# Problem Set 7

A *recurrence relation* is a useful mathematical tool to describe sequences[1] where the $n$:th number is given by some combination of earlier numbers in the sequence. For instance, the Fibonacci Sequence[2] is a fascinating sequence. The first few Fibonacci numbers are $(0, 1, 1, 2, 3, 5, 8, 12)$ and without previous knowledge it is difficult to see which number follows 12. The rule is that each number is given as a addition of two smaller Fibonacci numbers, subject to some initial conditions.

We can neatly describe all the Fibonacci numbers using the following recurrence relation[3]:

$$F_n = \begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad \text{for } n > 1 \end{cases}$$

**Question 1.** Provide the recurrence relation for the factorial function and explain how the output of the factorial function can be described as a sequence.

**Question 2.** We can also use recurrence relations to express the **time complexity** of certain algorithms. For instance, the general formula for the (worst-case) time complexity of the class of Decrease-by-one algorithms can be expressed as:

$$T(n) = T(n-1) + f(n)$$

where $f(n)$ is the time needed to reduce a problem of size $n$ to a smaller one, and to combine it to a larger solution.

Give the general formula for the time-complexity, together with instance of an algorithm, of the following classes of algorithms:

a. Decrease-by-a-Constant-Factor.

b. Divide-and-Conquer.

**Question 3.**

a. Given the list $[18, 5, 9, 1, 0, 3, 12, 6]$ draw the procedure of the Mergesort algorithm (Levitin Ch. 5.1) on this list.

b. In your own words, give a detailed description of how the `Merge` part of Mergesort works, and provide both a worst-case and best-case time complexity for this algorithm.

*Hint:* Assume you are given two lists: $low = [a_0, ..., a_{\lfloor n/2 \rfloor}]$, $high = [a_{\lfloor n/2 \rfloor + 1}, ..., a_n]$

c. Let $n$ some power of two. Sketch the state-space of the Mergesort algorithm on a list of $n$ elements where the value of each state is the size of the subproblem. For each level of this tree, provide the total time for the `Merge` algorithm over the states.

*Hint:* The root is a node with value $n$ and there should be $n$ leaf nodes of your tree, all with value 1.

[1] A sequence is an ordered collection of objects, for example: $(1, 2, 3, 4, 5)$, or every positive integer $(2, 4, 6, ...)$

[2] Fibonacci number were first described by Indian mathematicians in 200 BCE, later made famous in Western Europe around 1200 CE by the Italian mathematician Leonardo of Pisa, also known as Fibonacci (from *filius Bonacci* ('son of Bonacci')). The sequence converges to the so called *Golden ratio*.

[3] If you have never heard of recurrence relations there is a more thorough explanation given in Appendix B of Levitin, see "Sequences and Recurrence Relations", up until (and excluding) "Linear Second-Order Recurrences with Constant Coefficients", and "Common Recurrence Types in Algorithms", up until (and excluding) the "Smoothness rule and the Master Theorem."

d. Using what we know about the Binary Trees, determine the height of the tree.

With this information, combined with the total time to merge (for each level), what is the time complexity of Mergesort?

**Question 4.** *Alternating Glasses* (M. Gardner, *Scientific American* 1978)

a. There are $2n$ glasses standing next to each other in a row, the first $n$ of them filled with a soda drink and the remaining $n$ glasses empty. Design a pseudocode algorithm which results in the glasses alternating in a filled-empty-filled-empty pattern in the minimum number of glass moves.

b. Same problem as a. but now the glasses are randomly placed on the row. Reason about the worst-case time-complexity of your algorithm.