Institut für Telematik
Karlsruher Institut für Technologie (KIT)
Prof. Dr. M. Zitterbart
Dr. I. Baumgart
Dipl.-Inform. C. Haas

Algorithmen I
Sommersemester 2011

2. Übungsblatt
Sortieren und Folgen

Aufgabe 1

Suchen

(a) In Abbildung 1 ist der Pseudocode für das Sortierverfahren Counting-Sort abgebildet. Eine Eingabe für Counting-Sort besteht aus einem Feld $A[1..n]$, einem Hilfsfeld $B[1..n]$ und einem Wert k. Counting-Sort setzt voraus, dass für eine ganze Zahl k jedes der n Eingabeelemente eine ganze Zahl zwischen 0 und k ist. Erläutern Sie, wie ein Aufruf von CountingSort(A,B,6) auf dem Feld $A = [6,0,2,0,1,3,4,6,1,3,2]$ arbeitet. Geben Sie hierzu die Inhalte des Feldes A und der Hilfsfelder B und C nach jedem Durchlauf einer der **for** Schleifen an. Geben sie zusätzlich die erwartete Laufzeit von Counting-Sort an. (6P.)

(b) In der Vorlesung wurden verschiedene Sortierverfahren behandelt. Welche der behandelten vergleichsbasierten Sortierverfahren Bubblesort, Insertionsort, Selectionsort, Mergesort und Quicksort sind stabil ? Begründen Sie jeweils Ihre Antwort. (6P.)

(c) Geben Sie eine möglichst einfache Methode an, mit der jeder der nicht stabilen Algorithmen in einen stabilen Sortieralgorithmus transformiert werden kann. Geben Sie zusätzlich an, wieviel zusätzlichen Speicher und welche Laufzeit ihre Methode benötigt. (6P.)

Aufgabe 2

Stapel und Warteschlangen

(a) Nehmen Sie einen anfangs leeren Stapel S an, der durch ein Feld der Länge 3 realisiert wird. Zeichnen Sie den Zustand des Felds nach jeder der folgenden Stapeloperationen: *PUSH(S,4), PUSH(S,3), POP(S), POP(S), POP(S), PUSH(S,7), PUSH(S,9), POP(S), PUSH(S,12), PUSH(S,1), PUSH(S,5)*. Betrachten Sie auch auftretende Fehlersituationen. (Beim Auftreten eines Fehlers wird das Feld durch die Operation nicht verändert.) (4P.)

$Counting - Sort(A, B, k)$

```
1  // Sei C[0..k] ein neues Feld
2  for i = 0 to k
3        C[i] = 0
4  for j = 1 to A.länge
5        C[A[j]] = C[A[j]] + 1
6  // C[i] enthält nun die Anzahl der Elemente, die gleich i sind
7  for i = 1 to k
8        C[i] = C[i] + C[i − 1]
9  // C[i] enthält nun die Anzahl der Elemente, die kleiner oder gleich i sind
10 for j = A.länge downto 1
11       B[C[A[j]]] = A[j]
12       C[A[j]] = C[A[j]] − 1
```

Abbildung 1: Pseudocode für Counting-Sort

(b) Nehmen Sie eine anfangs leere Warteschlange Q an, die durch ein Feld der Länge 3 realisiert wird. Zeichnen Sie den Zustand des Felds nach jeder der folgenden Warteschlangenoperationen: *ENQUEUE(Q,8), ENQUEUE(Q,3), ENQUEUE(Q,5), DEQUEUE(Q), ENQUEUE(Q,2), ENQUEUE(Q,6), DEQUEUE(Q), ENQUEUE(Q,1), DEQUEUE(Q)*. Betrachten Sie auch auftretende Fehlersituationen. (Beim Auftreten eines Fehlers wird das Feld durch die Operation nicht verändert.) (4P.)

(c) Zeigen Sie, wie eine Warteschlange unter Verwendung von zwei Stapeln implementiert werden kann. Geben Sie die Laufzeit der Warteschlangen-Operationen im Worst-Case an. (6P.)

Aufgabe 3

Verkettete Listen

(a) Entwerfen Sie den Pseudocode für eine Methode *LIST-INSERT-SORTED(L,x)* die ein Element x in eine doppelt verkettete, zyklische Liste L einfügt, so dass die Elemente in

der Liste aufsteigend sortiert sind. **Annahme**: Die Liste ist vor dem Aufruf von LIST-INSERT-SORTED bereits sortiert. (6P.)

(b) Die Operation *UNION* auf dynamische Mengen erhält zwei nichtleere disjunkte Mengen S_1 und S_2 und gibt eine Menge $S = S_1 \cup S_2$ zurück, die aus allen Elementen von S_1 und S_2 besteht. Die Mengen S_1 und S_2 können durch diese Operation gelöscht werden. Zeigen Sie, wie UNION durch Verwendung einer geeigneten Listen-Datenstruktur realisiert werden kann, sodass die Laufzeit der Operation in $\mathcal{O}(1)$ liegt. (6P.)

(c) Eine *Doppelschlange (engl. Double-Ended Queue, kurz Deque)* hat die Operationen *PUSH-FRONT*, *PUSH-BACK*, *POP-FRONT* und *POP-BACK*. Lässt sich eine solche Deque mit einer einfach verketteten Liste konstruieren, so dass alle genannten Operationen nur konstante Zeit benötigen oder braucht man eine doppelt verkettete Liste? Geben Sie den Pseudecode für die genannten Operationen an. (10P.)

Aufgabe 4

Unbeschränkte Felder

Im Unterschied zur Vorlesung wird hier ein unbeschränktes Feld betrachtet, das sofort halbiert wird, wenn es nur noch zur Hälfte gefüllt ist.

Geben Sie eine Folge von n Operationen an, so dass ein derart schlecht implementiertes unbeschränktes Feld mit diesen Operationen einen Aufwand von $\Theta(n^2)$ verursacht. Zur Vereinfachung sei n eine Zweierpotenz. (6P.)

Abgabe des Übungsblattes bis Mittwoch, 11.5.2011 – 12 Uhr durch Einwurf in den entsprechenden Briefkasten im Gebäude 50.34, 1. UG.