Fundamental Algorithms 3

Exercise 1

Consider a partitioning algorithm that, in the worst case, will partition an array of \( m \) elements into two partitions of size \( \lfloor \epsilon m \rfloor \) and \( \lceil (1 - \epsilon)m \rceil \), where \( \epsilon \) is fixed, and \( 0 < \epsilon < 1 \). Show that a quicksort algorithm based on this partitioning has a worst-case complexity of \( O(n \log n) \).

Hint or solution: solve the recurrence by guessing the solution and finding the involved constants.

K-Exercise 2 (An Iterative MergeSort)

The following iterative implementation of the MergeSort algorithm is proposed:

\[
\begin{align*}
\text{ItMergeSort} (A: \text{Array}[0..n-1]) \{} \\
\quad \text{// } n \text{ assumed to be a power of 2: } n=2^k \\
\quad k := \log_2(n) \\
\quad \text{// } m := 2 \\
\quad \text{for } L \text{ from 1 to } k \text{ do } \{ \\
\quad \quad \text{for } i \text{ from 0 to } \lfloor n/m \rfloor - 1 \text{ do } \{ \\
\quad \quad \quad \text{MergeIP}(A[ i \times m \ldots i \times m+(m/2-1)], \\
\quad \quad \quad \quad A[ i \times m+(m/2) \ldots i \times m+(m-1)], \\
\quad \quad \quad \quad \quad \quad \quad A[ i \times m \ldots i \times m+(m-1)]); \\
\quad \quad \} \\
\quad \quad m := 2 \times m; \\
\quad \} \\
\}
\end{align*}
\]

The procedure MergeIP is equivalent to the procedure Merge discussed in the lecture, but can work directly on the array \( A \) (i.e., merges two adjacent subarrays of \( A \)).

a) Describe shortly and in plain words, how ItMergeSort compares to the recursive MergeSort implementation discussed in the lecture. For that purpose, draw a diagram that illustrates the sorting of an array \( A[0..7] \) for ItMergeSort.

b) Formulate a loop invariant for the \( L \)-loop of the algorithm, and prove its correctness.