Technische Universität München
Institut für Informatik
Scientific Computing in Computer Science

WS 2013/14
Worksheet
4.11.2013

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 ϵ 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:

```
ItMergeSort(A: Array[0..n-1]) {
   // n assumed to be a power of 2: n=2^k
   k := log2(n)
   //
   m := 2
   for L from 1 to k do {
      for i from 0 to (n/m)-1 do {
         MergeIP(A[i*m .. i*m+(m/2-1],
                 A[i*m+(m/2) .. i*m+(m-1],
                 A[i*m .. i*m+(m-1));
      };
      m := 2*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.