Entwurf und Analyse von Algorithmen

WS 1999/2000
Prof. Dr. K. Madlener
Übungsblatt 4

Aufgabe 4.1. Sei $V^*(n)$ die mittlere Anzahl von Vergleichen (unter der Annahme, dass jede Permutation der Elemente des Feldes gleichwahrscheinlich sei), die Algorithmus RANDOMSELECT aus der Vorlesung für Feldgröße n durchführt. Zeigen Sie, dass $V^*(n) \leq c \cdot n$ für eine (von n unabhängige) Konstante c ist.

Aufgabe 4.2. Betrachten Sie den folgenden Algorithmus STOOGESORT:

```
procedure StoogeSort(A, i, j)
begin
    if A[i] > A[j] then
        exchange A[i] ↔ A[j]
    if i + 1 ≥ j then
        return
    k := ⌊(j − i + 1)/3⌋
    StoogeSort(A, i, j − k)      # Erste zwei Drittel
    StoogeSort(A, i + k, j)      # Letzte zwei Drittel
    StoogeSort(A, i, j − k)      # Nochmals erste zwei Drittel
end
```

Zeigen Sie, dass der Aufruf STOOGESORT$(A, 1, n)$ das Eingabefeld $A[1..n]$ korrekt sortiert. Geben Sie eine Rekursionsgleichung für die Anzahl der Vergleiche an, die STOOGESORT im schlechtesten Fall braucht und geben Sie eine Lösung dieser Gleichung in der Θ-Notation an.
Ist dieser Sortierargorithmus eine Konkurrenz für die Ihnen bekannten Sortieralgorithmen?

Aufgabe 4.3. Betrachten Sie den folgenden Algorithmus COUNTINGSORT:

```
procedure CountingSort(A, B, k)
begin
    for i := 1 to k do
        C[i] := 0
    for j := 1 to length(A) do
        C[A[j]] := C[A[j]] + 1
    for i := 2 to k do
        C[i] := C[i] + C[i − 1]
    for j := length(A) downto 1 do
```

$B[C[A[j]]] := A[j]$
$C[A[j]] := C[A[j]] - 1$

Zeigen Sie, dass unter der Voraussetzung, dass Feld A n ganze Zahlen zwischen einschließlich 1 und einschließlich k enthält, nach dem Aufruf von CountingSort(A, B, k) das Feld B die Elemente von A nichtfallend sortiert enthält. Machen Sie sich die Arbeitsweise des Algorithmus an Hand des Feldes $A = \langle 7, 1, 3, 1, 2, 4, 5, 7, 2, 4, 3 \rangle$ klar.

Was ist die Zeitkomplexität dieses Algorithmus, wenn man Zuweisungen zählt? Wie sieht es im Falle $k \in O(n)$ aus? Machen Sie auch eine möglichst genaue Aussage über die Platzkomplexität.

Zeigen Sie, dass CountingSort ein stabiler Sortieralgorithmus ist. Einen Sortieralgorithmus nennt man stabil, wenn er die relative Reihenfolge gleicher Elemente unverändert lässt. Beispiel: Sortiert man obiges Feld, so erhält man $\langle 1, 1, 2, 2, 3, 3, 4, 4, 5, 7, 7 \rangle$; ein stabiler Sortieralgorithmus enthält beide 1en, 2en, 3en, 4en und 7en in der selben Reihenfolge, wie sie im Ausgangsfeld enthalten waren.

Wenn man die einleitende Zeile der letzten **for**-Schleife in

for $j := 1$ **to** length(A) **do**

ändert, ist der Algorithmus dann immer noch korrekt? Stabil?

Aufgabe 4.4. a) Zeichnen Sie die Entscheidungsbäume für das Sortieren eines Feldes mit vier unterschiedlichen ganzen Zahlen für Bubblesort (siehe Schöning, Seite 50), Heapsort (siehe Schöning, Seite 61 unten) und (nicht randomisierten) Quicksort (siehe Schöning, Seite 53, mit $x = a[\text{links}]$).

b) Geben Sie einen Algorithmus an, der fünf beliebige unterschiedliche ganze Zahlen mit höchstens sieben Vergleichen sortieren kann. Finden Sie auch einen Algorithmus, der zehn beliebige unterschiedliche ganze Zahlen mit höchstens 21 Vergleichen sortieren kann? Mit höchstens 22?

Informationen zur Vorlesung:
http://www-madlener.informatik.uni-kl.de/ag-madlener/lectures/eaa/ws1999-2000/eaa.html