AlgoExpert   Quad Layout   Swift   12px   Sublime   Monokai   00:00:

Our Solution(s)   Run Code

Solution 1   Solution 2

```
                    right.insert(value: value)
                } else {
                    right = BST(value: value)
                }
            }

            return self
        }

        // Average: O(log(n)) time | O(log(n)) space
        // Worst: O(n) time | O(n) space
        func contains(value: Int) -> Bool {
            if value < self.value {
                if let left = left {
                    return left.contains(value: value)
                } else {
                    return false
                }
            } else if value > self.value {
                if let right = right {
                    return right.contains(value: value)
                } else {
                    return false
                }
            } else {
                return true
            }
        }

        // Average: O(log(n)) time | O(log(n)) space
        // Worst: O(n) time | O(n) space
        func remove(value: Int?, parentNode: BST?) -> BST {
            if let valueToRemove = value, valueToRemove < self.value {
                if let left = left {
                    left.remove(value: value, parentNode: self)
                }
            } else if let valueToRemove = value, valueToRemove > self.value {
                if let right = right {
                    right.remove(value: value, parentNode: self)
                }
            } else {
                if let _ = left, let right = right {
                    self.value = right.getMinValue()

                    right.remove(value: self.value, parentNode: self)
                } else if parentNode === nil {
                    if let left = left {
                        self.value = left.value

                        right = left.right

                        self.left = left.left
                    } else if let right = right {
                        self.value = right.value

                        left = right.left

                        self.right = right.right
                    } else {
                        // This is a single-node tree; do nothing.
                    }
                } else if let parent = parentNode, let parentLeft = parent.left,
                    if let left = left {
                        parent.left = left
                    } else {
                        parent.left = right
```

Your Solutions   Run Code

Solution 1   Solution 2   Solution 3

```
class Program {
    class BST {
        var value: Int
        var left: BST?
        var right: BST?

        init(value: Int) {
            self.value = value
            left = nil
            right = nil
        }

        func insert(value: Int) -> BST {
            // Write your code here.
            return self
        }

        func contains(value: Int) -> Bool {
            // Write your code here.
            return false
        }

        func remove(value: Int?, parentNode: BST?) -> BST {
            // Write your code here.
            return self
        }
    }
}
```

Custom Output   Raw Output   Submit Code

```
                    }
                } else if let parent = parentNode, let parentRight = parentNode?.
                    if let left = left {
                        parent.right = left
                    } else {
                        parent.right = right
                    }
                }
            }

            return self
        }

        func getMinValue() -> Int {
            if let left = left {
                return left.getMinValue()
            } else {
                return value
            }
        }
    }
}
```

Run or submit code when you're ready.