Figure 19.1 Measuring the speed of an algorithm.

Suppose we want to know if our counting sort
is faster than our bubble sort.

Could we just code them, run them, and time their
execution with a stopwatch?

Perhaps, but the answer would be muddled by interfering factors:

- speed of computer
- choice of programming language
- choice of compiler and execution environment
- random events caused by other programs
  running during execution of the test
- quality of programming!

Instead, we isolate the "important operations"—that is,
the ones that take the longest—and *count* them.

What are these "important operations"?

- in scientific computing, floating-point operations
- in sorting, comparisons and data moves

For example, in counting sort, we had these statements:

```
if (list[j].compareTo(list[i]) < 0) …     // a comparison
sortedList[smallerCount] = list[i];     // a move
list[i] = sortedList[i];                // a move
```

Why are comparisons and moves so important?

- in a large list, data is kept on disk
- comparisons and moves require disk operations, which are
  a lot slower than operations involving RAM

```
int i = 0;     // conservatively, 0.000001 seconds
list[i] = sortedList[i];     // typically, 0.01 seconds
```

The disk operation is **10,000 times** slower!