Figure 19.3 Conclusions from the timing.

The numbers are rough, but they lead to a couple of broad conclusions:

1. "counting sort" just won't work on a PC for a list of $1,000,000$ items
2. we don't have really solid numbers, so we can only get a grasp on how things change as the list size grows; i.e. we don't care about exact values

The second conclusion means that we would like a functional form for the operation counts. Clearly,

$\boxed{\text{A list of } N \text{ items requires } N^2 \text{ comparisons and } 2N \text{ moves.}}$

Since the numbers are rough, we might as well ignore the $2N$ part; especially since as the list gets bigger it becomes relatively tiny.

So, we say that

- counting sort is *of order* N^2, or
- counting sort is $\mathcal{O}(N^2)$, or
- counting sort has $\mathcal{O}(N^2)$ *complexity*

"Complexity" in computing means *time taken*, not "complicatedness".

The first conclusion means that any $\mathcal{O}(N^2)$ sort algorithm will be unsatisfactory in the long run. Note that

- bubble sort is $\mathcal{O}(N^2/2)$
- insertion sort is $\mathcal{O}(N^2/4)$

The constant factors don't help. These algorithms are *all* unsatisfactory for large lists. So …

$\boxed{\text{… we need a better algorithm.}}$