Our Solution(s) Run Code

Solution 1 Solution 2

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

package main

import "math"

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) FindClosestValue(target int) int {
  return tree.findClosestValue(target, math.MaxInt32)
}

func (tree *BST) findClosestValue(target, closest int) int {
  currentnode := tree
  for currentnode != nil {
    if absdiff(target, closest) > absdiff(target, currentnode.value) {
      closest = currentnode.value
    }
    if target < currentnode.value {
      currentnode = currentnode.left
    } else if target > currentnode.value {
      currentnode = currentnode.right
    } else {
      break
    }
  }
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```go
package main

type BST struct {
  Value int

  Left  *BST
  Right *BST
}

func (tree *BST) FindClosestValue(target int) int {
  // Write your code here.
  return -1
}
```

Our Tests

Custom Output Submit Code

```
    tree := &BST{Value: nums[0]}
    for _, value := range values {
      tree.InsertValue(value)
    }
    return tree
}

func (tree *BST) InsertValue(int) {
  if value < tree.Value {
    if tree.Left == nil {
      tree.Left = &BST{Value: value}
    } else {
      tree.Left.InsertValue(value)
    }
  } else {
```

Run or submit code when you're ready.