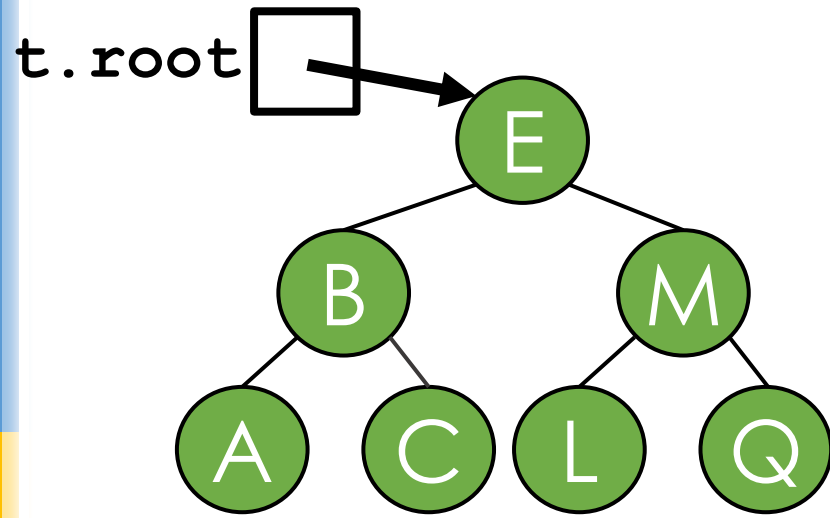# Binary Search Trees: Insert

Let's write it!

# Looking ahead…

- You will implement insert in a different kind of tree in the project, but the ideas are the same
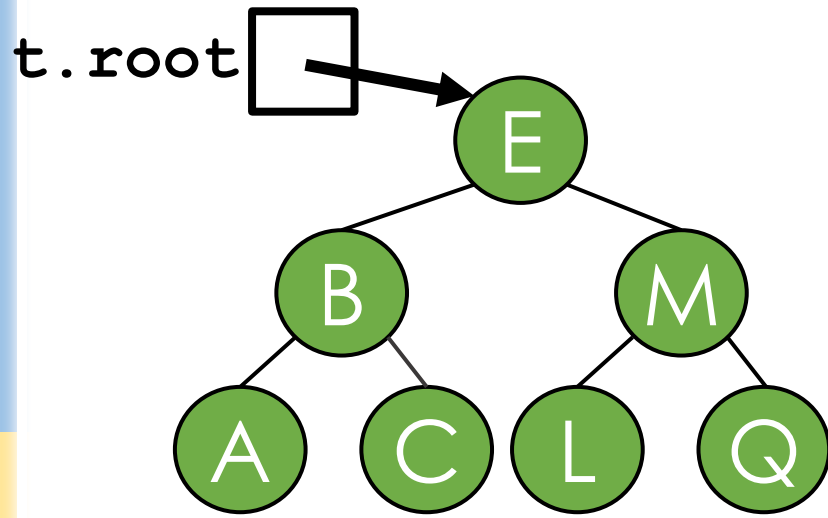
t.root

E

B          M

A     C     L     Q

```
public boolean insert(E toInsert)
{
```

find `toInsert`
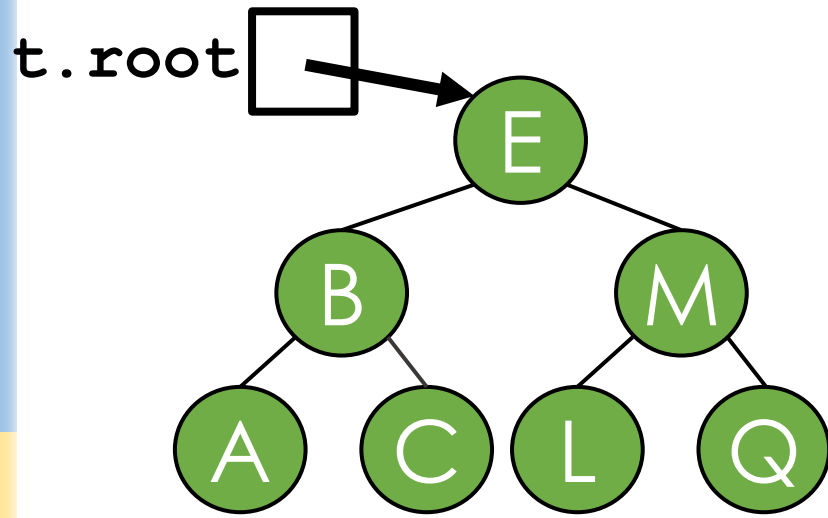
t.insert('D')

if it's not there, add it
where you end up

t.root



t.insert('D')

```java
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp =
        toInsert.compareTo(curr.getData());
    while (curr != null) {
        if (comp < 0)
            curr = curr.getLeft();
        else if (comp > 0)
            curr = curr.getRight();
        else
            return false;
    }

}
```

if it's not there, add it
where you end up

t.root



t.insert('D')

```java
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp =
        toInsert.compareTo(curr.getData());
    while (curr != null) {
        if (comp < 0)
            curr = curr.getLeft();
        else if (comp > 0)
            curr = curr.getRight();
        else
            return false;
    }
    curr = new TreeNode<E>(toInsert, curr);
}
```
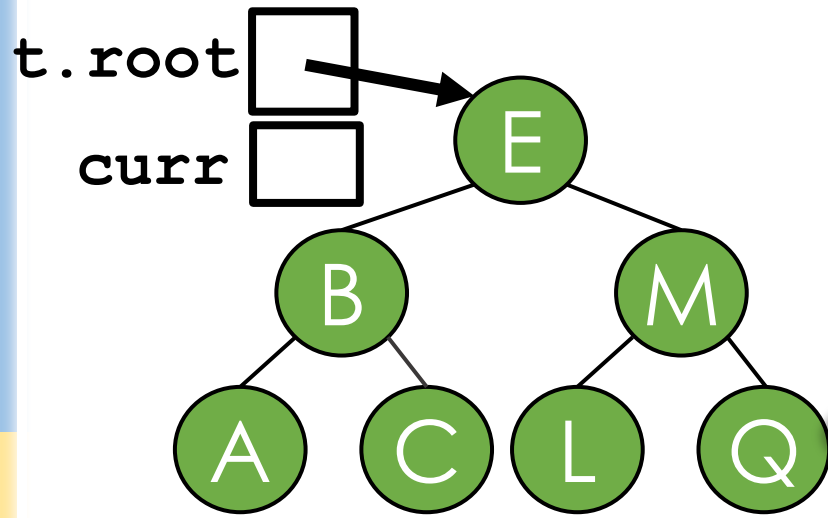
Does this work?

t.root ▢

curr ▢



t.insert('D')

```java
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp =
        toInsert.compareTo(curr.getData());
    while (curr != null) {
        if (comp < 0)
            curr = curr.getLeft();
        else if (comp > 0)
            curr = curr.getRight();
        else
            return false;
    }
    curr = new TreeNode<E>(toInsert, curr);
}
```
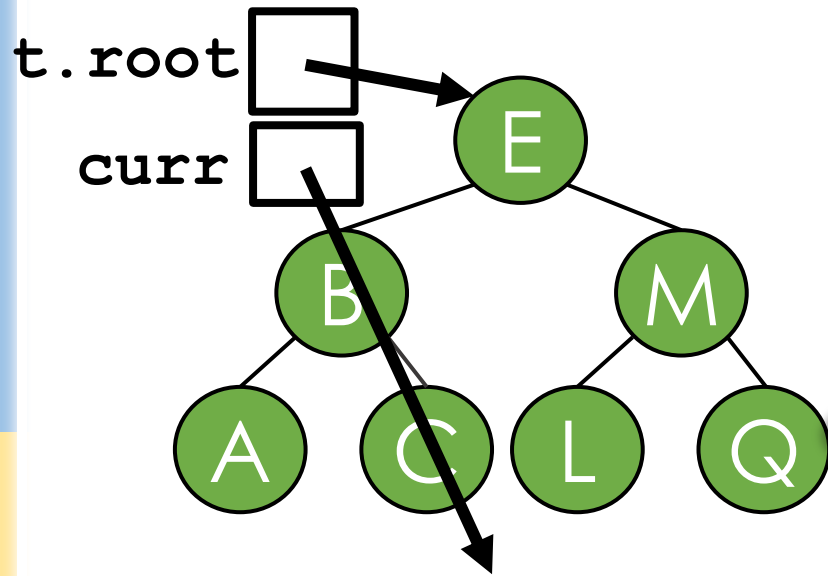
**Does this work? NO!**

```
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp =
        toInsert.compareTo(curr.getData());
    while (curr != null) {
        if (comp < 0)
            curr = curr.getLeft();
        else if (comp > 0)
            curr = curr.getRight();
        else
            return false;
    }
    curr = new TreeNode<E>(toInsert, curr);
}
```
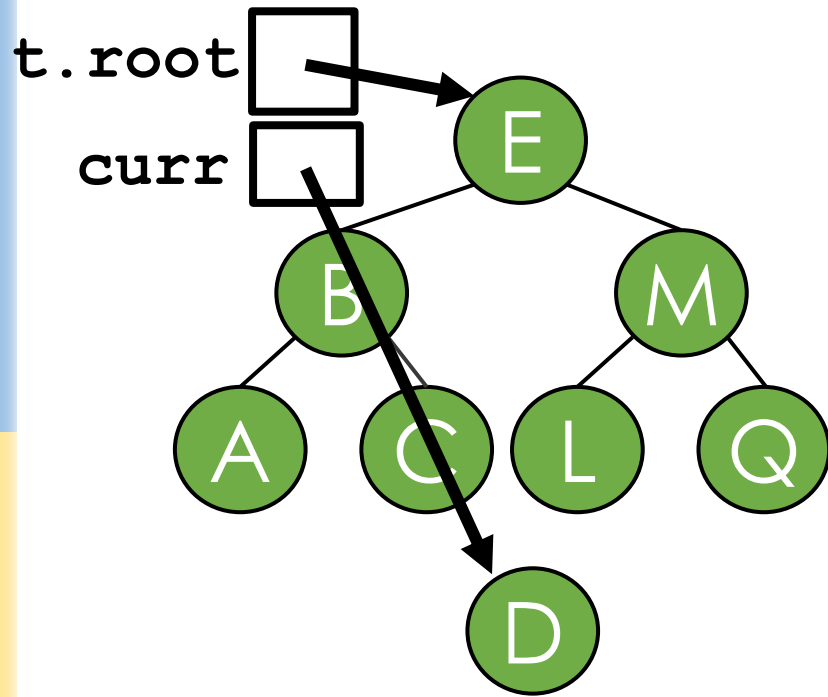
**Does this work? NO!**

t.root

curr



t.insert('D')

```java
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp =
        toInsert.compareTo(curr.getData());
    while (curr != null) {
        if (comp < 0)
            curr = curr.getLeft();
        else if (comp > 0)
            curr = curr.getRight();
        else
            return false;
    }
    curr = new TreeNode<E>(toInsert, curr);
}
```
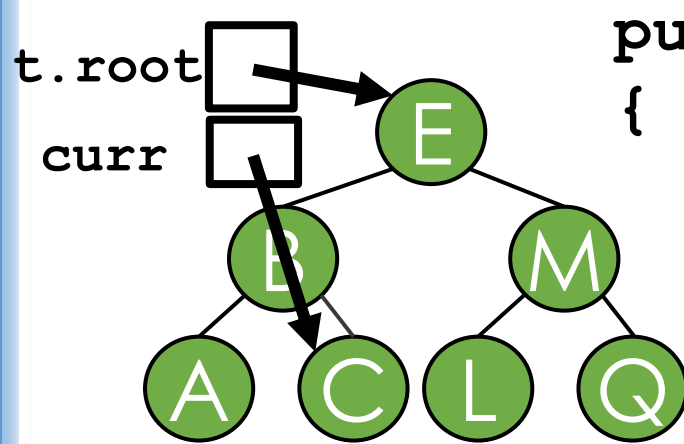
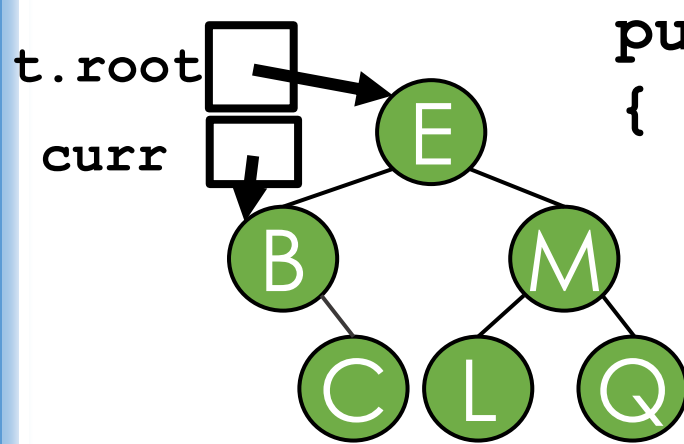**Does this work? NO!**

```
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp = toInsert.compareTo(curr.getData());
```

t.insert('D')

**Need to stop before we fall off the end**

t.root

curr



```
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp = toInsert.compareTo(curr.getData());
```

t.insert('D')

**Need to stop before
we fall off the end**

t.root

curr



```
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp = toInsert.compareTo(curr.getData());
    while (comp < 0 && curr.getLeft() != null ||
            comp > 0 && curr.getRight() != null)
```
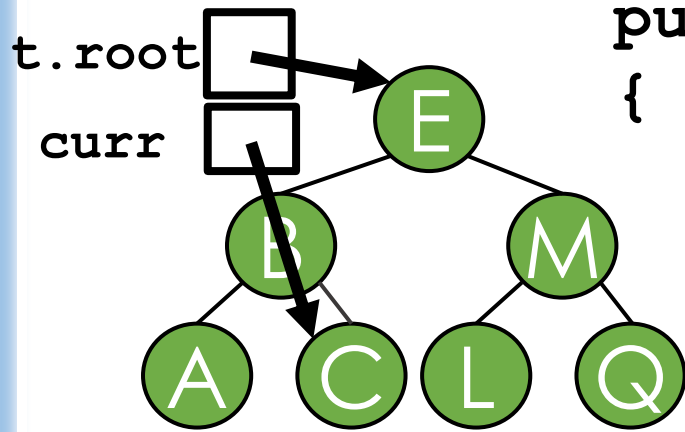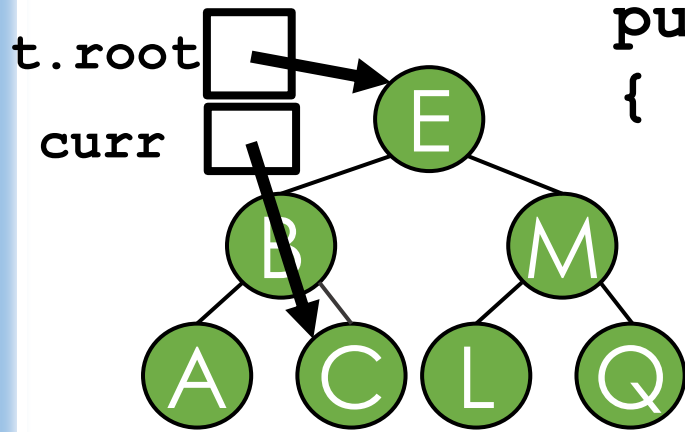
t.insert('D')

**Need to stop before we fall off the end**

```java
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp = toInsert.compareTo(curr.getData());
    while (comp < 0 && curr.getLeft() != null ||
              comp > 0 && curr.getRight() != null)
    {
        if (comp < 0) curr = curr.getLeft();
        else curr = curr.getRight();
```
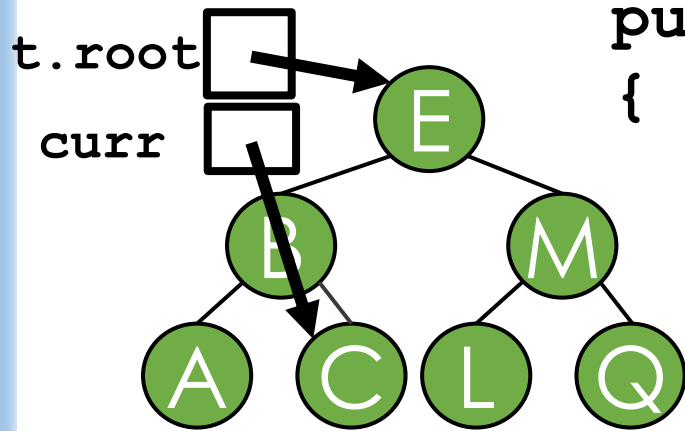
t.insert('D')

**Need to stop before we fall off the end**

```
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp = toInsert.compareTo(curr.getData());
    while (comp < 0 && curr.getLeft() != null ||
             comp > 0 && curr.getRight() != null)
    {
        if (comp < 0) curr = curr.getLeft();
        else curr = curr.getRight();
        comp = toInsert.compareTo(curr.getData());
    }
}
```
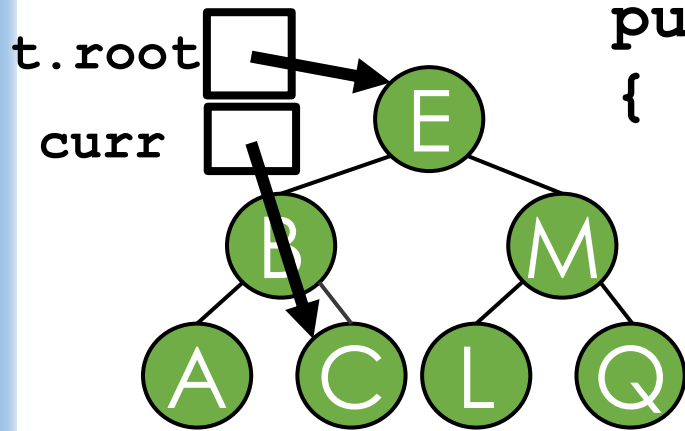
t.insert('D')

**What's next?**

t.root

curr

E

B

M

A

C

L

Q

```
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp = toInsert.compareTo(curr.getData());
    while (comp < 0 && curr.getLeft() != null ||
            comp > 0 && curr.getRight() != null)
    {
        if (comp < 0) curr = curr.getLeft();
        else curr = curr.getRight();
        comp = toInsert.compareTo(curr.getData());
    }
```
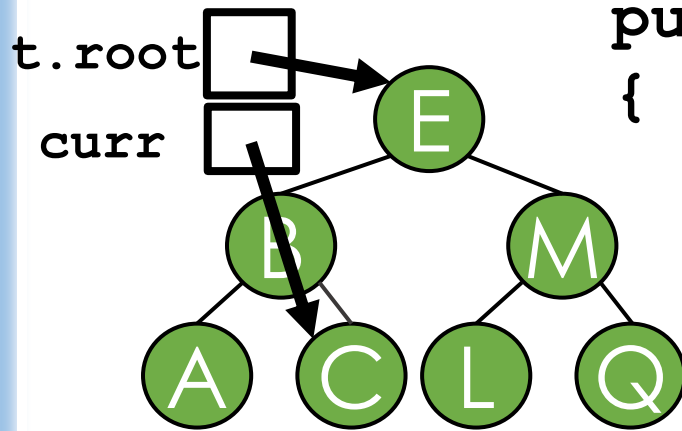
t.insert('D')   }

After the loop either:  (1) curr points to the last node,
or (2) we found the element

```java
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp = toInsert.compareTo(curr.getData());
    while (comp < 0 && curr.getLeft() != null ||
            comp > 0 && curr.getRight() != null)
    {
        if (comp < 0) curr = curr.getLeft();
        else curr = curr.getRight();
        comp = toInsert.compareTo(curr.getData());
    }
    if (comp < 0)
        curr.setLeftChild(toInsert, curr);
    else if (comp > 0)
        curr.setRightChild(toInsert, curr);
```
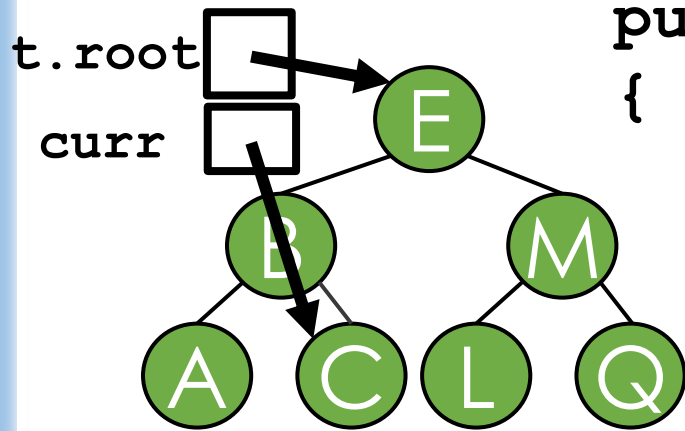
t.insert('D')

**After the loop either:  (1) curr points to the last node,**

```
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp = toInsert.compareTo(curr.getData());
    while (comp < 0 && curr.getLeft() != null ||
            comp > 0 && curr.getRight() != null)
    {
        if (comp < 0) curr = curr.getLeft();
        else curr = curr.getRight();
        comp = toInsert.compareTo(curr.getData());
    }
    if (comp < 0)
        curr.setLeftChild(toInsert, curr);
    else if (comp > 0)
        curr.setRightChild(toInsert, curr);
    else return false;
```
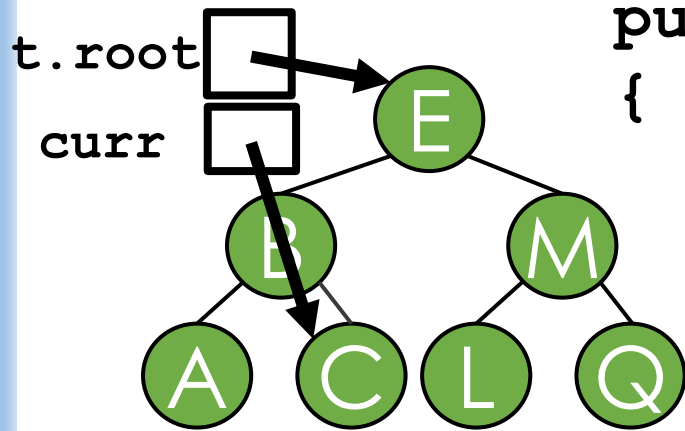
t.root

curr

t.insert('D')

**After the loop either:  (1) curr points to the last node, or (2) we found the element**

```java
public boolean insert(E toInsert)
{
    TreeNode<E> curr = root;
    int comp = toInsert.compareTo(curr.getData());
    while (comp < 0 && curr.getLeft() != null ||
           comp > 0 && curr.getRight() != null)
    {
        if (comp < 0) curr = curr.getLeft();
        else curr = curr.getRight();
        comp = toInsert.compareTo(curr.getData());
    }
    if (comp < 0)
        curr.setLeftChild(toInsert, curr);
    else if (comp > 0)
        curr.setRightChild(toInsert, curr);
    else return false;
    return true;
}
```