technische universität dortmund

Fakultät für Informatik
Lehrstuhl 11 / Algorithm Engineering

Petra Mutzel
Nicola Beume, Christian Bockermann, Christian Horoba,
Ingo Schulz, Dirk Sudholt, Christine Zarges

Sommersemester 2009

DAP2 Übung – Blatt 3

Ausgabe: 30. April, **Abgabe:** 7. Mai, 14:00 Uhr, **Block:** A

Aufgabe 3.1 (4 Punkte) Führe Quick-Sort am Beispiel der Eingabefolge

S, O, R, T, I, E, R, E, N

durch und stelle jeden Aufruf von `QuickSort` (analog zur Abbildung 3.4 auf Seite 49 im Skript) grafisch dar.

Aufgabe 3.2 (4 Punkte) Betrachte die Prozedur `Merge` des Merge-Sort-Algorithmus für zwei sortierte Teilfolgen der Länge h. Wie viele Schlüsselvergleiche werden im Best-Case und im Worst-Case benötigt? Gib die Anzahl exakt (also nicht in O-Notation) an. Gib für den Fall $h = 8$ jeweils ein Beispiel für einen Best-Case und einen Worst-Case an.

Hinweis: Beachte Anmerkung 3.3 auf Seite 42 im Skript.

Aufgabe 3.3 (4 Punkte) Berechne die exakte Anzahl (also nicht in O-Notation) der Schlüsselvergleiche, die Quick-Sort im Fall eines Eingabefeldes aus n *gleichen* Elementen benötigt.

Präsenzaufgabe 3.4 Wenn bekannt ist, dass die Eingabe viele gleiche Elemente enthält, kann man Quicksort verbessern. Gib eine Variante von Quick-Sort an, bei der sichergestellt ist, dass nicht mehrfach der gleiche Schlüssel als Pivotelement verwendet wird. Gib hierzu den Pseudocode des modifizierten Programms an. Erkläre dein Vorgehen und insbesondere deine Veränderungen im Vergleich zur `Partition`-Version aus dem Skript (S. 48, Listing 3.5).

Wie viele Schlüsselvergleiche benötigt der veränderte Algorithmus im Fall eines Feldes aus n gleichen Elementen? Gib die Anzahl in O-Notation an.

Hinweis: Passe die Funktion `Partition` so an, dass beim Aufruf von `Partition(ref A, l, r)` alle Vorkommen des Pivotelements im Bereich von `A[l]` bis `A[r]` richtig einsortiert werden. Du darfst `Partition` so verändern, dass mehr als ein Wert zurückgegeben wird.