bits of the dividend so that the resulting integer is greater than the divisor), carry down digits, etc. So our estimate is simply $(k-\ell+1)\ell$, which is $\leq k\ell$.

Example 8. Find an upper bound for the number of bit operations it takes to compute the binomial coefficient $\binom{n}{m}$.

Solution. Since $\binom{n}{m} = \binom{n}{n-m}$, without loss of generality we may assume that $m \leq n/2$. Let us use the following procedure to compute $\binom{n}{m} =$ $= n(n-1)(n-2)\cdots(n-m+1)/(2{\cdot}3\cdots m)$. We have $m-1$ multiplications followed by $m-1$ divisions. In each case the maximum possible size of the first number in the multiplication or division is $n(n-1)(n-2)\cdots(n-m+1) < n^m$, and a bound for the second number is n. Thus, by the same argument used in the solution to Example 6, we see that a bound for the total number of bit operations is $2(m-1)m([log_2 n]+1)^2$, which for large m and n is essentially $2m^2(log_2 n)^2$.

We now discuss a very convenient notation for summarizing the situation with time estimates.

The big-O notation. Suppose that $f(n)$ and $g(n)$ are functions of the positive integers n which take *positive* (but not necessarily integer) values for all n. We say that $f(n) = O(g(n))$ (or simply that $f = O(g)$) if there exists a constant C such that $f(n)$ is always less than $C \cdot g(n)$. For example, $2n^2 + 3n - 3 = O(n^2)$ (namely, it is not hard to prove that the left side is always less than $3n^2$).

Because we want to use the big-O notation in more general situations, we shall give a more all-encompassing definition. Namely, we shall allow f and g to be functions of several variables, and we shall not be concerned about the relation between f and g for small values of n. Just as in the study of limits as $n \longrightarrow \infty$ in calculus, here also we shall only be concerned with large values of n.

Definition. Let $f(n_1, n_2, \ldots, n_r)$ and $g(n_1, n_2, \ldots, n_r)$ be two functions whose domains are subsets of the set of all r-tuples of positive integers. Suppose that there exist constants B and C such that whenever all of the n_j are greater than B the two functions are defined and positive, and $f(n_1, n_2, \ldots, n_r) < C\, g(n_1, n_2, \ldots, n_r)$. In that case we say that f is *bounded* by g and we write $f = O(g)$.

Note that the "=" in the notation $f = O(g)$ should be thought of as more like a "<" and the big-O should be thought of as meaning "some constant multiple."

Example 9. (a) Let $f(n)$ be *any* polynomial of degree d whose leading coefficient is positive. Then it is easy to prove that $f(n) = O(n^d)$. More generally, one can prove that $f = O(g)$ in any situation when $f(n)/g(n)$ has a finite limit as $n \longrightarrow \infty$.

(b) If ϵ is any positive number, no matter how small, then one can prove that $log\, n = O(n^\epsilon)$ (i.e., for large n, the log function is smaller than any power function, no matter how small the power). In fact, this follows because $lim_{n\to\infty} \frac{log\, n}{n^\epsilon} = 0$, as one can prove using l'Hôpital's rule.