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

class Program {
  // O(n) time | O(n) space - where n is the length of the array
  static func minHeightBST(_ array: [Int]) -> BST? {
    return constructMinHeightBst(array, nil, 0, array.count - 1)
  }

  static func constructMinHeightBst(_ array: [Int], _ bst: BST?, _ startIdx: Int, _ endIdx: Int) -> BST? {
    if endIdx < startIdx {
      return nil
    }

    var tree = bst
    var midIdx = (startIdx + endIdx) / 2
    var newBstNode = BST(value: array[midIdx])
    if let t = tree {
      if array[midIdx] < t.value {
        t.left = newBstNode
      } else {
        t.right = newBstNode
      }
    }

    tree = newBstNode
    constructMinHeightBst(array, tree, startIdx, midIdx - 1)
    constructMinHeightBst(array, tree, midIdx + 1, endIdx)
    return tree
  }

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

    init(value: Int) {
      self.value = value
    }

    // We don't use this method for this solution.
    func insert(value: Int) {
      if value < self.value {
        if let left = self.left {
          left.insert(value: value)
        } else {
          left = BST(value: value)
        }
      } else {
        if let right = self.right {
          right.insert(value: value)
        } else {
          right = BST(value: value)
        }
      }
    }
  }
}
```