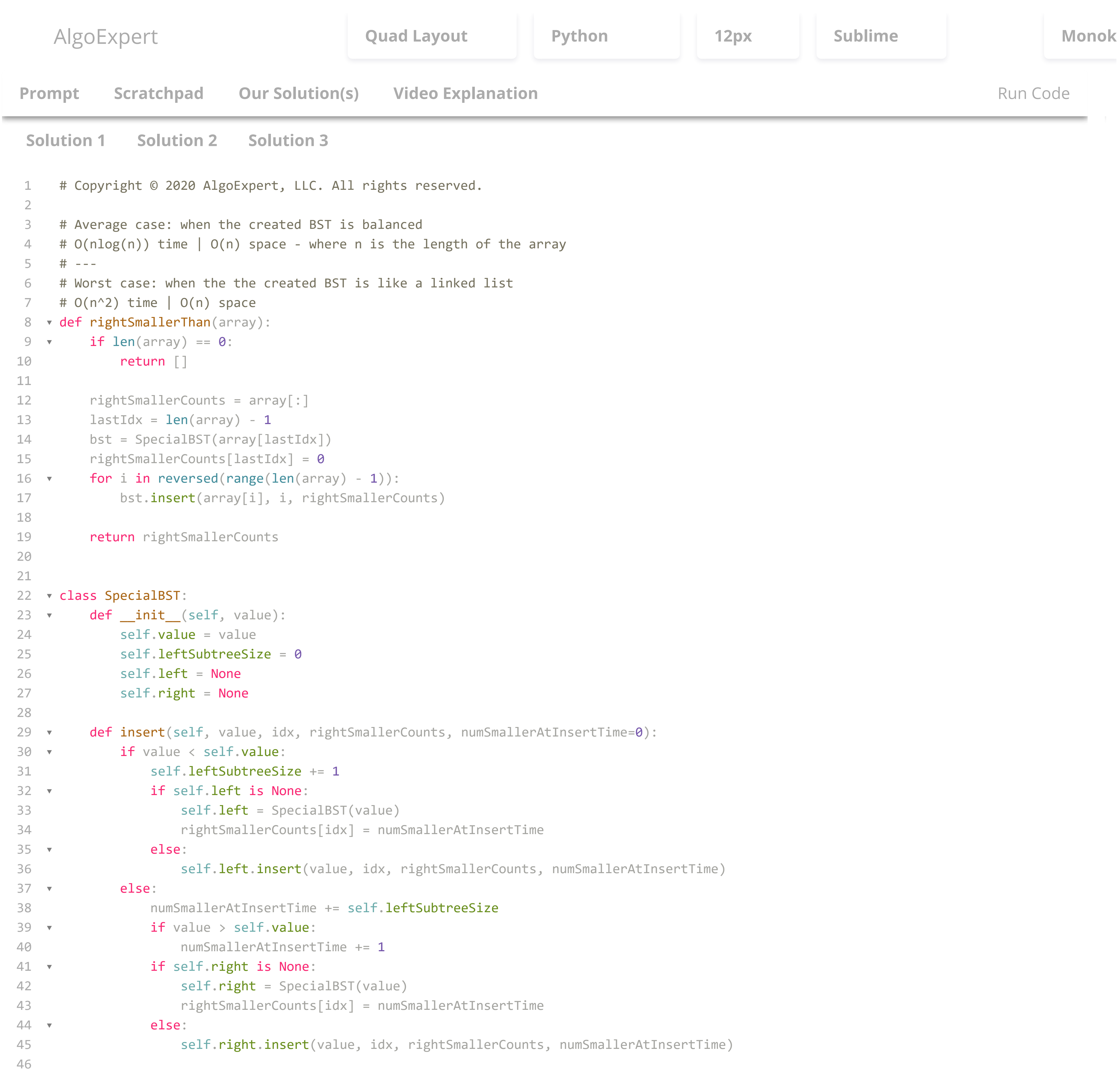

```python
# Copyright © 2020 AlgoExpert, LLC. All rights reserved.

# Average case: when the created BST is balanced
# O(nlog(n)) time | O(n) space - where n is the length of the array
# ---
# Worst case: when the the created BST is like a linked list
# O(n^2) time | O(n) space
def rightSmallerThan(array):
    if len(array) == 0:
        return []

    rightSmallerCounts = array[:]
    lastIdx = len(array) - 1
    bst = SpecialBST(array[lastIdx])
    rightSmallerCounts[lastIdx] = 0
    for i in reversed(range(len(array) - 1)):
        bst.insert(array[i], i, rightSmallerCounts)

    return rightSmallerCounts


class SpecialBST:
    def __init__(self, value):
        self.value = value
        self.leftSubtreeSize = 0
        self.left = None
        self.right = None

    def insert(self, value, idx, rightSmallerCounts, numSmallerAtInsertTime=0):
        if value < self.value:
            self.leftSubtreeSize += 1
            if self.left is None:
                self.left = SpecialBST(value)
                rightSmallerCounts[idx] = numSmallerAtInsertTime
            else:
                self.left.insert(value, idx, rightSmallerCounts, numSmallerAtInsertTime)
        else:
            numSmallerAtInsertTime += self.leftSubtreeSize
            if value > self.value:
                numSmallerAtInsertTime += 1
            if self.right is None:
                self.right = SpecialBST(value)
                rightSmallerCounts[idx] = numSmallerAtInsertTime
            else:
                self.right.insert(value, idx, rightSmallerCounts, numSmallerAtInsertTime)
```