CSCI 241

Lecture 13b

Balance Factor

Balanced Binary Search Trees

Goals

- Know why we want our BSTs to be balanced.
- Be able to calculate the balance factor of a tree.

Back to BSTs

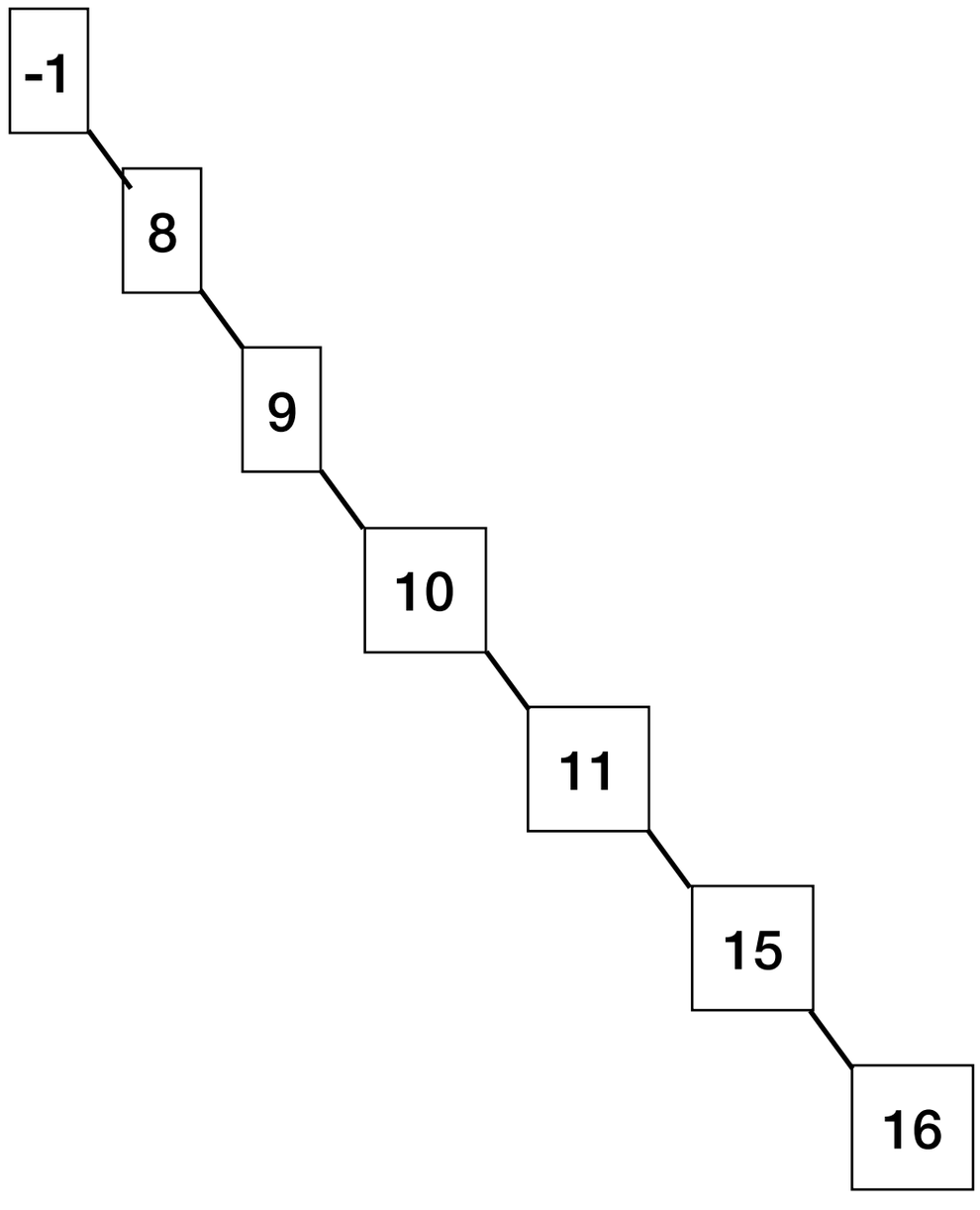
Long ago, we built some trees:

```
t = new BST();
t.insert(-1)
t.insert(8)
t.insert(9)
t.insert(10)
t.insert(11)
t.insert(15)
t.insert(16)
t.insert(16)
```

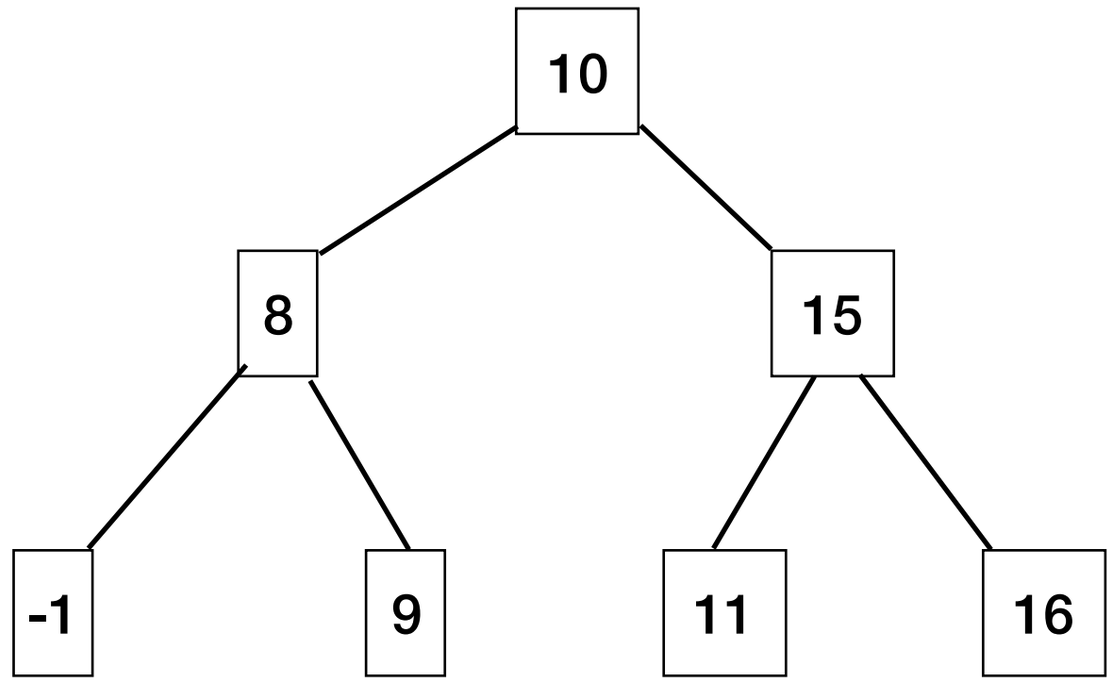
```
t = new BST();
t.insert(10)
t.insert(15)
t.insert(16)
t.insert(8)
t.insert(16)
t.insert(9)
t.insert(11)
t.insert(-1)
```

Same values, different trees

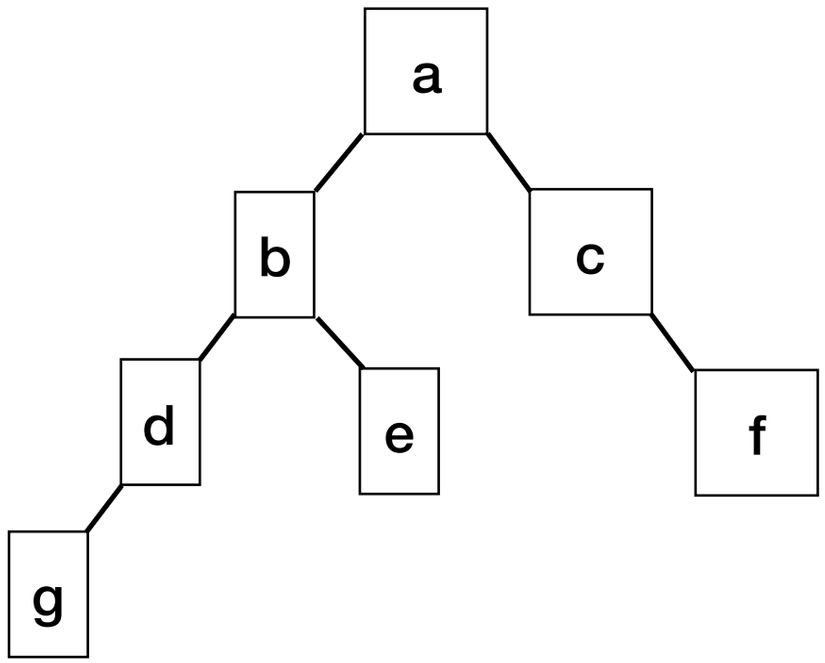
Bad tree =(



Good tree =)



Quantifying Badness: Balance Factor



Height(t): path length from t's deepest descendant (leaf) to t's root.

Height(n): height of the subtree rooted at n

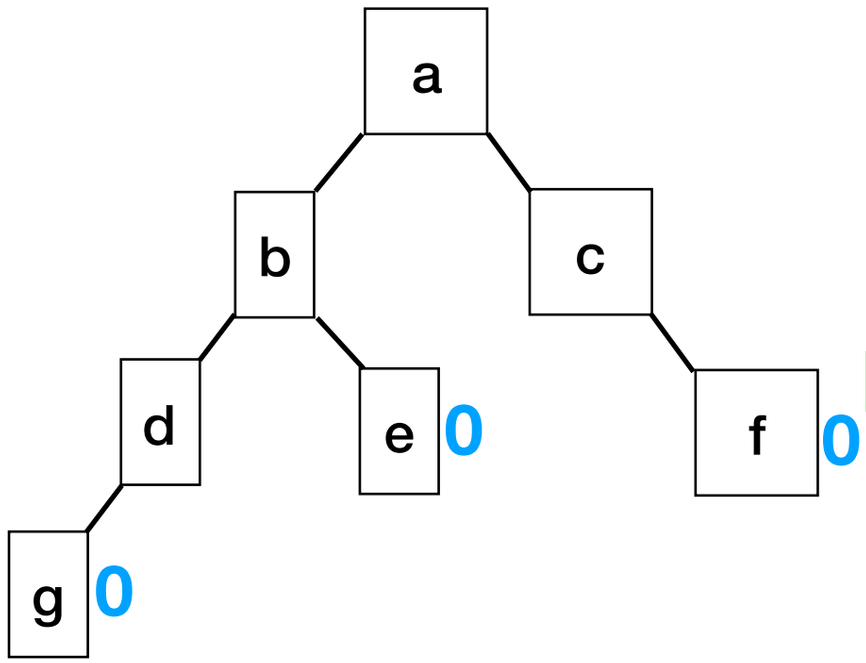
Balance(n): height(n.right) - height(n.left)

```
height(null) = -1
height(n) = 1 + max(height(n.left), height(n.right)
```

Quantifying Badness: Balance Factor

Height(t): path length from t's deepest descendant (leaf) to t's root.

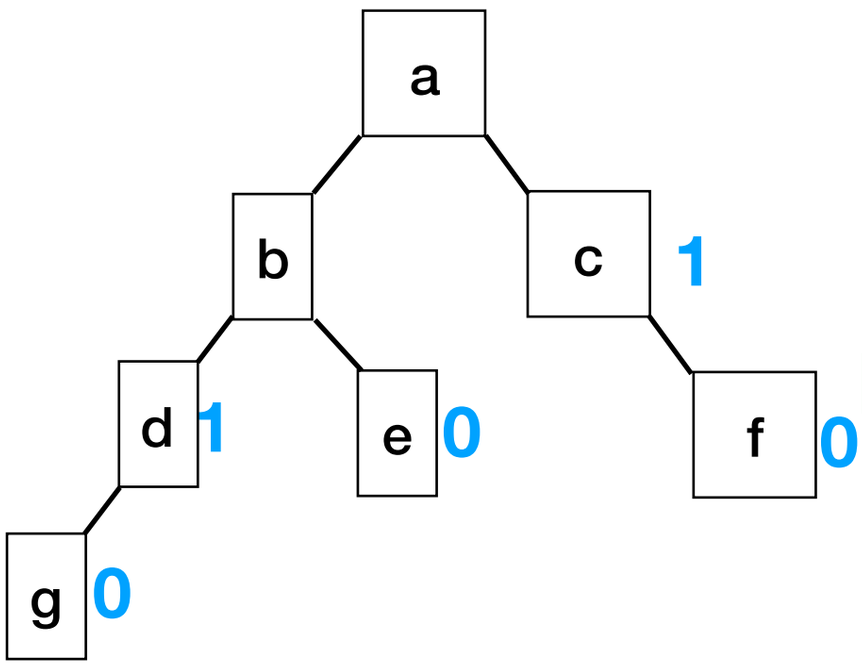
Height(n): height of the subtree rooted at n



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

```
height(null) = -1
height(n) = 1 + max(height(n.left), height(n.right)
```

Quantifying Badness: Balance Factor



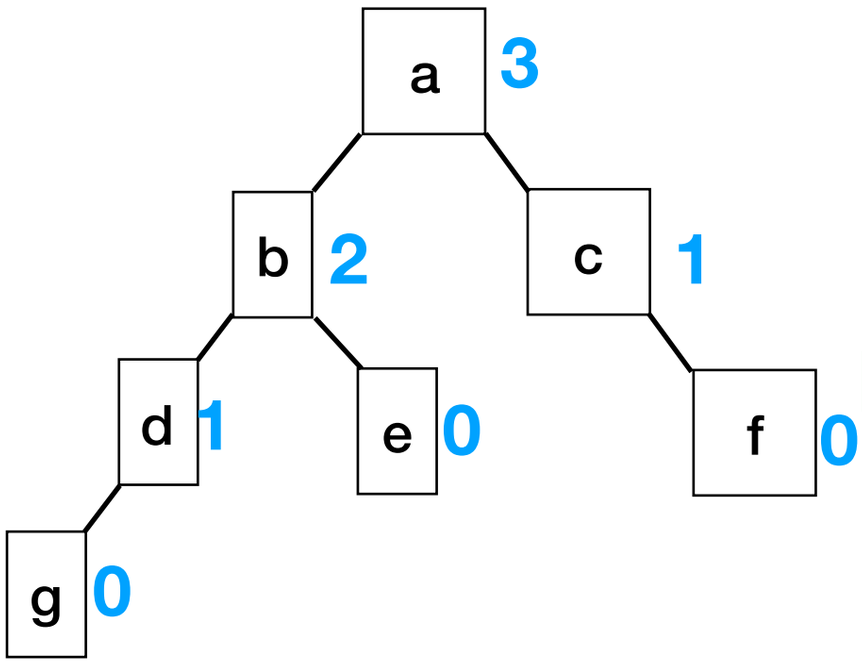
Height(t): path length from t's deepest descendant (leaf) to t's root.

Height(n): height of the subtree rooted at n

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

```
height(null) = -1
height(n) = 1 + max(height(n.left), height(n.right)
```

Quantifying Badness: Balance Factor



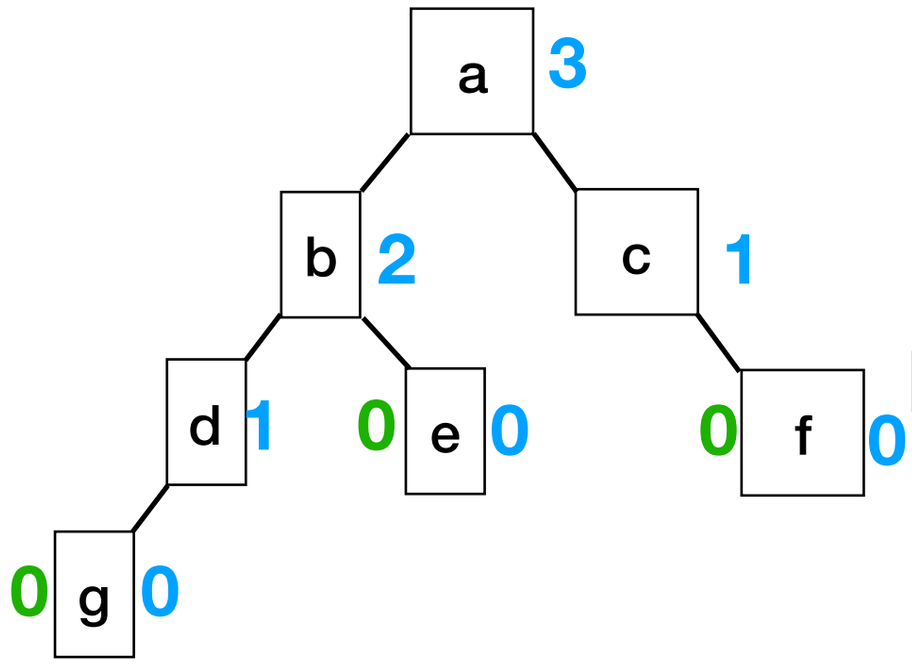
Height(t): path length from t's deepest descendant (leaf) to t's root.

Height(n): height of the subtree rooted at n

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

```
height(null) = -1
height(n) = 1 + max(height(n.left), height(n.right)
```

Quantifying Badness: Balance Factor



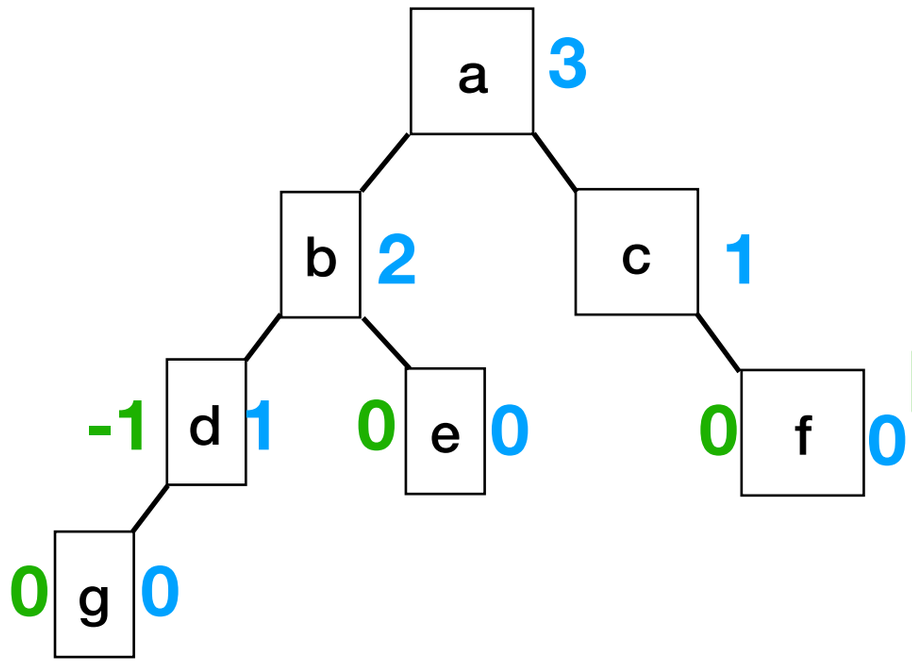
Height(t): path length from t's deepest descendant (leaf) to t's root.

Height(n): height of the subtree rooted at n

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

```
height(null) = -1
height(n) = 1 + max(height(n.left), height(n.right)
```

Quantifying Badness: Balance Factor



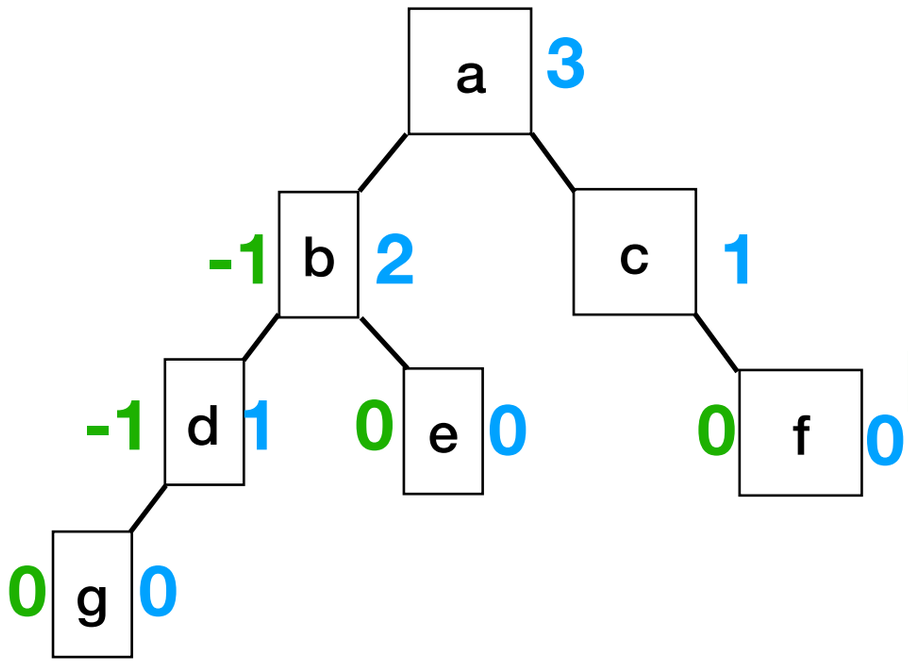
Height(t): path length from t's deepest descendant (leaf) to t's root.

Height(n): height of the subtree rooted at n

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

```
height(null) = -1
height(n) = 1 + max(height(n.left), height(n.right)
```

Quantifying Badness: Balance Factor



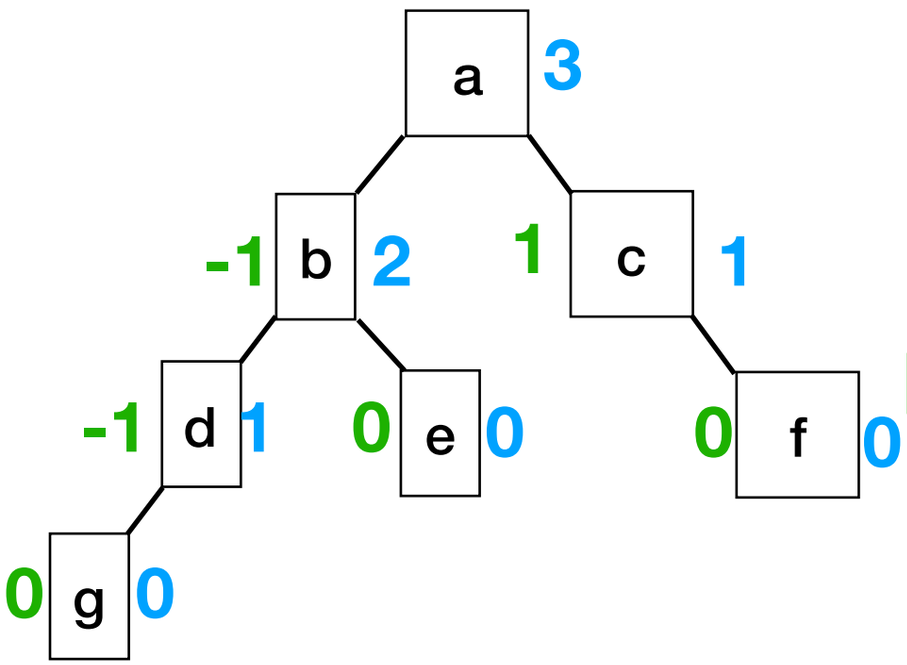
Height(t): path length from t's deepest descendant (leaf) to t's root.

Height(n): height of the subtree rooted at n

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

```
height(null) = -1
height(n) = 1 + max(height(n.left), height(n.right)
```

Quantifying Badness: Balance Factor



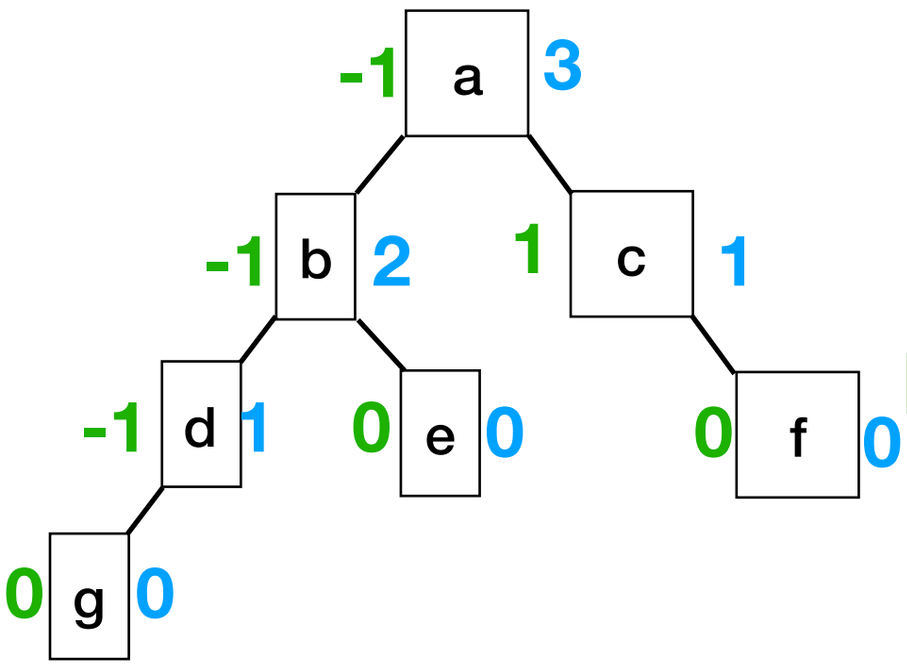
Height(t): path length from t's deepest descendant (leaf) to t's root.

Height(n): height of the subtree rooted at n

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

```
height(null) = -1
height(n) = 1 + max(height(n.left), height(n.right)
```

Quantifying Badness: Balance Factor



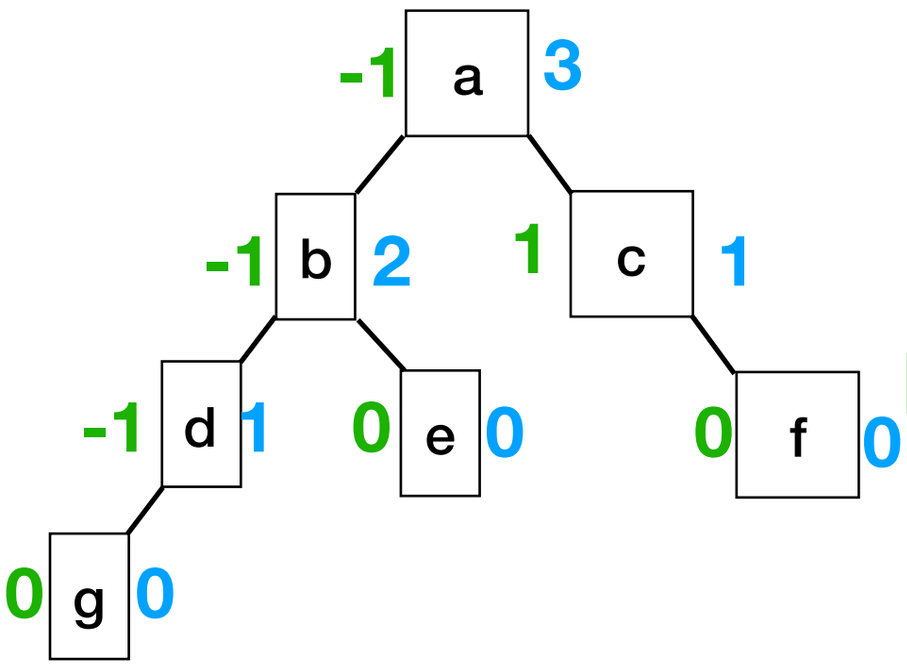
Height(t): path length from t's deepest descendant (leaf) to t's root.

Height(n): height of the subtree rooted at n

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

```
height(null) = -1
height(n) = 1 + max(height(n.left), height(n.right)
```

Quantifying Badness: Balance Factor



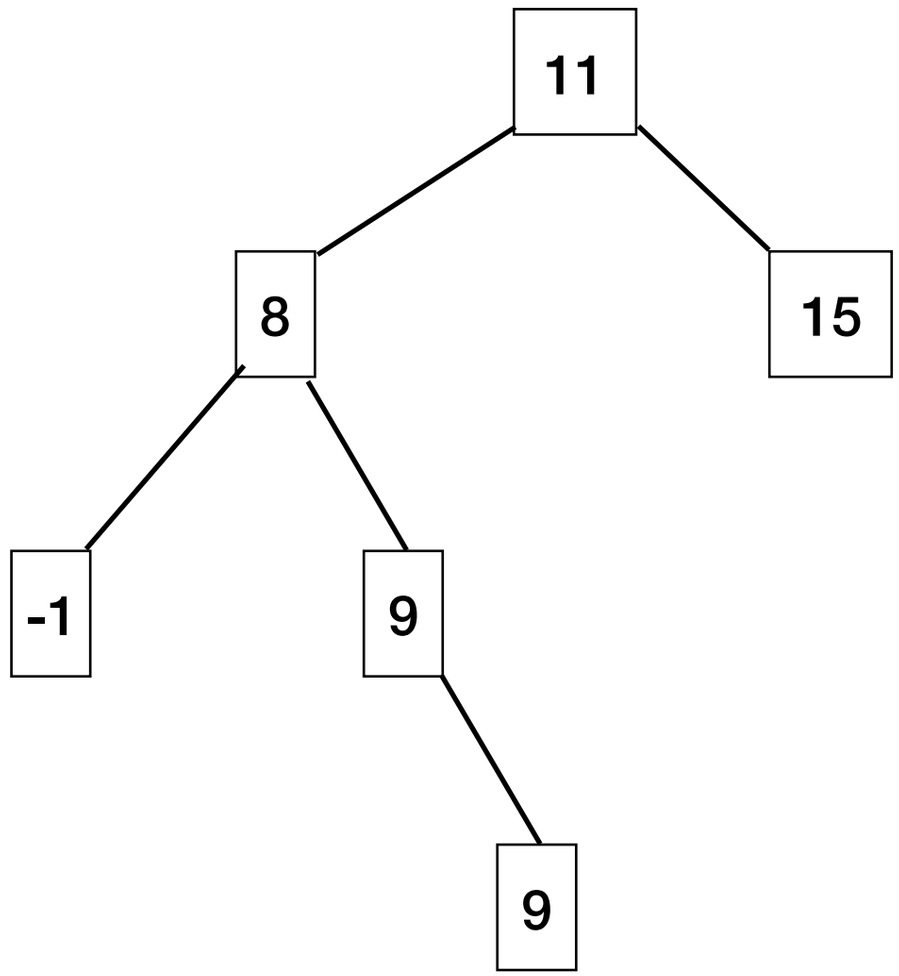
Height(t): path length from t's deepest descendant (leaf) to t's root.

Height(n): height of the subtree rooted at n

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

```
height(null) = -1
height(n) = 1 + max(height(n.left), height(n.right)
```

Balance Factor: Another Example



Balance Factor: Another Example