

algorithm is nearly always far too slow to be used in practice, and even a much faster processor makes hardly any difference (see Example 2.4).

We say that an *algorithm* is *feasible* if it is fast enough to be used in practice. Likewise, we say that a *problem* is *feasible* if it can be solved by a feasible algorithm.

Algorithms of complexity up to $O(n \log n)$ are feasible. Algorithms of complexity $O(n^2)$ or $O(n^3)$ might be feasible, but only for small values of n . Algorithms of complexity $O(2^n)$ are infeasible, except possibly for very small values of n .

EXAMPLE 2.4 Growth rates

Suppose that we are given three algorithms that solve the same problem, with complexities $O(n)$, $O(n^2)$, and $O(2^n)$, respectively. Measurement shows that their actual running times on a particular processor are $0.1n$ seconds, $0.01n^2$ seconds, and 0.0001×2^n seconds, respectively, where n is the number of data items to be processed.

The following table shows the largest values of n for which the problem can be solved in a second, a minute, and an hour:

Algorithm	Running time (seconds)	Maximum n in 1 second	Maximum n in 1 minute	Maximum n in 1 hour
A	$0.1n$	10	600	36 000
B	$0.01n^2$	10	77	600
C	0.0001×2^n	9	15	21

In one second, all three algorithms can process the same amount of data (by coincidence). But there the similarity ends. In one minute, Algorithm A can process by far the most data, and Algorithm C the least. If an hour is allowed, Algorithm A is out of sight!

How much difference does it make if we use a processor that is ten times faster? This reduces each algorithm's running time by a factor of ten. The effects are as follows:

Algorithm	Running time (seconds)	Maximum n in 1 second	Maximum n in 1 minute	Maximum n in 1 hour
A	$0.01n$	100	6 000	360 000
B	$0.001n^2$	31	244	1 897
C	0.00001×2^n	13	19	25

Ironically, Algorithm A (already the fastest) benefits the most, and Algorithm C (already the slowest) benefits the least, from using the faster processor! Algorithm A can now process ten times as much data in any given time, Algorithm B about three times as much data, and Algorithm C only three extra data items.

Even if we handicap Algorithm A by leaving it to run on the slower processor, in as little as a minute it beats Algorithm B and outclasses Algorithm C.

The moral of Example 2.4 is this. Constant factors are important only when comparing algorithms of the same time complexity, such as two $O(n)$ algorithms, or two $O(n^2)$ algorithms. But an $O(n)$ algorithm will beat an $O(n^2)$ algorithm, sooner or later, regard-