expected, it might be silly to expend a great deal of effort to design a clever algorithm. On the other hand, there is a large market these days for rewriting programs that were written five years ago based on a no-longer-valid assumption of small input size. These programs are now too slow because they used poor algorithms. For large amounts of input, algorithm 4 is clearly the best choice (although algorithm 3 is still usable).

Second, the times given do not include the time required to read the input. For algorithm 4, the time merely to read the input from a disk is likely to be an order of magnitude larger than the time required to solve the problem. This is typical of many efficient algorithms. Reading the data is generally the bottleneck; once the data are read, the problem can be solved quickly. For inefficient algorithms this is not true, and significant computer resources must be used. Thus, it is important that, whenever possible, algorithms be efficient enough not to be the bottleneck of a problem.

Notice that for algorithm 4, which is linear, as the problem size increases by a factor of 10, so does the running time. Algorithm 2, which is quadratic, does not display this behavior; a tenfold increase in input size yields roughly a hundredfold (10^2) increase in running time. And algorithm 1, which is cubic, yields a thousandfold (10^3) increase in running time. We would expect algorithm 1 to take nearly 9,000 seconds (or two and a half hours) to complete for $N = 100{,}000$. Similarly, we would expect algorithm 2 to take roughly 333 seconds to complete for $N = 1{,}000{,}000$. However, it is possible that algorithm 2 could take somewhat longer to complete due to the fact that $N = 1{,}000{,}000$ could also yield slower memory accesses than $N = 100{,}000$ on modern computers, depending on the size of the memory cache.

Figure 2.3 shows the growth rates of the running times of the four algorithms. Even though this graph encompasses only values of N ranging from 10 to 100, the relative

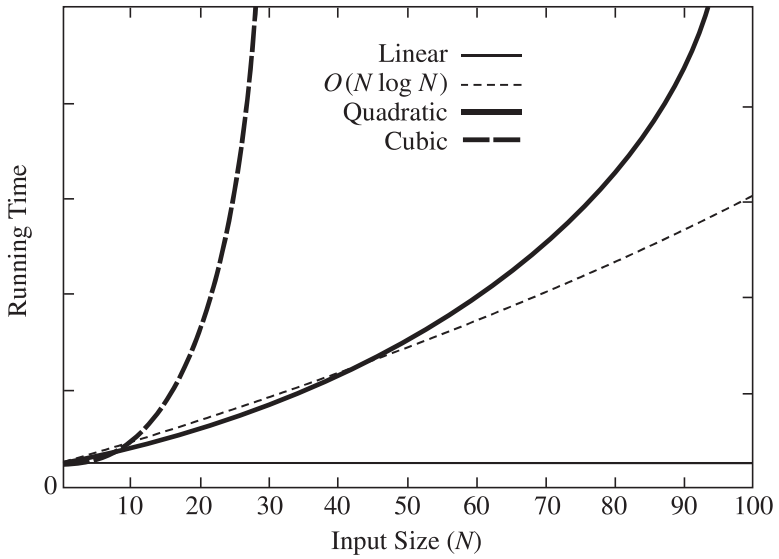

Figure 2.3 Plot (N vs. time) of various algorithms