Daily Coding Problem #165

Problem

This problem was asked by Google.

Given an array of integers, return a new array where each element in the new array is the number of smaller elements to the right of that element in the original input array.

For example, given the array `[3, 4, 9, 6, 1]`, return `[1, 1, 2, 1, 0]`, since:

- There is 1 smaller element to the right of 3
- There is 1 smaller element to the right of 4
- There are 2 smaller elements to the right of 9
- There is 1 smaller element to the right of 6
- There are no smaller elements to the right of 1

Solution

A naive solution for this problem would simply be to create a new array, and for each element count all the smallest elements to the right of it.

```python
def smaller_counts_naive(lst):
    result = []
    for i, num in enumerate(lst):
        count = sum(val < num for val in lst[i + 1:])
        result.append(count)
    return result
```

This takes O(n^2) time. Can we do this any faster?

To speed this up, we can try the following idea:

- Iterate backwards over the input list
- Maintain a sorted list `s` of the elements we've seen so far
- Look at `s` to see where the current element would fit in

The index will be how many elements on the right are smaller.

```python
import bisect

def smaller_counts(lst):
    result = []
    s = []
    for num in reversed(lst):
        i = bisect.bisect_left(s, num)
        result.append(i)
        bisect.insort(s, num)
    return list(reversed(result))
```

Now this only takes O(n log n) time and O(n) space.