Analysis of Recursive algorithms

- Read 2.4, pages 70-76
  - Study the algorithm in Example 1, for computing the factorial function.
    - What do we consider as the problem's size? Why is this not entirely correct?
    - What do we consider as the algorithm's basic operation?
    - Explain the recurrence relation, at the top of page 71, for the number $M(n)$ of multiplications needed to compute the n-th fibonacci number.
    - What else do we need in addition to the recurrence relation, in order to compute $M(n)$? Where is that number coming from?
    - Why is $M(0) = 0$?
    - Use the *method of backward substitutions* to solve this recurrence relation.
  - What are the five steps followed in the plan to analyze the time efficiency of recursive algorithms? Where does it differ with the plan for non-recursive algorithms?
  - Study the Towers of Hanoi recursive algorithm in Example 2.
    - Explain the recurrence relation for the number $M(n)$ of moves needed, described at the top of page 74.
    - Use the *method of backward substitutions* to solve this recurrence relation.
    - Look at the tree-based description at the bottom of page 75. How does that computation relate to the computation of $M(n)$ you just did? Do they count the same things?
  - Study the "number of digits in binary representation" algorithm of example 3.
    - Explain why this algorithm would correctly compute the number of binary digits that the input number's representation uses.
    - Explain why it is enough to consider the algorithm's time efficiency in the case where n is a power of 2.
    - Explain the final formula for the time efficiency class of this algorithm.
  - Practice problems: 2.4.1, 2.4.3, 2.4.4, 2.4.9
  - Challenge: 2.4.13, 2.4.14