Fundamental Algorithms 3

Exercise 1 (Worst-Case)

Consider a partitioning algorithm that, in the worst case, will partition an array of \( m \) elements into two partitions of size \( \lfloor \varepsilon m \rfloor \) and \( \lceil (1 - \varepsilon) m \rceil \), where \( \varepsilon \) is fixed and \( 0 < \varepsilon < 1 \). Show that a QUICKSORT algorithm based on this partitioning has a worst-case complexity of \( O(n \log n) \) (in terms of comparisons between array elements). \textit{Hint:} Solve the recurrence by guessing the solution and finding the involved constants.

Exercise 2 (Iterative MergeSort)

The following iterative implementation of the MERGESORT algorithm is proposed. The procedure \textit{MergeIP} is equivalent to the procedure \textit{Merge} discussed in the lecture, but can work directly on the array \( A \) (i.e., merges two adjacent sub-arrays of \( A \)).

\begin{algorithm}
\caption{MERGE\textsc{SortIt}}
\label{alg:merge-sort-it}
\begin{algorithmic}
\STATE \textbf{Input:} \( A \): Array of size \( n = 2^k \)
\STATE \textbf{Result:} Array \( A \) sorted
\STATE \( k \leftarrow \log_2(n) \);
\STATE \( m \leftarrow 2 \);
\FOR {\( L = 1 \) to \( k \)}
\FOR {\( i = 0 \) to \( (n/m) - 1 \)}
\STATE \textit{MergeIP}(\( A[i \cdot m .. i \cdot m + (m/2) - 1] \), \( A[i \cdot m + (m/2) .. i \cdot m + (m - 1)] \), \( A[i \cdot m .. i \cdot m + (m - 1)] \));
\ENDFOR
\STATE \( m \leftarrow 2 \cdot m \);
\ENDFOR
\end{algorithmic}
\end{algorithm}

1. Describe shortly and in plain words, how \textsc{MergeSortIt} compares to the recursive \textsc{MergeSort} implementation discussed in the lecture. For that purpose, draw a diagram that illustrates the sorting of some array with length 8 for \textsc{MergeSortIt}.

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