```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 []

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

    rightSmallerCounts = array[:]
    getRightSmallerCounts(bst, rightSmallerCounts)
    return rightSmallerCounts


def getRightSmallerCounts(bst, rightSmallerCounts):
    if bst is None:
        return
    rightSmallerCounts[bst.idx] = bst.numSmallerAtInsertTime
    getRightSmallerCounts(bst.left, rightSmallerCounts)
    getRightSmallerCounts(bst.right, rightSmallerCounts)


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

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