Technische Universität München
Institut für Informatik
Theoretical Computer Science

WS 2017/18

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 ε 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). *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 MergeIP is equivalent to the procedure Merge discussed in the lecture, but can work directly on the array A (i.e., merges two adjacent sub-arrays of A).

Algorithm 1: MergeSortIt

Input: A: Array of size $n = 2^k$
Result: Array A sorted
$k \leftarrow \log_2(n)$;
$m \leftarrow 2$;
for $L = 1$ **to** k **do**
  for $i = 0$ **to** $(n/m) - 1$ **do**
    `MergeIP(`$A[i \cdot m \,..\, i \cdot m + (m/2) - 1]$,
      $A[i \cdot m + (m/2) \,..\, i \cdot m + (m-1)]$,
      $A[i \cdot m \,..\, i \cdot m + (m-1)]$`)`;
  end
  $m \leftarrow 2 \cdot m$;
end

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

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