General Plan for Analyzing the Time Efficiency of Recursive Algorithms

1. Decide on a parameter (or parameters) indicating an input's size.

2. Identify the algorithm's basic operation.

3. Check whether the number of times the basic operation is executed can vary on different inputs of the same size; if it can, the worst-case, average-case and best-case efficiencies must be investigated separately.

4. Set up a recurrence relation, with an appropriate initial condition, for the number of times the basic operation is executed.

5. Solve the recurrence or, at least, ascertain the order of growth of its solution.

Analysis of Recursive Algorithms

What is a recursive algorithm?

Example: Factorial

$n!$ = 1*2*3...n and 0! = 1 (called initial case)
So the recursive defintiion $n!$ = n*(n-1)!

Algorithm $F(n)$
    if n = 0 **then** return 1 // base case
    else F(n-1)*n // recursive call

Basic operation is multiplication during the recursive call

Formula for multiplication

$$M(n) = M(n-1) + 1$$

is a recursive formula too.

We need the initial case which corresponds to the base case

$M(0) = 0$

There are no multiplications

Solve by the method of *backward substitutions*

$M(n) = M(n\text{-}1) + 1$

$= [M(n\text{-}2) + 1] + 1 = M(n\text{-}2) + 2$ substituted $M(n\text{-}2)$ for $M(n\text{-}1)$

$= [M(n\text{-}3) + 1] + 2 = M(n\text{-}3) + 3$ substituted $M(n\text{-}3)$ for $M(n\text{-}2)$

... a pattern evolves

$= M(0) + n$

$= n$

Therefore $M(n) \varepsilon \Theta(n)$