

Tree Memory Diagram

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



Stack

Name	Value
------	-------

Heap

- Build a tree
- Visit every node with a post-order traversal

in/out

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}
```

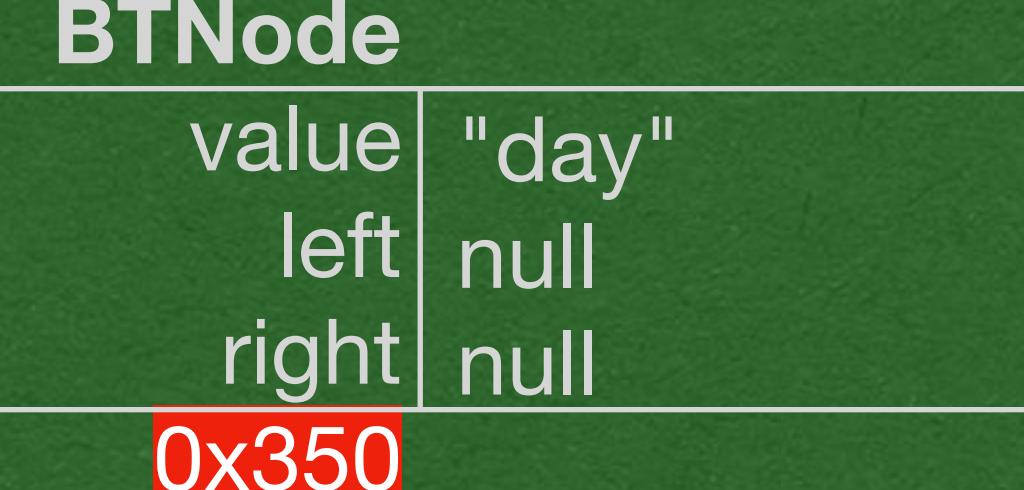
```
def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```

day

in/out

- Create the root node with the value "day"

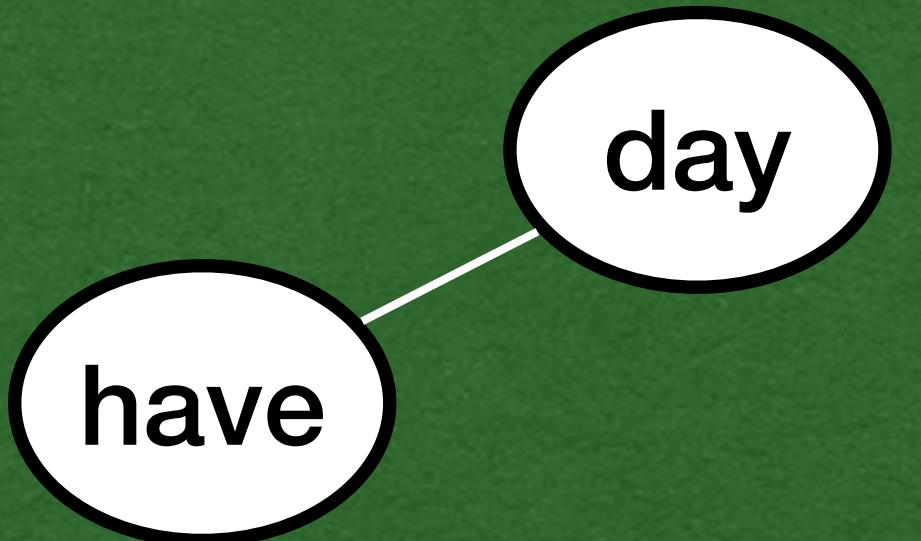
Stack		Heap
Name	Value	
root	0x350	
this	0x350	
value	"day"	
left	null	
right	null	
BTNode		
value	"day"	
left	null	
right	null	
	0x350	



```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out

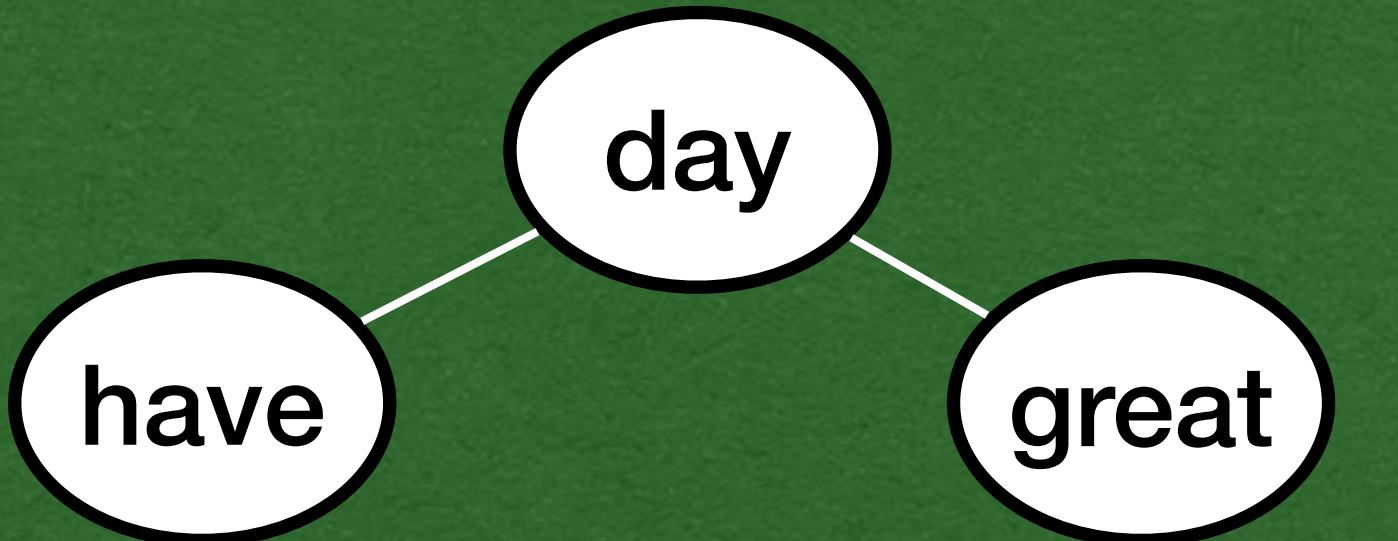
- Set the left child of the root to a node with "have"

Stack		Heap
Name	Value	
root	0x350	BTNode value "day" left null right null
this	0x350	
value	"day"	BTNode value "have" left null right null
left	null	
right	null	BTNode value "have" left null right null
this	0x200	
value	"have"	BTNode value "have" left null right null
left	null	
right	null	BTNode value "have" left null right null
	0x200	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out

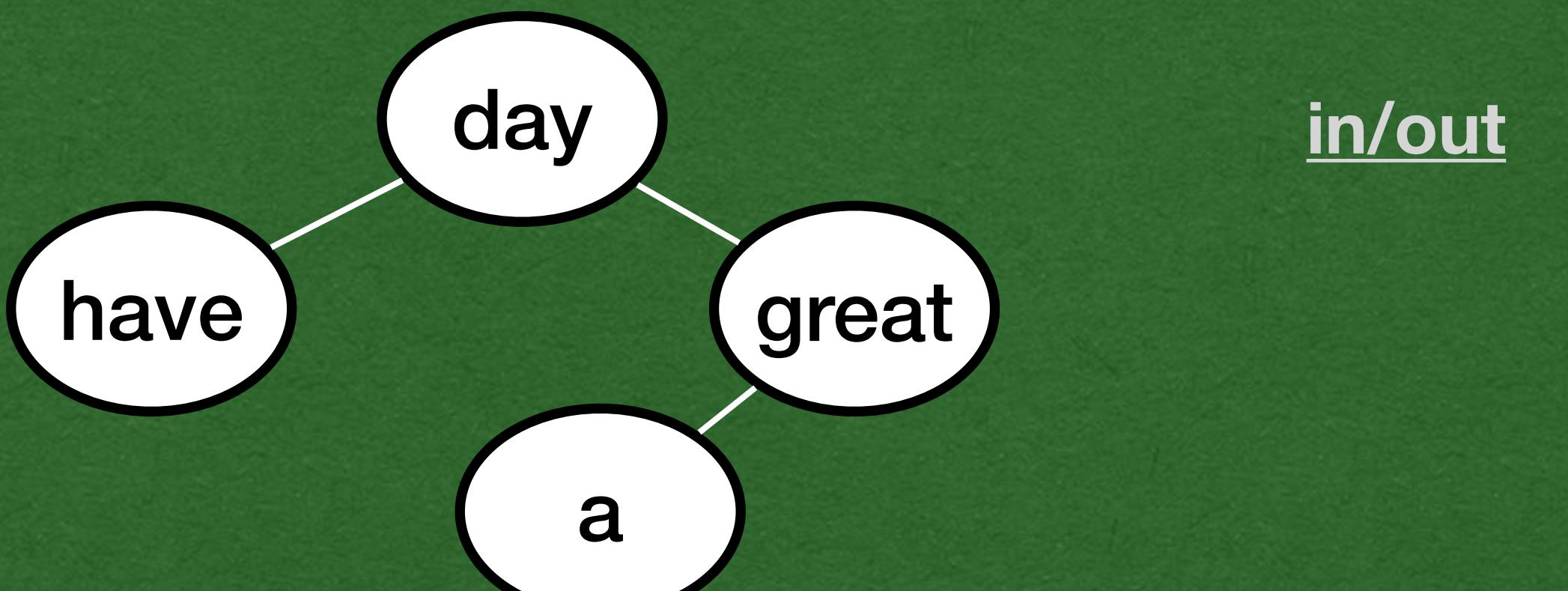
- Set the right child of the root to a node with "great"

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	BTNode
this	0x200	value "have"
value	"have"	left null
left	null	right null
right	null	BTNode
this	0x480	value "great"
value	"great"	left null
left	null	right null
right	null	0x200

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



- Set the left child of the right child of the root to a node with "a"

Stack

Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null

Heap

BTNode
value "day"
left null 0x200
right null 0x480

BTNode
value "have"
left null
right null

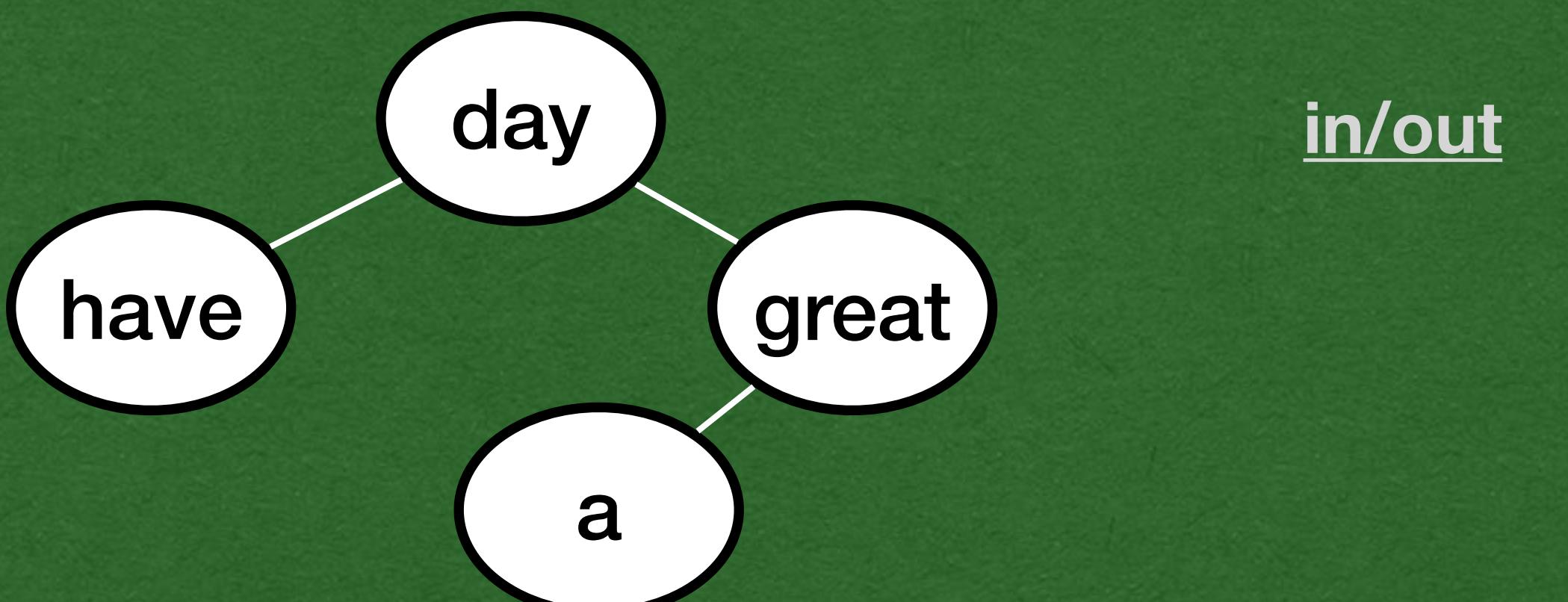
BTNode
value "great"
left null 0x936
right null

BTNode
value "a"
left null
right null

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



- Time to start the post-order traversal

Stack

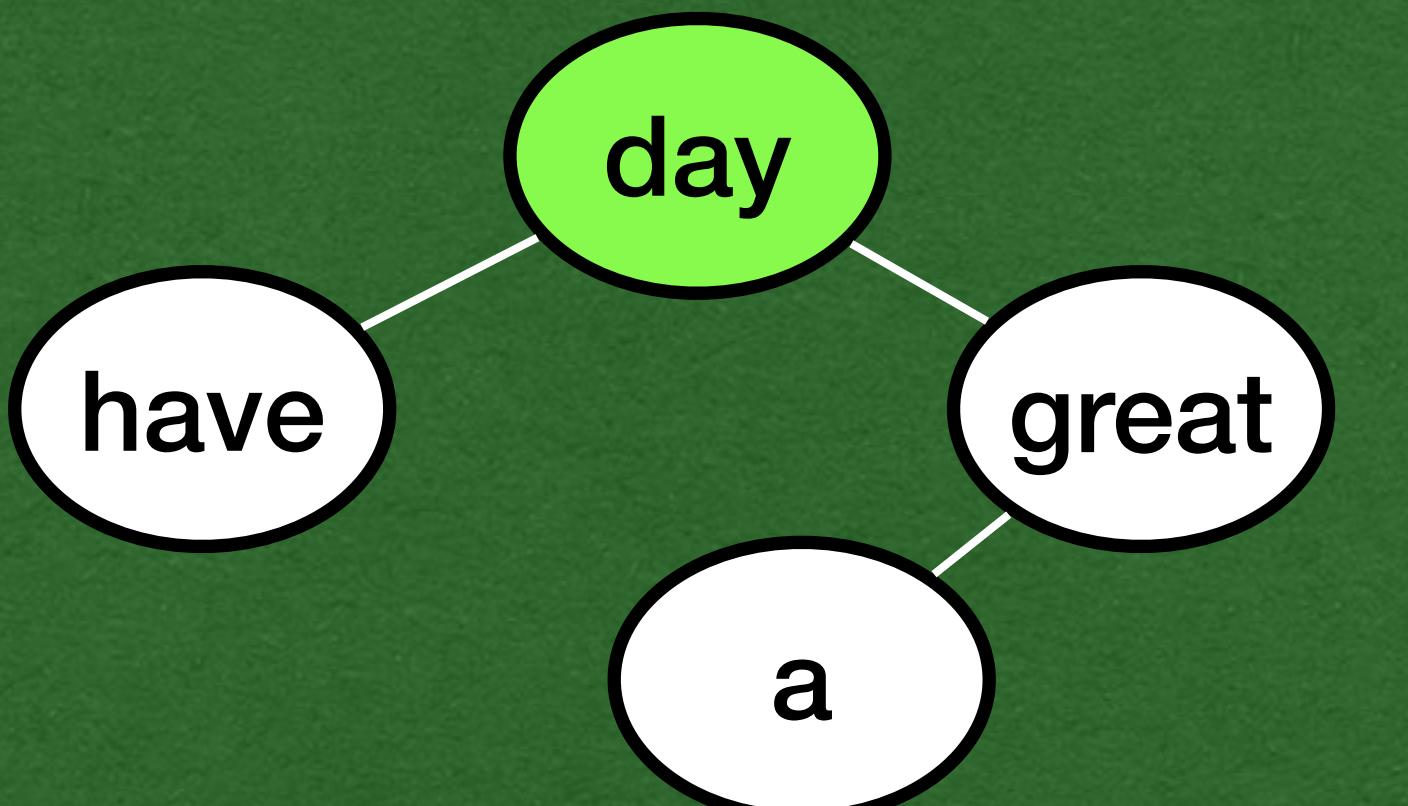
Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null

Heap

BTNode
value "day"
left null 0x200
right null 0x480 0x350
BTNode
value "have"
left null
right null
0x200
BTNode
value "great"
left null 0x936
right null 0x480
BTNode
value "a"
left null
right null
0x936

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {  
    if (node != null) {  
        traversal(node.left)  
        traversal(node.right)  
        print(node.value + " ")  
    }  
}  
  
def main(args: Array[String]): Unit = {  
    val root = new BTNode("day", null, null)  
    root.left = new BTNode("have", null, null)  
    root.right = new BTNode("great", null, null)  
    root.right.left = new BTNode("a", null, null)  
    traversal(root)  
}
```



in/out

- Initial call of traversal is on the root node (0x350)

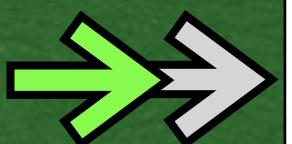
Stack

Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
traversal	
node	0x350

Heap

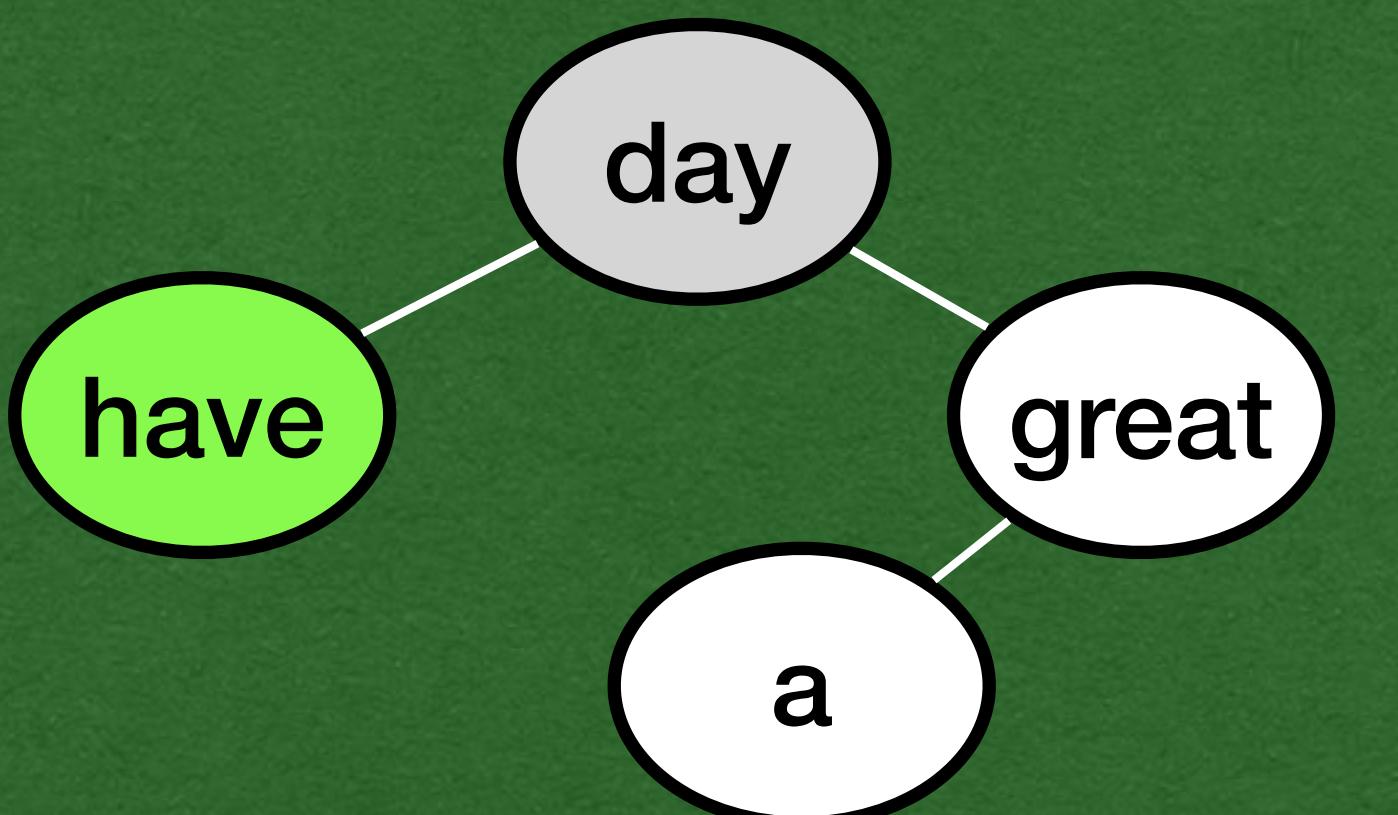
BTNode	
value	"day"
left	null 0x200
right	null 0x480
0x350	
BTNode	
value	"have"
left	null
right	null
0x200	
BTNode	
value	"great"
left	null
right	null
0x480	
BTNode	
value	"a"
left	null 0x936
right	null
0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```



```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out

- Make a recursive call on the left child first

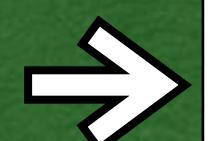
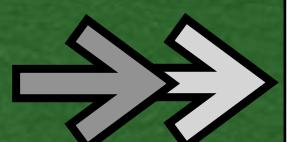
Stack

Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200

Heap

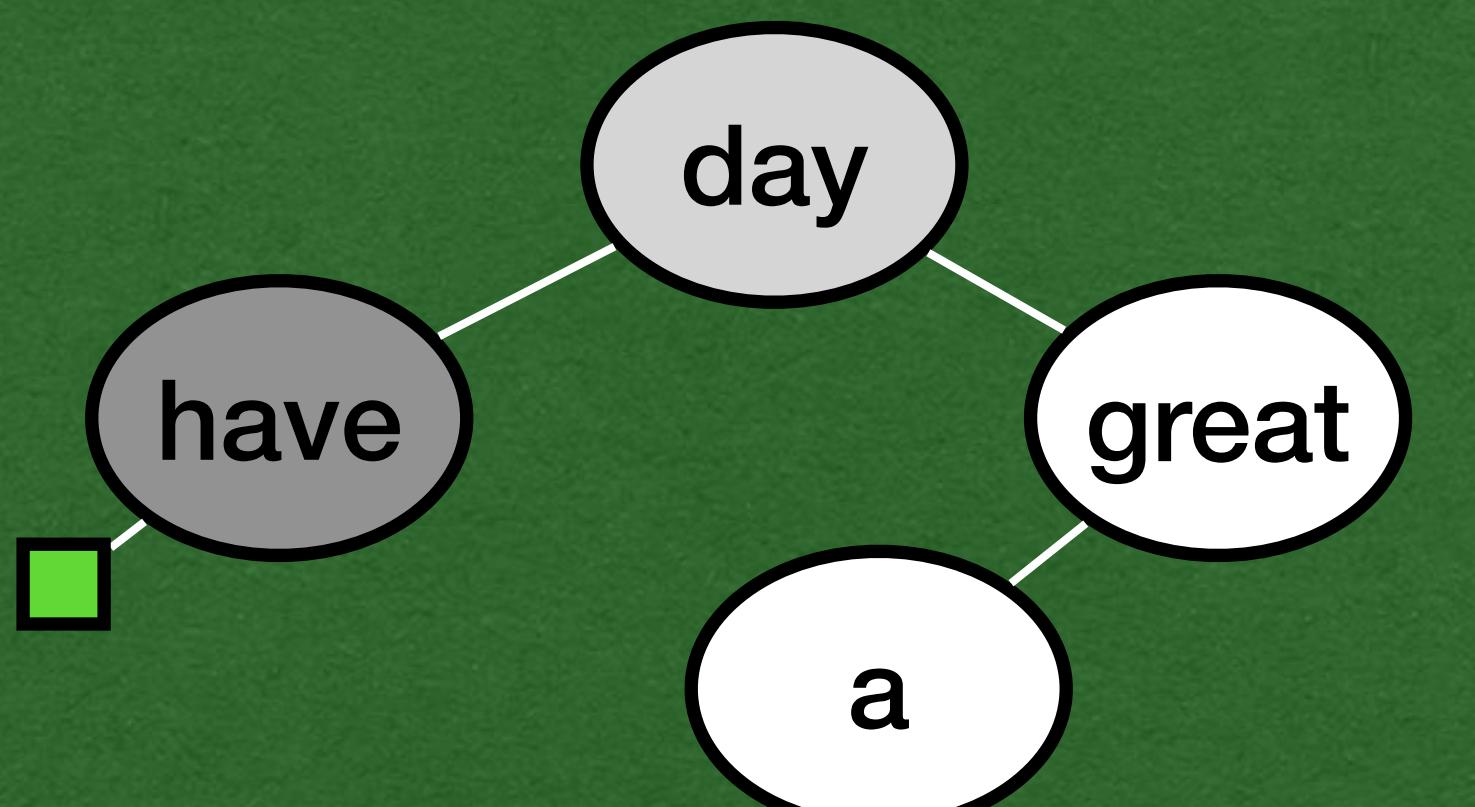
BTNode
value "day"
left null 0x200
right null 0x480 0x350
BTNode
value "have"
left null
right null
0x200
BTNode
value "great"
left null 0x936
right null 0x480 0x200
BTNode
value "a"
left null
right null
0x936

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```



```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out

- Left node makes another recursive call on its left child

Stack

Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null

Heap

BTNode	
value	"day"
left	null 0x200
right	null 0x480

BTNode	
value	"have"
left	null
right	null

BTNode	
value	"great"
left	null 0x936
right	null

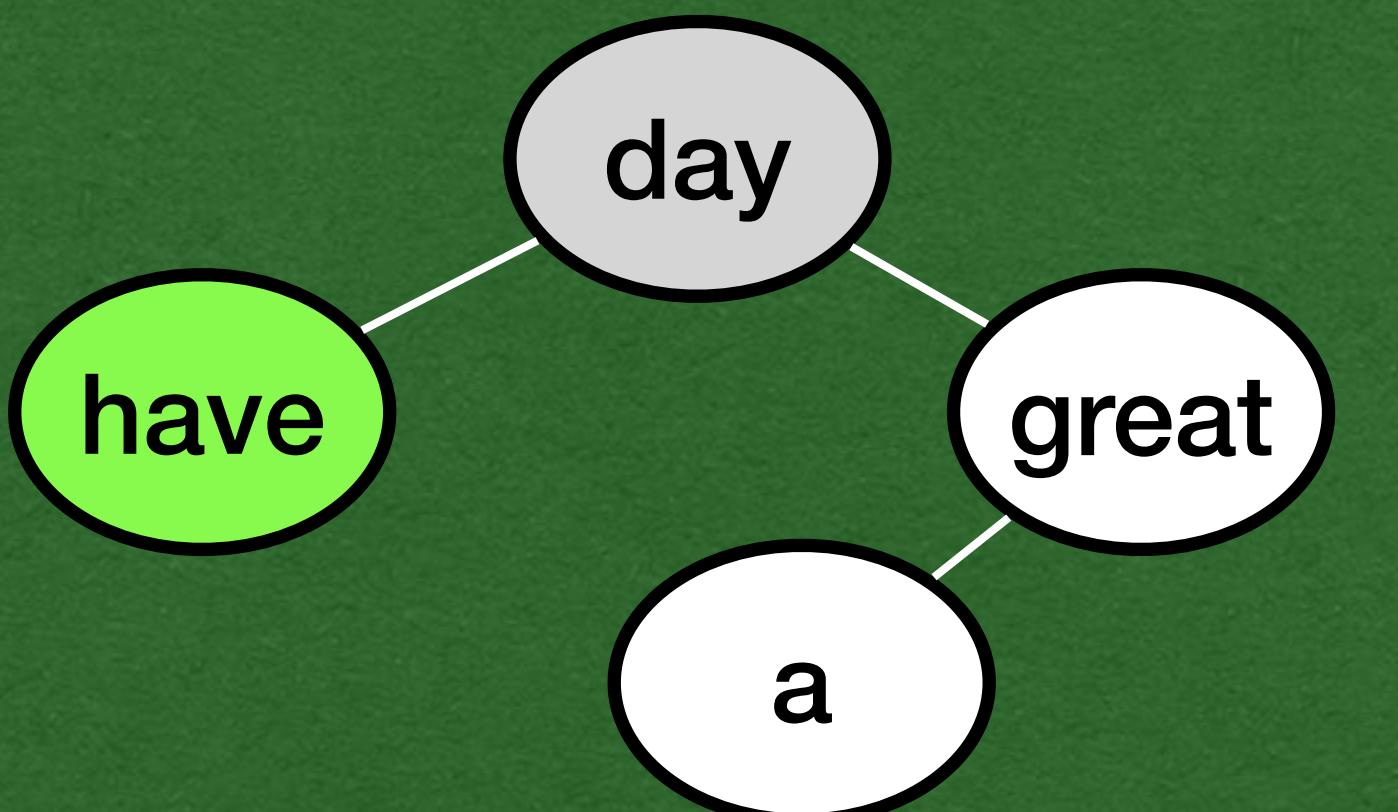
BTNode	
value	"a"
left	null
right	null

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```



```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out

- A null node is our base case
- Do nothing and return to the previous frame on the stack

Stack

Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null

Heap

BTNode	
value	"day"
left	null 0x200
right	null 0x480

BTNode	
value	"have"
left	null
right	null

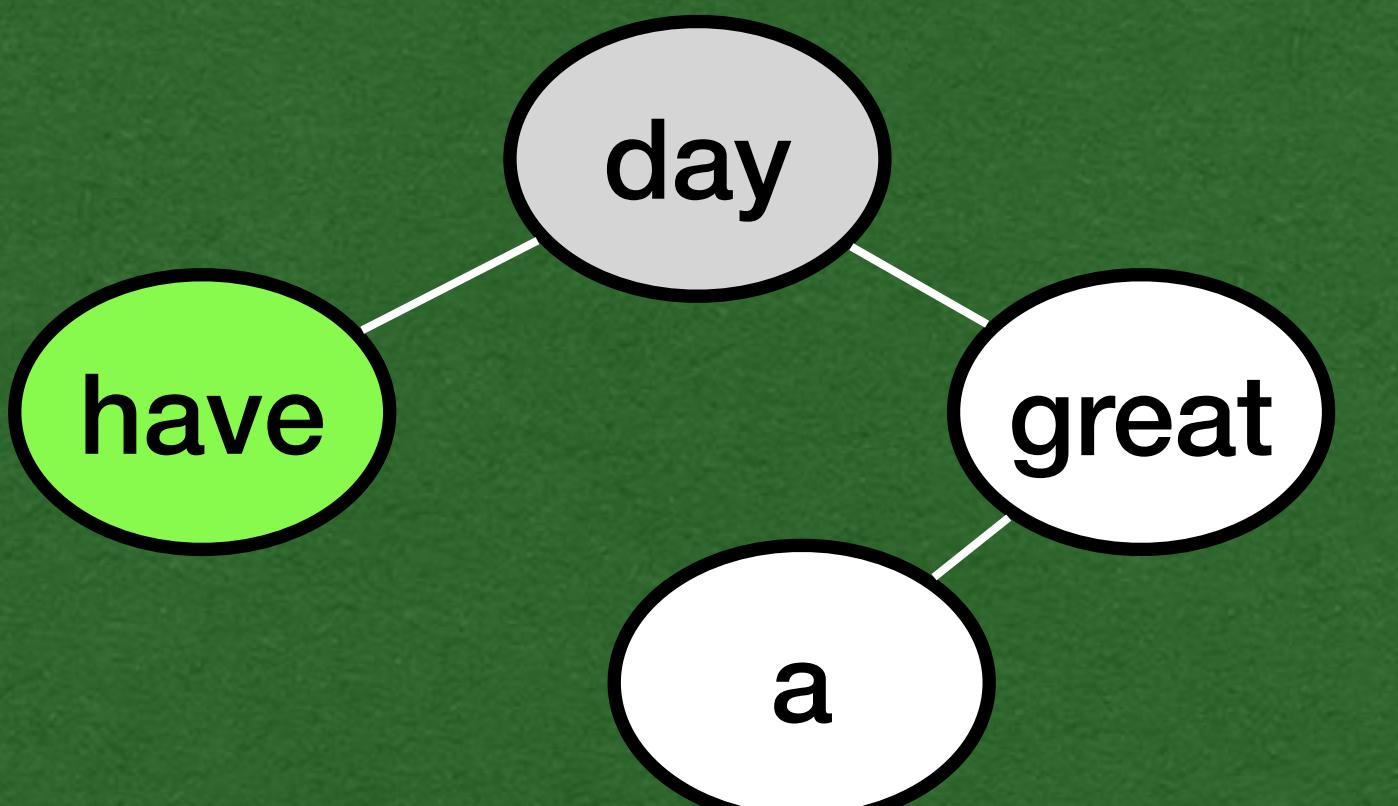
BTNode	
value	"great"
left	null 0x936
right	null

BTNode	
value	"a"
left	null
right	null

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out

- This frame "remembers" that it made a left recursive call and needs to make the right recursive call

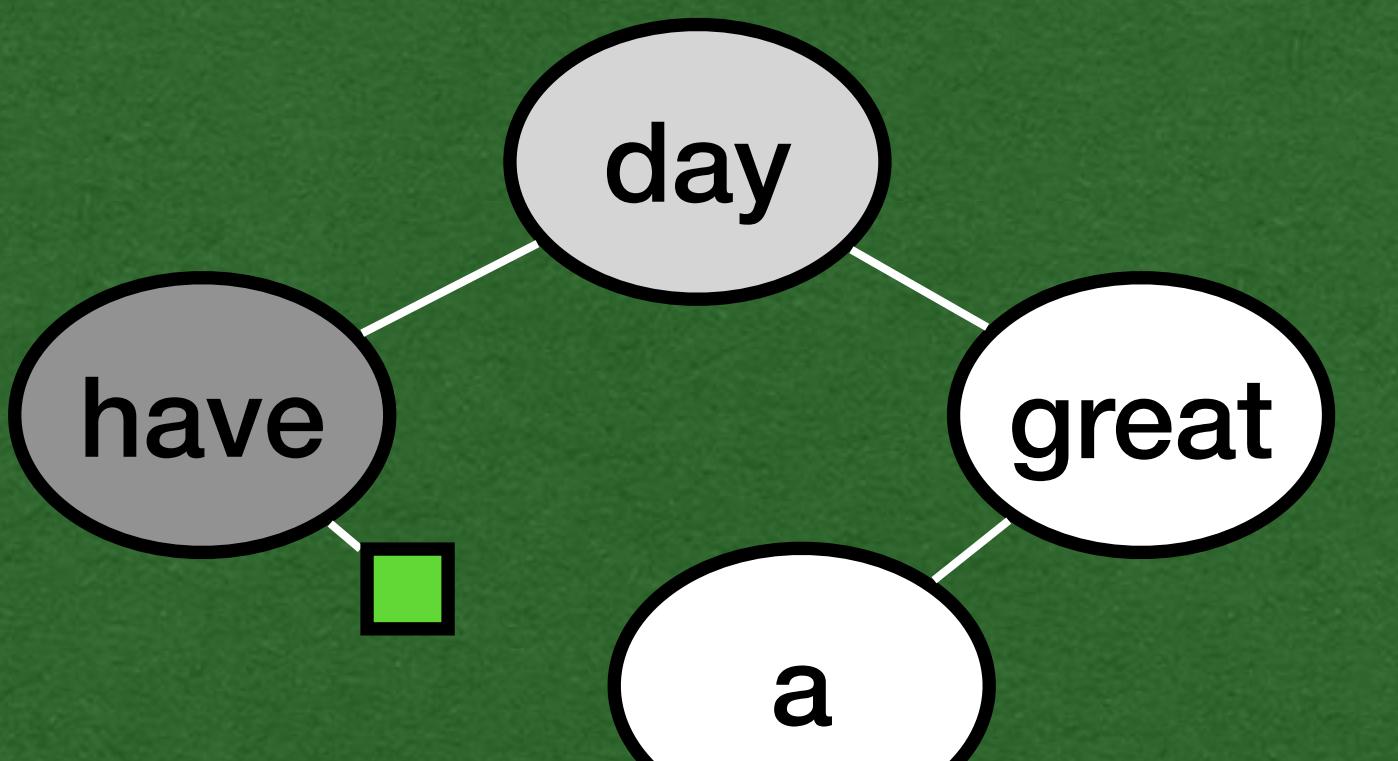
Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	0x200
node	0x200	0x480
node	null	0x936

BTNode	value "a"	0x936
value	"a"	0x936
left	null	0x936
right	null	0x936

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out

- Right recursive call is also for a null node

Stack

Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null
node	null

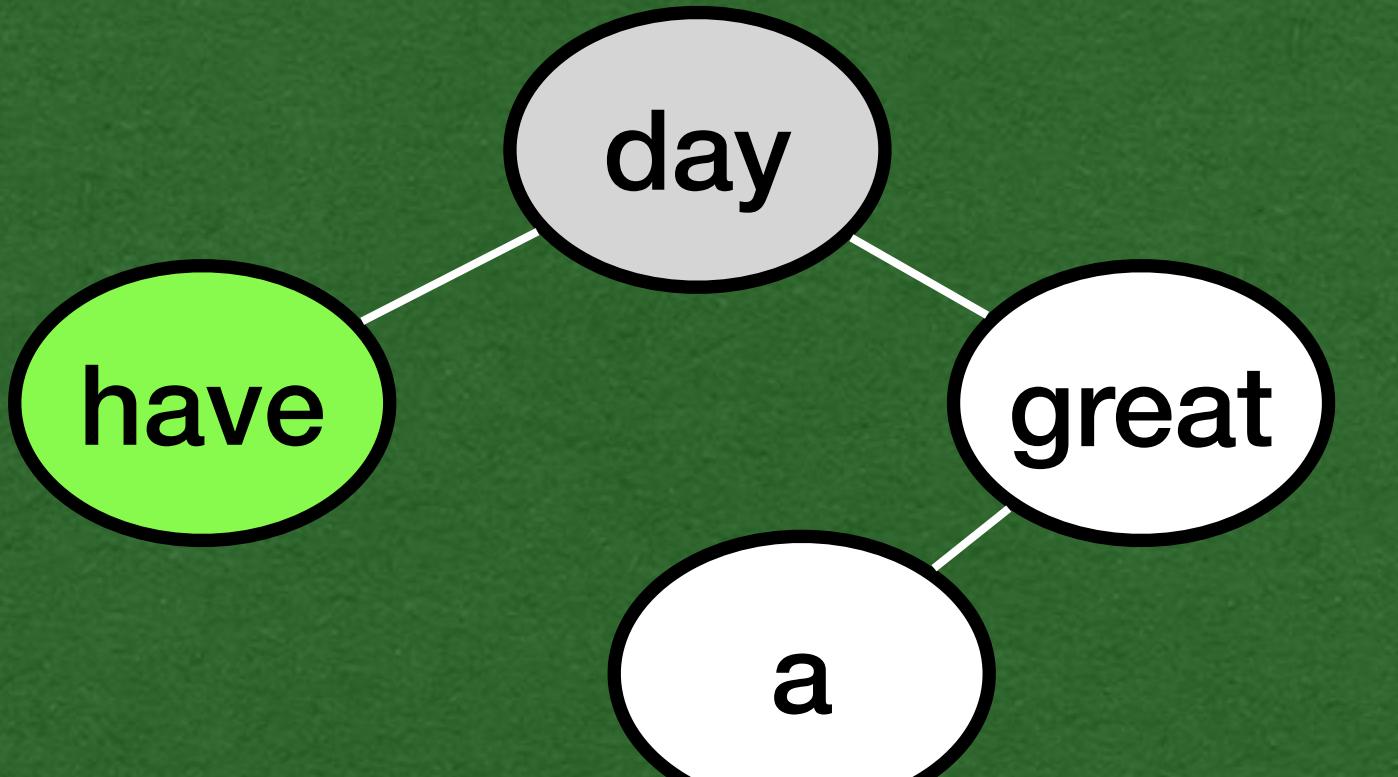
Heap

BTNode	
value	"day"
left	null 0x200
right	null 0x480
0x350	
BTNode	
value	"have"
left	null
right	null
0x200	
BTNode	
value	"have"
left	null
right	null
0x480	
BTNode	
value	"great"
left	null 0x936
right	null
0x936	
BTNode	
value	"a"
left	null
right	null
0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out

Stack	
Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null
node	null

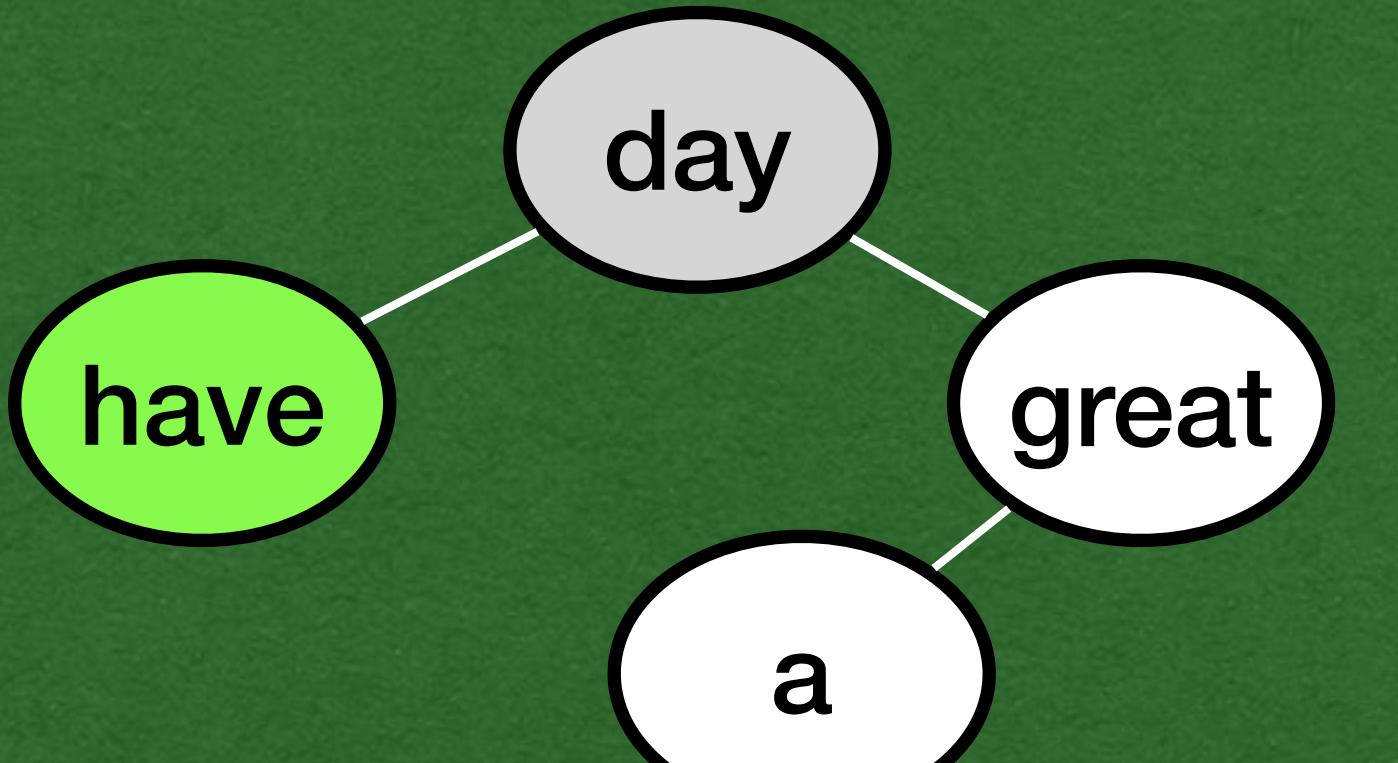
Heap	
BTNode	
value	"day"
left	null 0x200
right	null 0x480
0x350	
BTNode	
value	"have"
left	null
right	null
0x200	
BTNode	
value	"great"
left	null
right	null
0x480	
BTNode	
value	"a"
left	null
right	null
0x936	
traversal	
traversal	
traversal	
traversal	

- Return to the previous frame again
- "Remembers" that it made the right recursive call

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out
have

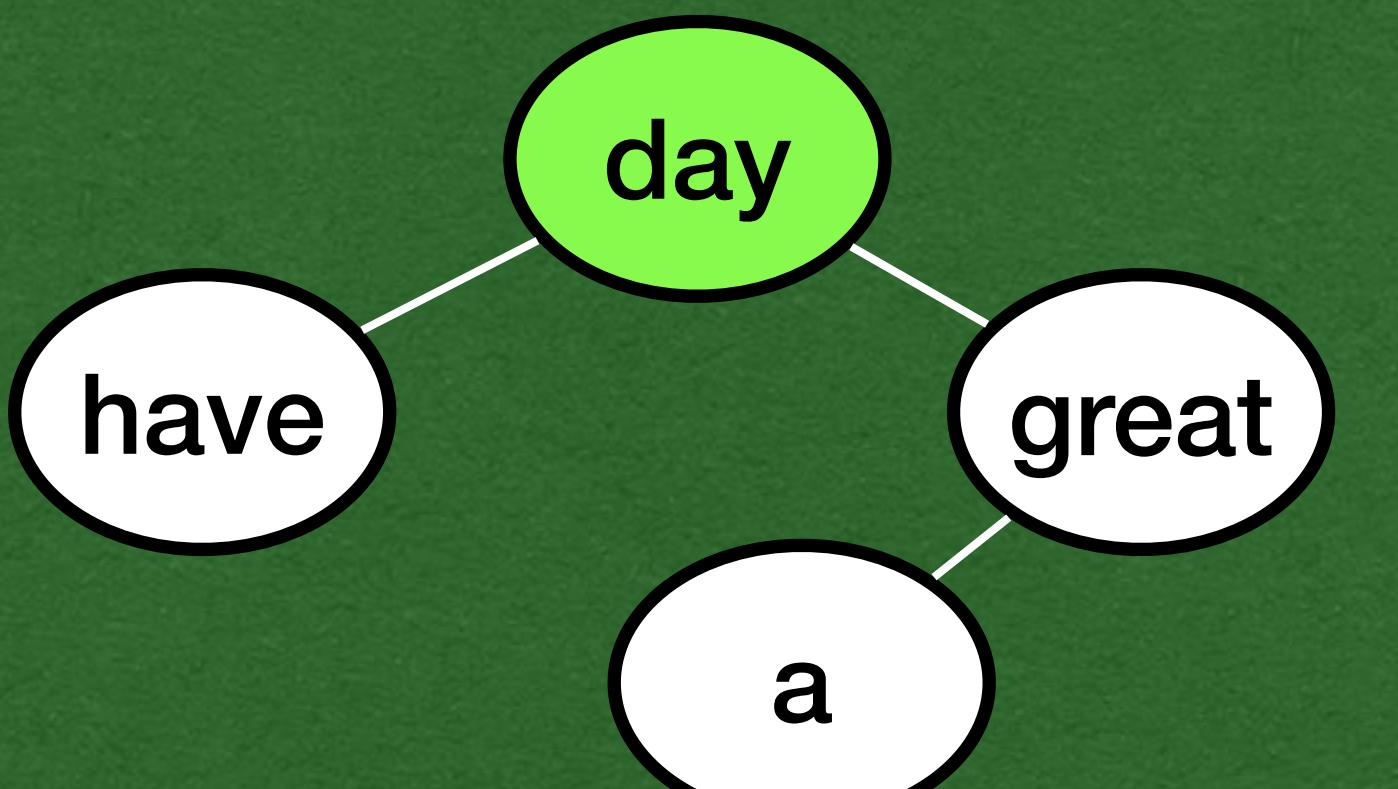
- Print "have" to the screen
- This stack frame is done and returns

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	0x200
node	0x200	0x480
node	null	
node	null	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out
have

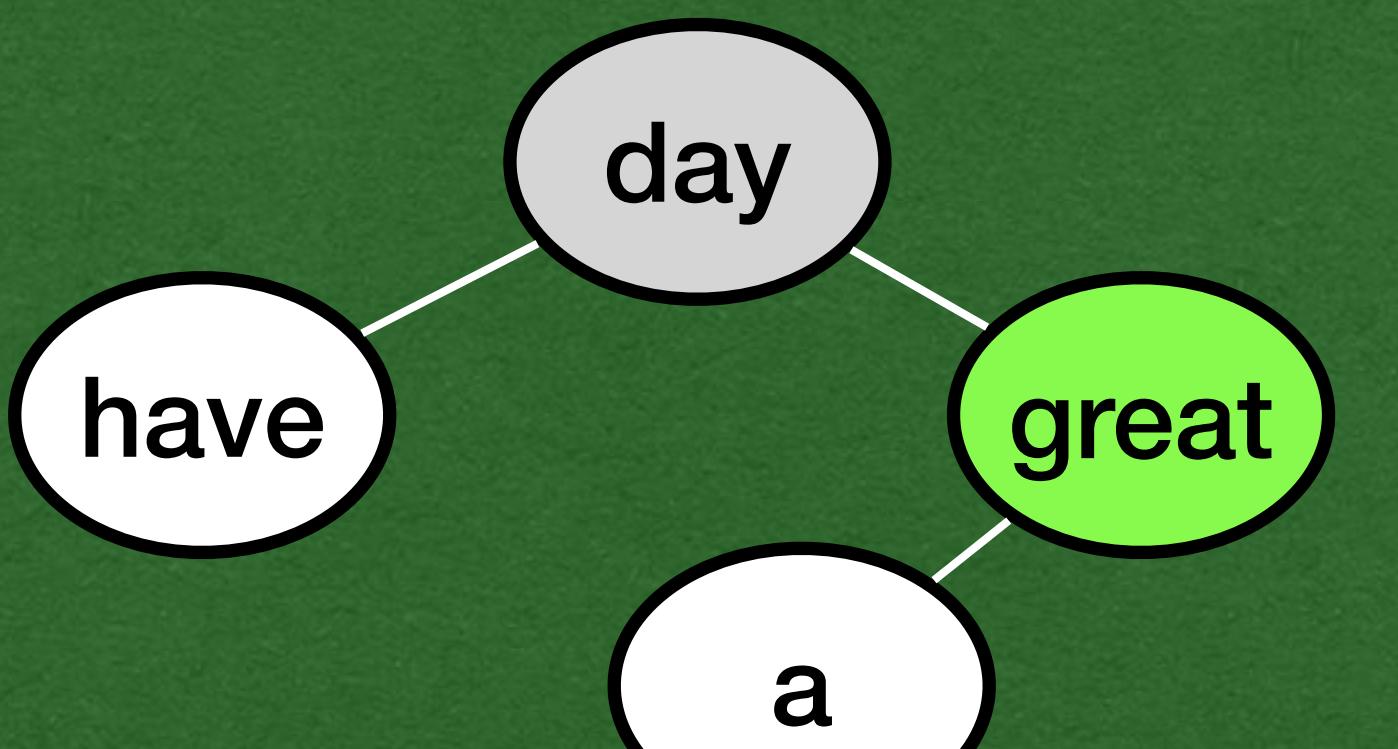
- First recursive call finally regains control (is at the top of the stack)

Stack		Heap
Name	Value	
root	0x350	BTNode value "day" left null right null
this	0x350	BTNode value "have" left null right null
value	"day"	BTNode value "have" left null right null
left	null	BTNode value "have" left null right null
right	null	BTNode value "have" left null right null
this	0x200	BTNode value "great" left null right null
value	"have"	BTNode value "great" left null right null
left	null	BTNode value "great" left null right null
right	null	BTNode value "great" left null right null
this	0x480	BTNode value "a" left null right null
value	"great"	BTNode value "a" left null right null
left	null	BTNode value "a" left null right null
right	null	BTNode value "a" left null right null
traversal	0x936	BTNode value "a" left null right null
node	0x350	BTNode value "a" left null right null
node	0x200	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```

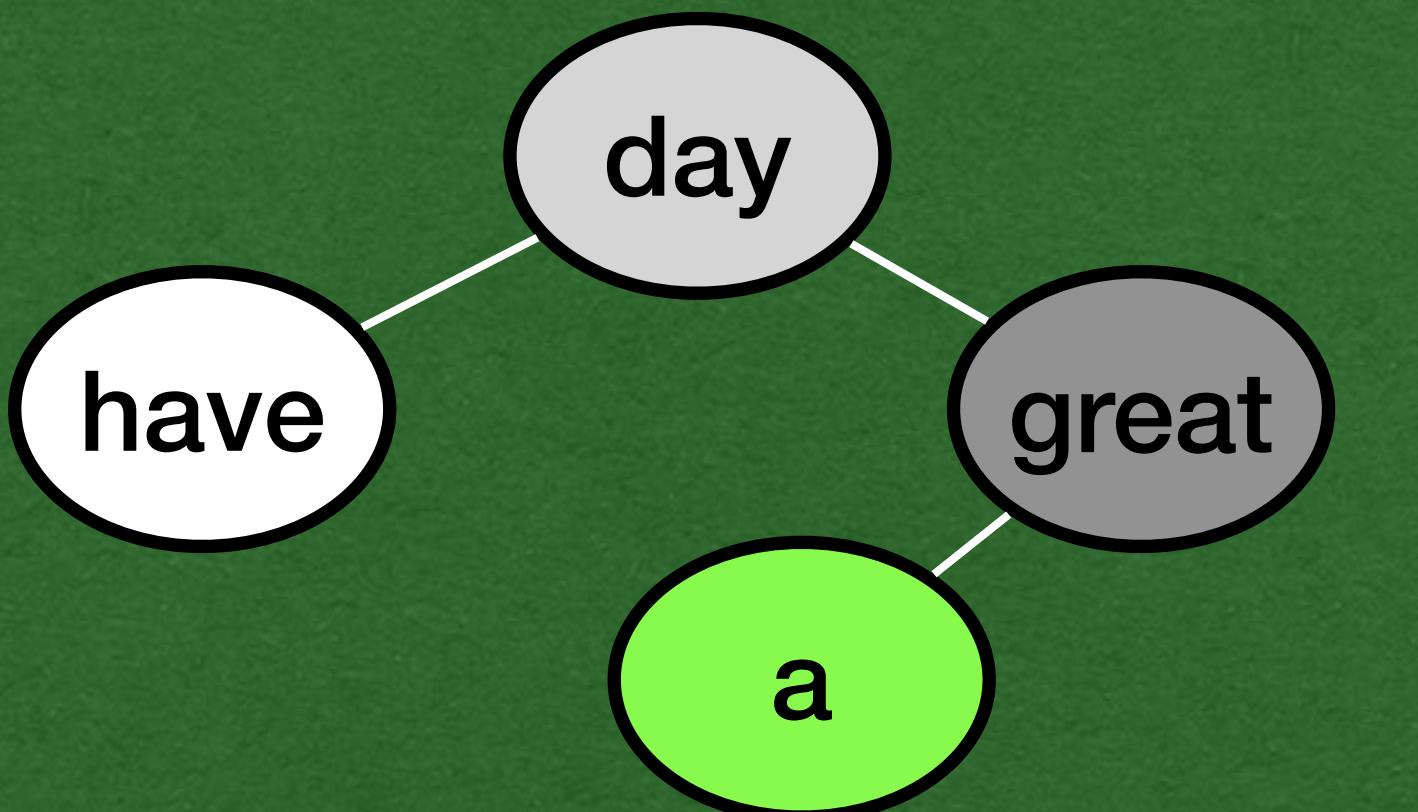


- Make a recursive call to the right of the root

Stack		Heap
Name	Value	
root	0x350	BTNode value "day" left null right null
this	0x350	BTNode value "have" left null right null
value	"day"	BTNode value "have" left null right null
left	null	BTNode value "have" left null right null
right	null	BTNode value "have" left null right null
this	0x200	BTNode value "great" left null right null
value	"have"	BTNode value "great" left null right null
left	null	BTNode value "great" left null right null
right	null	BTNode value "great" left null right null
this	0x480	BTNode value "a" left null right null
value	"great"	BTNode value "a" left null right null
left	null	BTNode value "a" left null right null
right	null	BTNode value "a" left null right null
traversal	0x936	BTNode value "a" left null right null
node	0x350	BTNode value "a" left null right null
node	0x200	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null
node	0x480	BTNode value "a" left null right null

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {  
    if (node != null) {  
        traversal(node.left)  
        traversal(node.right)  
        print(node.value + " ")  
    }  
}  
  
def main(args: Array[String]): Unit = {  
    val root = new BTNode("day", null, null)  
    root.left = new BTNode("have", null, null)  
    root.right = new BTNode("great", null, null)  
    root.right.left = new BTNode("a", null, null)  
    traversal(root)  
}
```



- Recursive call to the left

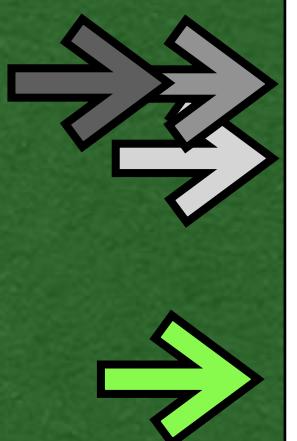
Stack

Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936

Heap

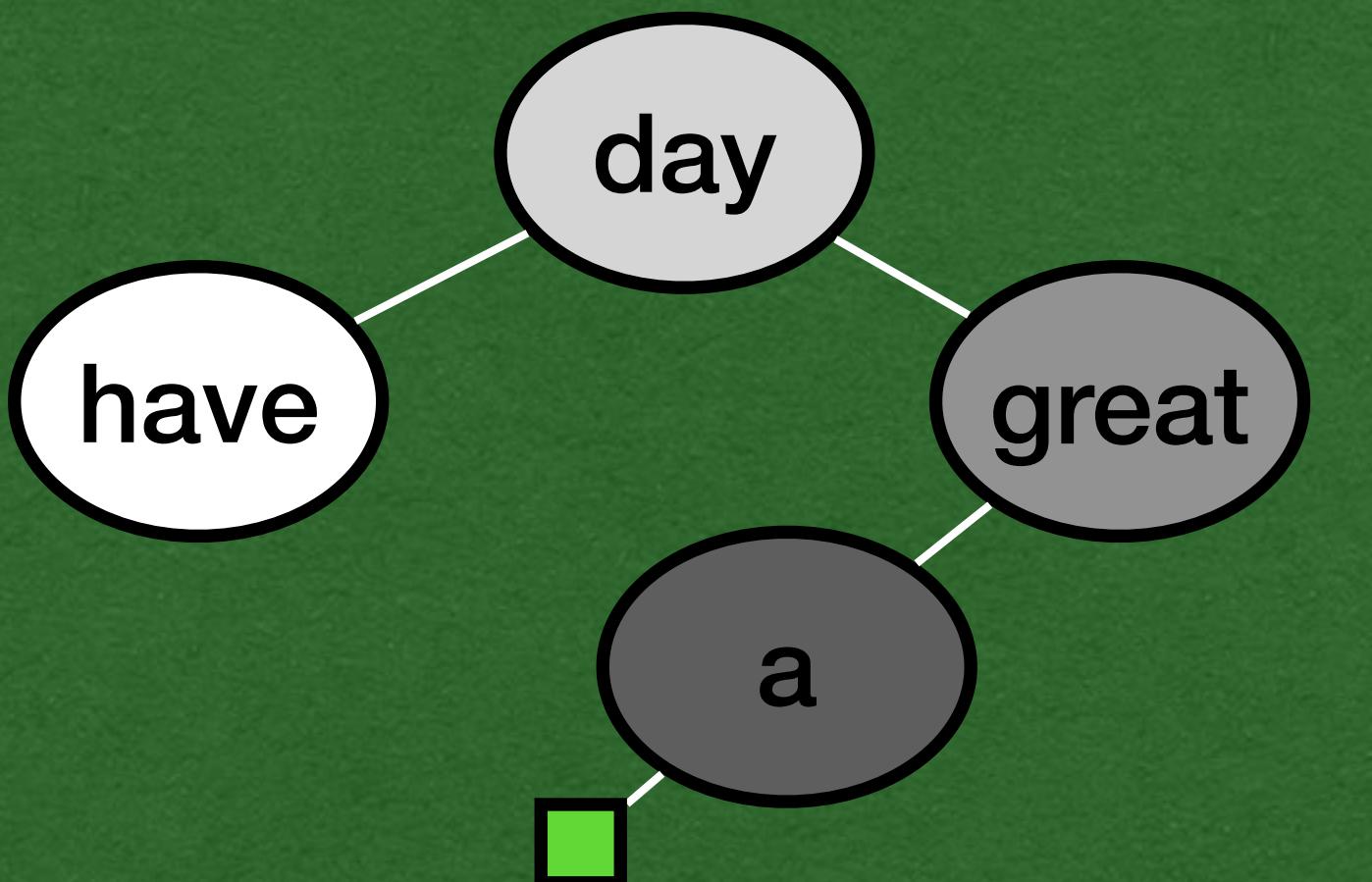
BTNode	
value	"day"
left	null 0x200
right	null 0x480
0x350	
BTNode	
value	"have"
left	null
right	null
0x200	
BTNode	
value	"great"
left	null
right	null
0x480	
BTNode	
value	"a"
left	null
right	null
0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```



```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out
have

- Recursive call of null to the left

Stack

Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null

Heap

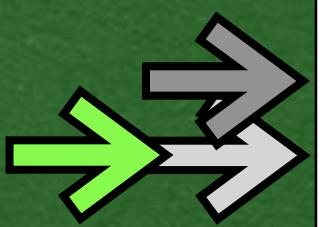
BTNode	
value	"day"
left	null 0x200
right	null 0x480

BTNode	
value	"have"
left	null
right	null

BTNode	
value	"great"
left	null 0x936
right	null

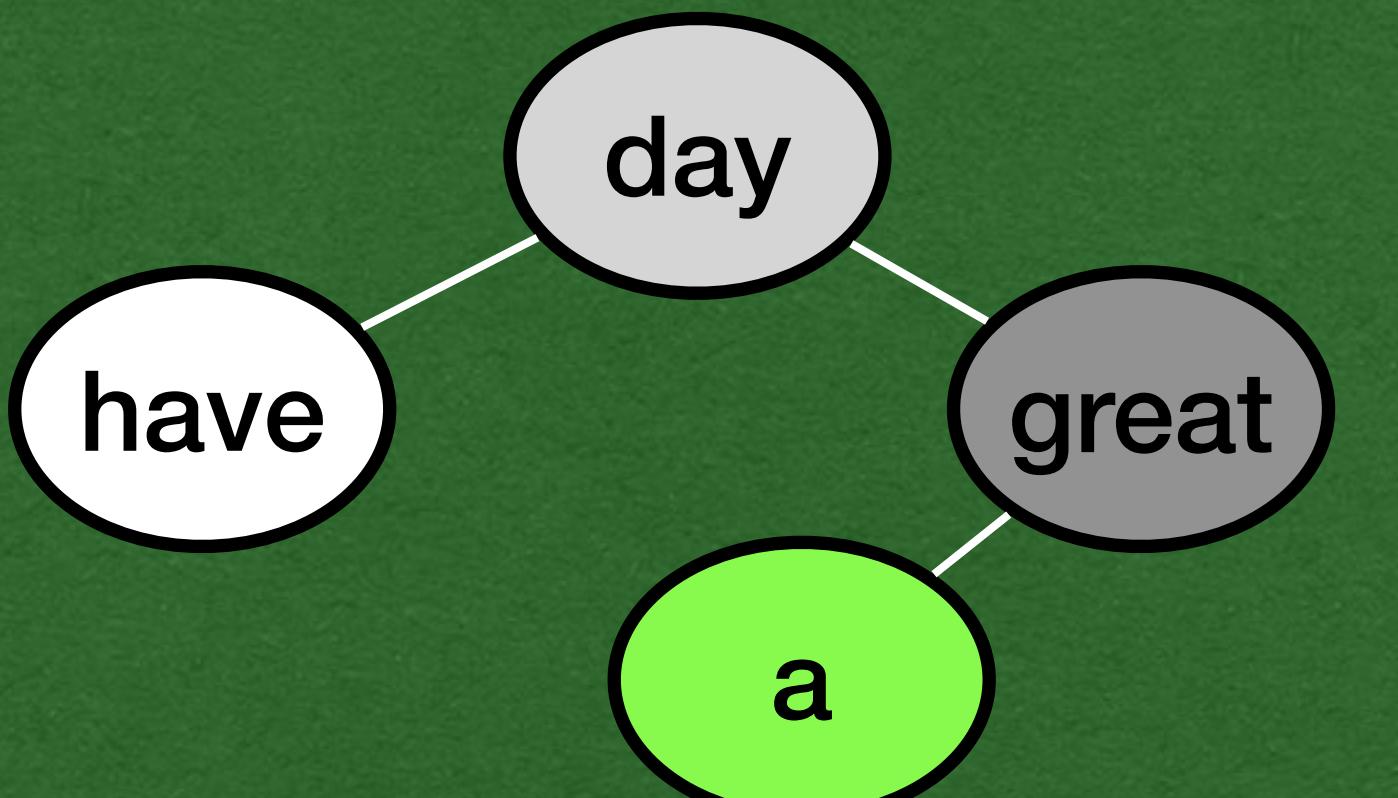
BTNode	
value	"a"
left	null
right	null

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```



```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out
have

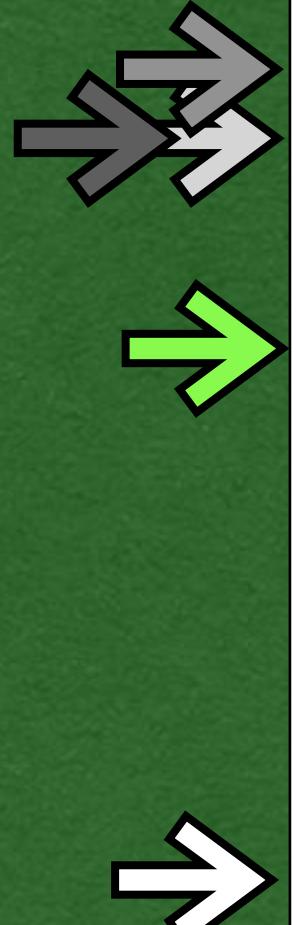
- Return to the previous frame and make the right recursive call

Stack

Name	Value	Heap
root	0x350	BTNode value "day" left null right null
this	0x350	BTNode value "have" left null right null
value	"day"	BTNode value "have" left null right null
left	null	BTNode value "have" left null right null
right	null	BTNode value "have" left null right null
this	0x200	BTNode value "great" left null right null
value	"have"	BTNode value "great" left null right null
left	null	BTNode value "great" left null right null
right	null	BTNode value "great" left null right null
this	0x480	BTNode value "a" left null right null
value	"great"	BTNode value "a" left null right null
left	null	BTNode value "a" left null right null
right	null	BTNode value "a" left null right null
traversal	0x936	BTNode value "a" left null right null
node	0x350	BTNode value "a" left null right null
node	0x200	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null
traversal	0x480	BTNode value "a" left null right null
node	0x936	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null

Heap

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```



```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```

in/out
have

Stack

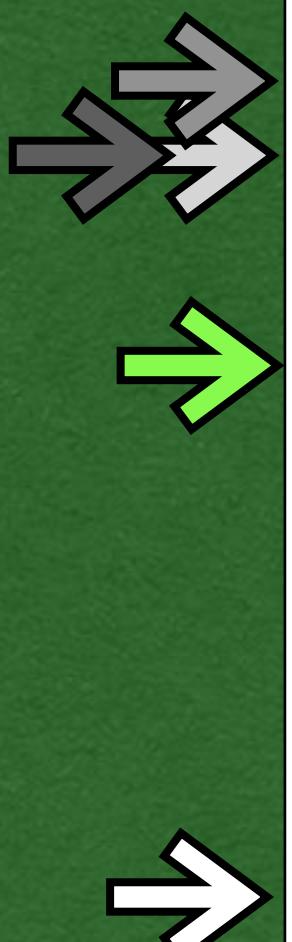
Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null

Heap

BTNode	
value	"day"
left	null 0x200
right	null 0x480
0x350	
BTNode	
value	"have"
left	null
right	null
0x200	
BTNode	
value	"great"
left	null
right	null
0x480	
BTNode	
value	"a"
left	null
right	null
0x936	

- Make the right recursive call of null

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```



```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```

in/out
have

Stack

Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null

Heap

BTNode	
value	"day"
left	null 0x200
right	null 0x480

BTNode	
value	"have"
left	null
right	null

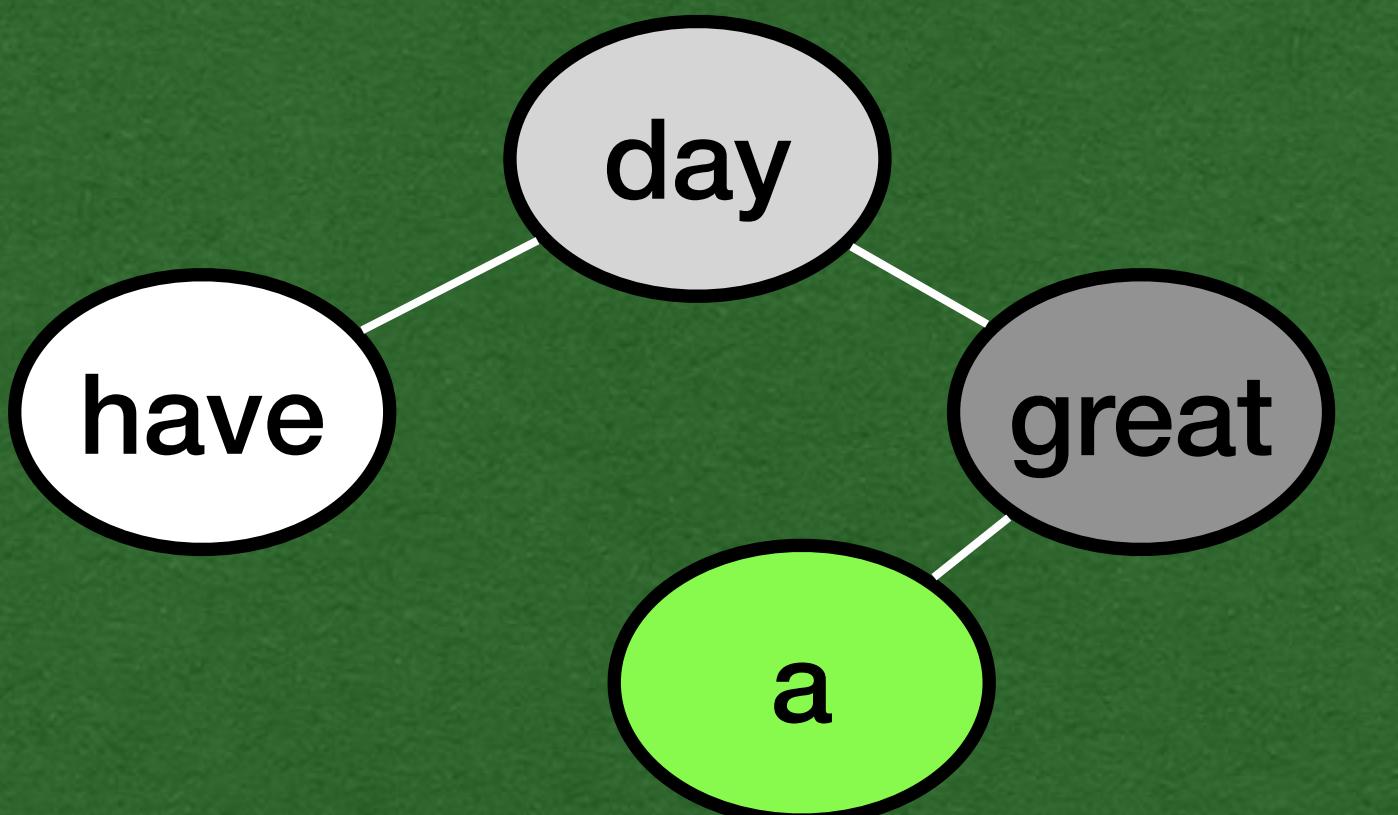
BTNode	
value	"great"
left	null 0x936
right	null

BTNode	
value	"a"
left	null
right	null

- Notice that all 4 stack frames remember what they need to do next when they regain control

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {  
    if (node != null) {  
        traversal(node.left)  
        traversal(node.right)  
        print(node.value + " ")  
    }  
}  
  
def main(args: Array[String]): Unit = {  
    val root = new BTNode("day", null, null)  
    root.left = new BTNode("have", null, null)  
    root.right = new BTNode("great", null, null)  
    root.right.left = new BTNode("a", null, null)  
    traversal(root)  
}
```



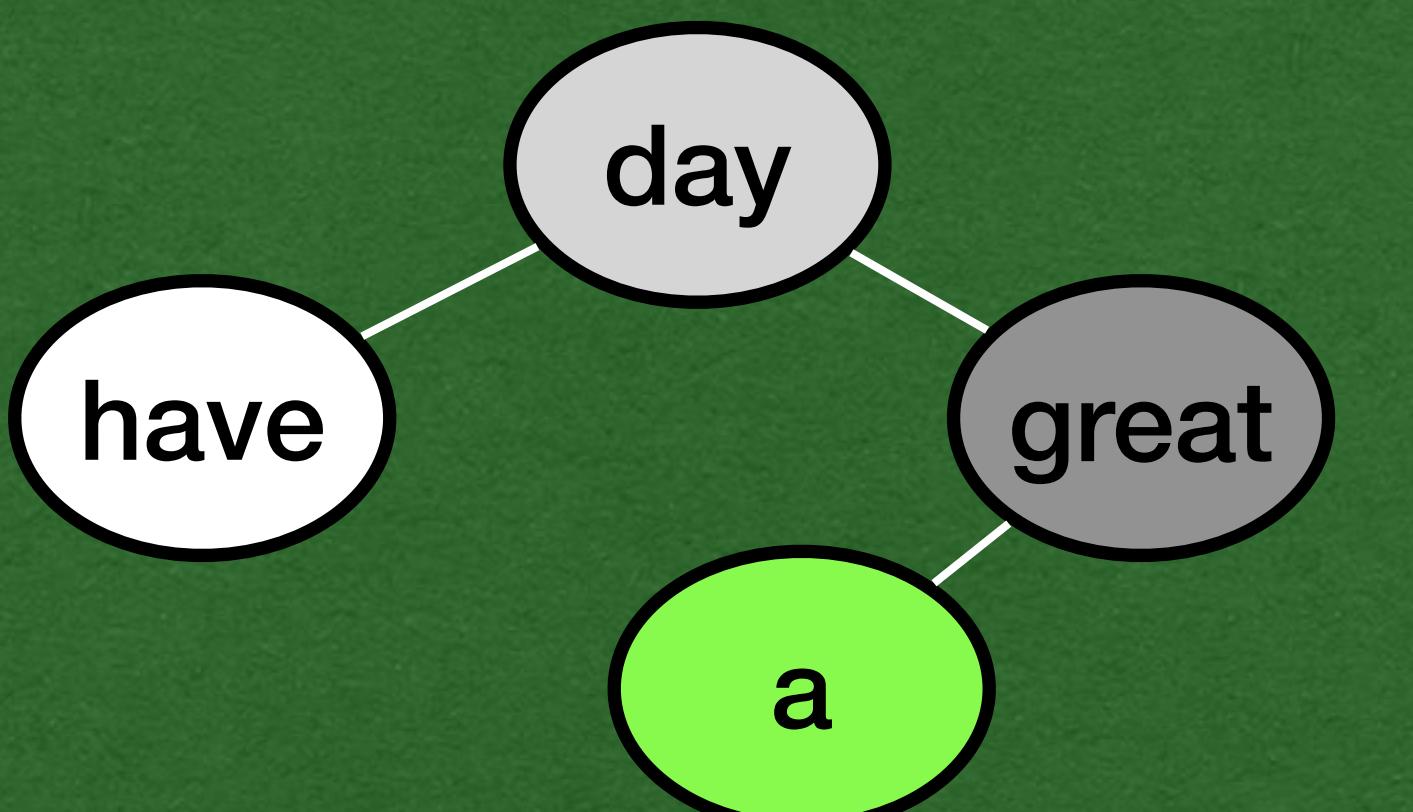
- This stack frame already made both recursive calls

Stack		Heap
Name	Value	
root	0x350	BTNode value "day" left null right null
this	0x350	BTNode value "have" left null right null
value	"day"	BTNode value "have" left null right null
left	null	BTNode value "have" left null right null
right	null	BTNode value "have" left null right null
this	0x200	BTNode value "great" left null right null
value	"have"	BTNode value "great" left null right null
left	null	BTNode value "great" left null right null
right	null	BTNode value "great" left null right null
this	0x480	BTNode value "a" left null right null
value	"great"	BTNode value "a" left null right null
left	null	BTNode value "a" left null right null
right	null	BTNode value "a" left null right null
traversal	0x936	BTNode value "a" left null right null
node	0x350	BTNode value "a" left null right null
node	0x200	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null
traversal	0x480	BTNode value "a" left null right null
node	0x936	BTNode value "a" left null right null
traversal	0x936	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null
traversal	0x936	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null
traversal	0x936	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
    if (node != null) {
        traversal(node.left)
        traversal(node.right)
        print(node.value + " ")
    }
}

def main(args: Array[String]): Unit = {
    val root = new BTNode("day", null, null)
    root.left = new BTNode("have", null, null)
    root.right = new BTNode("great", null, null)
    root.right.left = new BTNode("a", null, null)
    traversal(root)
}
```



- Print "a" to the screen

Stack

Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null

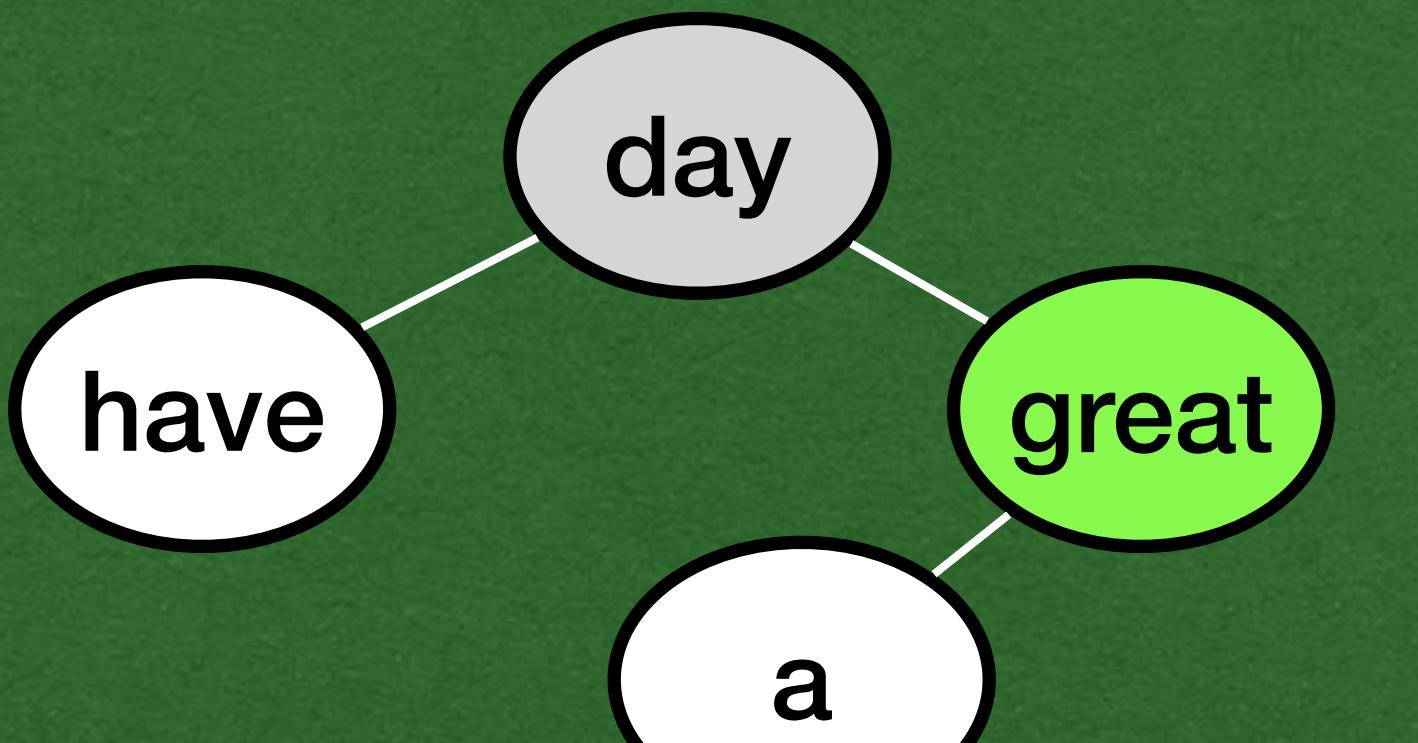
Heap

BTNode
value "day" left null right null 0x350
value "have" left null right null 0x200
value "great" left null right null 0x480
value "a" left null right null 0x936
value "day" left null right null 0x350
value "have" left null right null 0x200
value "great" left null right null 0x480
value "a" left null right null 0x936

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out
have a

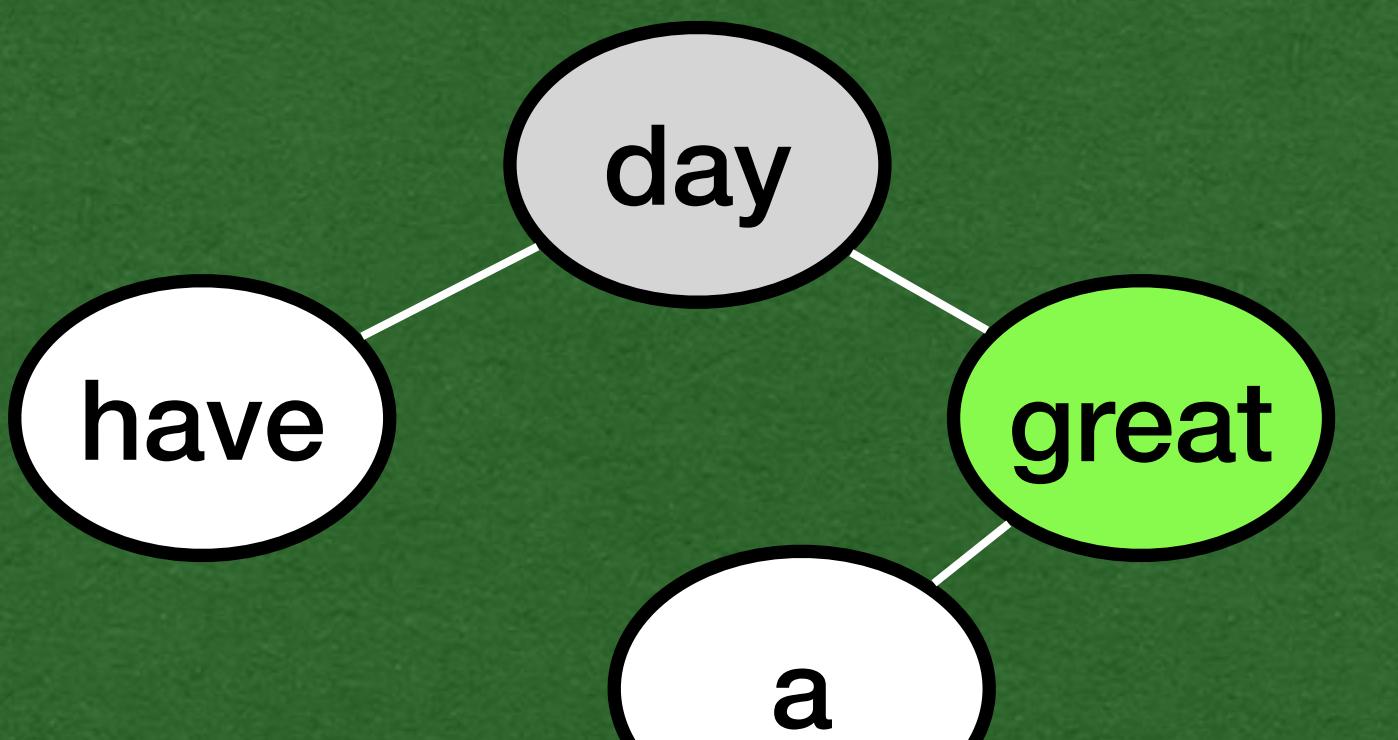
- Return back to the previous stack frame

Stack		Heap
Name	Value	
root	0x350	BTNode value "day" left null right null
this	0x350	BTNode value "have" left null right null
value	"day"	BTNode value "have" left null right null
left	null	BTNode value "have" left null right null
right	null	BTNode value "have" left null right null
this	0x200	BTNode value "great" left null right null
value	"have"	BTNode value "great" left null right null
left	null	BTNode value "great" left null right null
right	null	BTNode value "great" left null right null
this	0x480	BTNode value "a" left null right null
value	"great"	BTNode value "a" left null right null
left	null	BTNode value "a" left null right null
right	null	BTNode value "a" left null right null
traversal	0x936	BTNode value "a" left null right null
node	0x350	BTNode value "a" left null right null
node	0x200	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null
traversal	0x480	BTNode value "a" left null right null
node	0x936	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null
traversal	0x936	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null
traversal		

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
    if (node != null) {
        traversal(node.left)
        traversal(node.right)
        print(node.value + " ")
    }
}

def main(args: Array[String]): Unit = {
    val root = new BTNode("day", null, null)
    root.left = new BTNode("have", null, null)
    root.right = new BTNode("great", null, null)
    root.right.left = new BTNode("a", null, null)
    traversal(root)
}
```



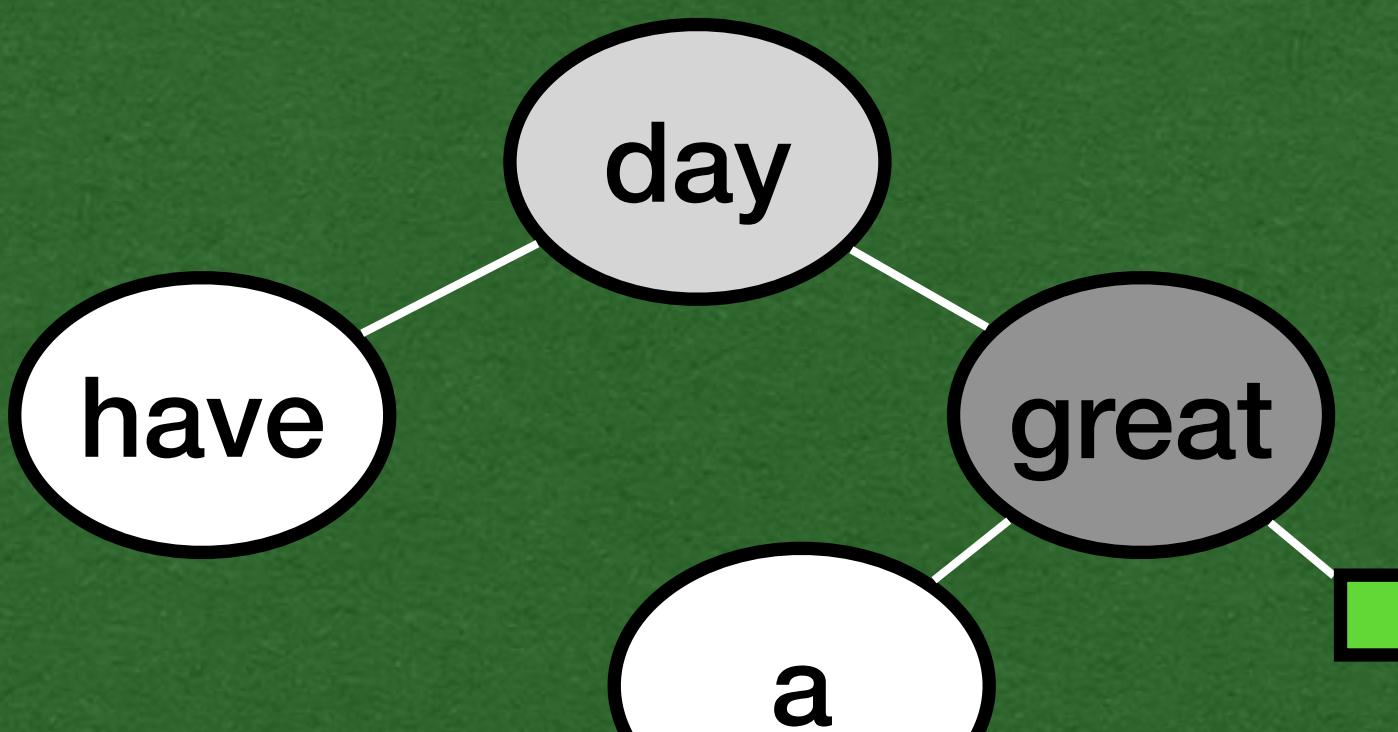
- Make right recursive call

Stack		Heap
Name	Value	
root	0x350	
this	0x350	
value	"day"	
left	null	
right	null	
this	0x200	
value	"have"	
left	null	
right	null	
this	0x480	
value	"great"	
left	null	
right	null	
this	0x936	
value	"a"	
left	null	
right	null	
node	0x350	
node	0x200	
node	null	
node	null	
node	0x480	
node	0x936	
node	null	
node	null	

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
    if (node != null) {
        traversal(node.left)
        traversal(node.right)
        print(node.value + " ")
    }
}

def main(args: Array[String]): Unit = {
    val root = new BTNode("day", null, null)
    root.left = new BTNode("have", null, null)
    root.right = new BTNode("great", null, null)
    root.right.left = new BTNode("a", null, null)
    traversal(root)
}
```



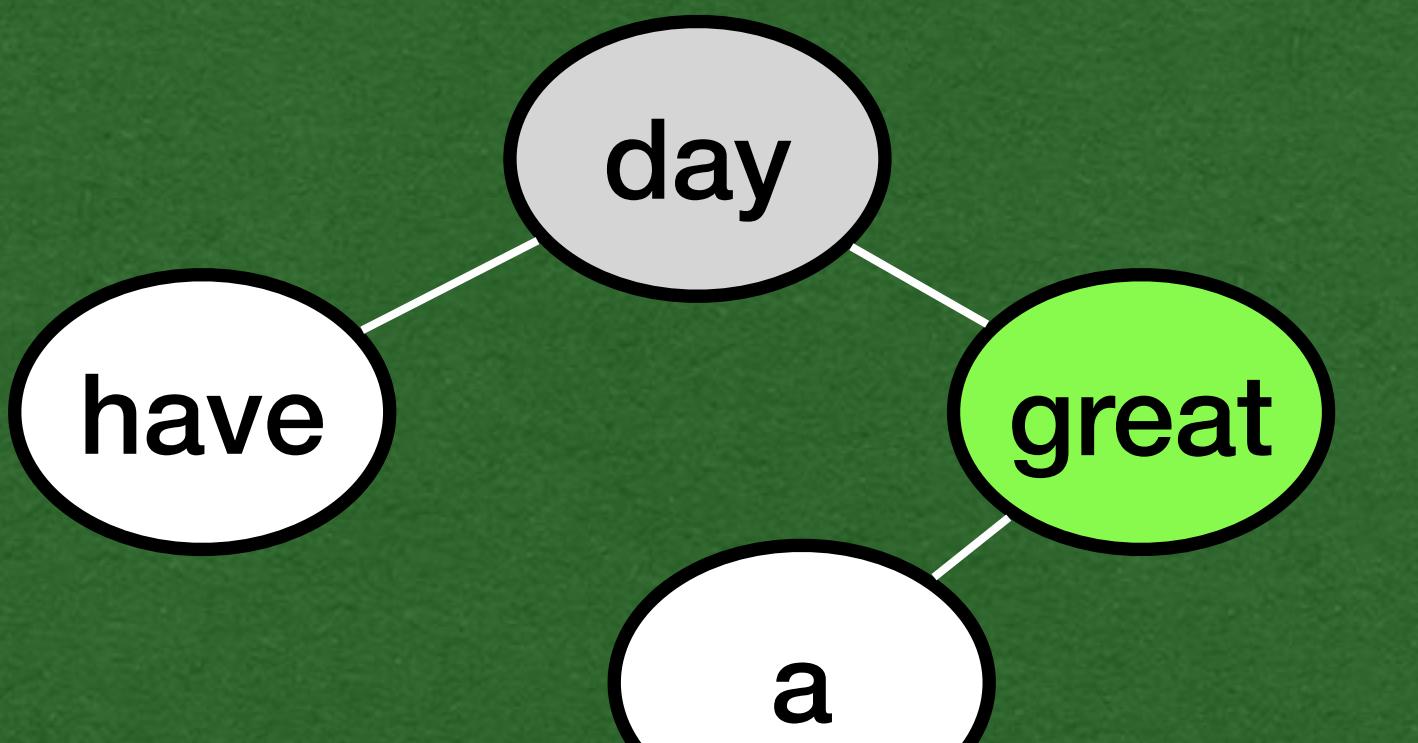
- Base case of null
 - Just return

Stack		Heap	
Name	Value		
root	0x350	BTNode	
this	0x350	value	"day"
value	"day"	left	null
left	null	right	null
right	null	0x350	
BTNode		BTNode	
this	0x200	value	"have"
value	"have"	left	null
left	null	right	null
right	null	0x200	
BTNode		BTNode	
this	0x480	value	"great"
value	"great"	left	null
left	null	right	null
right	null	0x480	
BTNode		BTNode	
this	0x936	value	"a"
value	"a"	left	null
left	null	right	null
right	null	0x936	
BTNode		BTNode	
traversal	0x350	value	"great"
traversal	0x200	left	null
traversal	null	right	null
traversal	null	0x480	
traversal	0x480	BTNode	
traversal	0x936	value	"a"
traversal	null	left	null
traversal	null	right	null
traversal	null	0x936	
traversal	0x936	BTNode	
traversal	null	value	"day"
traversal	null	left	null
traversal	null	right	null
traversal	null	0x350	
traversal	0x350	BTNode	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



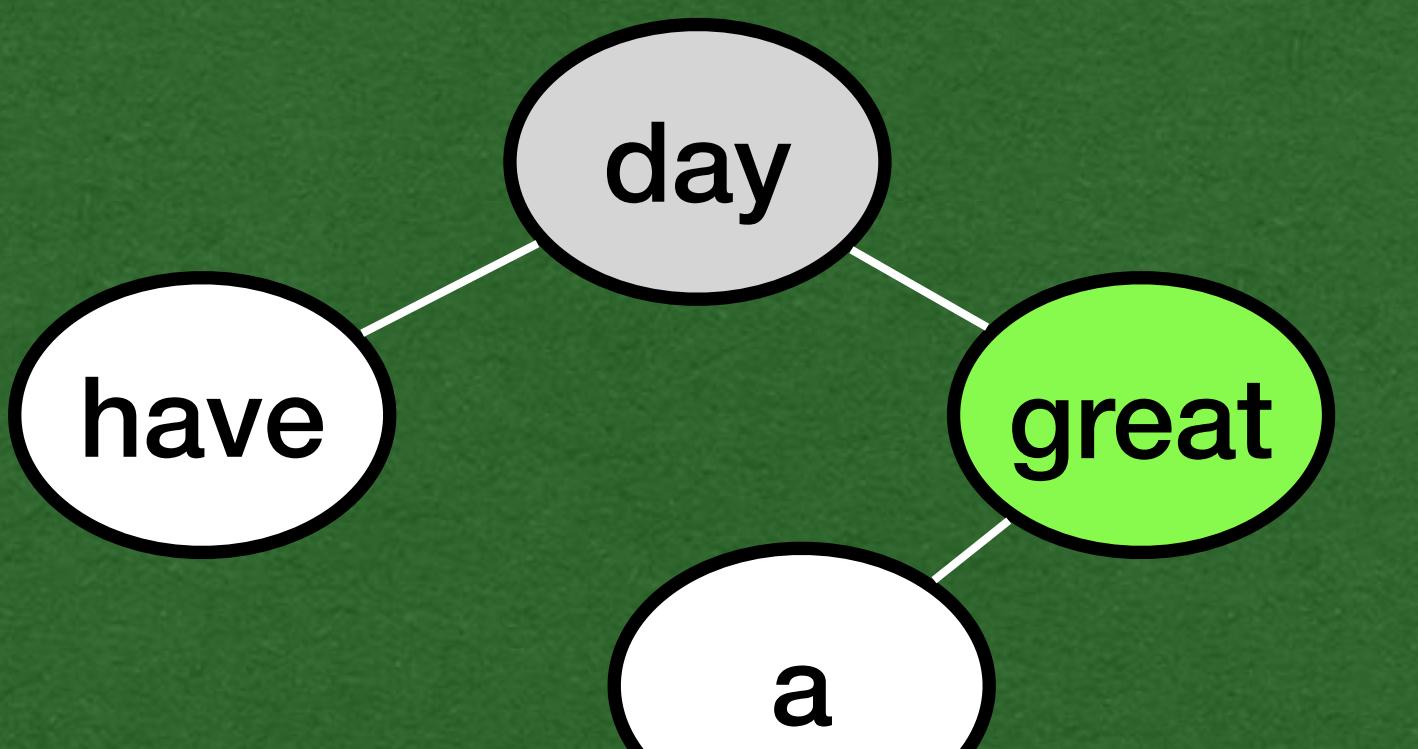
- Stack frame done with both recursive calls

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	traversal
node	0x200	traversal
node	null	traversal
node	null	traversal
node	0x480	traversal
node	0x936	traversal
node	null	traversal
node	null	traversal
node	0x936	traversal
node	null	traversal
node	null	traversal
node	null	traversal

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out
have a great

- Print "great" to the screen

Stack

Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null
node	0x936

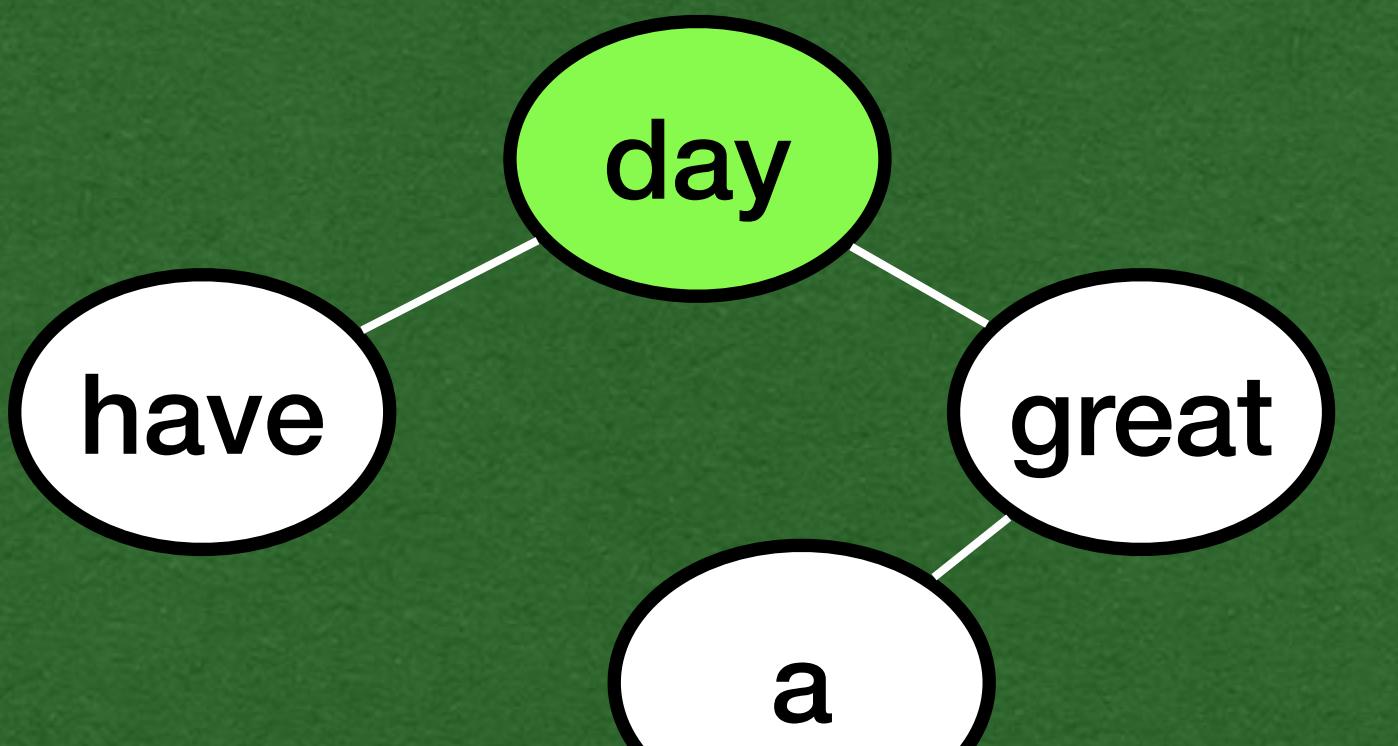
Heap

BTNode	
value	"day"
left	null 0x200
right	null 0x480
0x350	
BTNode	
value	"have"
left	null
right	null
0x200	
BTNode	
value	"have"
left	null
right	null
0x480	
BTNode	
value	"great"
left	null 0x936
right	null
0x936	
BTNode	
value	"a"
left	null
right	null
0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



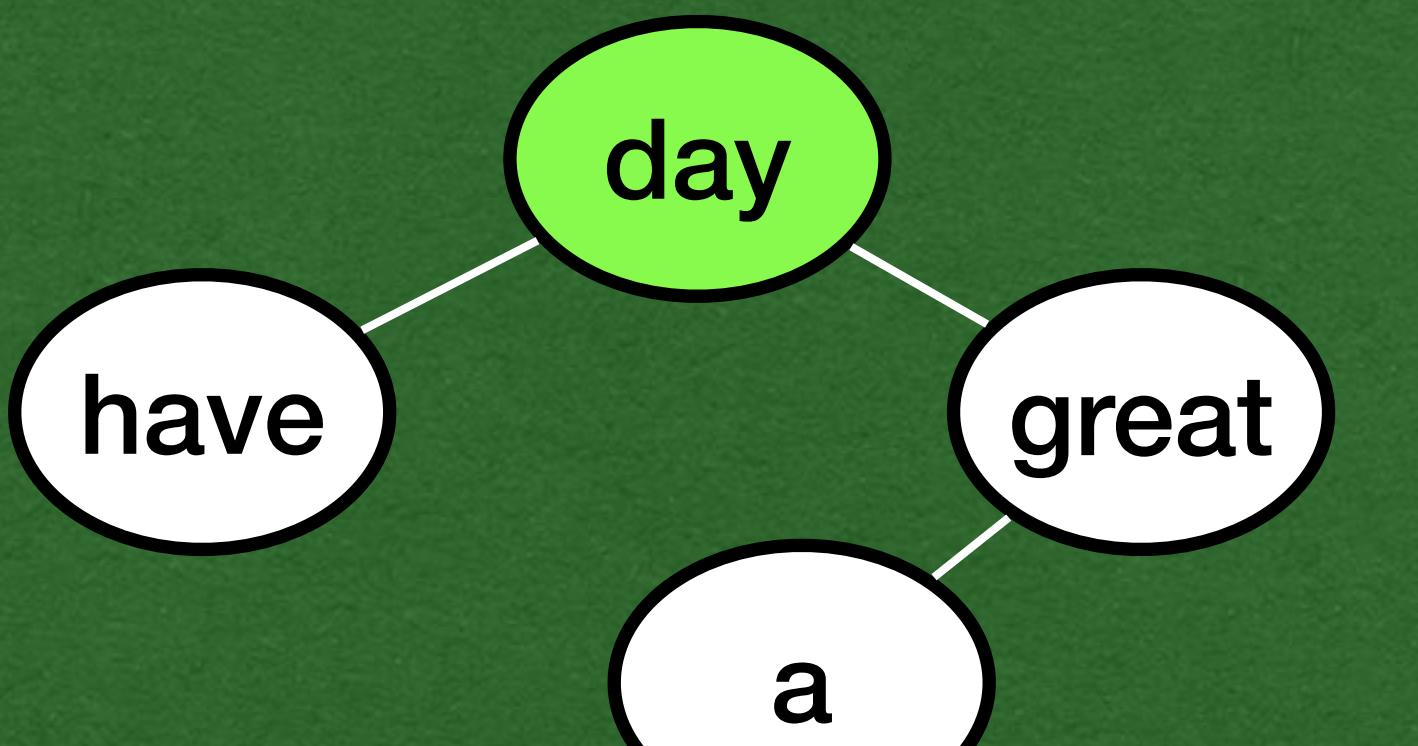
in/out
have a great

- Root node is now done with both recursive calls

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left 0x200
right	null	right 0x480
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	0x200
node	0x200	traversal
node	null	traversal
node	null	traversal
node	0x480	traversal
node	0x936	traversal
node	null	traversal
node	0x480	traversal
node	0x936	traversal
node	null	traversal
node	null	traversal
node	null	traversal
node	0x936	traversal

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {  
    if (node != null) {  
        traversal(node.left)  
        traversal(node.right)  
        print(node.value + " ")  
    }  
}  
  
def main(args: Array[String]): Unit = {  
    val root = new BTNode("day", null, null)  
    root.left = new BTNode("have", null, null)  
    root.right = new BTNode("great", null, null)  
    root.right.left = new BTNode("a", null, null)  
    traversal(root)  
}
```



in/out
have a great day

- Print "day" to the screen

Stack

Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null

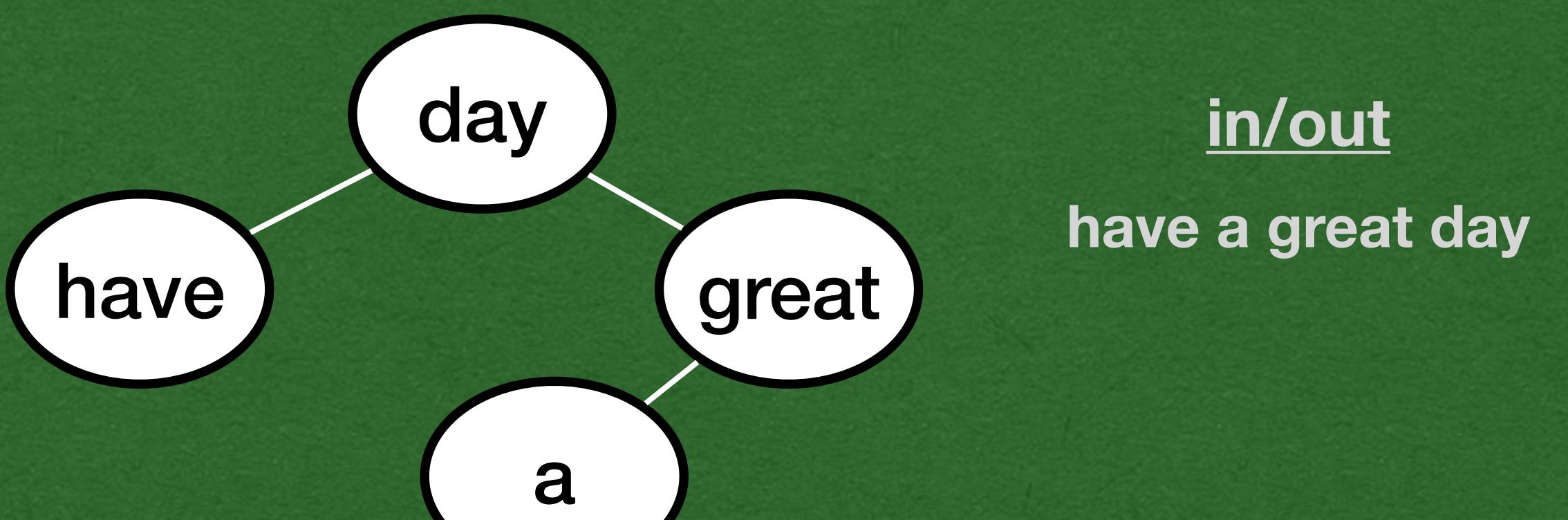
Heap

BTNode	
value	"day"
left	null 0x200
right	null 0x480
0x350	
BTNode	
value	"have"
left	null
right	null
0x200	
BTNode	
value	"great"
left	null 0x936
right	null 0x480
0x480	
BTNode	
value	"a"
left	null
right	null
0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



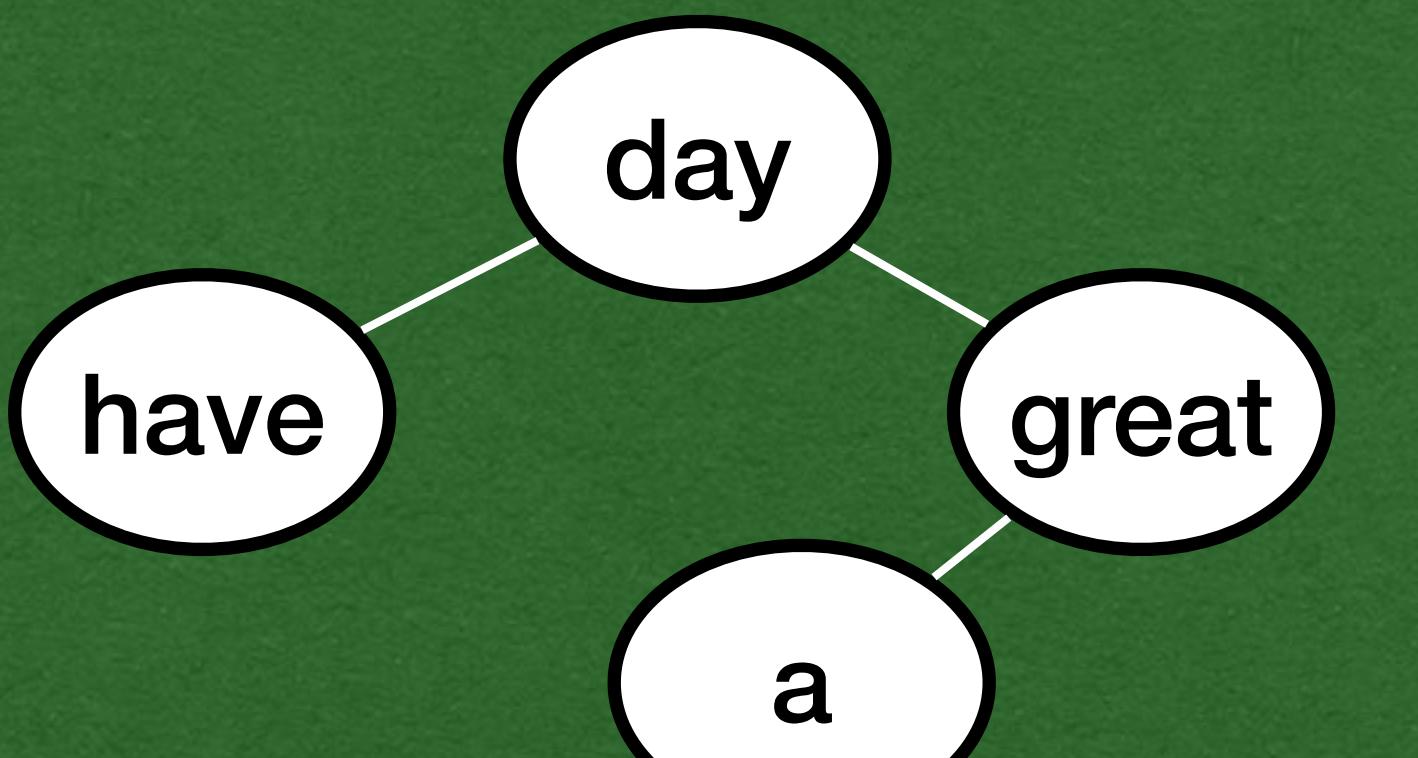
- The traversal is over
- All stack frames have returned

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
traversal	0x350	
traversal	0x200	
traversal	null	
traversal	null	
traversal	0x480	
traversal	0x936	
traversal	null	
traversal	null	
traversal	null	
traversal	0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("day", null, null)
  root.left = new BTNode("have", null, null)
  root.right = new BTNode("great", null, null)
  root.right.left = new BTNode("a", null, null)
  traversal(root)
}
```



in/out
have a great day

- Program ends

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "day"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "have"
left	null	left null
right	null	right null
this	0x936	0x200
value	"a"	value "have"
left	null	left null
right	null	right null
traversal	0x350	BTNode
traversal	0x200	value "great"
traversal	null	left null
traversal	null	right null
traversal	0x480	0x480
traversal	0x936	value "a"
traversal	null	left null
traversal	null	right null
traversal	0x936	0x936