Johannes Blömer
u.v.a.m.

Paderborn, den 29. April

Musterlösungen zur Vorlesung

Datenstrukturen und Algorithmen

SS 2004

Blatt 1

AUFGABE 1:
Die Anzahl der Sekunden, die man mit INSERTION-SORT auf Computer A zum Sortieren von 10^7 Elementen benötigt, ist gegeben durch

$$\frac{8 \cdot 10^{14}}{10^9} = 8 \cdot 10^5.$$

Die Anzahl der Sekunden, die man mit MERGE-SORT auf Computer B zum Sortieren von 10^7 Elementen benötigt, ist gegeben durch

$$\frac{64 \cdot 10^7 \cdot \log(10^7)}{5 \cdot 10^6} \approx 2977.$$

Mit MERGE-SORT benötigt man selbst auf dem langsameren Computer B deutlich weniger Zeit als mit INSERTION-SORT. Schnelle Algorithmen sind also sehr sinnvoll.

AUFGABE 2:
Durch Ausprobieren findet man

$$100 \cdot 14^2 = 19600 > 16384 = 2^{14}$$

und

$$100 \cdot 15^2 = 22500 < 32768 = 2^{15}.$$

Da sowohl $100n^2$ als 2^n streng mononton wachsen, ist $n = 15$ die gesuchte Zahl.

AUFGABE 3:
Hier ist zunächst der Algorithmus in Pseudocode.

SEARCH(A, v)

```
1 for i ← 1 to length[A]
2     do if A[i] = v
3           then return i
4 return NIL
```

Der Algorithmus durchläuft in der Schleife in den Zeile 1-4 das Array und überprüft dabei, ob eines der Arrayelement das gesuchte v ist. Hat der Algorithmus das Element v an Position i des Arrays gefunden, wird der Index i ausgegeben (Zeilen 3 und 4). Der **return**-Befehl terminiert den Algorithmus dann sofort. Ist keines der Arrayelemente das gesuchte v, so wird in der letzten Zeile des Algorithmus NIL ausgegeben, wie in der Problembeschreibung verlangt.

AUFGABE 4:
Wir beschreiben zunächst den Algorithmus wieden in Pseudocode.

SEARCH-MIN(A)

```
1 min ← 1
2 for i ← 2 to length[A]
3     do if A[i] < A[min]
4           then min ← i
5 return min
```

Um die Korrektheit des Algorithmus zu beweisen benutzen wir die folgende Invariante für die **for**-Schleife in den Zeilen 2-5.

Invariante: Vor dem Schleifendurchlauf mit Index i, ist $A[min]$ das kleinste Element im Array $A[1..i-1]$.

Nun zeigen wir Initialisierung, Erhaltung und Terminierung.

Initialisierung: Der kleinste Index für die Schleife ist $i = 2$. Unmittelbar davor ist $min = 1$ und $A[min] = A[1]$ ist sicherlich das kleinste Element in $A[1..i-1] = A[1]$, denn dieses Array besteht nur aus $A[1]$.

Erhaltung: Beim Schleifendurchlauf für Index i wird überprüft, ob $A[i]$ kleiner ist als das bisherige Minimum $A[min]$ (Zeile 4). Ist dieses der Fall, muss das Minimum durch $A[i]$ ersetzt werden. Dieses geschieht in Zeile 5. Im Fall $A[i] \geq A[min]$ bleibt das Minimum unverändert. Damit ist die Invariante auch nach Durchlauf der Schleife für Index i erfüllt.

Terminierung: Die Invariante sagt, dass nach Durchlauf der Schleife für Index $i = n$ (vor Durchlauf mit Index $i = n + 1$), $A[min]$ das Minimum im Array $A[1..n]$, also im gesamten Eingabearray ist. Damit ist der Algorithmus korrekt.