AlgoExpert Quad Layout Go 12px Sublime Monokai 00:00:0

Our Solution(s) Run Code

Solution 1 Solution 2

```go
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

package main

type BST struct {
  Value int

  Left  *BST
  Right *BST
}

// Average: O(log(n)) time | O(1) space
// Worst: O(n) time | O(1) space
func (tree *BST) Insert(value int) *BST {
  current := tree
  for {
    if value < current.Value {
      if current.Left == nil {
        current.Left = &BST{Value: value}
        break
      } else {
        current = current.Left
      }
    } else {
      if current.Right == nil {
        current.Right = &BST{Value: value}
        break
      } else {
        current = current.Right
      }
    }
  }
  return tree
}

// Average: O(log(n)) time | O(1) space
// Worst: O(n) time | O(1) space
func (tree *BST) Contains(value int) bool {
  current := tree
  for current != nil {
    if value < current.Value {
      current = current.Left
    } else if value > current.Value {
      current = current.Right
    } else {
      return true
    }
  }
  return false
}

// Average: O(log(n)) time | O(1) space
// Worst: O(n) time | O(1) space
func (tree *BST) Remove(value int) *BST {
  tree.remove(value, nil)
  return tree
}

func (tree *BST) remove(value int, parent *BST) {
  current := tree
  for current != nil {
    if value < current.Value {
      parent = current
      current = current.Left
    } else if value > current.Value {
      parent = current
      current = current.Right
    } else {
      if current.Left != nil && current.Right != nil {
        current.Value = current.Right.getMinValue()
        current.Right.remove(current.Value, current)
      } else if parent == nil {
        if current.Left != nil {
          current.Value = current.Left.Value
          current.Right = current.Left.Right
          current.Left = current.Left.Left
        } else if current.Right != nil {
          current.Value = current.Right.Value
          current.Left = current.Right.Left
          current.Right = current.Right.Right
        } else {
          // This is a single-node tree; do nothing.
        }
      } else if parent.Left == current {
        if current.Left != nil {
          parent.Left = current.Left
        } else {
          parent.Left = current.Right
        }
      } else if parent.Right == current {
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```go
package main

// Do not edit the class below except for
// the insert, contains, and remove methods.
// Feel free to add new properties and methods
// to the class.
type BST struct {
  Value int

  Left  *BST
  Right *BST
}

func (tree *BST) Insert(value int) *BST {
  // Write your code here.
  // Do not edit the return statement of this method.
  return tree
}

func (tree *BST) Contains(value int) bool {
  // Write your code here.
  return false
}

func (tree *BST) Remove(value int) *BST {
  // Write your code here.
  // Do not edit the return statement of this method.
  return tree
}

```

Custom Output Raw Output Submit Code

```go
        if current.Left != nil {
          parent.Right = current.Left
        } else {
          parent.Right = current.Right
        }
      }
      break
    }
  }
}

func (tree *BST) getMinValue() int {
  if tree.Left == nil {
    return tree.Value
  }
  return tree.Left.getMinValue()
}

```

Run or submit code when you're ready.