1. Derive the formulas for

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                temp = arr[j]
                arr[j] = arr[j+1]
                arr[j+1] = temp
    return arr
```

Considering the above implementation of bubble sort (taken from the Sorting #1 slideshow):

(i) Number of comparisons

The outer loop runs a total of n times. The inner loop runs $n-i-1$ times. This works out to a pattern like the following: $n-1$ for i = 0. $n-2$ for i = 1 and so on. The result is an arithmetic series as follows: $(n-1)+(n-2)+(n-3)+...+3+2+1+0$. The sum of this series is given as $\frac{n(n-1)}{2}$, which is the number of comparisons. This number does not change for different arrangements of the array elements.

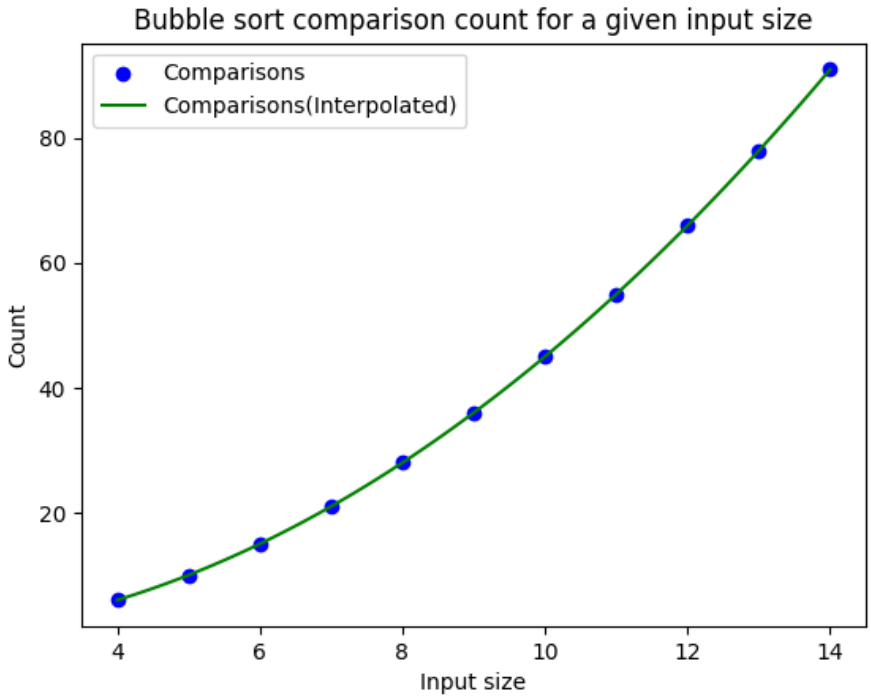
(ii) Average-case number of swaps for bubble sort

Firstly, the worst case number swaps occurs when the array is sorted backwards, in which case a swap is done for every pair of numbers. The first run through will result in $n-1$ swaps, the next $n-2$, then $n-3$ and so on. This matches the arithmetic series we found in the number of comparisons part, the sum of which is as follows: $(n-1)+(n-2)+(n-3)+...+3+2+1+0=\frac{n(n-1)}{2}$ swaps.

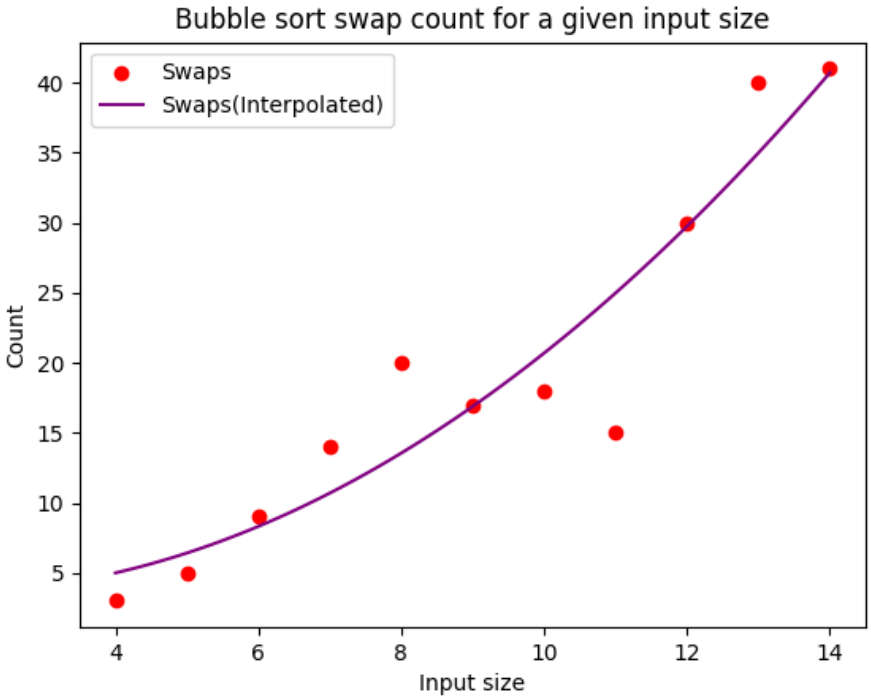
In the average case, only half of the comparisons will result in a swap, which results in $\frac{n(n-1)}{2}\times\frac{1}{2}=\frac{n(n-1)}{4}$ swaps.

4.

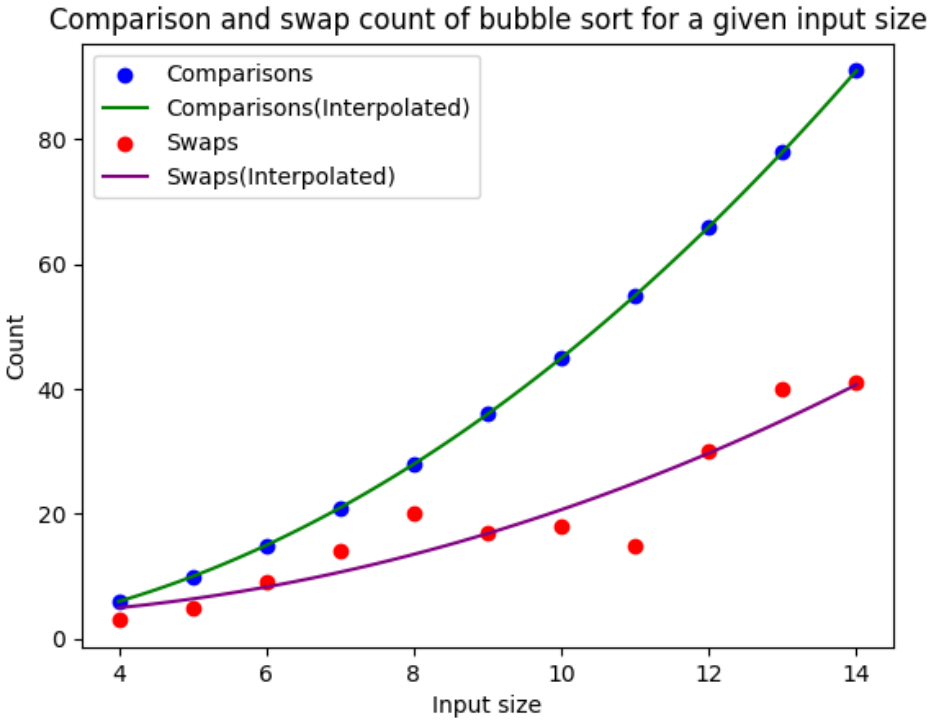
comparisons



swaps



both



Discussion:
The plots match my complexity analysis. As expected, both functions are quadratic. As well, the number of swaps is roughly half of the number of comparisons, which matches their respective functions of $\frac{n(n-1)}{2}$ and $\frac{n(n-1)}{4}$. The number of comparisons is constant for any case, whereas the number of swaps varies depending on the slight differences in the distributions.