Lösungen zu „Fundamente der Informatik – Ablaufmodellierung, Algorithmen und Datenstrukturen“

Im Lehrbuch „Fundamente der Informatik – Ablaufmodellierung, Algorithmen und Datenstrukturen“ von Peter Hubwieser und Gerd Aiglstorfer (erschienen im Oldenbourg Verlag München, 2004, ISBN 3-486-27572-0) schließt jedes Kapitel mit einigen Aufgaben.

Die folgenden Lösungsvorschläge wurden von Gerd Aiglstorfer erstellt. Weitere Informationen und Hilfestellungen zum Lehrbuch finden Sie unter http://www.aigl.de.

Inhalt

1 Lösungsvorschläge zu „Hashing“

1.1 Lösungsvorschlag Aufgabe 12.1

Wir berechnen $h(k) = k^2 \; mod \; 7$ für die Werte $1, \ldots, 6$:

$1^2 \bmod 7 = 1$
$2^2 \bmod 7 = 4$
$3^2 \bmod 7 = 2$
$4^2 \bmod 7 = 2$
$5^2 \bmod 7 = 4$
$6^2 \bmod 7 = 1.$

Die Hashfunktion hat für diese Eingabewerte nur den Wertebereich 1, 2 und 4. Eine gute Hahsfunktion verteilt jedoch alle Werte gleichmäßig über den Wertebereich, um möglichst wenig Adresskollisionen zu verursachen.

Eingabewert $k' \geq 7$ stellen wir mit $k' = 7 \cdot x + k$ ($x \geq 0$ und $0 < k \leq 6$) dar. Durch Einsetzen erhalten wir:

$$(7x + k)^2 \bmod 7 = (7^2x^2 + 7xk + k^2) \bmod 7 = k^2 \bmod 7.$$

Die Summanden 7^2x^2 und $7xk$ sind gerade Vielfache von 7. Somit bildet die Hashfunktion alle Eingabewerte auf 1, 2 und 4 ab. Die Hashadressen 0, 3, 5 und 6 sind von dieser Funktion nicht erreichbar. $h(k)$ ist damit keine gut gewählte Hashfunktion.

1.2 Lösungsvorschlag Aufgabe 12.2

Die ASCII-Darstellung ist eine eindeutige Darstellung. Für jeden Buchstaben existiert eine entsprechende Zahldarstellung. Dadurch lassen sich die Zeichenfolgen der Schlüsselwörter eindeutig in nicht identische Zahlen umwandeln. Nun wird nur noch eine mathematische Funktion benötigt, die diese Zahlen ohne Kollision auf die Indexwerte der Hashtabelle abbildet. Perfektes Hashing über die ASCII-Darstellung von Schlüsselwörtern ist also möglich.

1.3 Lösungsvorschlag Aufgabe 12.3

Nach der Definition von universellem Hashing wird die Größe der Hashtabelle benötigt, welche im Rahmen der Aufgabe nicht angegeben ist. Wir kennen jedoch den Betrag der Klasse $\boldsymbol{H}$ sowie die Anzahl der Funktionen, die für zwei verschiedene Schlüssel Adresskollisionen verursachen. Damit berechnen wir:

$$\frac{\left|\{h \in \mathbf{H} : h(k_1) = h(k_2)\}\right|}{\left|\mathbf{H}\right|} = \frac{4}{8} = \frac{1}{2} \leq \frac{1}{t}.$$

Das zugrunde liegende Verfahren wäre also für $t \leq 2$ universell. Die Verwendung einer Hashtabelle der Größe 2 erscheint jedoch wenig sinnvoll, wodurch die beschriebene Klasse ***H*** nicht brauchbar für universelles Hashing ist.

1.4 Lösungsvorschlag Aufgabe 12.4

Teilaufgabe a)
Die Hashadressen für *h(k)* ergeben sich wie folgt:

Tab. 1.1 *Separate Verkettung*

k	135	102	28	14	61	6	94
h(k)	3	3	6	3	6	6	6

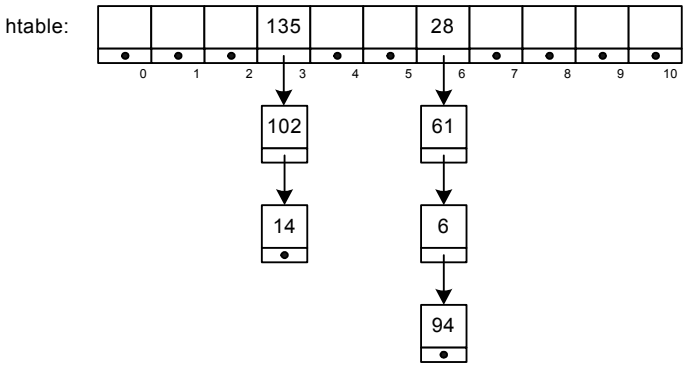

Abb. 1.1 *Separate Verkettung*

Teilaufgabe b)
Wir berechnen zuerst die Hashadressen:

Tab. 1.2 *Lineares Sondieren (1. Teil)*

k	135	102	28	14
h(k)	3	3	6	3
(h(k) – s(j, k)) mod 11	–	(3 – 1) mod 11 = 2	–	(3 – 1) mod 11 = 2 (3 – 2) mod 11 = 1
Speicheradresse	3	2	6	1

Tab. 1.3 *Lineares Sondieren (2. Teil)*

k	61	6	94
h(k)	6	6	6
(h(k) – s(j, k)) mod 11	(6 – 1) mod 11 = 5	(6 – 1) mod 11 = 5	(6 – 1) mod 11 = 5
		(6 – 2) mod 11 = 4	(6 – 2) mod 11 = 4
			(6 – 3) mod 11 = 3
			(6 – 4) mod 11 = 2
			(6 – 5) mod 11 = 1
			(6 – 6) mod 11 = 0
Speicheradresse	5	4	0

In der dritten Zeile wird jeweils die Sondierungsfolge bis zum ersten freien Platz in der Hashtabelle berechnet, und zwar genau dann, wenn die Hash- bzw. Speicheradresse (4. Zeile) bereits durch einen vorigen Schlüssel *k* belegt ist.

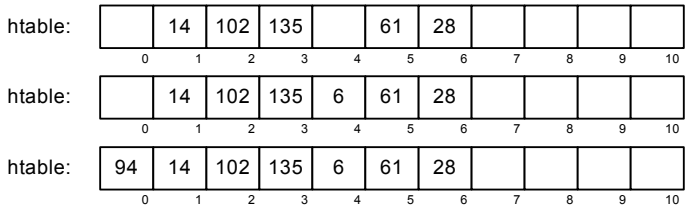

Abb. 1.2 *Lineares Sondieren*

Beim Einfügen der Schlüssel 6 und 94 kann der Effekt der primären Häufung beobachtet werden: bereits mit Schlüsseln belegte Bereiche wachsen schneller, wobei dies umso stärker auftritt, wenn sich größere Bereiche zusammenschließen (wie beim Einfügen des Schlüssels 6 der Fall).

Teilaufgabe c)
Wiederum geben wir die Hashadressen an:

Tab. 1.4 *Quadratisches Sondieren (1. Teil)*

k	135	102	28	14
h(k)	3	3	6	3
(h(k) – s(j, k)) mod 11	–	(3 + 1) mod 11 = 4	–	(3 + 1) mod 11 = 4
				(3 – 1) mod 11 = 2
Speicheradresse	3	4	6	2

Tab. 1.5 *Quadratisches Sondieren (2. Teil)*

k	61	6	94
h(k)	6	6	6
(h(k) – s(j, k)) mod 11	(6 + 1) mod 11 = 7	(6 + 1) mod 11 = 7 (6 – 1) mod 11 = 5	(6 + 1) mod 11 = 7 (6 – 1) mod 11 = 5 (6 + 4) mod 11 = 10
Speicheradresse	7	5	10

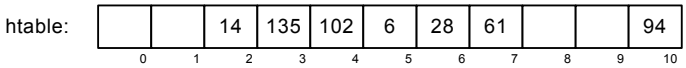

Abb. 1.3 *Quadratisches Sondieren*

Bei der sekundären Häufung – wie in der Abbildung zu erkennen ist – treten ebenfalls Zusammenschlüsse von bereits belegten Bereichen auf. Dies geschieht jedoch nicht so stark wie im Falle der primären Häufung. Vergleichen Sie dazu auch die Anzahl der benötigten Sondierungsschritte bei linearem und quadratischem Sondieren (beachten Sie dabei aber auch, dass die Anzahl der Sondierungsschritte umso größer ist, je höher der Belegungsfaktor ist).