Prof. Andreas Goerdt
Frank Schädlich

TU Chemnitz
Sommersemester 2003

Theoretische Informatik III

4. Übung - Lösungsvorschläge

1. Aufgabe

$$i = 1 + \log^* n$$

2. Aufgabe

Es soll gezeigt werden, daß die amortisierte Laufzeit einer Operation $\sigma_i = \texttt{Delete}(x)$ unter der Heuristik *MF* (Move to Front) höchsten zweimal so groß ist wie bei der „Superheuristik“ A. Dazu wurde in der Vorlesung folgende Potentialfunktion Φ für die Datenstruktur der selbstorganisierenden Liste eingeführt.

$$\Phi(S, S') = \text{Anzahl der Inversionen zwischen } S \text{ und } S'$$

Dabei ist S die Liste, die bei der Ausführung der Operationsfolge σ mit *MF* entstand und S' die entsprechende Liste für Heuristik A. Eine Inversion tritt genau dann auf, wenn ein Element x in S vor (nach) y und in S' nach (vor) y steht.

Zu zeigen ist:

$$a_i \leq 2t'_i$$

Sei k die Position von x (zu löschendes Element) in S bzw. j in S':

$$\begin{aligned}
a_i &= k + \Phi_i(S, S') - \Phi_{i-1}(S, S') \\
&= k + r - p \\
t'_i &= j + z \quad .
\end{aligned}$$

$$\text{Dabei sind:} \quad \begin{aligned}
k &= \text{Anzahl Schritte bis } x \text{ in } S \text{ gefunden} \\
a_i &= \text{amortisierte Kosten der Operation } i \text{ der Heuristik } MF \\
t'_i &= \text{Kosten der Operation } i \text{ der Heuristik } A \\
r &= \text{Anzahl neuer Inversionen} \\
p &= \text{Anzahl Inversionen, die verschwinden} \\
j &= \text{Anzahl Schritte bis } x \text{ in } S' \text{ gefunden} \\
z &= \text{Anzahl von Transpositionen der Heuristik } A.
\end{aligned}$$

Offensichtlich kann allein durch das Entfernen von x aus den Listen keine neue Inversion entstehen. Es ist allerdings möglich, daß sich durch die Transpostionen der Heuristik A neue Inversionen bilden. Das können maximal z sein. Es gilt also $r \leq z$. Da für a_i eine obere Abschätzung gesucht wird, reicht es, für p eine untere anzugeben. Bei `Delete` können nur Inversionen mit x verschwinden. Steht in einer der beiden Listen S oder S' das Element x weiter hinten als in der anderen, so können nicht alle Elemente, die in dieser Liste vor x standen, auch in der anderen vor x stehen. Es gibt also mindestens $p \geq |k-j|$ Inversionen, die bei `Delete` verschwinden.

Für $k > j$ gilt dann:

$$\begin{aligned} p &\geq k-j \\ a_i &= k+r-p \\ &\leq k+z-k+j \\ &= j+z = t'_i \\ a_i &\leq t_i < 2t'_i \end{aligned}$$

q.e.d.

Und für $j \geq k$:

$$\begin{aligned} p &\geq j-k \\ a_i &= k+r-p \\ k-p &\leq j \text{ (wegen } k \leq j \text{ und } p \geq 0) \\ a_i &= k+z-p \leq j+z = t'_i \\ a_i &\leq t'_i < 2t'_i \end{aligned}$$

q.e.d.

3. Aufgabe

Operation	**Worst-Case-Laufzeit**
`Delete`(x)	maximal Länge der Liste
Find(x)	maximal Länge der Liste
Insert(x)	genau Länge der Liste (wenn x noch nicht in Liste)

`Insert`(x) verursacht also immer die größten Kosten von allen drei Operationen, wenn x sich nicht in der Liste befindet. Eine ungünstigste Folge von Operation wird also aus m `Inserts` von noch nicht vorhanden Elementen bestehen. Zum Beispiel `Insert`(1) bis `Insert`(m). Dabei wird beim i-ten `Insert` das Element gesucht wozu $i-1$ Schritte notwendig sind. Nach dem Einfügen am Ende der Liste (ein Schritt) wird das Element mit i Transpositionen an den Anfang der Liste bewegt (i Schritte). Für das Einfügen eines Elements beträgt der Aufwand also $2i$. Eine Folge m solcher `Inserts` auf anfangs leerer Liste hat dann die Kosten:

$$\begin{aligned} \text{Kosten} &= \sum_{i=1}^{m} 2m \\ &= 2\frac{m(m+1)}{2} \\ &= m^2 + m \end{aligned}$$

Die Worst-Case-Laufzeit ist also in $\mathrm{O}(m^2)$.

4. Aufgabe

Bei folgender Anweisungssequenz liegen die Kosten unter der Heuristik *TR* (nach `Insert` und `Find` je ein Transpose) nicht in O(Kosten unter *MF*).

```
1 FOR i = 1 to m DO
2     Insert(i)
3 Find(1)
4 FOR i = 1 to m*m DO
5     Find(m)
6     Find(1)
```

Entsprechend den Heuristiken enstehen dann folgende Listen:

Operation	Heuristik MF	Heuristik TR
m Inserts	m m-1 — 3 2 1	2 3 — m-1 m 1
Find(1)	1 m m-1 — 3 2	2 3 — m-1 1 m
Find(m)	m 1 m-1 — 3 2	2 3 — m-1 m 1
Find(1)	1 m m-1 — 3 2	2 3 — m-1 1 m
	gesuchte Elemente am Anfang, Find in O(1)	gesuchte Elemente am Ende, Find in O(m)

Die Kosten für Zeile 1 bis 3 sind bei beiden Heuristiken gleich in O(m^2), da für die `Inserts` $\frac{1}{2}m(m+1)$ Schritte und für `Find`(1) m Schritte benötigt werden. Offensichtlich arbeiten die `Finds` aus der letzten Schleife bei *MF* nur auf den ersten beiden und bei *TR* nur auf den letzten beiden Elementen. Bei *TR* muß jedesmal die gesamte Liste durchsucht werden, so daß pro `Find` also $\Theta(m)$ Kosten anfallen, bei MF aber nur O(1). Insgesamt ergibt sich für *MF* also ein Aufwand von O(m^2) und für *TR* von $\Theta(m^3)$, da die letzten beiden `Finds` m^2 mal ausgeführt werden. Die Kosten unter *TR* liegen also nicht in O(Kosten unter *MF*).