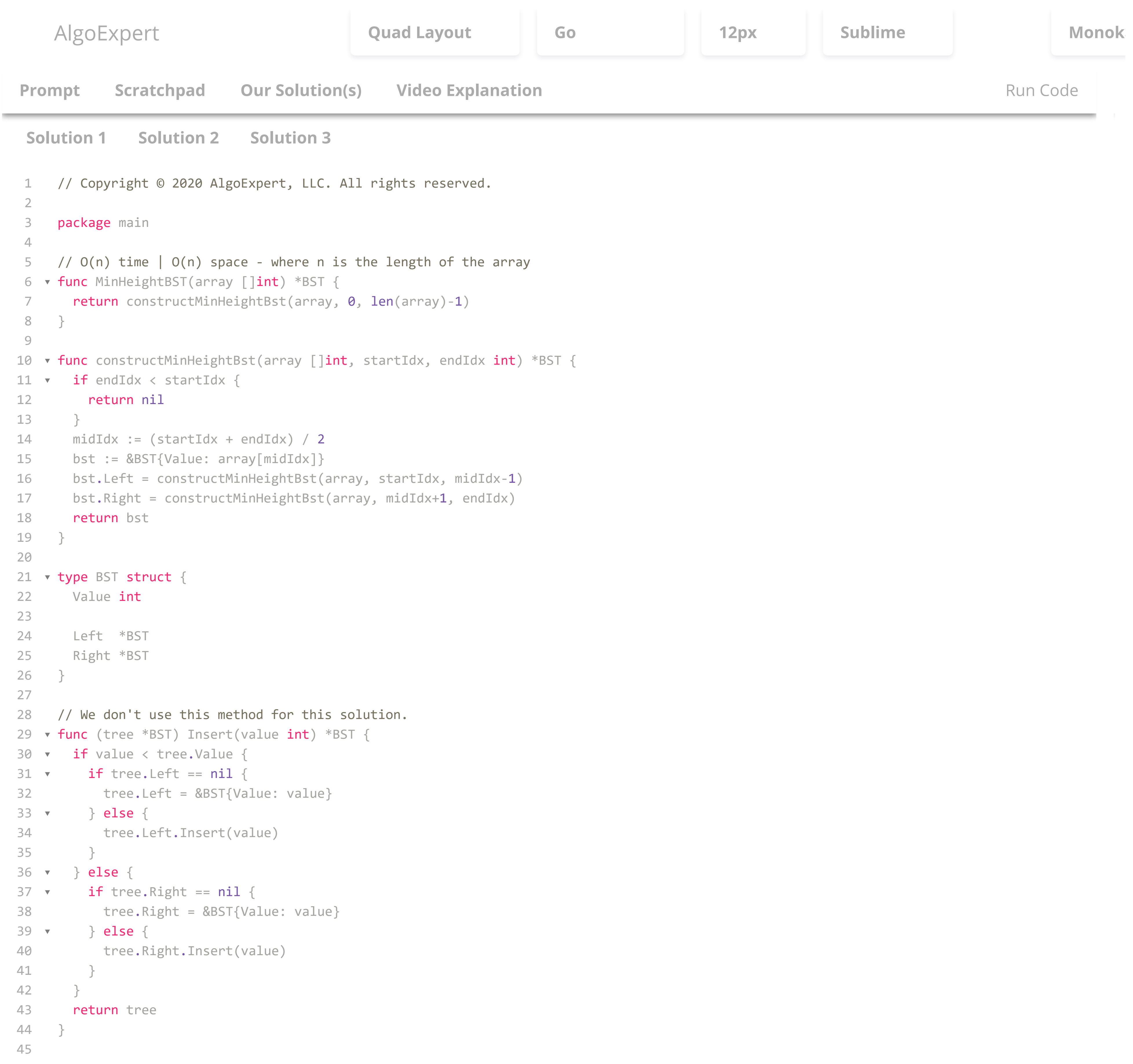

```go
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

package main

// O(n) time | O(n) space - where n is the length of the array
func MinHeightBST(array []int) *BST {
  return constructMinHeightBst(array, 0, len(array)-1)
}

func constructMinHeightBst(array []int, startIdx, endIdx int) *BST {
  if endIdx < startIdx {
    return nil
  }
  midIdx := (startIdx + endIdx) / 2
  bst := &BST{Value: array[midIdx]}
  bst.Left = constructMinHeightBst(array, startIdx, midIdx-1)
  bst.Right = constructMinHeightBst(array, midIdx+1, endIdx)
  return bst
}

type BST struct {
  Value int

  Left  *BST
  Right *BST
}

// We don't use this method for this solution.
func (tree *BST) Insert(value int) *BST {
  if value < tree.Value {
    if tree.Left == nil {
      tree.Left = &BST{Value: value}
    } else {
      tree.Left.Insert(value)
    }
  } else {
    if tree.Right == nil {
      tree.Right = &BST{Value: value}
    } else {
      tree.Right.Insert(value)
    }
  }
  return tree
}

```