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

in/out

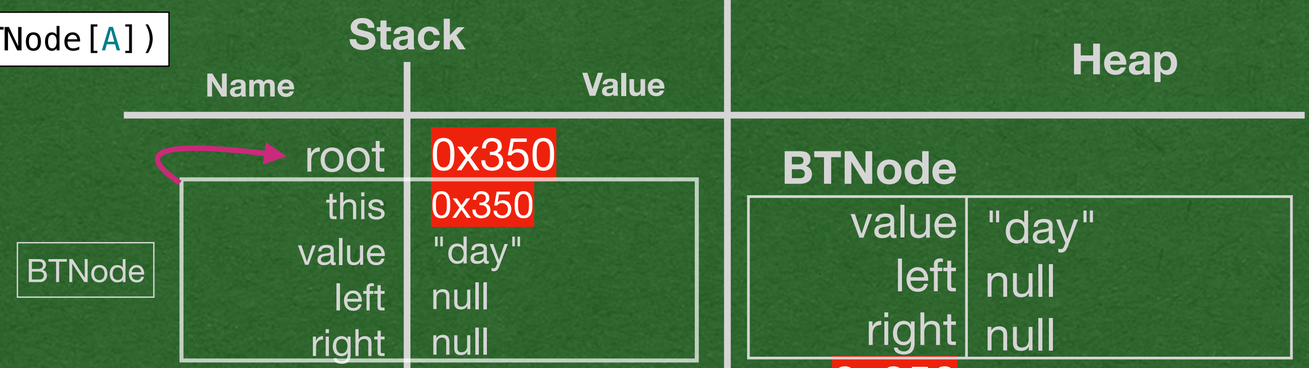
- Build a tree
- Visit every node with a post-order traversal

```
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)
}
```

day

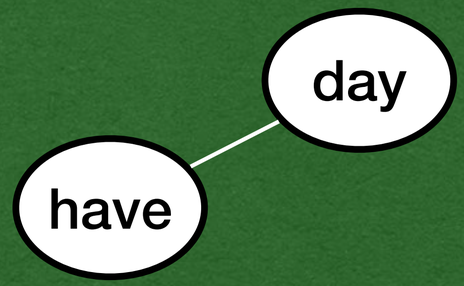
in/out



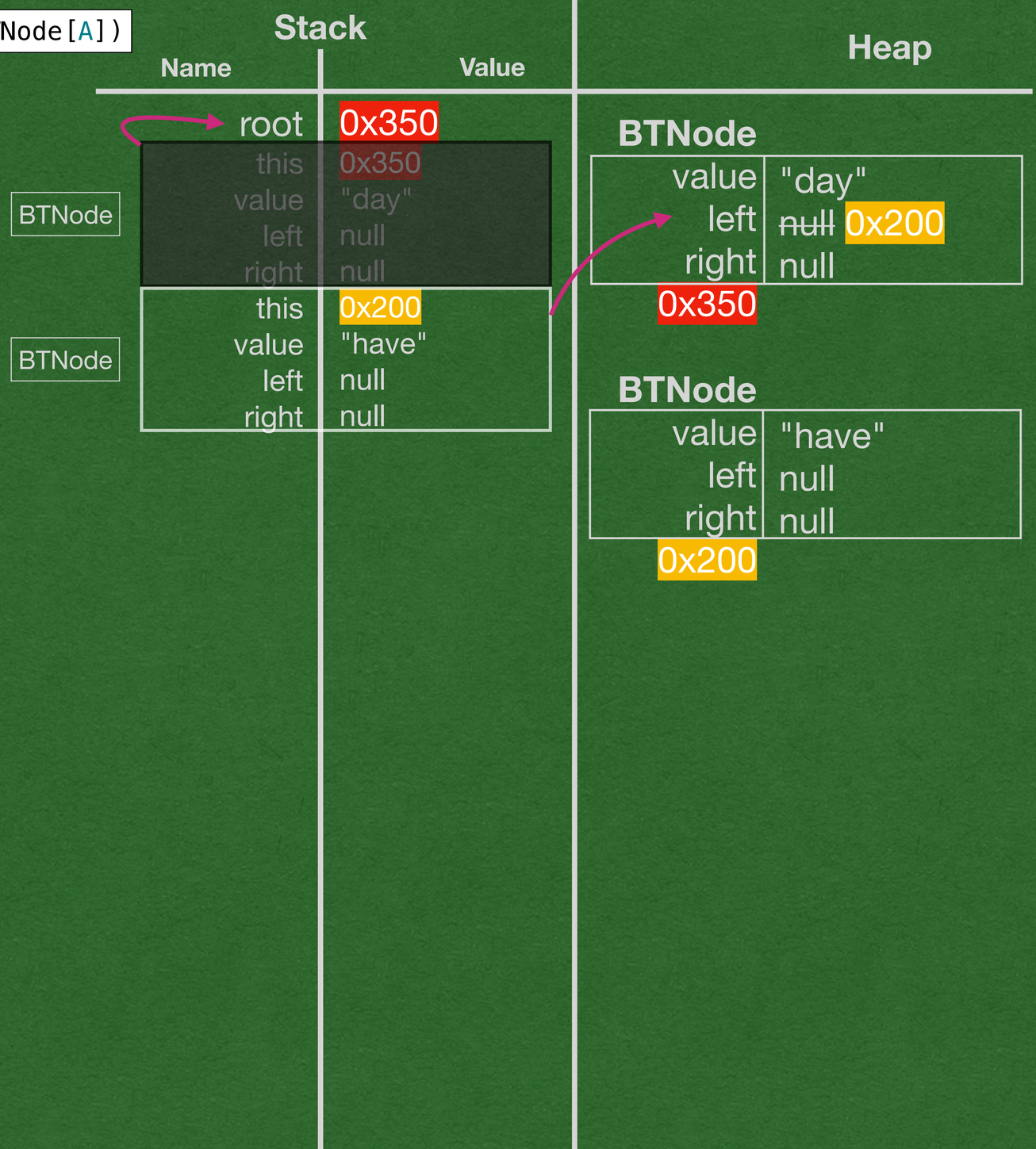
- Create the root node with the value "day"

```
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)
}
```



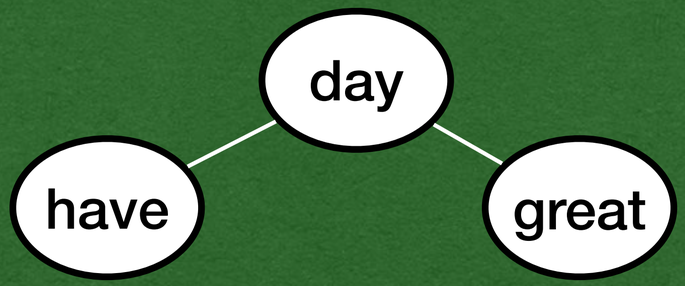
in/out



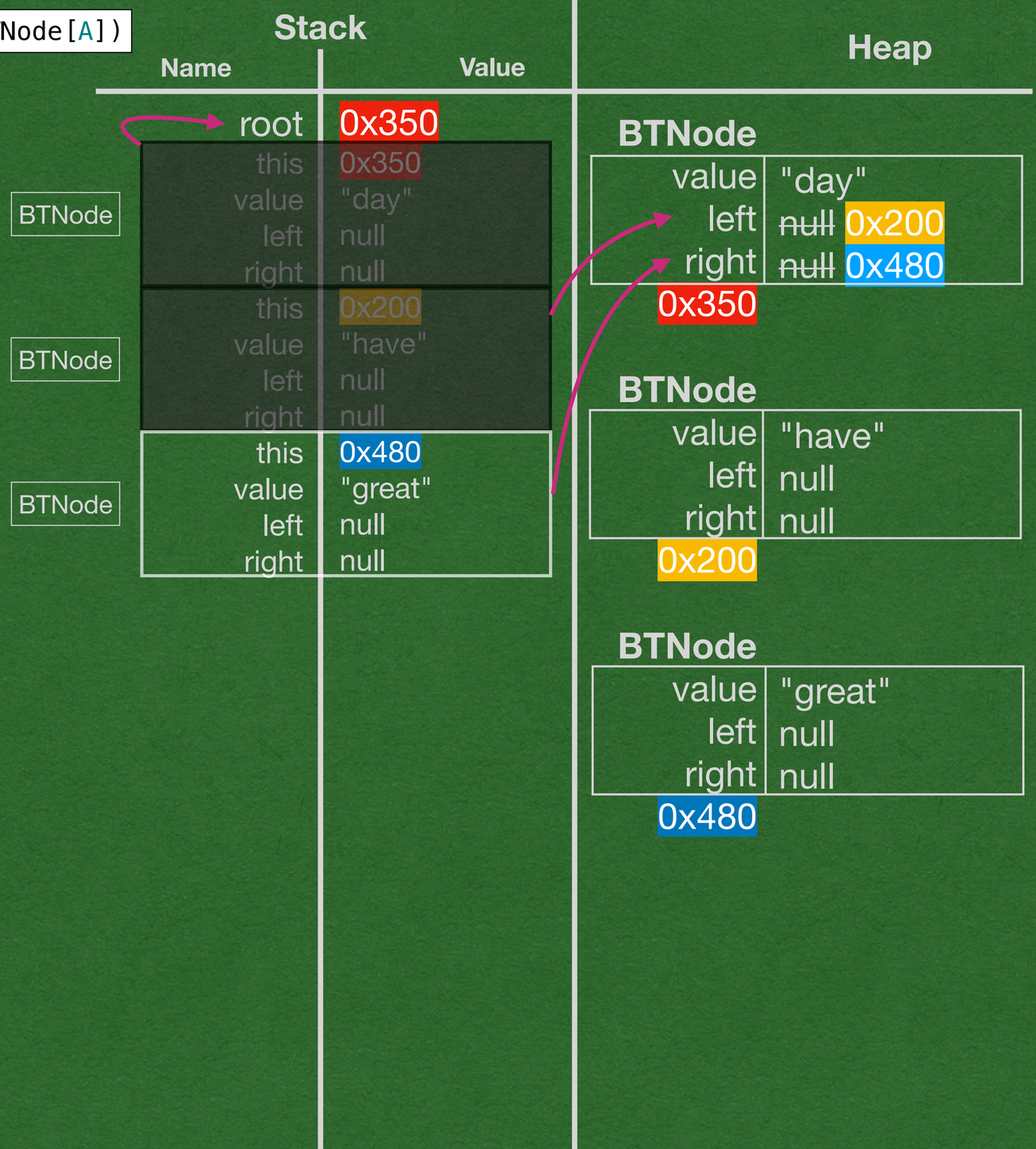
- Set the left child of the root to a node with "have"

```
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)
}
```



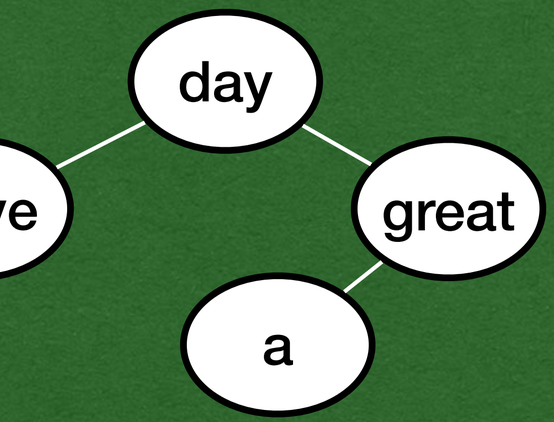
in/out



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

```
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 right child of the root to a node with "a"

Stack

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

Heap

BTNode 0x350

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

BTNode 0x200

value	"have"
left	null
right	null

BTNode 0x480

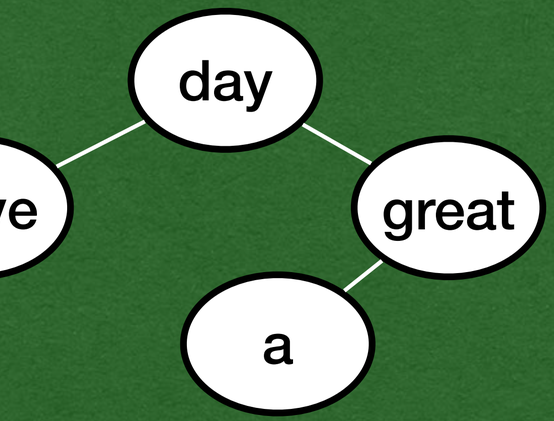
value	"great"
left	~~null~~ 0x936
right	null

BTNode 0x936

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

- Time to start the post-order traversal

Stack

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

Heap

BTNode 0x350

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

BTNode 0x200

value	"have"
left	null
right	null

BTNode 0x480

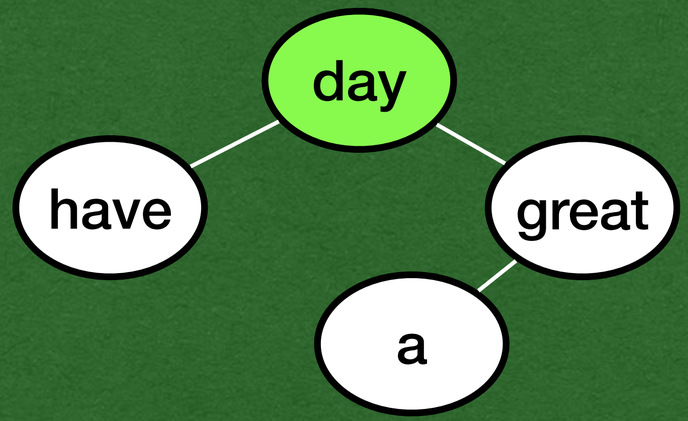
value	"great"
left	~~null~~ 0x936
right	null

BTNode 0x936

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

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

Stack

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

Heap

BTNode 0x350

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

BTNode 0x200

value	"have"
left	null
right	null

BTNode 0x480

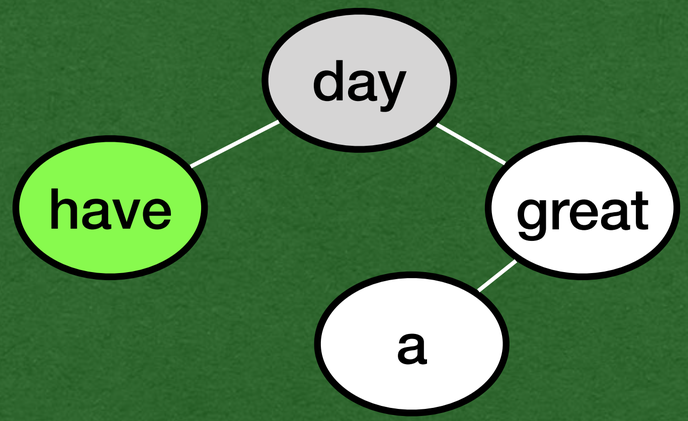
value	"great"
left	~~null~~ 0x936
right	null

BTNode 0x936

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

- Make a recursive call on the left child first

Stack

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

Heap

BTNode
value "day"
left ~~null~~ 0x200
right ~~null~~ 0x480
0x350

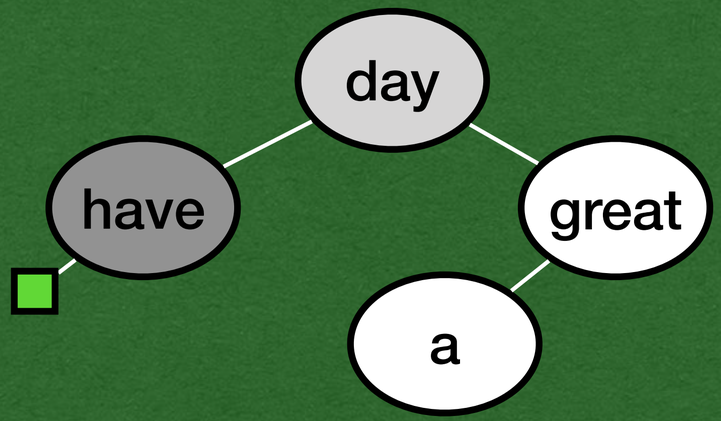
BTNode
value "have"
left null
right null
0x200

BTNode
value "great"
left ~~null~~ 0x936
right null
0x480

BTNode
value "a"
left null
right null
0x936

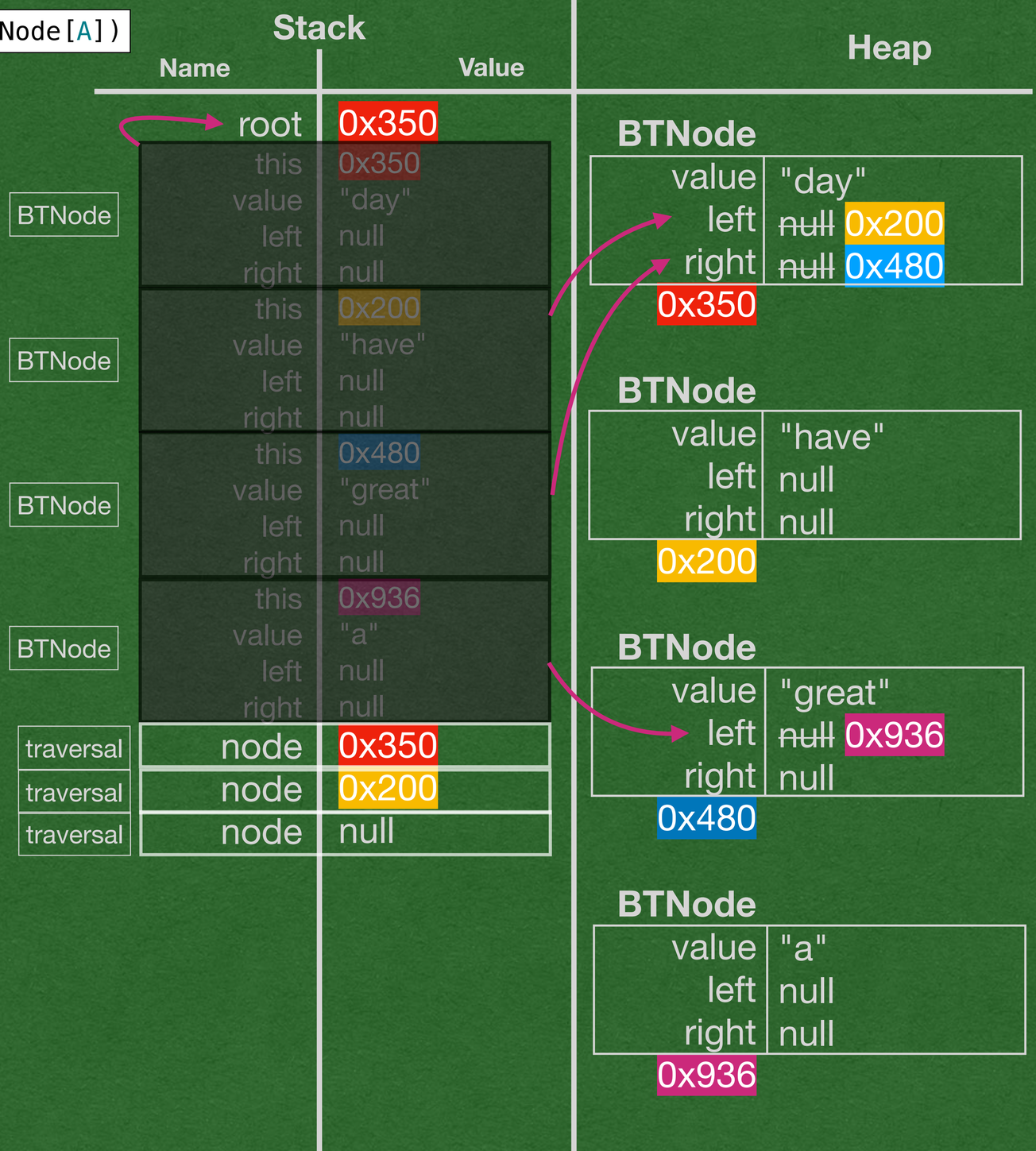
```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
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)
}
```



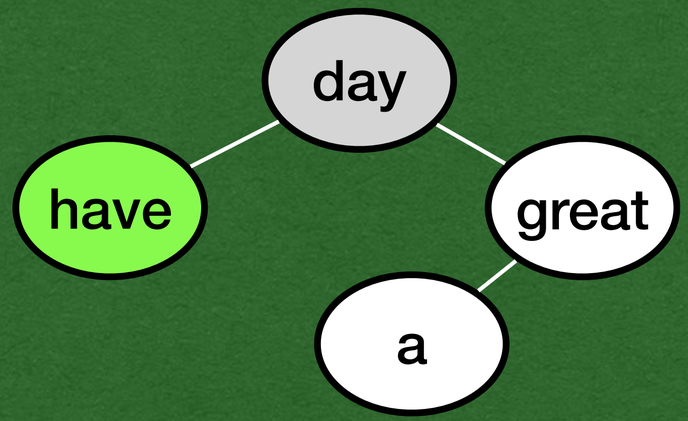
in/out

- Left node makes another recursive call on its left child



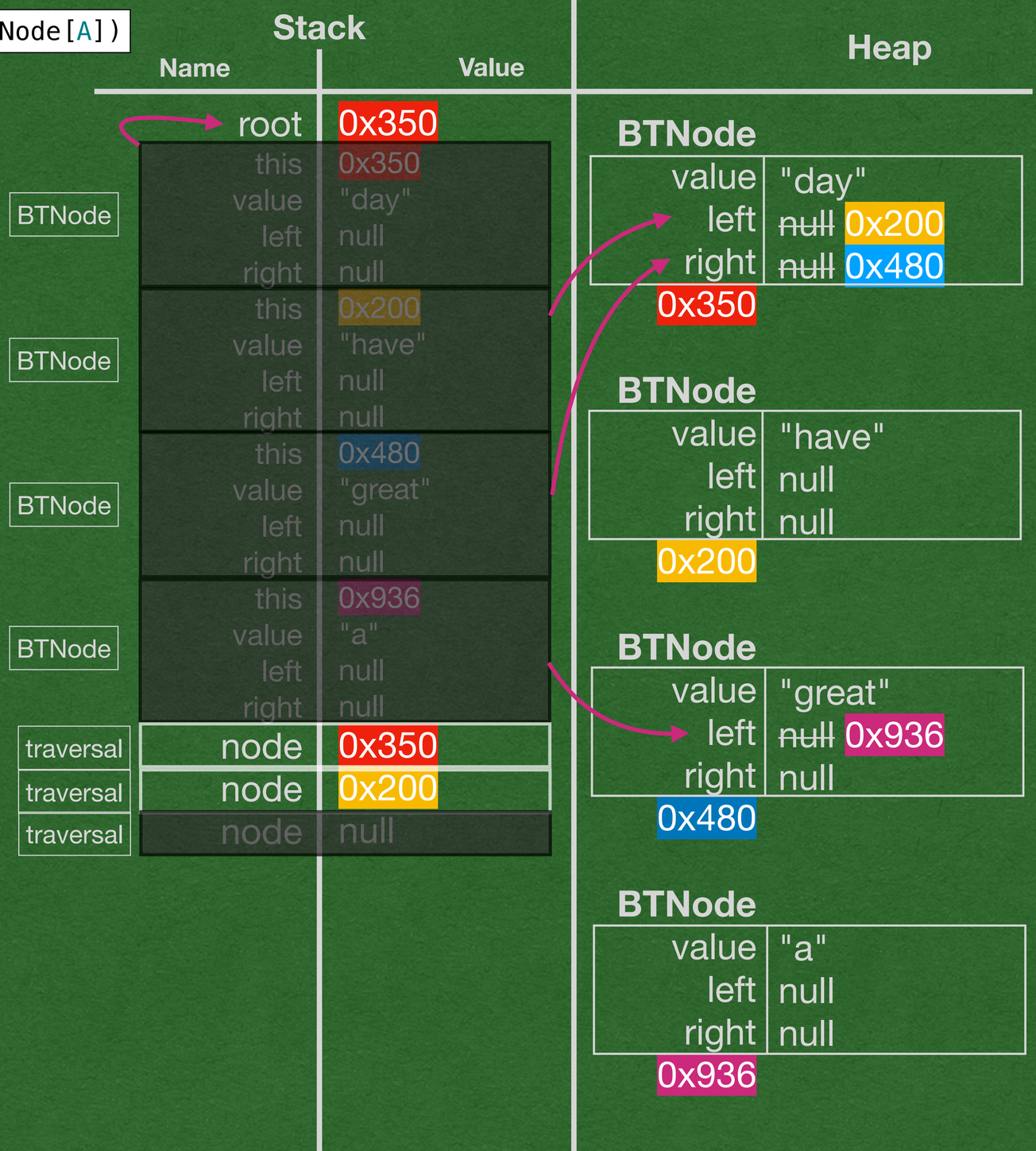
```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
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)
}
```



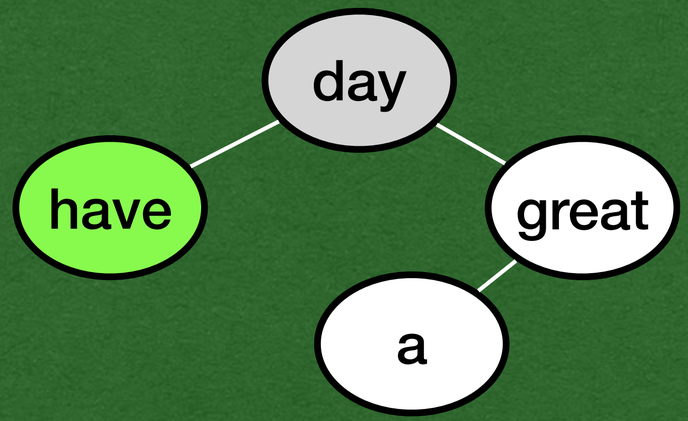
in/out

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



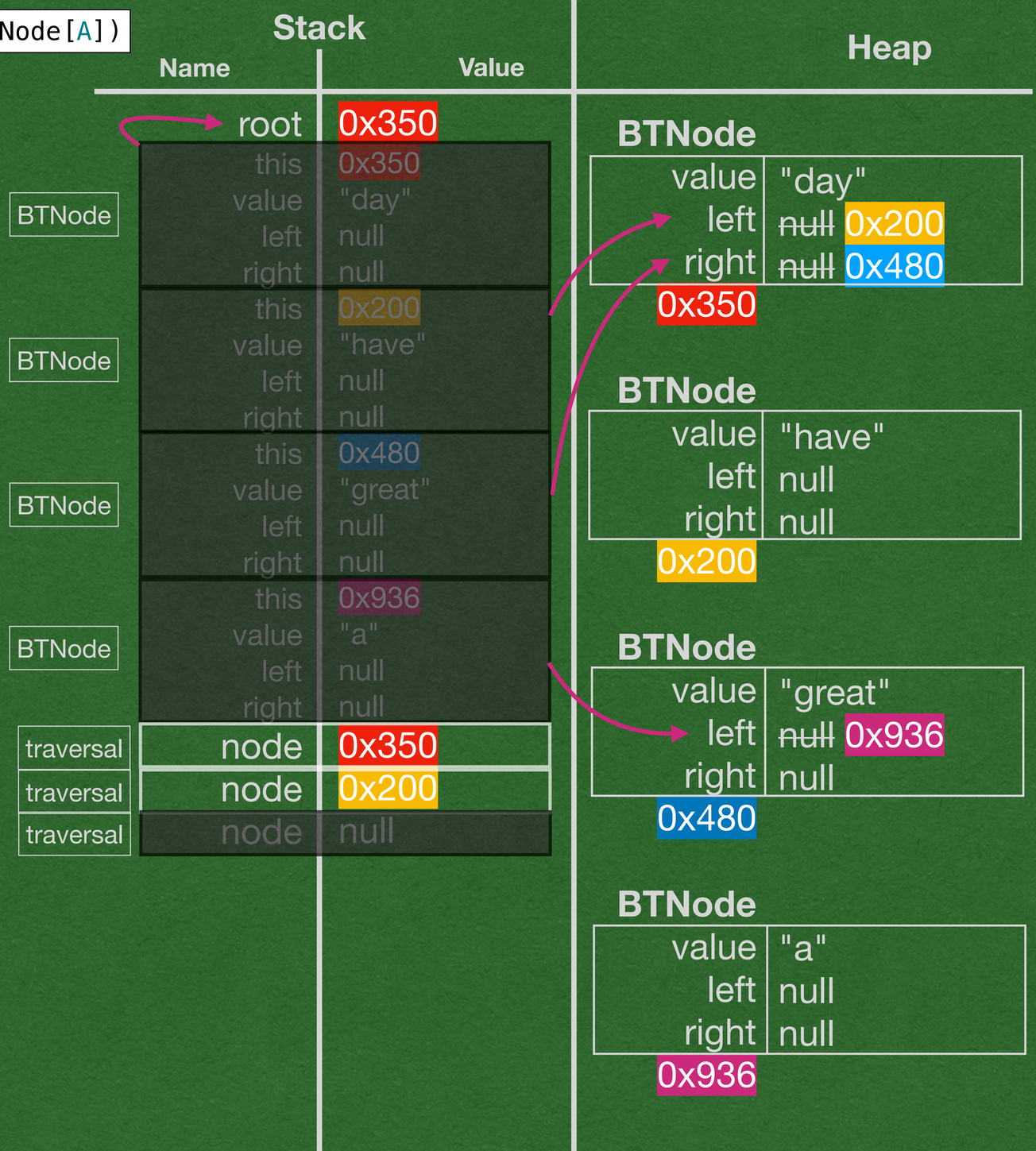
```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
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)
}
```



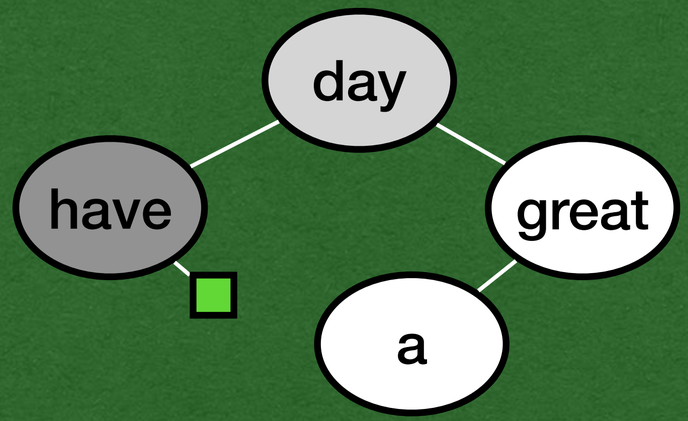
in/out

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



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

Heap

BTNode 0x350

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

BTNode 0x200

value	"have"
left	null
right	null

BTNode 0x480

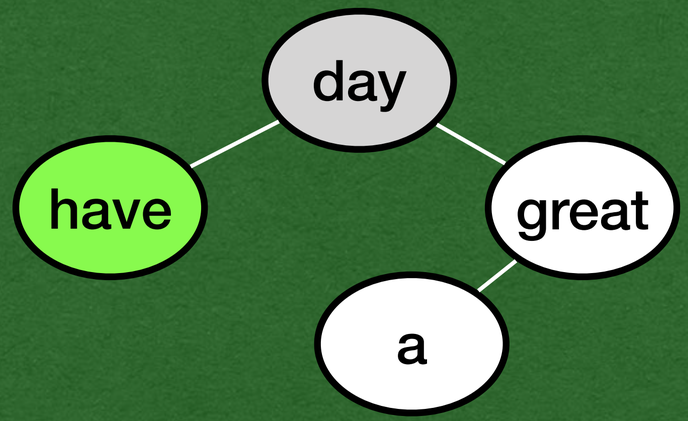
value	"great"
left	~~null~~ 0x936
right	null

BTNode 0x936

value	"a"
left	null
right	null

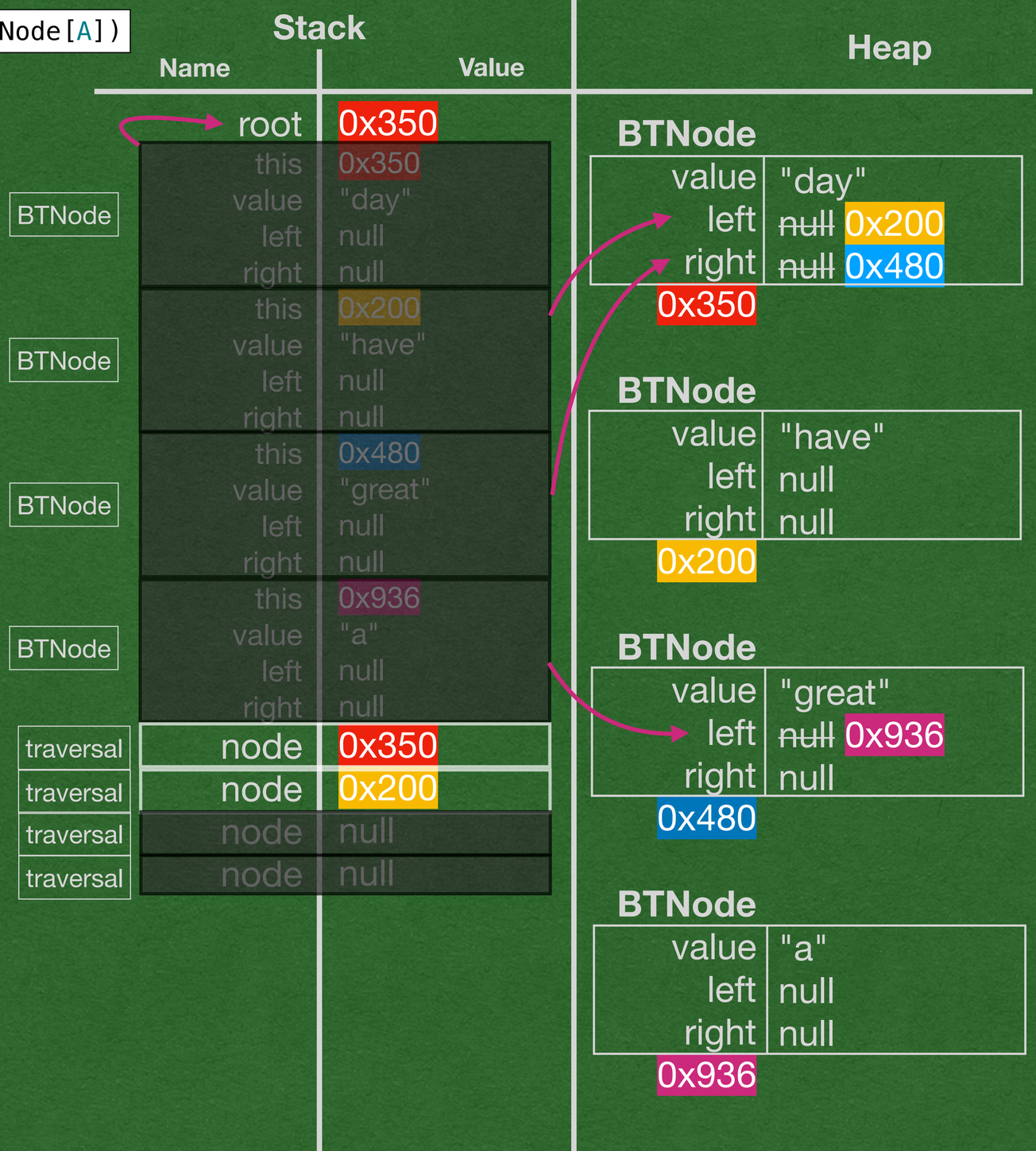
```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
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)
}
```



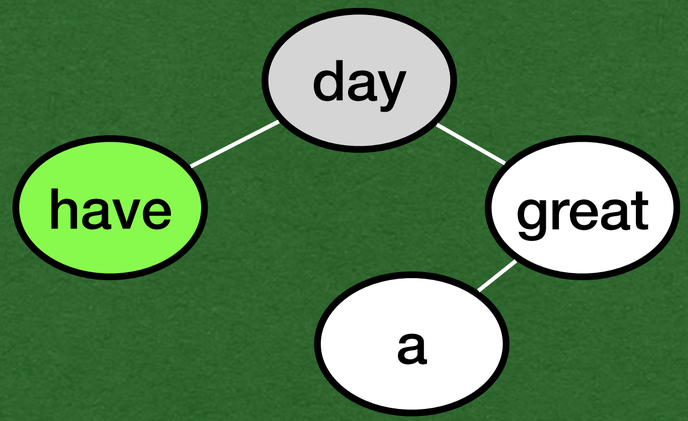
in/out

- 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

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

Heap

BTNode 0x350

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

BTNode 0x200

value	"have"
left	null
right	null

BTNode 0x480

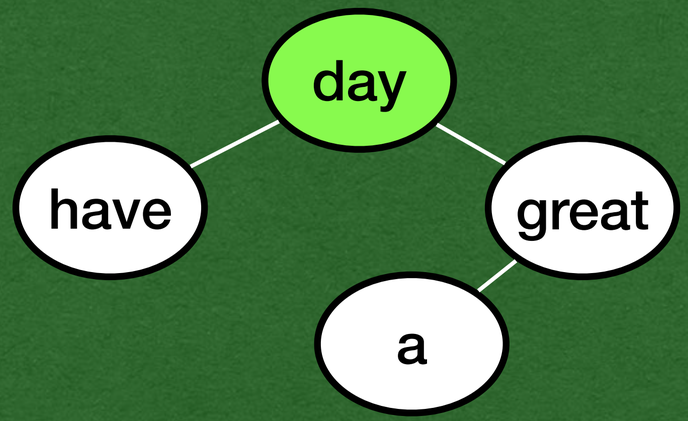
value	"great"
left	~~null~~ 0x936
right	null

BTNode 0x936

value	"a"
left	null
right	null

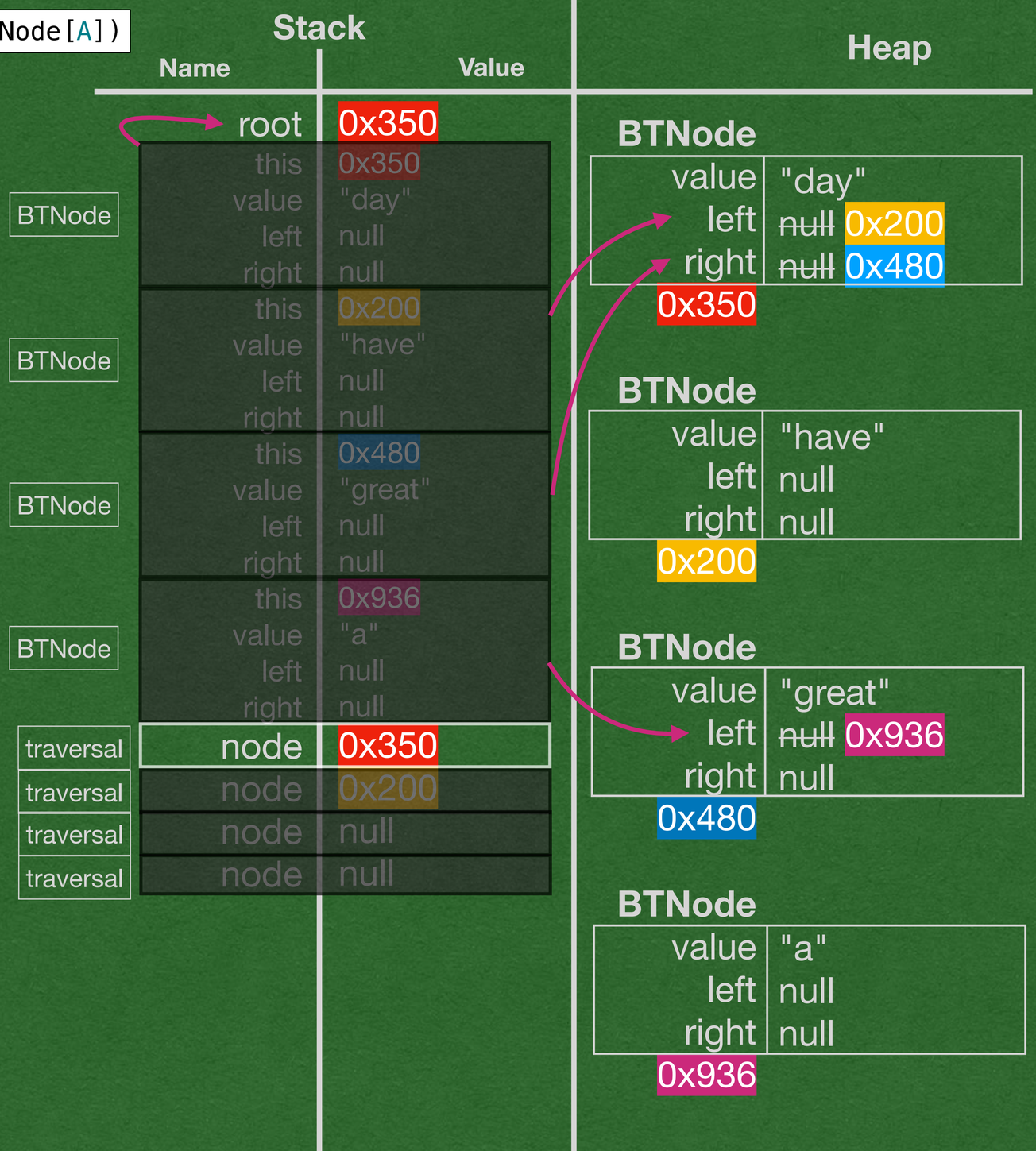
```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
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

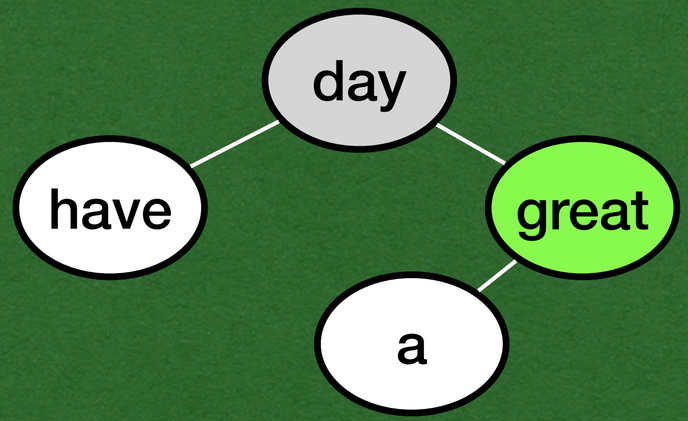
have



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

```
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

- Make a recursive call to the right of the root

Stack

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

Heap

BTNode (0x350)

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

BTNode (0x200)

value	"have"
left	null
right	null

BTNode (0x480)

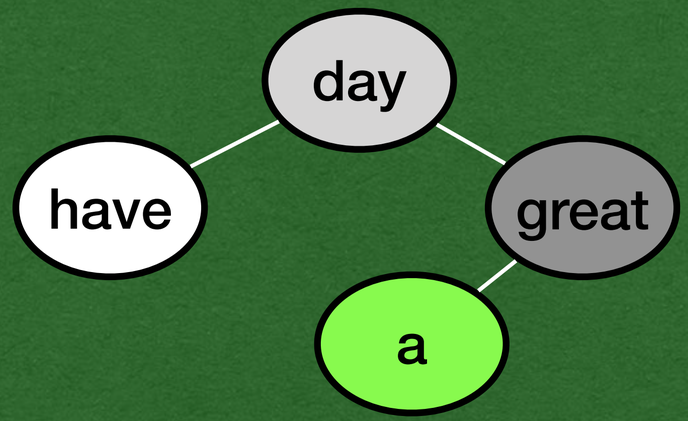
value	"great"
left	~~null~~ 0x936
right	null

BTNode (0x936)

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

- Recursive call to the left

Stack

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

Heap

BTNode 0x350

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

BTNode 0x200

value	"have"
left	null
right	null

BTNode 0x480

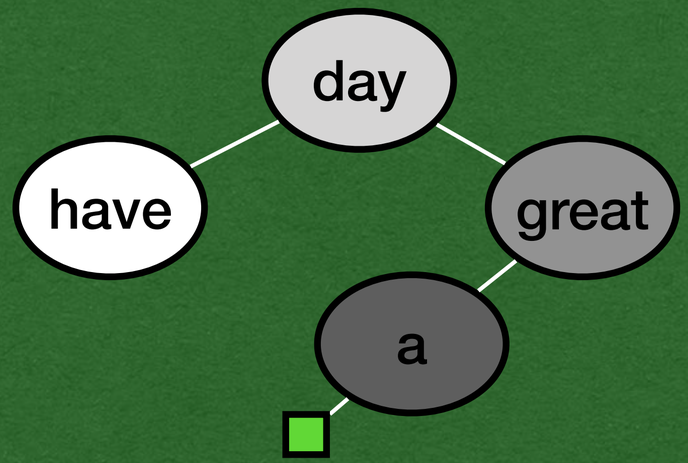
value	"great"
left	~~null~~ 0x936
right	null

BTNode 0x936

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

- Recursive call of null to the left

Stack

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

Heap

BTNode 0x350

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

BTNode 0x200

value	"have"
left	null
right	null

BTNode 0x480

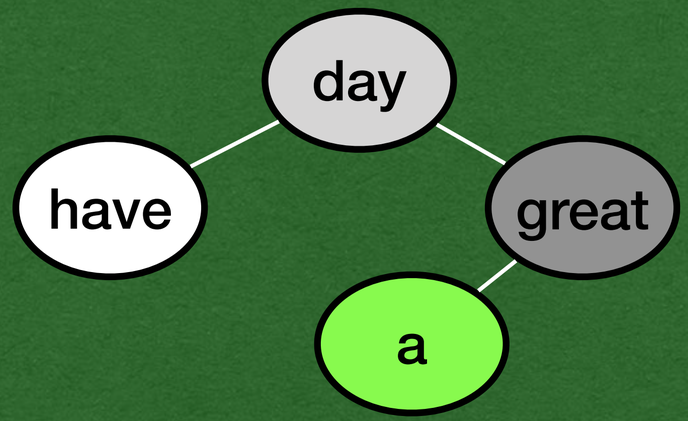
value	"great"
left	~~null~~ 0x936
right	null

BTNode 0x936

value	"a"
left	null
right	null

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

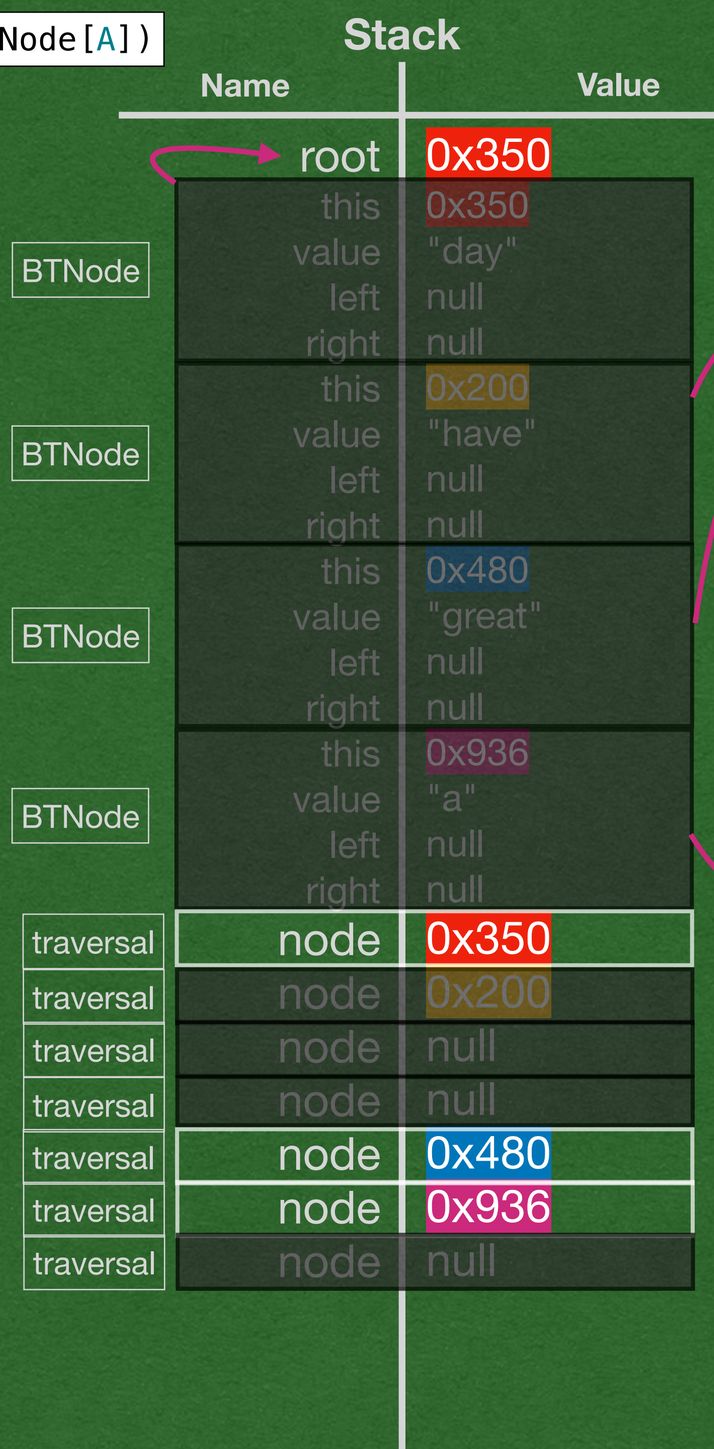
```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

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)
}
```

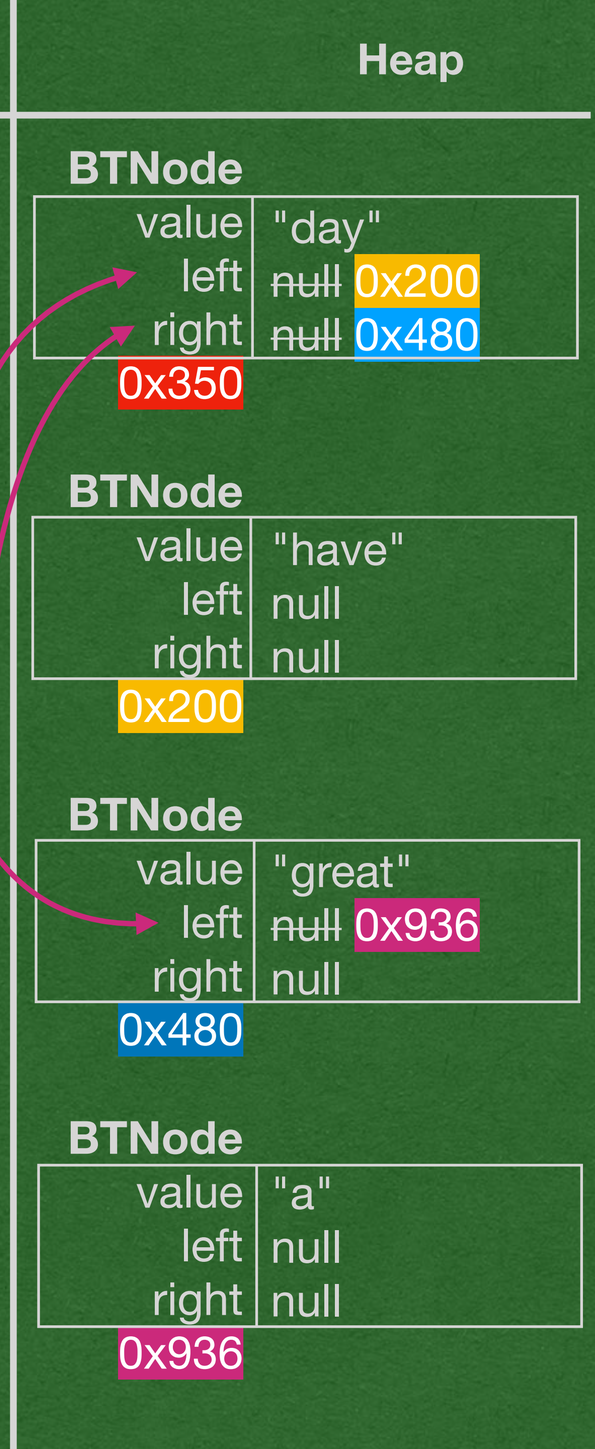


in/out

have

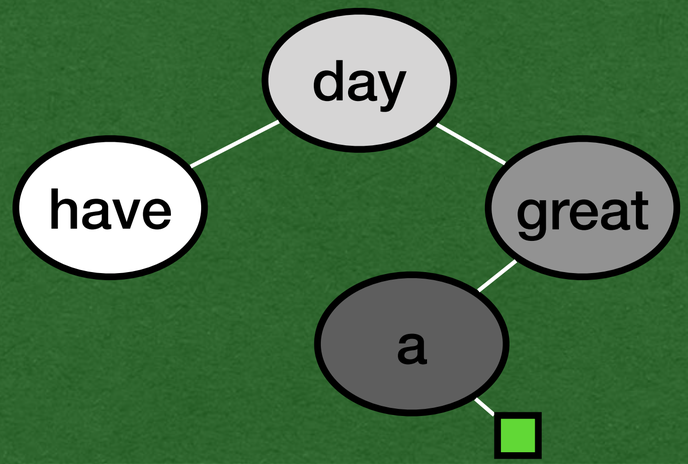
- Return to the previous frame and make the right recursive call





```
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

- Make the right recursive call of null

Stack

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

Heap

BTNode 0x350

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

BTNode 0x200

value	"have"
left	null
right	null

BTNode 0x480

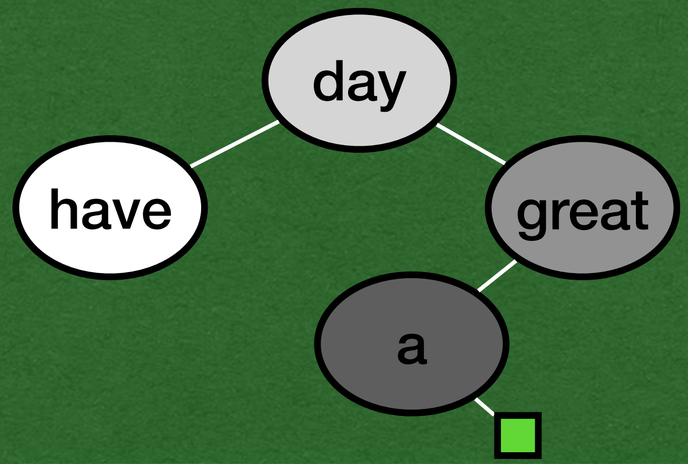
value	"great"
left	~~null~~ 0x936
right	null

BTNode 0x936

value	"a"
left	null
right	null

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

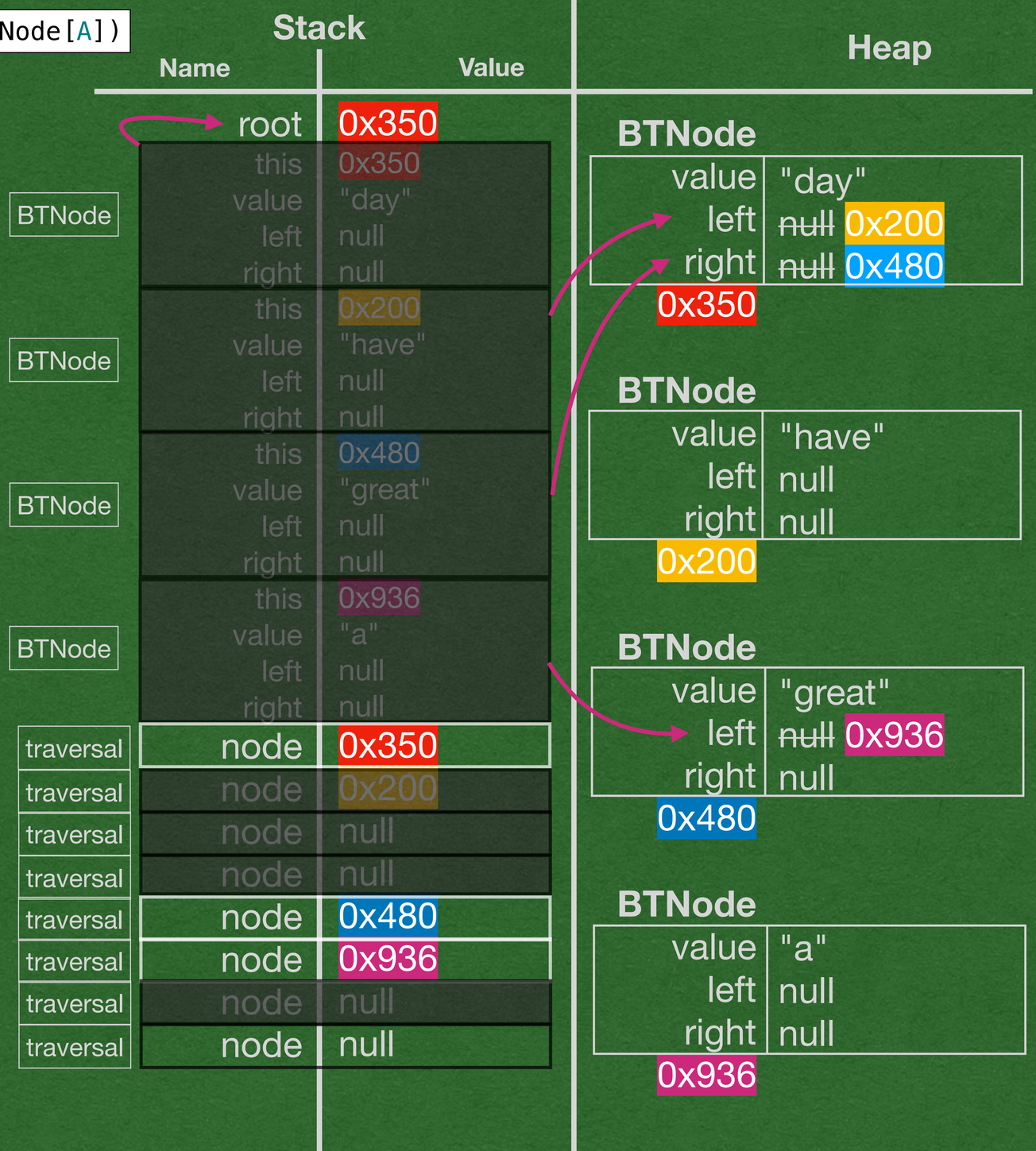
```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

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

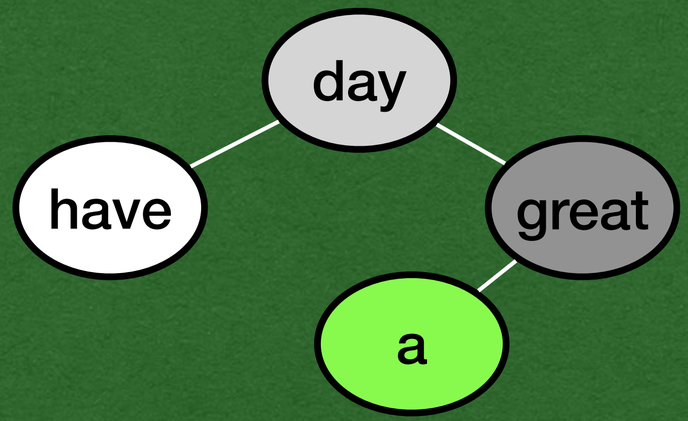
have

- 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)
}
```



in/out

have

- This stack frame already made both recursive calls

Stack

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

Heap

BTNode 0x350

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

BTNode 0x200

value	"have"
left	null
right	null

BTNode 0x480

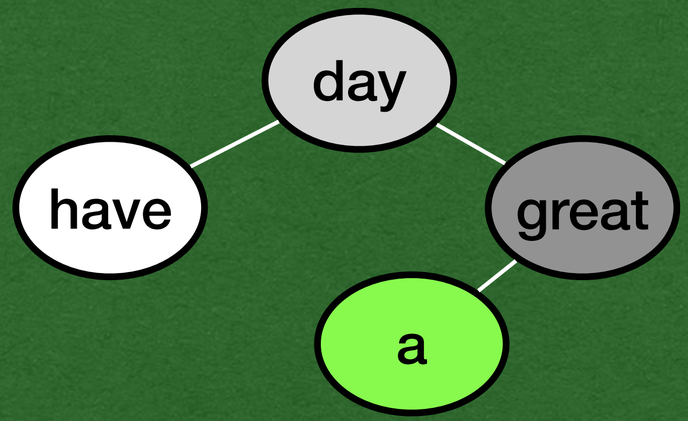
value	"great"
left	~~null~~ 0x936
right	null

BTNode 0x936

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 a

- Print "a" to the screen

Stack

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

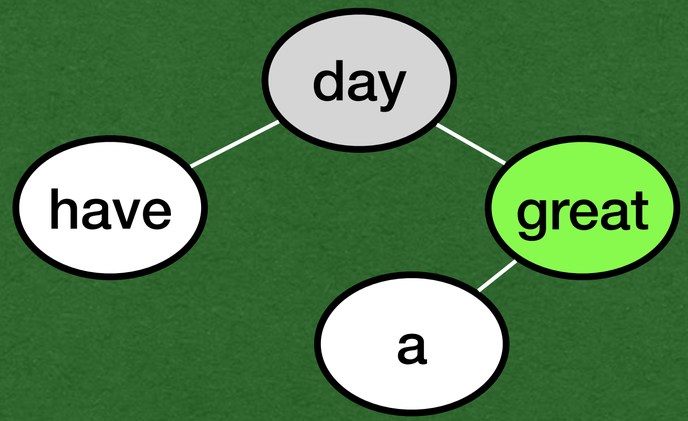
BTNode

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

- Return back to the previous stack frame

Stack

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

Heap

BTNode 0x350

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

BTNode 0x200

value	"have"
left	null
right	null

BTNode 0x480

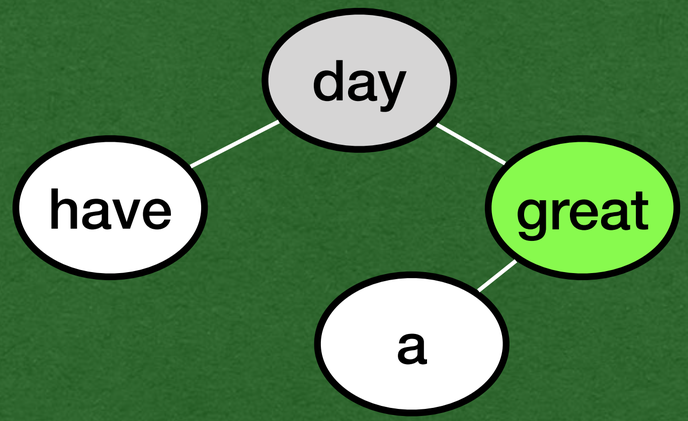
value	"great"
left	~~null~~ 0x936
right	null

BTNode 0x936

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 a

- Make right recursive call

Stack

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

Heap

BTNode 0x350

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

BTNode 0x200

value	"have"
left	null
right	null

BTNode 0x480

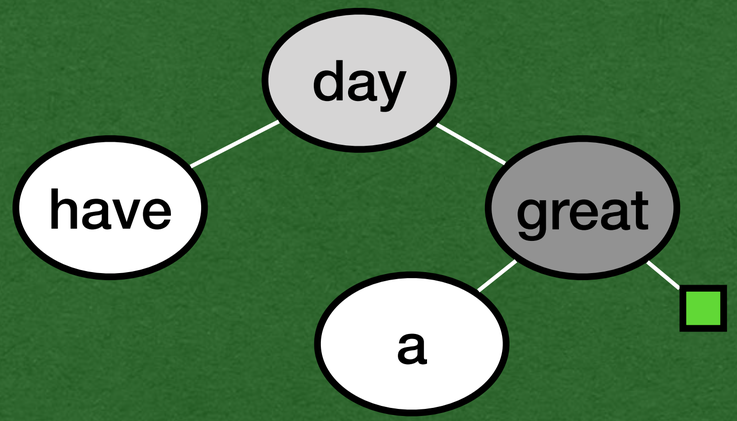
value	"great"
left	~~null~~ 0x936
right	null

BTNode 0x936

value	"a"
left	null
right	null

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

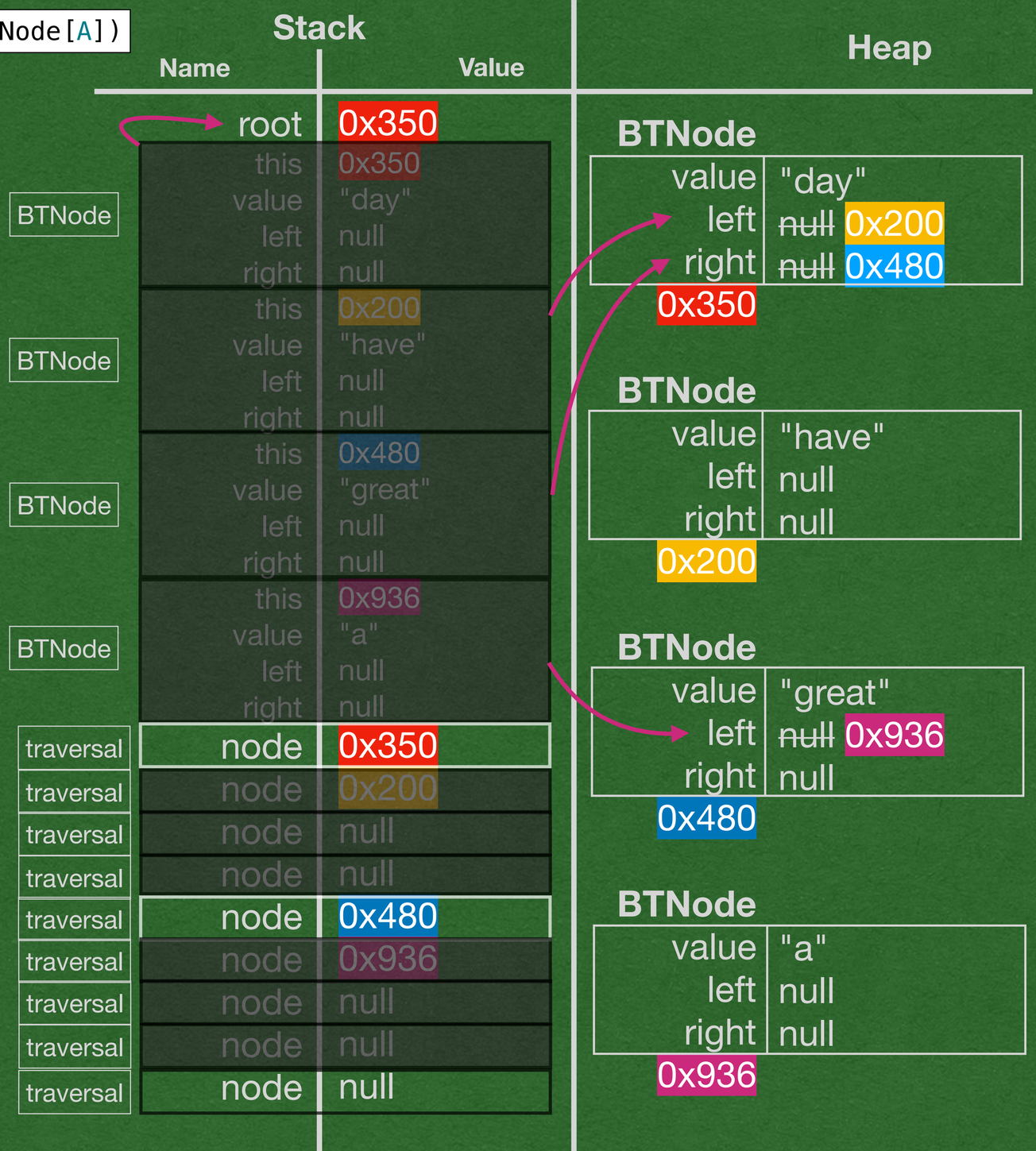
```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

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

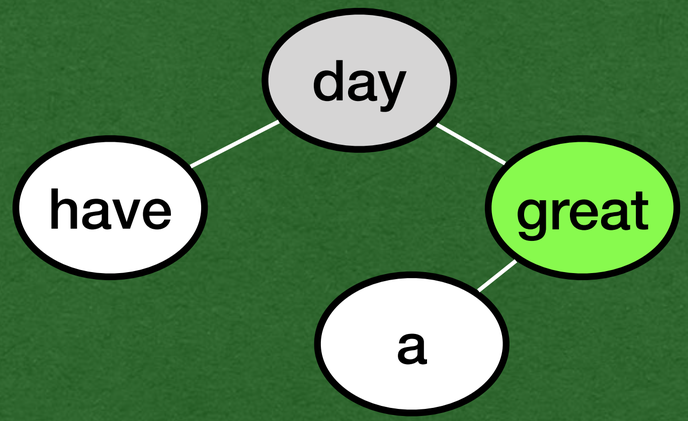
have a

- Base case of null
- Just return



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

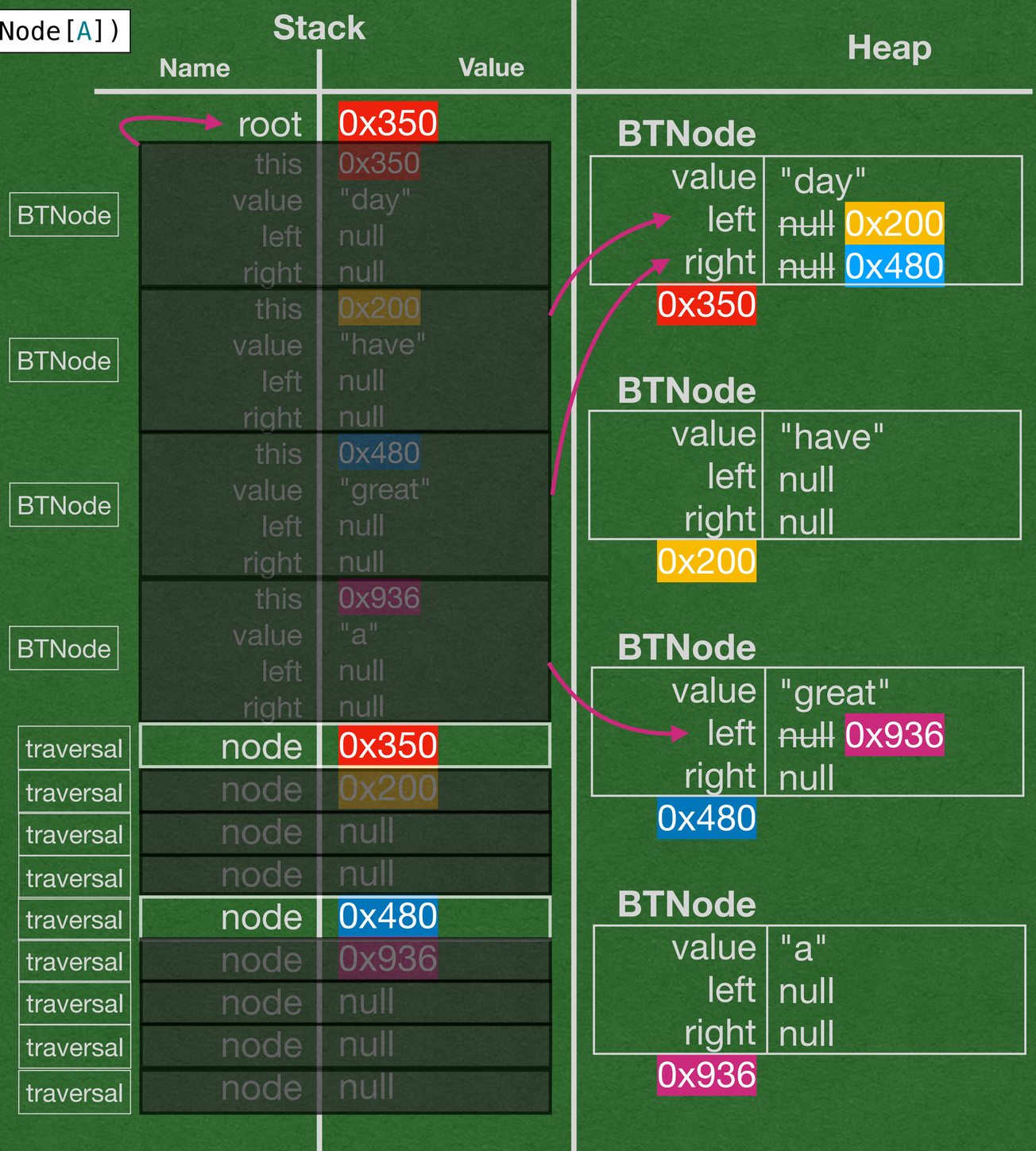
```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

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

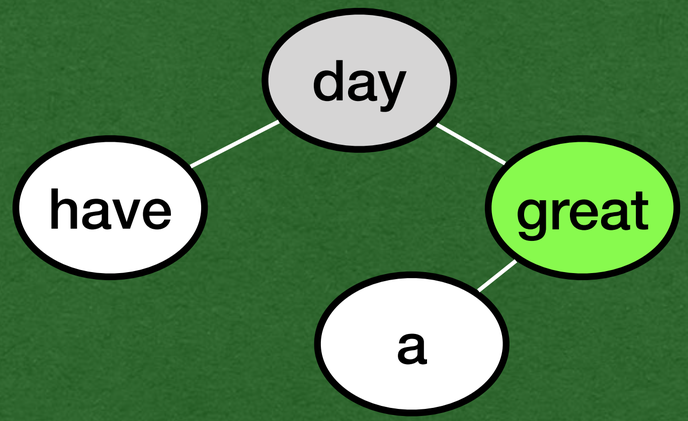
have a

- Stack frame done with both recursive calls



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

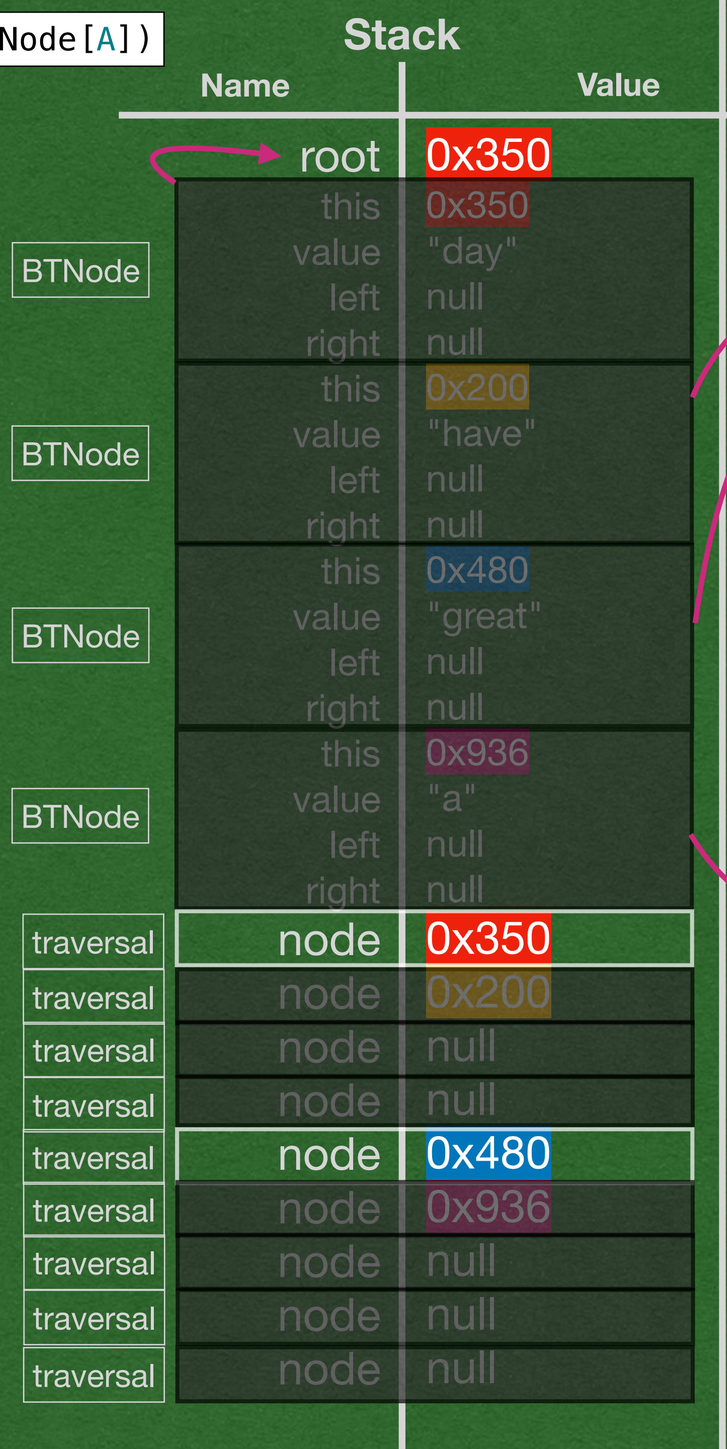
```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

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)
}
```

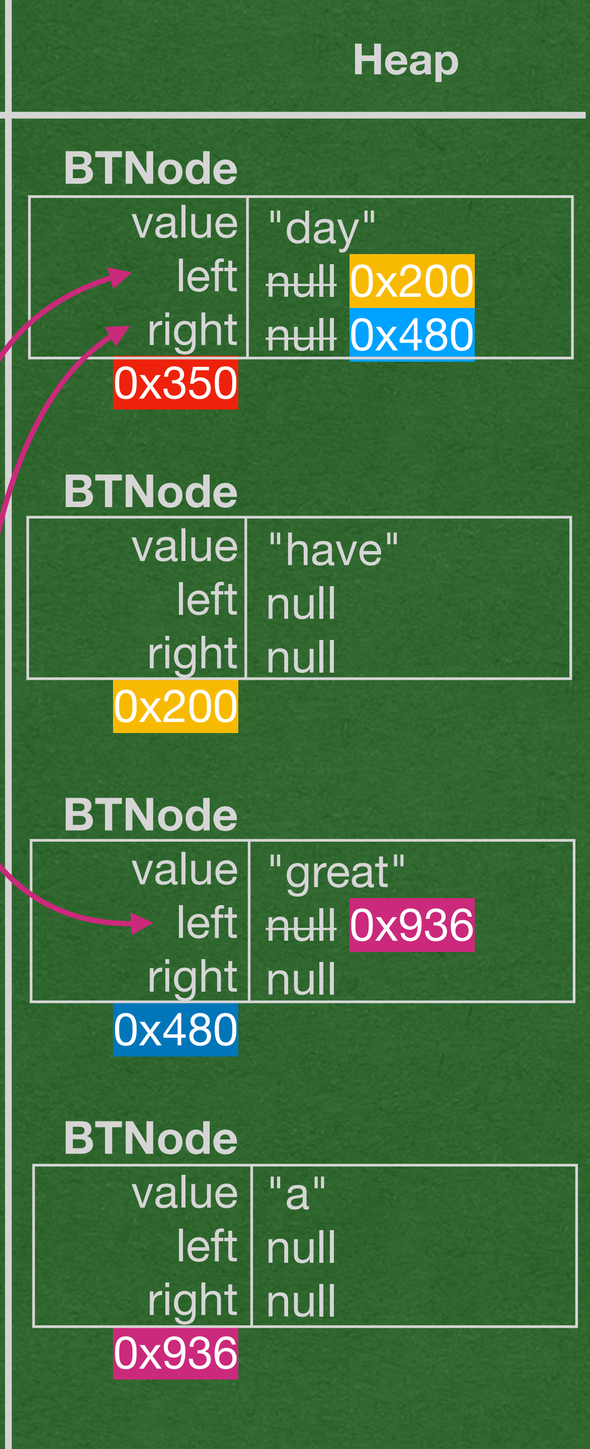


in/out

have a great

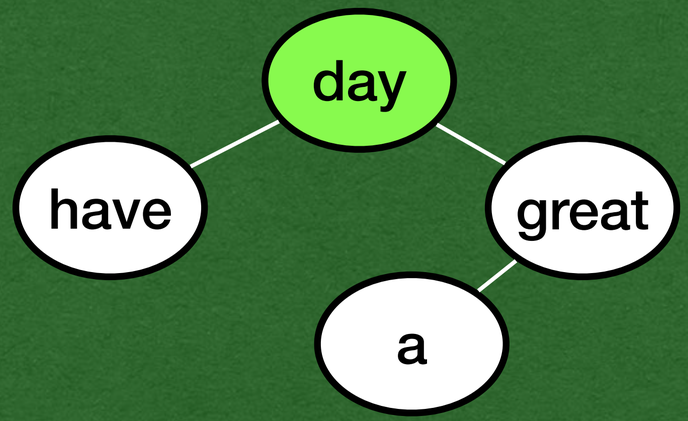
- Print "great" to the screen





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

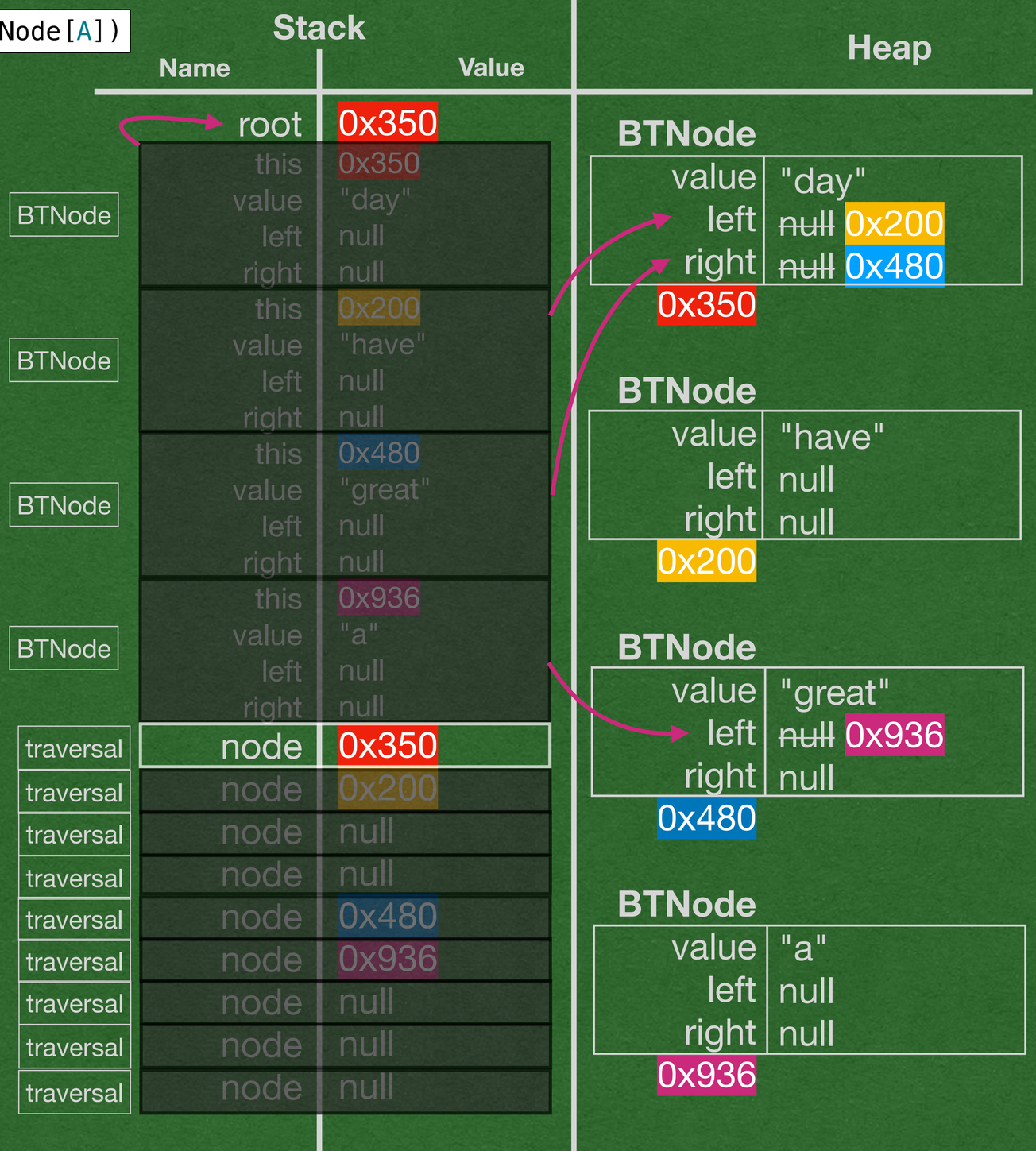
```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

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

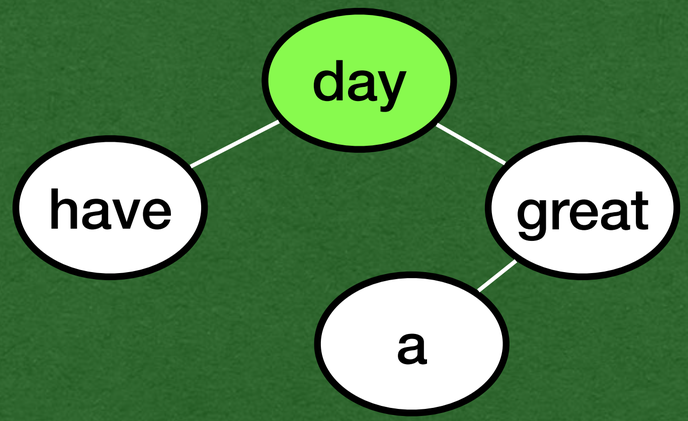
have a great

- Root node is now done with both recursive calls



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

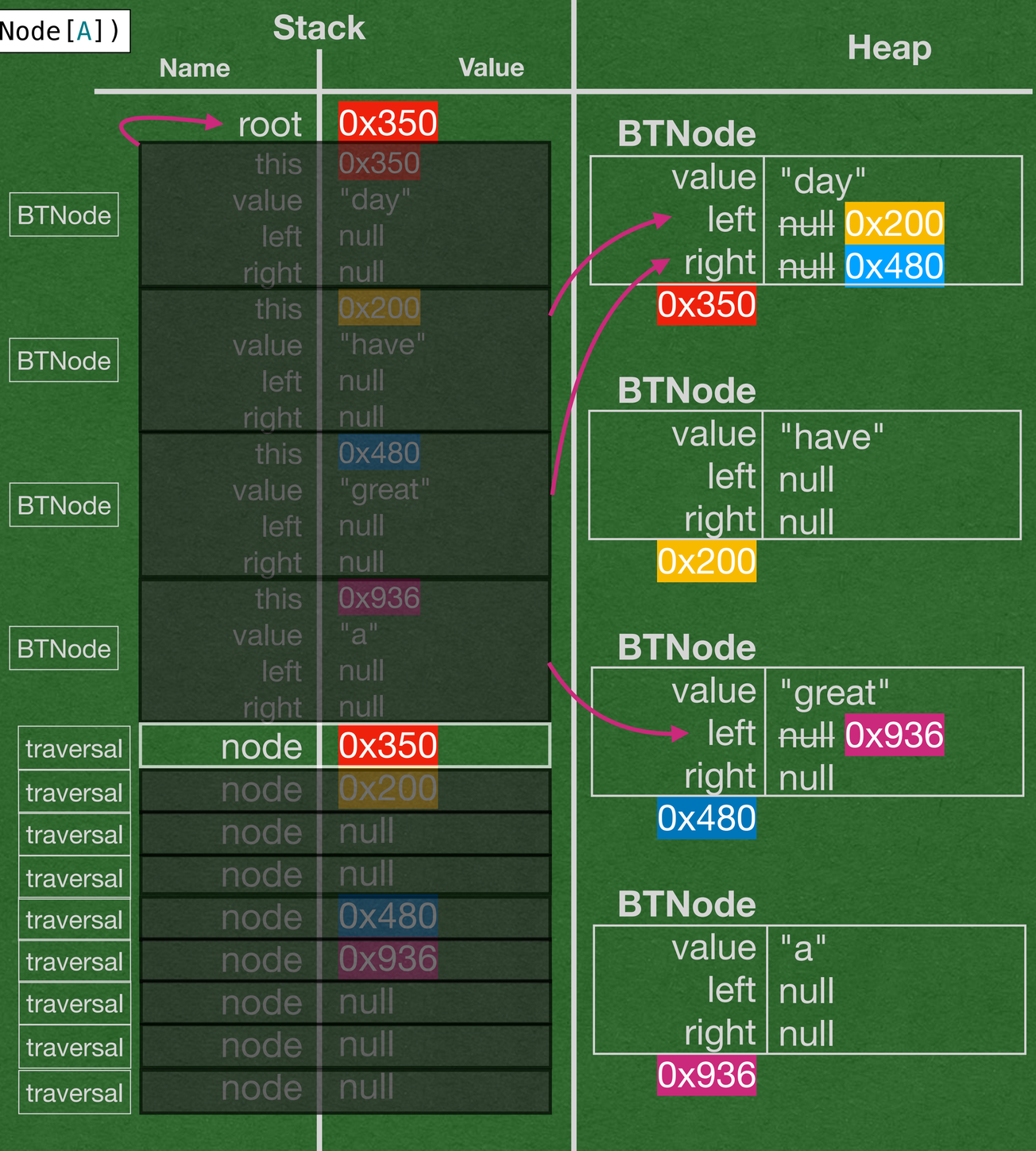
```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

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

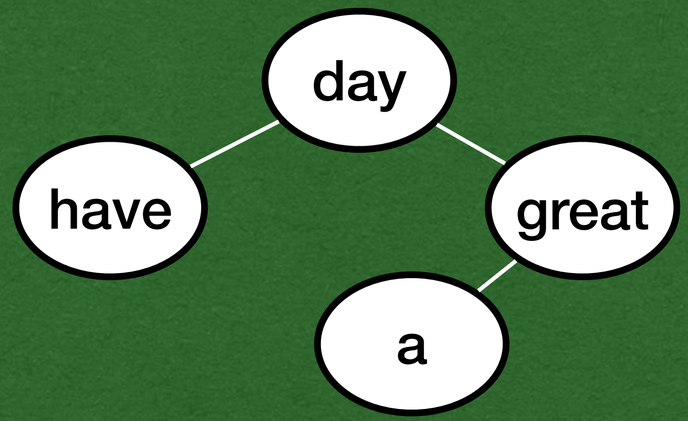
have a great day

- Print "day" to the screen



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

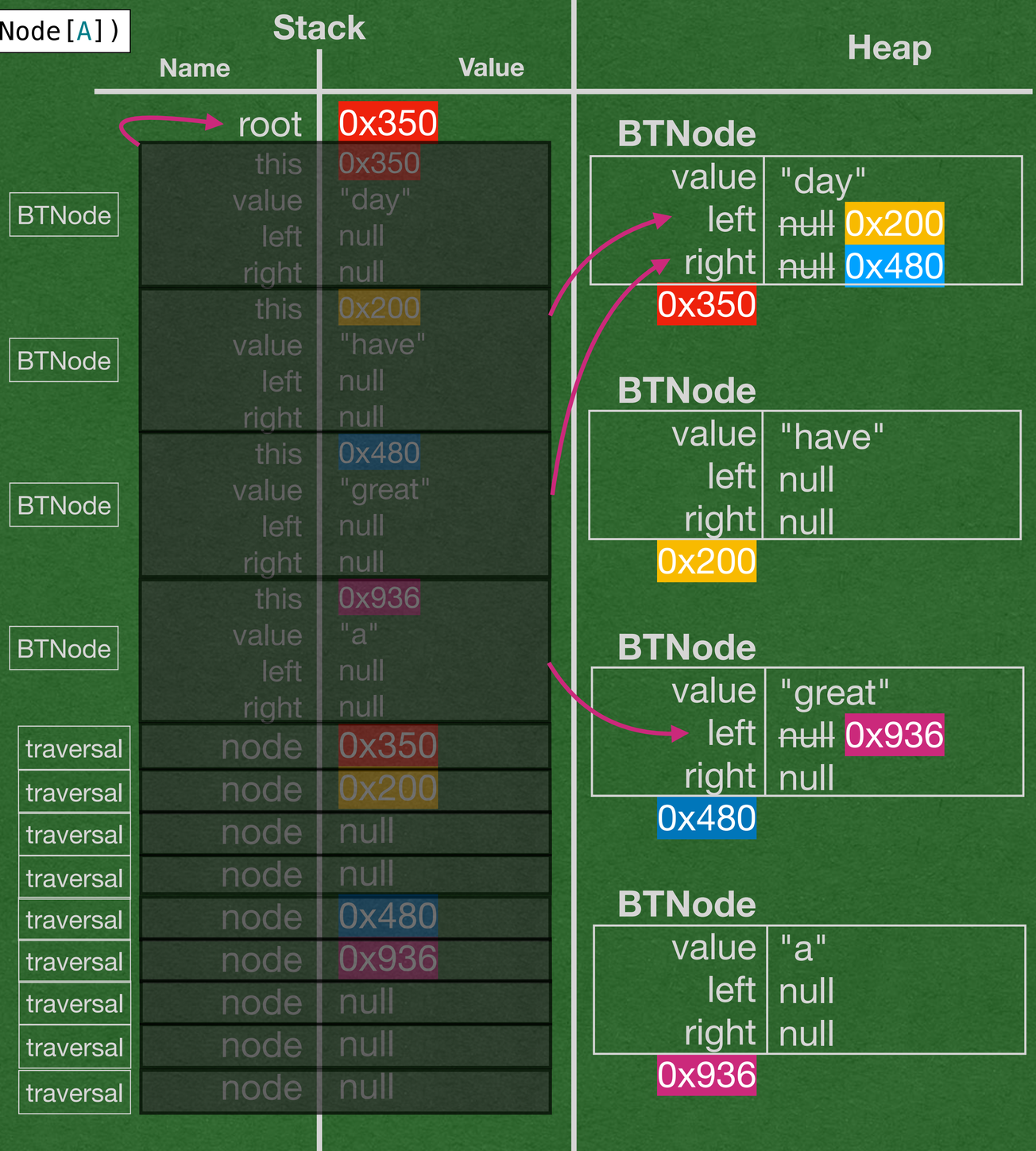
```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

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

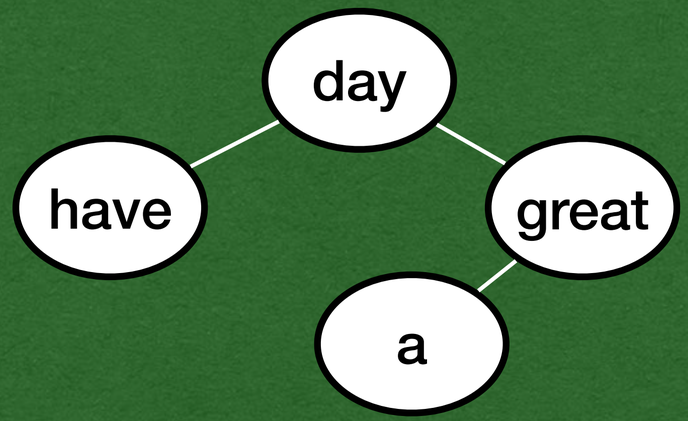
have a great day

- The traversal is over
- All stack frames have returned



```
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