Figure 19.2 The speed of “counting sort”.

We need to count data comparisons and moves.

Listing the relevant parts only,

```
for ( each item i in the list ) {
    int smallerCount = 0;        // we could omit this

    for ( each item j in the list )
        if ( item j < item i )        // COMPARISON
            smallerCount++;

    sortedList(smallerCount) = list[i];     // MOVE 1
}
for ( each item i in the list )
    list[i] = sortedList[i];                // MOVE 2
```

Some example counts and times:

list size	**# of moves and comparisons**				**execution time**
	COMPARE	**MOVE 1**	**MOVE 2**	**TOTAL**	
5	5^2	5	5	35	too small to measure
10	10^2	10	10	120	too small to measure
100	100^2	100	100	10,200	just noticeable
1,000	$1,000^2$	1,000	1,000	1,002,000	˜2 hours @ .01 secs/op
1,000,000	$1,000,000^2$	1,000,000	1,000,000	$\tilde{}1 \times 10^{12}$	˜300 years @ .01 secs/op

Note that:

- execution times are “rough” estimates for a modern PC
- the Toronto phone book has ˜1,000,000 names
- even at main-memory speed (.000001 secs/op),
  it would take us ˜12 days to sort the Toronto phone book!