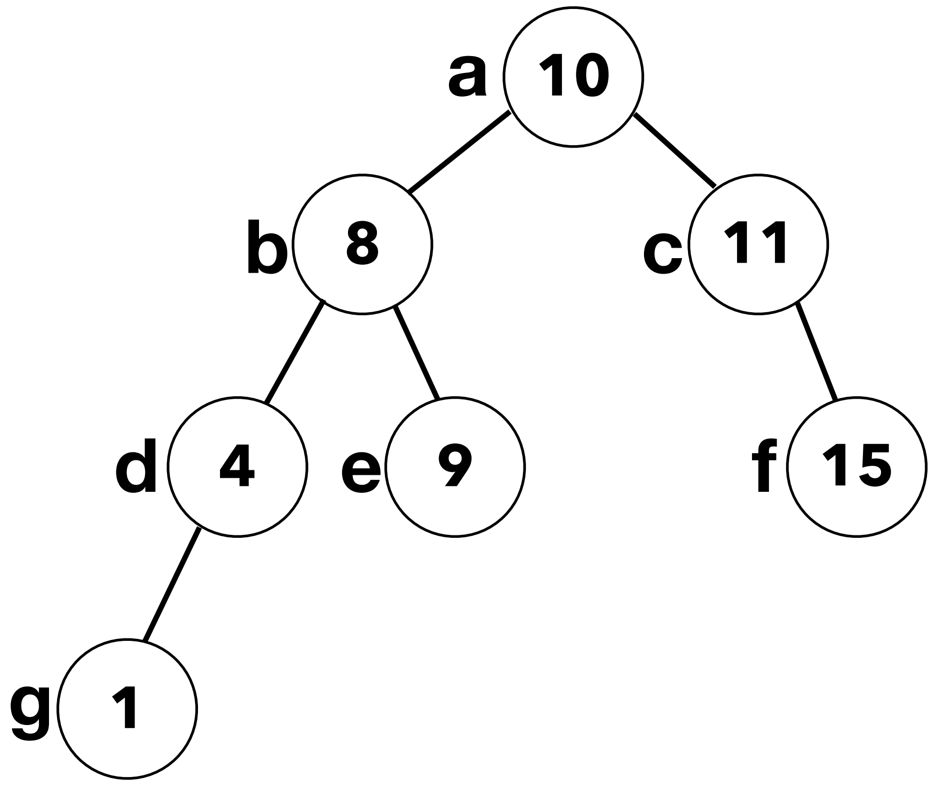
Refresher: BST Insertion

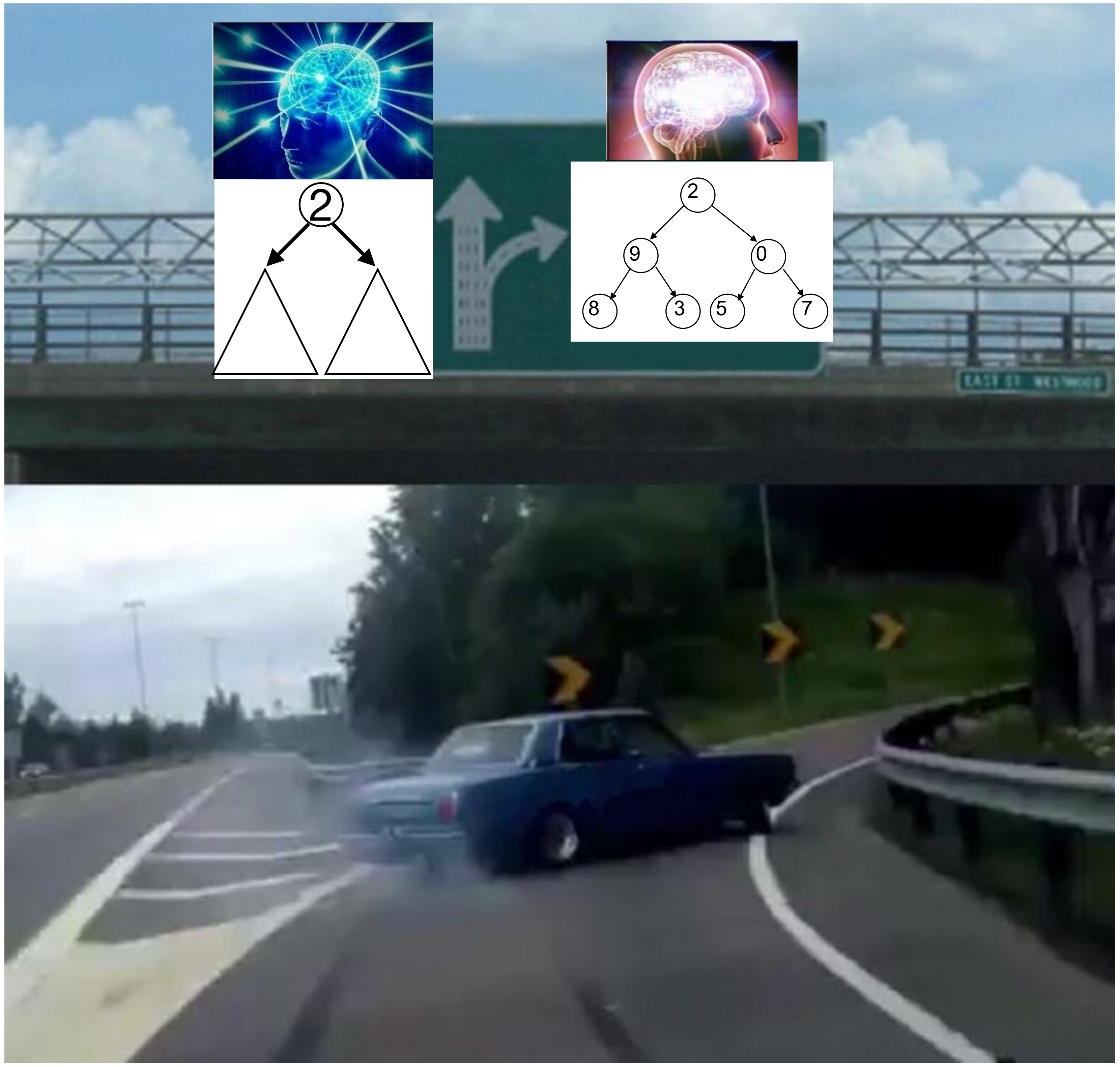
```
/* insert a node with value v into the
 * tree rooted at n. pre: n is not null. */
insert(Node n, int v):
  if n.value == v: return // (duplicate)
  if v < n.value:
    if n has left:
      insert(n.left, v)
    else:
      // attach new node w/ value v to n.left
  else: // v > n.value
    if n has right:
      insert(n.right, v)
    else:
      // attach new node w/ value v to n.right
```

AVL Insertion

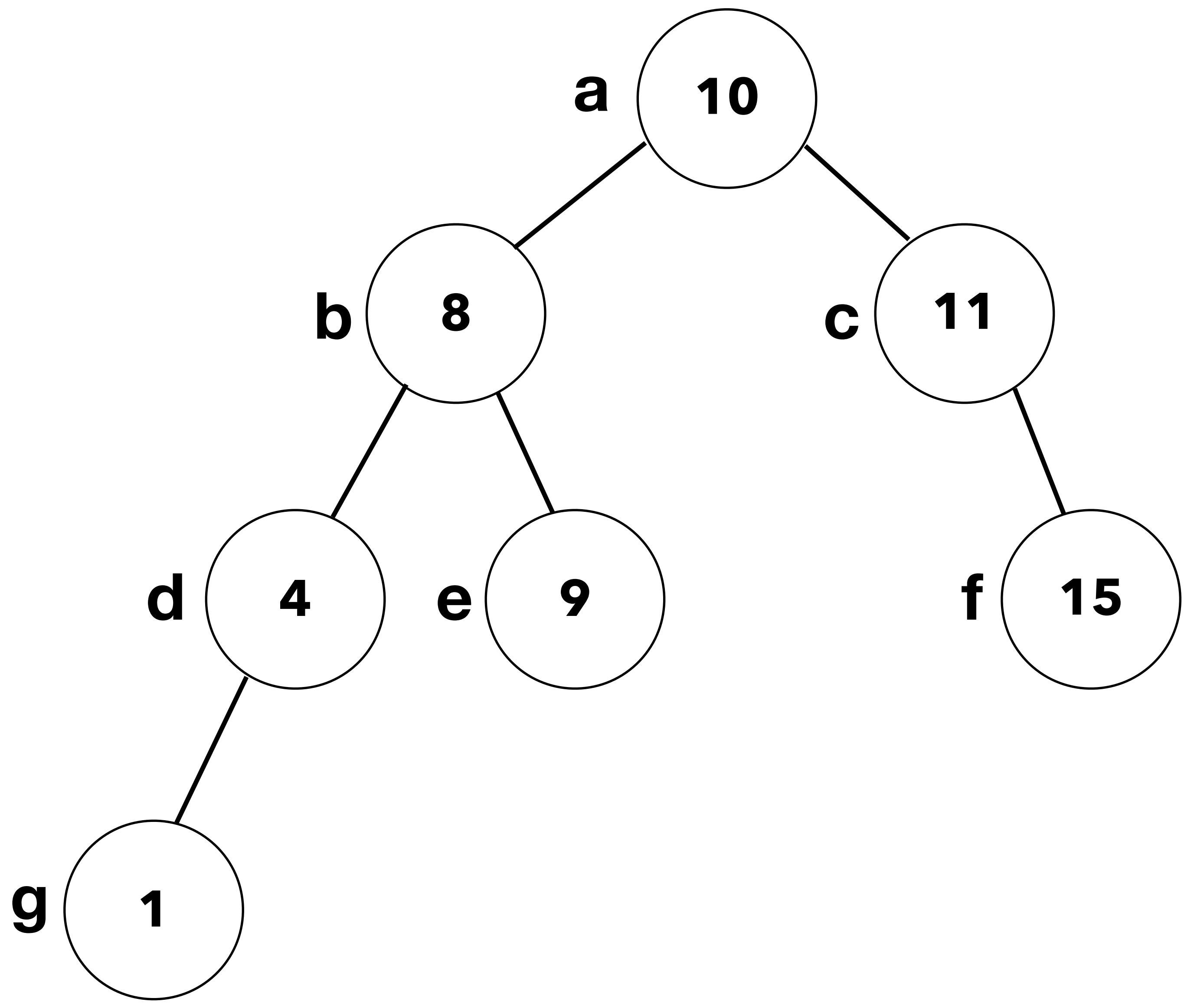
```
/* insert a node with value v into the
 * tree rooted at n. pre: n is not null. */
insert(Node n, int v):
  if n.value == v: return // (duplicate)
  if v < n.value:
    if n has left:
      insert(n.left, v)
    else:
      // attach new node w/ value v to n.left
  else: // v > n.value
    if n has right:
      insert(n.right, v)
    else:
      // attach new node w/ value v to n.right
  rebalance(n);  ←
```

AVL Insertion



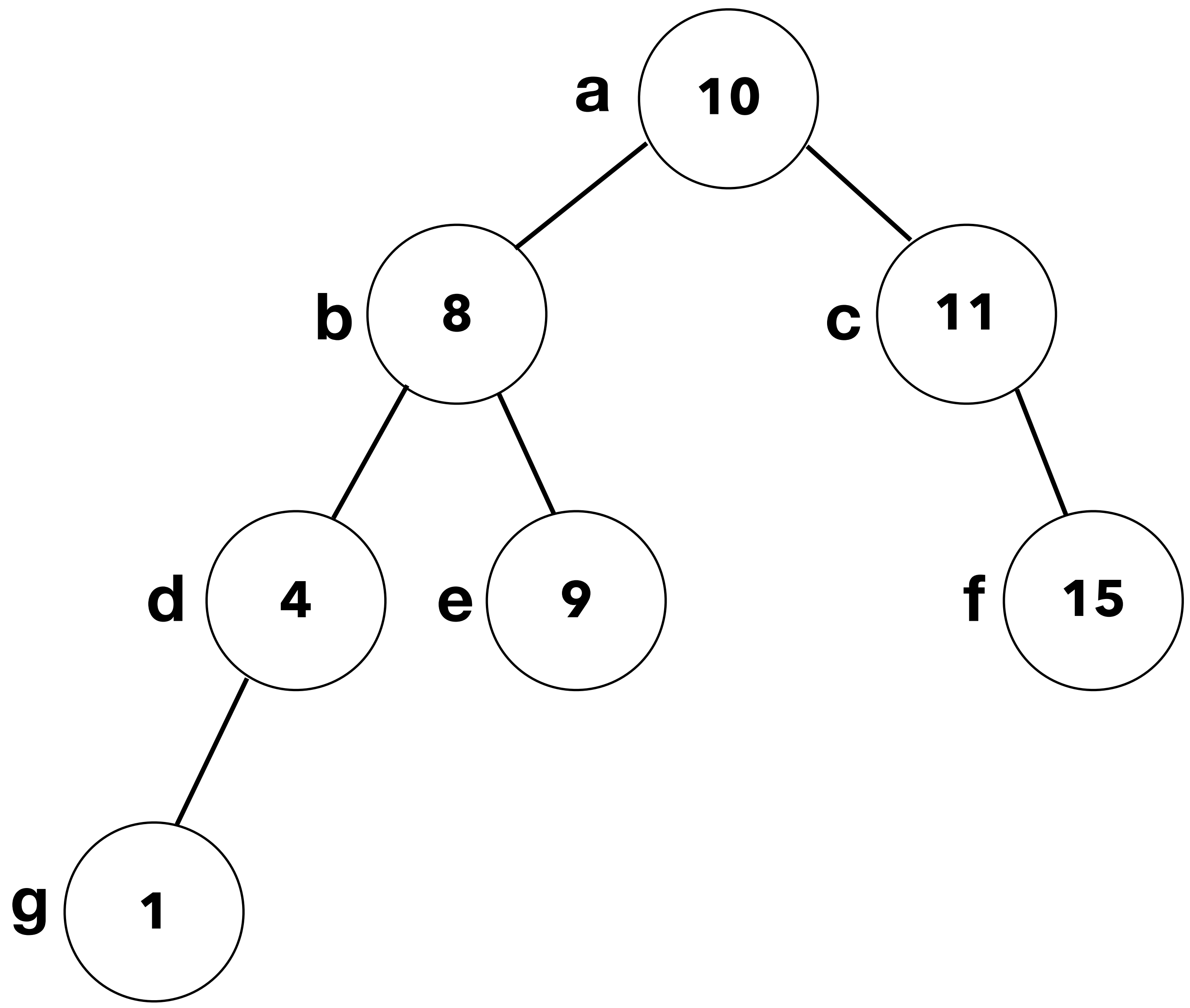


AVL Insertion



First: is this an AVL tree?

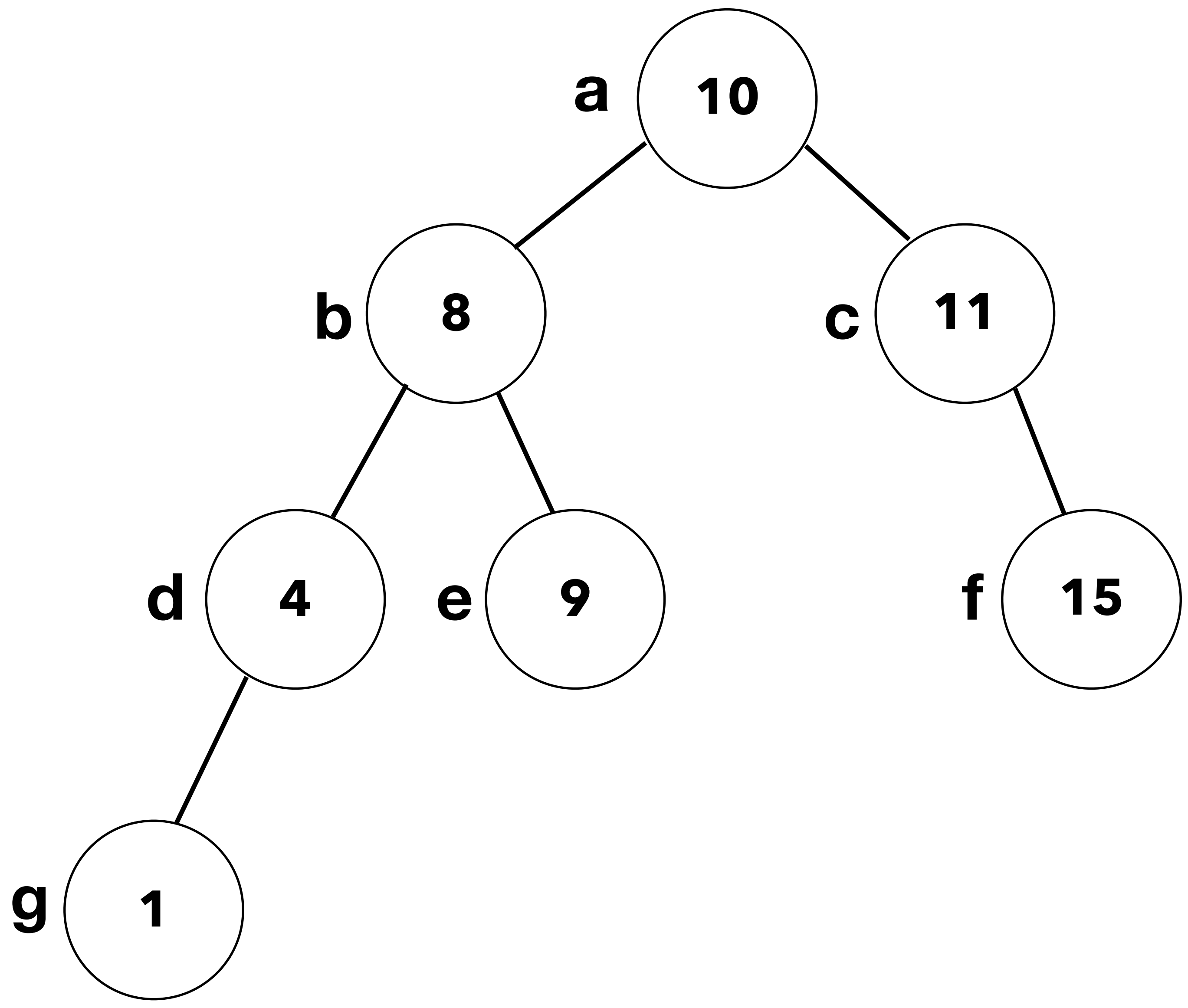
AVL Insertion



```
insert(a, 16)
```

```
insert(Node n, int v):
  //…(other cases
  else: // v > n.value
    if n has right:
      insert(n.right, v)
    else:
      // attach new node
  rebalance(n);
```

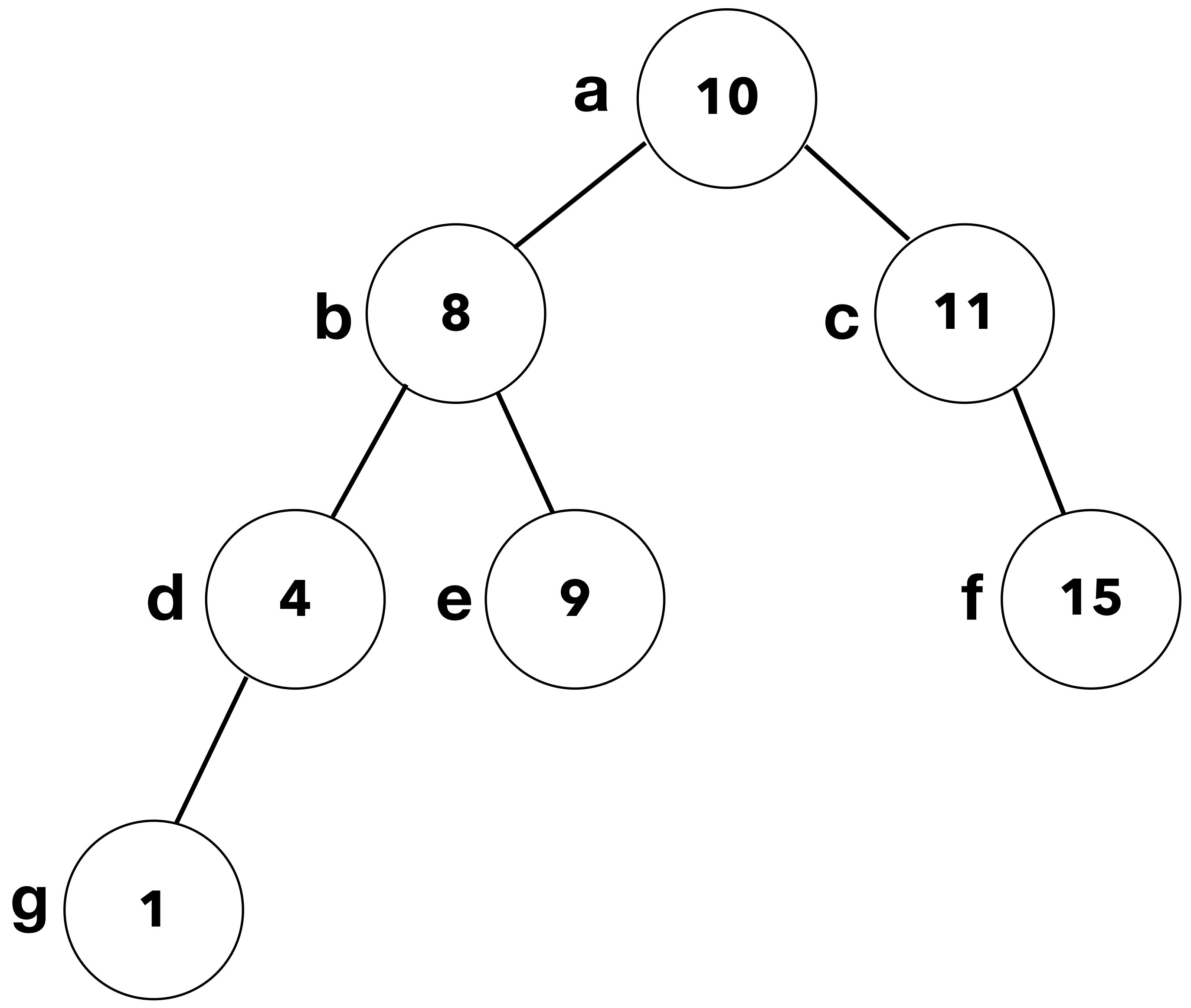
AVL Insertion



```
insert(a, 16)
=>insert(c, 16)





  rebalance(a)
```

```
insert(Node n, int v):
  //…(other cases
  else: // v > n.value
    if n has right:
      insert(n.right, v)
    else:
      // attach new node
  rebalance(n);
```

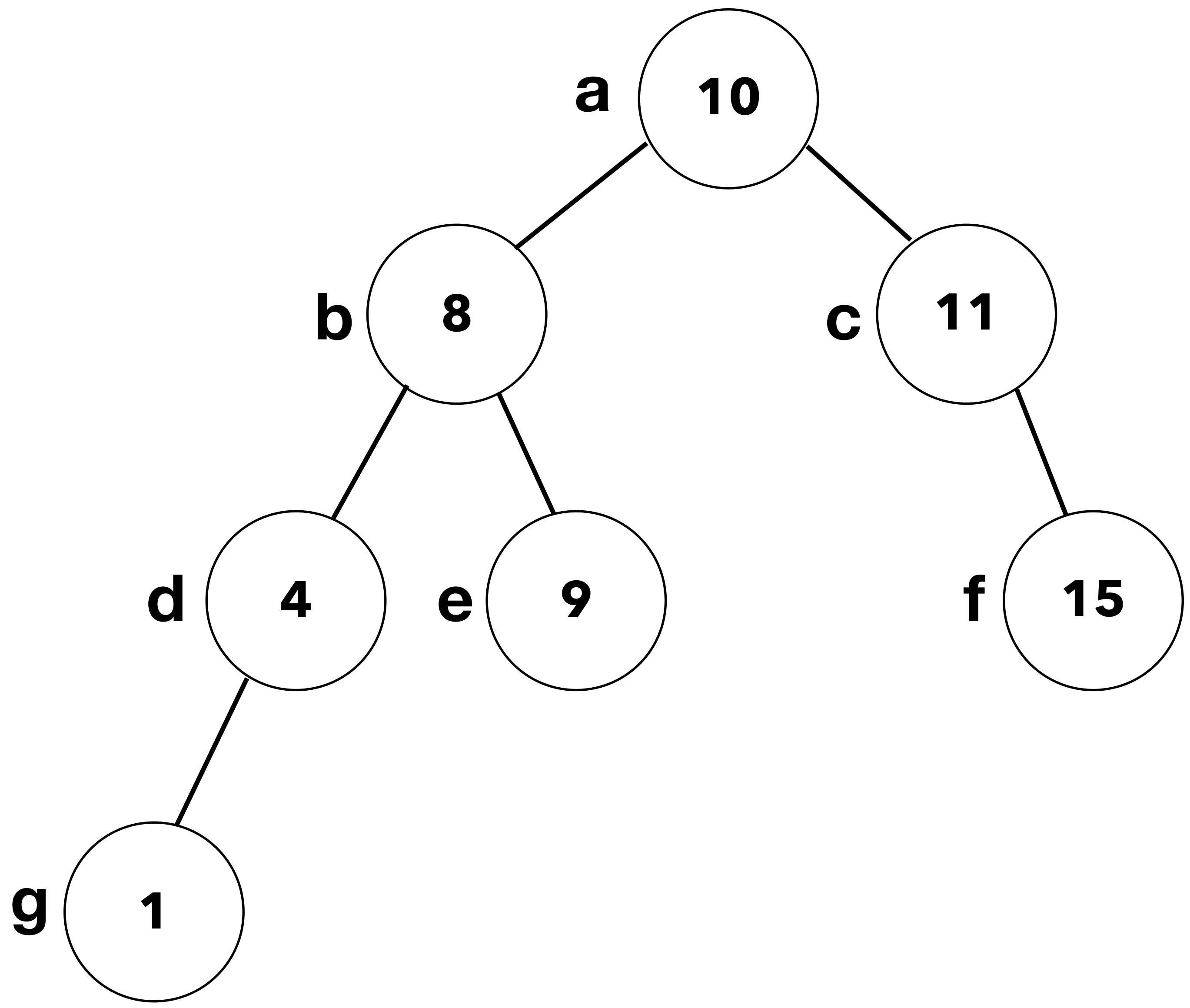
AVL Insertion



```
insert(a, 16)
=>insert(c, 16)
  =>insert(f, 16)



    rebalance(c)
  rebalance(a)
```

```
insert(Node n, int v):
  //…(other cases
  else: // v > n.value
    if n has right:
      insert(n.right, v)
    else:
      // attach new node
  rebalance(n);
```

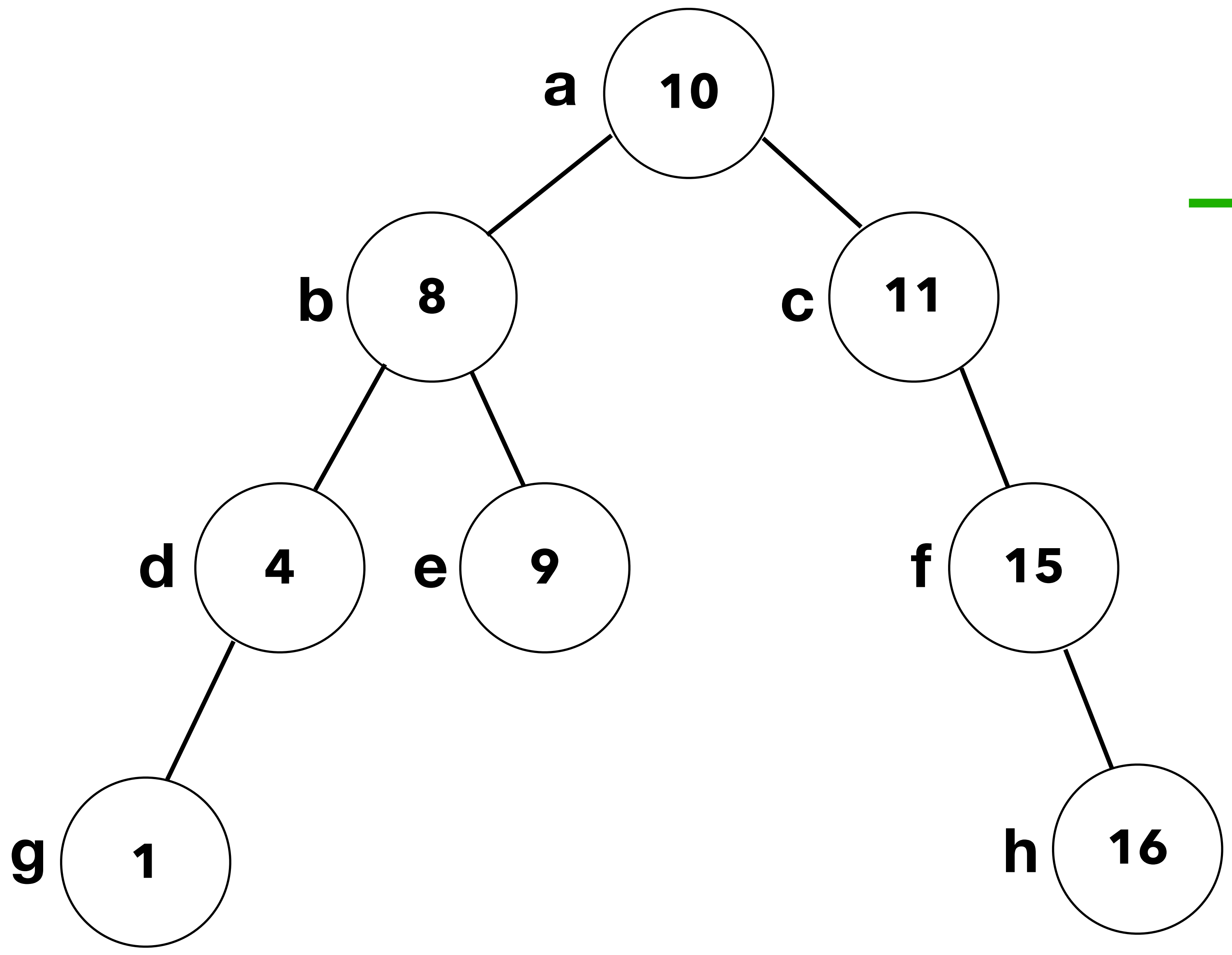
AVL Insertion



```
insert(a, 16)
=>insert(c, 16)
  =>insert(f, 16)
    =>attach new node
      rebalance(f)
    rebalance(c)
  rebalance(a)
```

```
insert(Node n, int v):
  //…(other cases
  else: // v > n.value
    if n has right:
      insert(n.right, v)
    else:
      // attach new node
  rebalance(n);
```

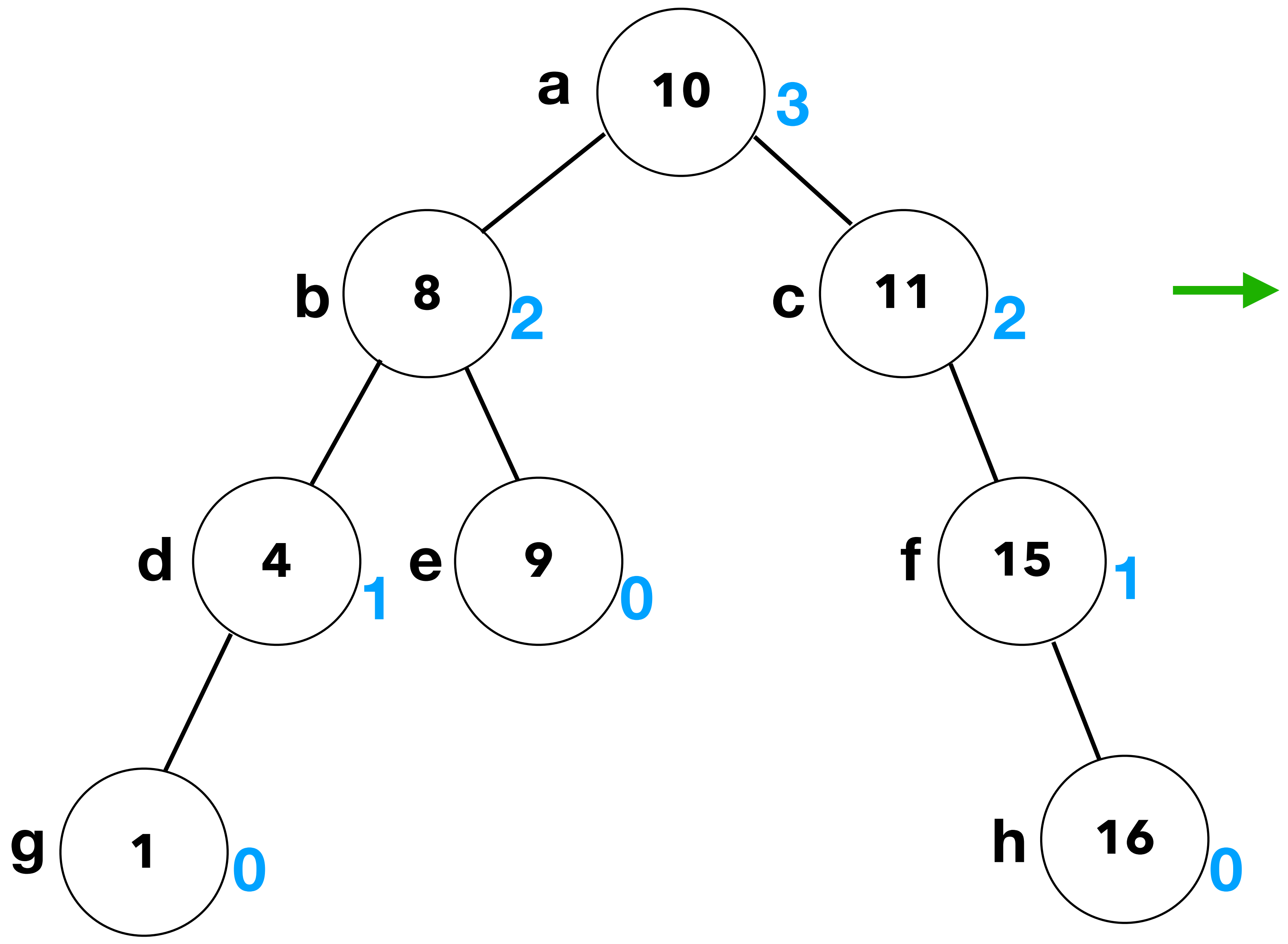
AVL Insertion



```
insert(a, 16)
=>insert(c, 16)
  =>insert(f, 16)
    =>attach new node
      rebalance(f)
    rebalance(c)
  rebalance(a)
```

```
insert(Node n, int v):
  //…(other cases
  else: // v > n.value
    if n has right:
      insert(n.right, v)
    else:
      // attach new node
  rebalance(n);
```

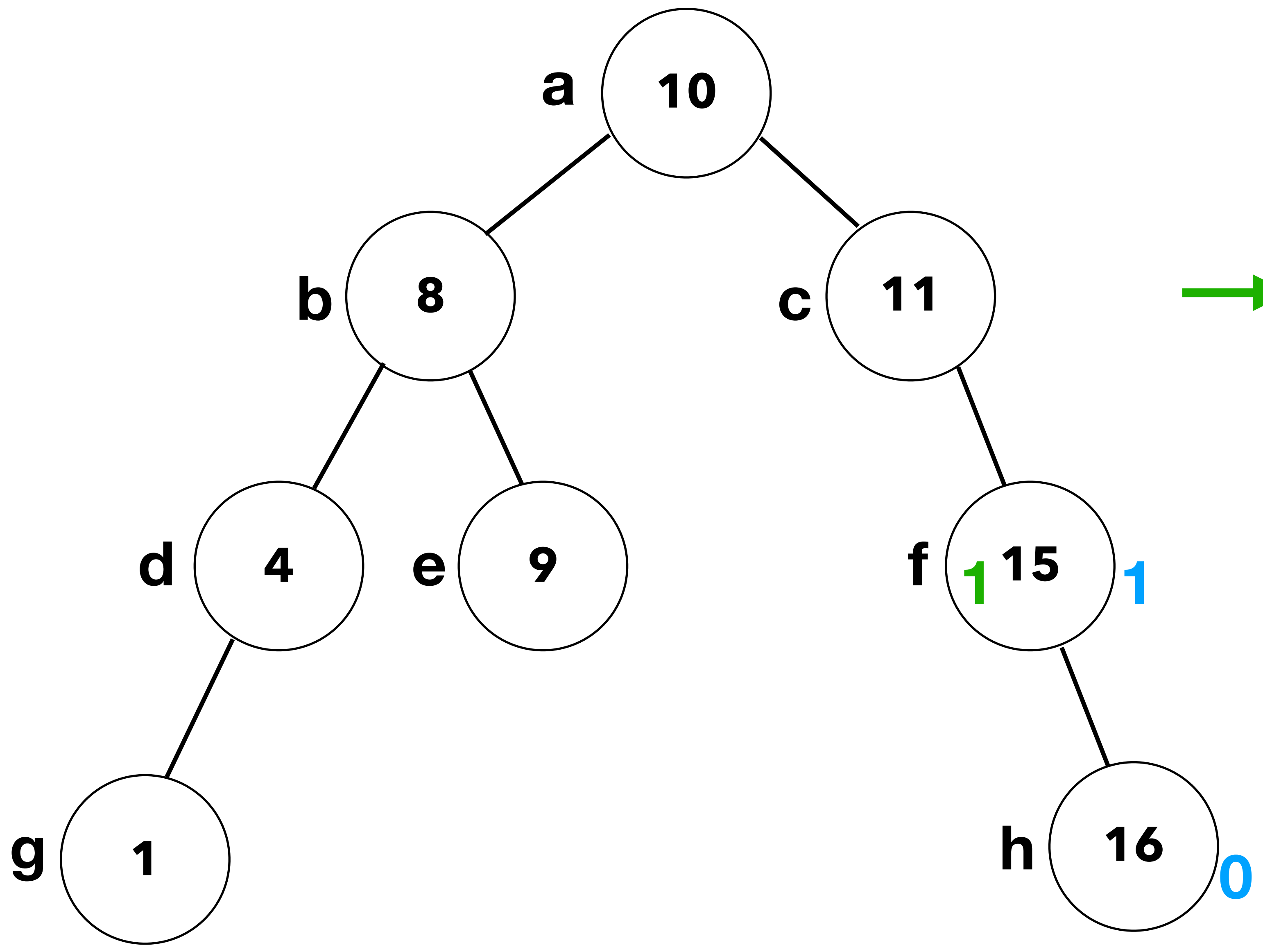
AVL Insertion



```
insert(a, 16)
=>insert(c, 16)
  =>insert(f, 16)
    =>attach new node
      rebalance(f)
    rebalance(c)
  rebalance(a)
```

```
insert(Node n, int v):
  //…(other cases
  else: // v > n.value
    if n has right:
      insert(n.right, v)
    else:
      // attach new node
  rebalance(n);
```

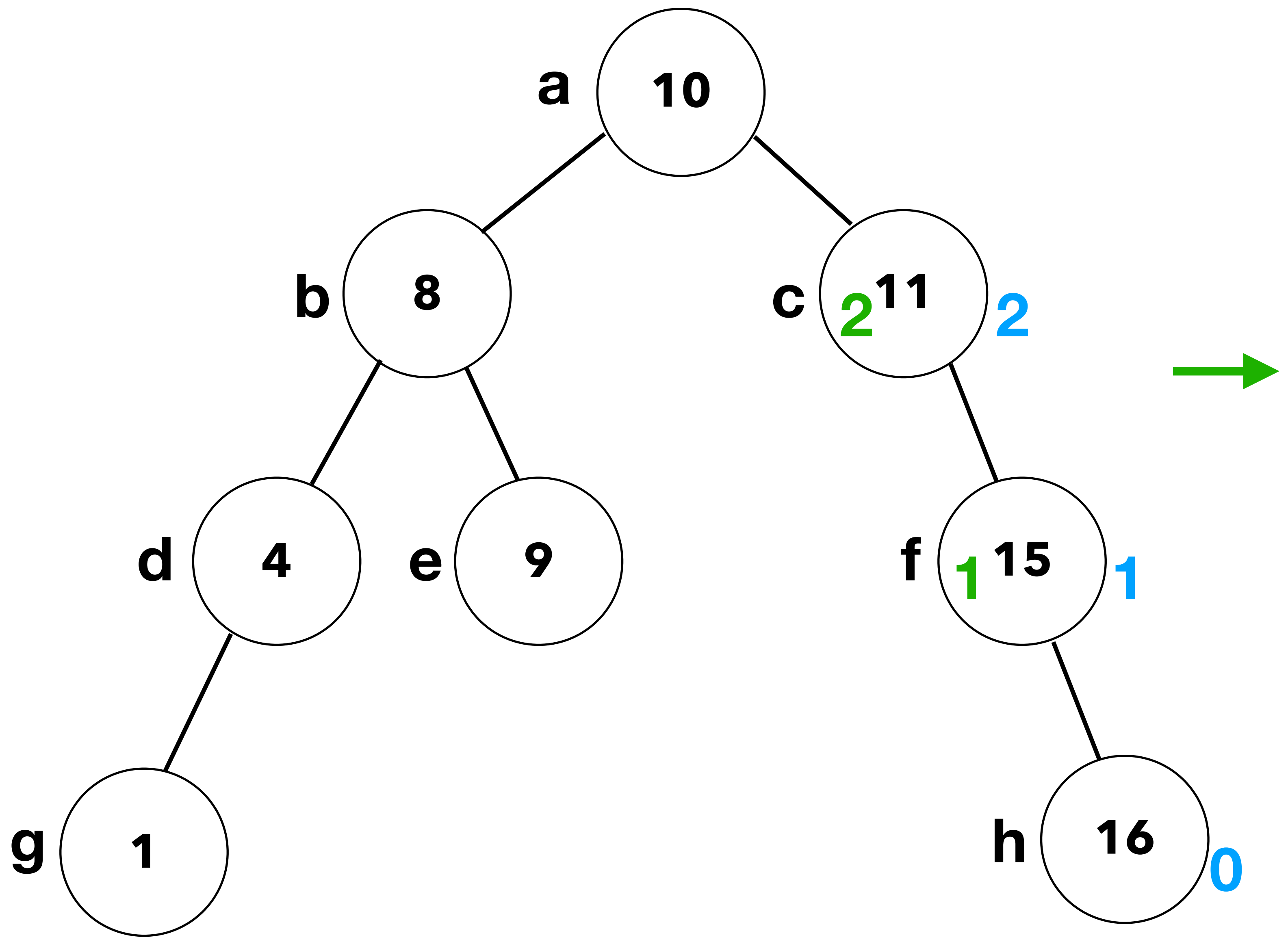
AVL Insertion



```
insert(a, 16)
=>insert(c, 16)
  =>insert(f, 16)
    =>attach new node
     (already balanced)(f)
    rebalance(c)
  rebalance(a)
```

```
insert(Node n, int v):
  //…(other cases
  else: // v > n.value
    if n has right:
      insert(n.right, v)
    else:
      // attach new node
  rebalance(n);
```

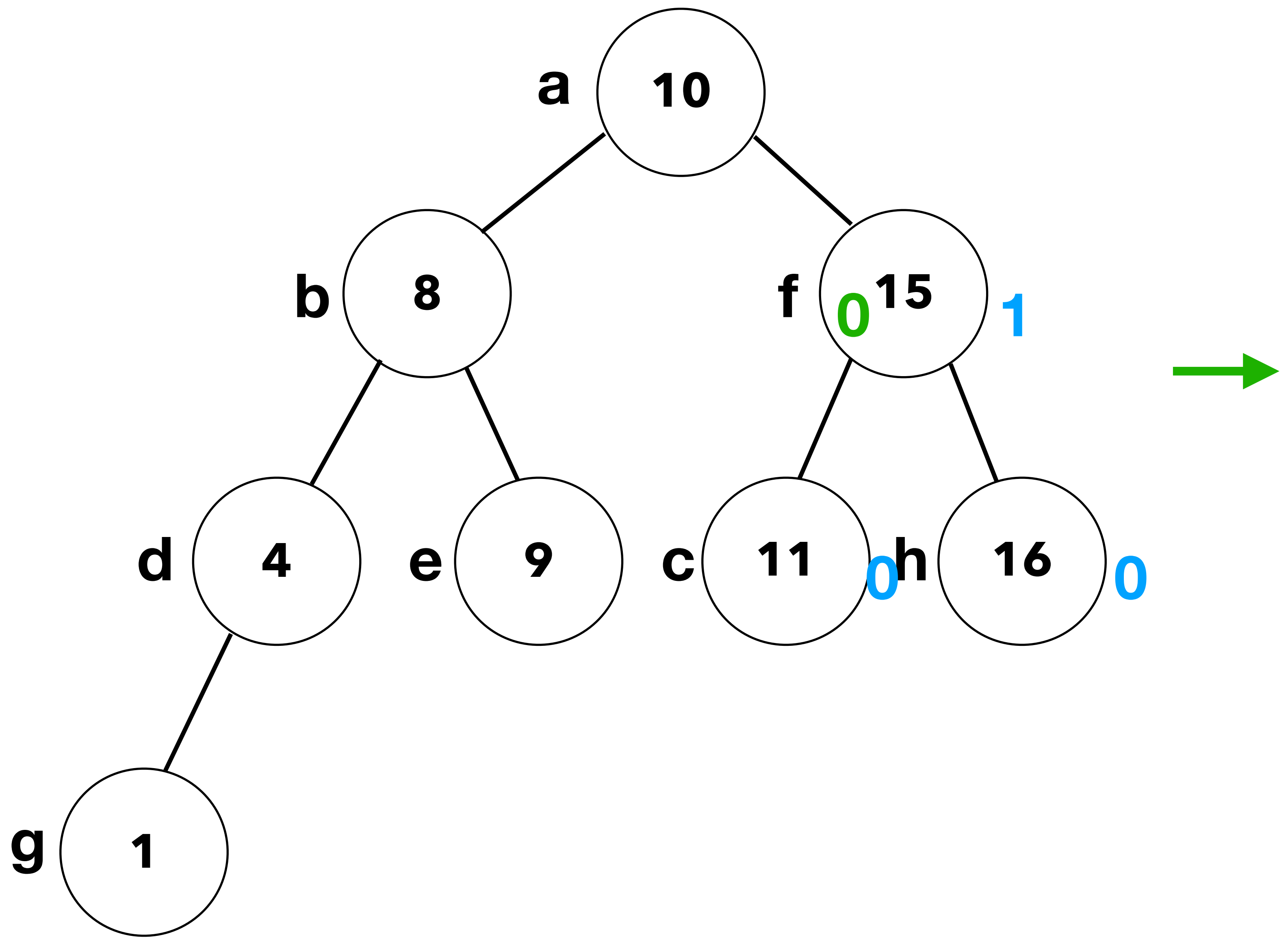
AVL Insertion



```
insert(a, 16)
=>insert(c, 16)
  =>insert(f, 16)
    =>attach new node
    (already balanced)(f)
   (perform rotation)(c)
  rebalance(a)
```

```
insert(Node n, int v):
  //…(other cases
  else: // v > n.value
    if n has right:
      insert(n.right, v)
    else:
      // attach new node
  rebalance(n);
```

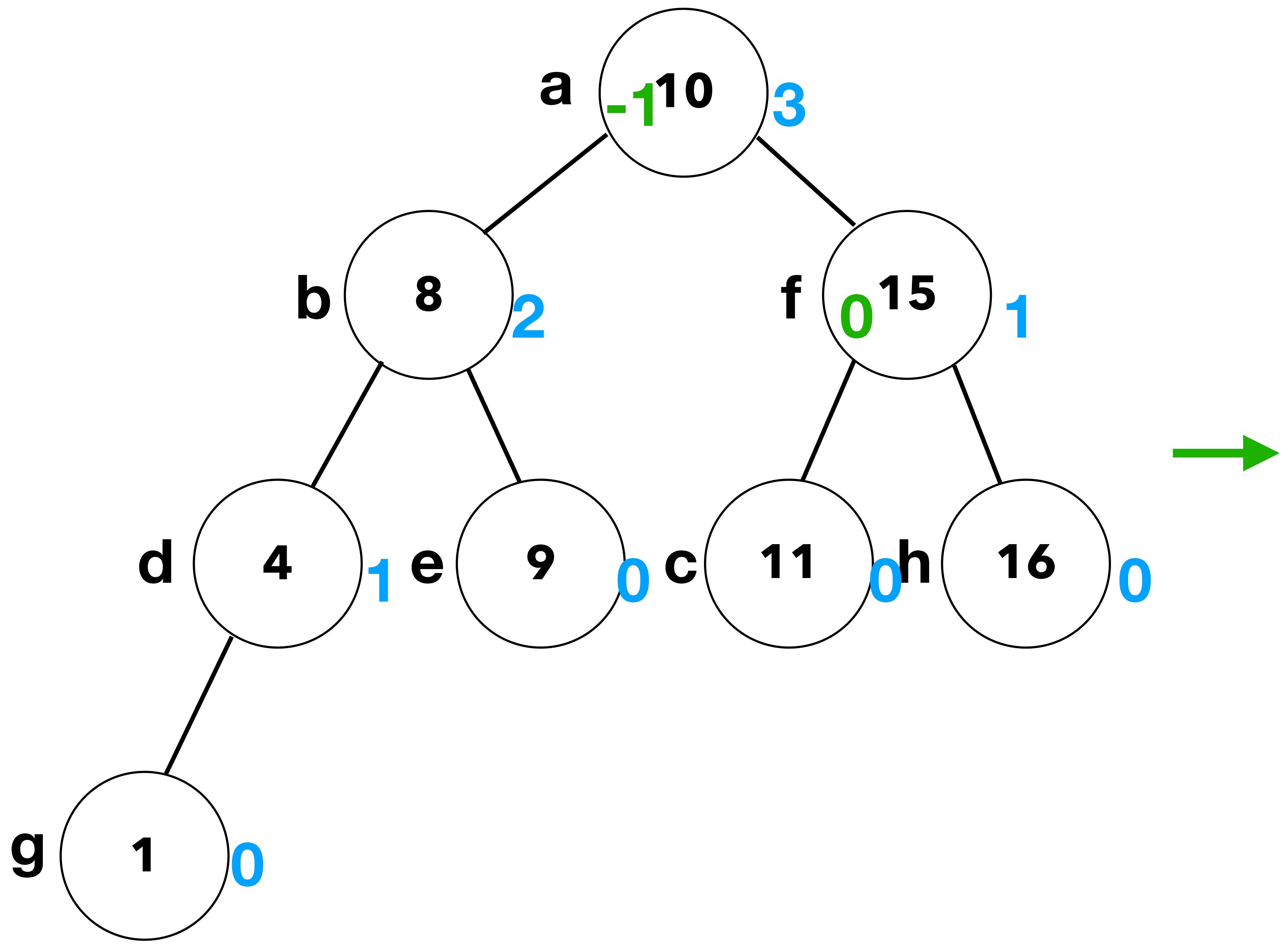
AVL Insertion



```
insert(a, 16)
=>insert(c, 16)
  =>insert(f, 16)
    =>attach new node
     (already balanced)(f)
   (perform rotation)(c)
  rebalance(a)
```

```
insert(Node n, int v):
  //…(other cases
  else: // v > n.value
    if n has right:
      insert(n.right, v)
    else:
      // attach new node
  rebalance(n);
```

AVL Insertion



```
insert(a, 16)
=>insert(c, 16)
  =>insert(f, 16)
    =>attach new node
    (already balanced)(f)
   (perform rotation)(c)
 (already balanced)(a)
```

```
insert(Node n, int v):
  //…(other cases
  else: // v > n.value
    if n has right:
      insert(n.right, v)
    else:
      // attach new node
  rebalance(n);
```