Johannes Blömer
u.v.a.m.

Paderborn, den 21. Juni 2004

Musterlösungen zur Vorlesung

Datenstrukturen und Algorithmen

SS 2004

Blatt 8

AUFGABE 1:
In einer einfach verketteten Liste L besitzt jedes Objekt x neben seinen Satellitendaten noch einen Verweis $next[x]$ auf seinen Nachfolger in der Liste L. Ist x das letzte Objekt, so ist $next[x] = \text{NIL}$. Der Zugriff auf eine einfach verkettete Liste L erfolgt über einen Verweis $head[L]$ auf das erste Objekt der Liste. Ist die Liste leer, so gilt $head[L] = \text{NIL}$.
Einen Stack realisieren wir nun mithilfe einer einfach verketteten Liste, indem wir bei der Operation Push das neue Element am Anfang der Liste einfügen. Die Pop-Operation können wir dann realisieren, indem wir das erste Objekt der Liste aus der Liste entfernen. Hier sind die beiden Operationen in Pseudocode. Dabei nennen wir die Push-Operation List-Push und die Pop-Opertion List-Pop.

List-Push(L, x)

```
1 next[x] ← head[L]
2 head[L] ← x
```

List-Pop(L)

```
1 if head[L] = NIL
2    then error "Stack Unterfluss"
3    else  x ← head[L]
4          head[L] ← next[x]
```

AUFGABE 2:
Um das maximale Element zu finden, durchsuchen wir das Array, solange bis wir ein Feld gefunden haben, das nicht den NIL-Verweis speichert. Wir beginnen dabei mit dem letzten Feld des Arrays. Das maximale Element ist dann die Position des zuletzt besuchten Feldes. Wir nehmen an, dass die Tabelle beim direkten Adressieren durch ein Array T gegeben ist, dessen Länge mit der Anzahl der möglichen Schlüsseln übereinstimmt. Wird kein Feld gefunden, dass einen Verweis ungleich dem NIL-Verweis enthält, so ist die Tabelle leer und es existiert auch kein Maximunm. Hier ist zunächst der Algorithmus.

Maximum-Direct-Adress

```
1 k ← length[T]
2 while T[k] = NIL und k ≠ −1
3         do  k ← k − 1
4 if k ≠ −1
5    then return k
6    else  error "Tabelle leer"
```

Die Korrektheit ergibt sich aus den Bemerkungen oben. Um die Laufzeit zu analysieren, bemerken wir, dass der Algorithmus jedes Feld des Arrays T höchstens einmal anschaut. Pro Feld wird dabei konstante Zeit benötigt. Damit ist die Laufzeit $\mathcal{O}(n)$, wobei $n = length[T]$. Bei einem leeren Array T wird aber auch das vollständige Array durchsucht. Damit ist in diesem Fall die Laufzeit $\Omega(n)$. Insgesamt ist die Laufzeit damit $\Theta(n)$.

AUFGABE 3:
Es gilt

$$5 \mod 9 = 5, \quad 28 \mod 9 = 1, \quad 19 \mod 9 = 1, \quad 15 \mod 9 = 6$$

$$20 \mod 9 = 2, \quad 33 \mod 9 = 6, \quad 12 \mod 9 = 3$$

$$17 \mod 9 = 8, \quad 10 \mod 9 = 1$$

Damit ergibt sich die Hashtabelle in Abbildung 1

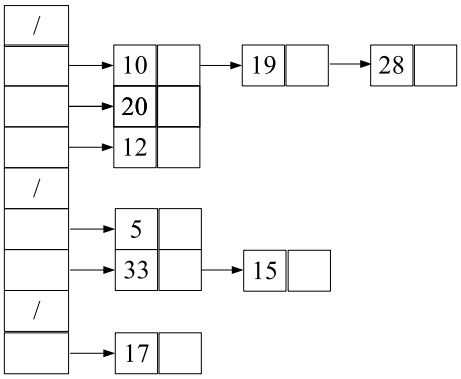

Abbildung 1: Hashtabelle zu Aufgabe 3

AUFGABE 4:
Sei X eine Zufallsvariable, die die Anzahl der Kollisionen zählt. Gesucht ist also der Erwartungswert $\mathrm{E}[X]$. Weiter definieren wir für jedes geordnete Paar (k, l) von Schlüsseln eine Zufallsvariable X_{kl}. Dabei gelte

$$X_{kl} = \begin{cases} 1, & \text{falls } h_2(k) = h_2(l) \\ 0, & \text{sonst} \end{cases}$$

Damit gilt

$$X = \sum_{\{(k,l)|k \neq l\}} X_{kl}$$

und, da der Erwartungswert linear ist,

$$\mathrm{E}[X] = \sum_{\{(k,l)|k \neq l\}} \mathrm{E}[X_{kl}]$$

Für je zwei Schlüssel $k, l \in \{0, 1, \ldots, m-1\}$ gilt nun unter der Annahme des einfachen uniformen Hashings

$$\Pr(h_2(k) = h_2(l)) = \frac{1}{m},$$

denn $h_2(k)$ kann jeden beliebigen Wert u annehmen und die Wahrscheinlichkeit, dass dann auch $h_2(l)$ den Wert u annimmt, ist $1/m$. Damit gilt für alle Schlüsselpaare $(k, l), k \neq l$,

$$\Pr[X_{kl} = 1] = \frac{1}{m}$$

und somit

$$\mathrm{E}[X_{kl}] = \frac{1}{m}.$$

Nun gibt es genau $n(n-1)$ viele geordnete Paare (k, l) von unterschiedlichen Schlüsseln. Damit erhalten wir schließlich

$$\mathrm{E}[X] = \sum_{\{(k,l)|k \neq l\}} \mathrm{E}[X_{kl}] = \sum_{\{(k,l)|k \neq l\}} \frac{1}{m} = \frac{n(n-1)}{m}.$$