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:

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

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.