Problem: Calculate compound interest.
Assume $P_0 = 10,000$, interest rate 11%.
First technique: Recurrence Equation
$P_n = P_{n-1} \times 1.11$
Big O: $O(n)$ (n is the number of years)
$P_1 = P_0 \times 1.11$
$\vdots$
$P_{30} = P_{29} \times 1.11$
$P_{100} = P_0 \times 1.11^{100}$

"Shortcut" for $\sum_{i=1}^{n} i$ with $O(1)$ complexity (closed form)

Inverting the order of terms of the sum and adding them to the original sum, we will get n elements. Each element is going to equal $n + 1$. Since we added the same sum in the opposite order, we have to divide the final result by 2.

So, a formula for this kind of sum is $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

1 Section 3. Algorithms

procedure max($a_1, a_2, a_3, ..., a_n$: integers)
    max := a_1
    for 1:= 2 to n
        if $a_i >$ max then
            max := a_i
    Big O is $O(n)$

function exponential(b: positive integer, a: non-negative integer)
    if $a = 0$ then
        result := 1
    else
        result := b · exponential($b,\ a - 1$)
    return result
    Big O is $O(a)$

function fastExpo(b: positive integer, a: non-negative integer)
    if a = 0 then
        result := 1
    else if e % 2 = 0 then
        half := fastExpo(b, a/2)
        result := half · half
    else
        result := b · fastExpo(b, a-1)
    return result

Example: $3^{12} = 3^6 \cdot 3^6 = g \cdot g$
Big O is $O(\log n)$
The fastExpo function is an example of "divide-and-conquer".

There are also, for instance, dynamic programming (example: longest common subsequence), and greedy algorithm.