be content to use the rougher estimates above for the time needed for a multiplication.

In general, when estimating the number of bit operations required to do something, the first step is to decide upon and write down an outline of a detailed procedure for performing the task. An explicit step-by-step procedure for doing calculations is called an *algorithm.* Of course, there may be many different algorithms for doing the same thing. One may choose to use the one that is easiest to write down, or one may choose to use the fastest one known, or else one may choose to compromise and make a trade-off between simplicity and speed. The algorithm used above for multiplying n by m is far from the fastest one known. But it is certainly a lot faster than repeated addition (adding n to itself m times).

Example 10. Estimate the time required to convert a k-bit integer to its representation in the base 10.

Solution. Let n be a k-bit integer written in binary. The conversion algorithm is as follows. Divide $10 = (1010)_2$ into n. The remainder — which will be one of the integers 0, 1, 10, 11, 100, 101, 110, 111, 1000, or 1001 — will be the ones digit d_0. Now replace n by the quotient and repeat the process, dividing that quotient by $(1010)_2$, using the remainder as d_1 and the quotient as the next number into which to divide $(1010)_2$. This process must be repeated a number of times equal to the number of decimal digits in n, which is $\left[\frac{log\, n}{log\, 10}\right]+1 = O(k)$. Then we're done. (We might want to take our list of decimal digits, i.e., of remainders from all the divisions, and convert them to the more familiar notation by replacing $0, 1, 10, 11, \ldots, 1001$ by $0, 1, 2, 3, \ldots, 9$, respectively.) How many bit operations does this all take? Well, we have $O(k)$ divisions, each requiring $O(4k)$ operations (dividing a number with at most k bits by the 4-bit number $(1010)_2$). But $O(4k)$ is the same as $O(k)$ (constant factors don't matter in the big-O notation), so we conclude that the total number of bit operations is $O(k) \cdot O(k) = O(k^2)$. If we want to express this in terms of n rather than k, then since $k = O(log\, n)$, we can write

$$\text{Time(convert } n \text{ to decimal)} = O(log^2 n).$$

Example 11. Estimate the time required to convert a k-bit integer n to its representation in the base b, where b might be very large.

Solution. Using the same algorithm as in Example 10, except dividing now by the ℓ-bit integer b, we find that each division now takes longer (if ℓ is large), namely, $O(k\ell)$ bit operations. How many times do we have to divide? Here notice that the number of base-b digits in n is $O(k/\ell)$ (see Example 9(c)). Thus, the total number of bit operations required to do all of the necessary divisions is $O(k/\ell) \cdot O(k\ell) = O(k^2)$. This turns out to be the same answer as in Example 10. That is, our estimate for the conversion time does not depend upon the base to which we're converting (no matter how large it may be). This is because the greater time required to find each digit is offset by the fact that there are fewer digits to be found.