Johannes Blömer
u.v.a.m.

Paderborn, den 5.Juni 2004

Musterlösungen zur Vorlesung

Datenstrukturen und Algorithmen

SS 2004

Blatt 6

AUFGABE 1:

Der Entscheidungsbaum für Quicksort bei $n = 3$ ist in Abbildung 1 dargestellt.

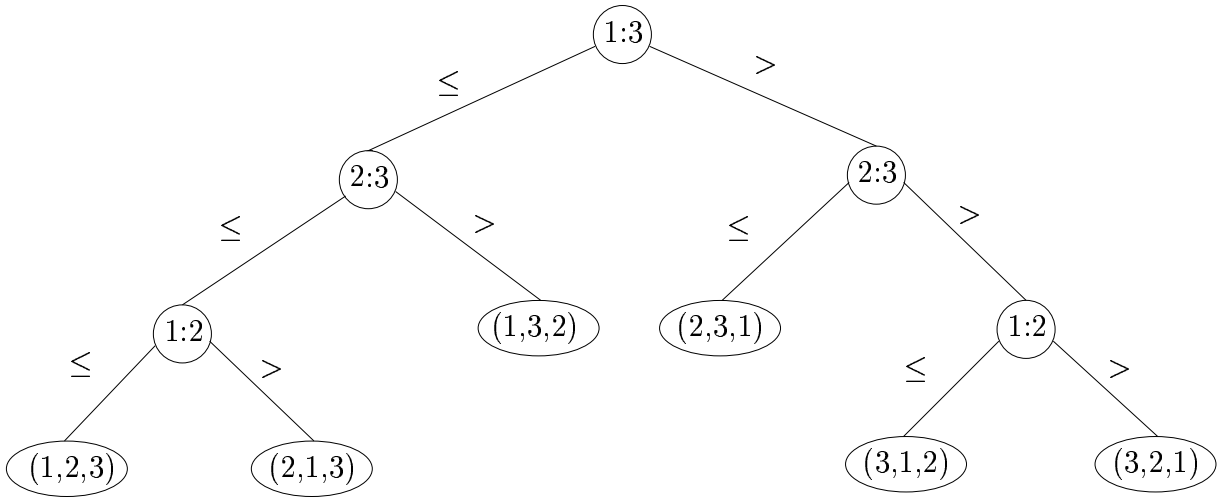

Abbildung 1: Entscheidungsbaum für Quicksort bei $n = 3$

AUFGABE 2:

Wir behaupten, dass jedes Blatt im Entscheidungsbaum eines Vergleichssortierers bei n Eingabezahlen Tiefe mindestens $n - 1$ besitzt. Äquivalent zu dieser Behauptung ist, dass unabhängig von der speziellen Eingabe, die korrekte Sortierung von n Zahlen $a_1, \ldots, a_n$ mindestens $n - 1$ Vergleiche benötigt.

Wir beweisen diese Behauptung durch vollständige Induktion über n.

Induktionsanfang: Für $n = 1$ besagt die Behauptung, dass mindestens 0 Vergleiche zur Sortierung einer Zahl benötigt werden. Dieses ist sicherlich richtig.

Induktionsschritt: Nehmen wir an, dass wir bereits wissen, dass zur Sortierung von $m, m < n$, Zahlen durch eine Vergleichssortierer mindestens $m - 1$ Vergleiche benötigt werden (*Induktionsvoraussetzung*). Wir müssen zeigen, dass zur Sortierung von n durch einen Vergleichssortierer dann mindestens $n - 1$ Vergleiche benötigt werden. Seien $a_1, \ldots, a_n$ die zu sortierenden Zahlen. Sei bei einem beliebigen Vergleichssortierer I die Menge der Indizes $i, 1 \leq i \leq n-1$, so dass der Vergleichssortierer a_n und a_i vergleicht. Sei $J := \{1, \ldots, n-1\} \setminus I$ die Menge der übrigen Indizes. Weiter setzen wir $k := |I|$. Dann ist $|J| = n - k - 1$.

Nun muss der Vergleichssortierer insbesondere auch die Zahlen $a_j, j \in J$, korrekt sortieren. Da $|J| \leq n-1$ wird er hierzu nach Induktionsvoraussetzung mindestens $n-k-2$ Vergleiche benötigen, an denen nur Zahlen $a_j, j \in J$, beteiligt sind. Ausserdem wird jedes $a_i, i \in I$, mit a_n verglichen. Dieses liefert k zusätzliche Vergleiche. Insgesamt benutzt der Vergleichssortierer damit bereits mindestens $n - k - 2 + k = n - 2$ Vergleiche. Aber der Vergleichssortierer muss auch noch mindestens ein

$a_j, j \in J$, mit einem Element $a_i, i \in I \cup \{n\}$, vergleichen, denn sonst könnte der Vergleichssortierer noch nichts über das Verhältnis der Zahlen $a_j, j \in J$, zu den Zahlen $a_i, i \in I \cup \{n\}$, wissen. Damit erhalten wir insgesamt mindestens $n-2+1 = n-1$ Vergleiche und unsere Behauptung ist bewiesen. Im Vergleichsbaum für Insertion-Sort etwa gibt es auch Blätter mit Tiefe $n-1$. Damit ist die oben bewiesene Schranke von $n-1$ auch optimal.

AUFGABE 3:
Wir bezeichnen mit C_u die Anzahl der Eingabezahlen, die höchstens so groß wie u sind, $0 \leq u \leq k$. Mit dieser Notation ist dann die Anzahl der Eingabezahlen in einem Intervall $[a..b]$ gegeben durch $C_b - C_{a-1}$. Sind somit die Werte $C_u, u = -1, 0, \ldots, k$ bekannt, so kann die Anzahl der Eingabezahlen im Intervall $[a..b]$ in Zeit $\mathcal{O}(1)$ bestimmt werden.
Es gilt sicherlich $C_{-1} = 0$. Die weiteren Werte C_u können wir nun mit Hilfe des folgenden Algorithmus SEQUENCE-COUNT bestimmen, der aus der ersten 8 Zeilen des Algorithmus COUNTING-SORT besteht. Wir nehmen an, dass die Eingabezahlen in einem Array A gespeichert sind. Weiter sollen die Werte $C_u, u = 0, \ldots, k$ in einem Array C gespeichert werden.

SEQUENCE-COUNT(A)

```
1 for i ← 0 to k
2     do C[i] ← 0
3 for j ← 1 to length[A]
4     do C[A[j]] ← C[A[j]] + 1
5 ▷ C[i] enthält nun die Anzahl der Elemente A[j] mit Wert i.
6 for i ← 1 to k
7     do C[i] ← C[i] + C[i − 1]
8 ▷ C[i] enthält nun die Anzahl der Elemente A[j] mit Wert höchstens i.
```

In der Vorlesung hatten wir bereits gesehen, dass COUNTING-SORT Zeit $\mathcal{O}(n+k)$ benötigt. Damit gilt dieses auch für SEQUENCE-COUNT, der ja nur einen Teil von COUNTING-SORT darstellt. Ausserdem hatten wir uns auch bereits überzeugt, dass in Counting-Sort zunächst die Anzahl der Eingabezahlen mit Wert höchstens i bestimmt werden. Diese Argumente zeigen natürlich auch die Korrektheit von SEQUENCE-COUNT.