Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Institut für Theoretische Informatik
Peter Widmayer
Thomas Tschager

9. März 2016

Datenstrukturen & Algorithmen Lösungen zu Blatt 2 FS 16

Lösung 2.1 *Asymptotische Laufzeit einschätzen.*

a) Die äussere Schleife wird $\Theta(\log n)$ Mal ausgeführt. Die innere Schleife (Zeilen 2 und 3) wird $\Theta(n)$ Mal durchlaufen. Insgesamt ist die Laufzeit daher in $\Theta(n \log n)$.

b) Die äussere Schleife wird $\Theta(n)$ Mal ausgeführt. Die innere Schleife (Zeilen 2 bis 9) in jedem Durchlauf $i \leq n$ Mal durchlaufen, also insgesamt $\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \in \Theta(n^2)$ mal. Die while-Schleife (Zeile 4 und 5) wird jeweils $\Theta(\sqrt{n})$ Mal durchlaufen. Insgesamt ist die Laufzeit daher in $\Theta(n^2 \cdot \sqrt{n})$.

c) Der Aufwand für die Zeilen 1 bis 8 ohne den rekursiven Aufruf in Schritt 6 ist linear in n, d.h. er beträgt maximal cn für eine geeignet gewählte Konstante $c > 0$. Die Gesamtlaufzeit (mit Berücksichtigung der rekursiven Aufrufe) ist dann durch

$$cn + c \cdot \frac{n}{2} + c \cdot \frac{n}{2^2} + ... = cn \left(1 + \frac{1}{2} + \frac{1}{2^2} + ...\right) \leq 2cn \tag{1}$$

nach oben beschränkt und beträgt damit ebenfalls $\Theta(n)$.

Lösung 2.2 *Rekursionsgleichungen.*

Mittels Teleskopieren kommt man auf folgende Vermutung:

$$\begin{aligned}
T(n) &= a \cdot T\left(\frac{n}{b}\right) + c \cdot n + d && (2)\\
&= a \cdot \left(a \cdot T\left(\frac{n}{b^2}\right) + c \cdot \frac{n}{b} + d\right) + c \cdot n + d && (3)\\
&= a \cdot \left(a \cdot \left(a \cdot T\left(\frac{n}{b^3}\right) + c \cdot \frac{n}{b^2} + d\right) + c \cdot \frac{n}{b} + d\right) + c \cdot n + d && (4)\\
&= a^3 \cdot T\left(\frac{n}{b^3}\right) + cn \cdot \frac{a^2}{b^2} + cn \cdot \frac{a}{b} + cn + da^2 + da + d && (5)\\
&= a^{\log_b(n)} e + cn \cdot \sum_{i=0}^{\log_b(n)-1} \left(\frac{a}{b}\right)^i + d \cdot \sum_{i=0}^{\log_b(n)-1} a^i. && (6)
\end{aligned}$$

Durch Anwendung der geometrischen Summenformel erhalten wir für $a \neq b, a \neq 1$ die geschlossene Form

$$T(n) = a^{\log_b(n)} e + cn \cdot \frac{\left(\frac{a}{b}\right)^{\log_b(n)} - 1}{\frac{a}{b} - 1} + d \cdot \frac{a^{\log_b(n)} - 1}{a - 1}. \tag{7}$$

Wir beweisen mittels vollständiger Induktion:

Induktionsverankerung: Die Vermutung ist korrekt für $n = 1$, denn $T(1) = a^0 e + c \cdot 0 + d \cdot 0 = e$.

Induktionsannahme: Wir nehmen an, dass unsere Vermutung für $T(n/b)$ korrekt ist, also (es gilt $\log_b(n/b) = \log_b(n) - 1$)

$$T(n/b) = a^{\log_b(n)-1}e + \frac{cn}{b} \cdot \frac{\left(\frac{a}{b}\right)^{\log_b(n)-1} - 1}{\frac{a}{b} - 1} + d \cdot \frac{a^{\log_b(n)-1} - 1}{a - 1}. \tag{8}$$

Induktionsschritt: Wir zeigen unter Verwendung der Induktionsannahme, dass unsere Vermutung auch für $T(n)$ stimmt:

$$\begin{aligned}
T(n) &= aT(n/b) + cn + d \\
&\overset{\text{Ind.-Ann.}}{=} a\left(a^{\log_b(n)-1}e + \frac{cn}{b} \cdot \frac{\left(\frac{a}{b}\right)^{\log_b(n)-1} - 1}{\frac{a}{b} - 1} + d \cdot \frac{a^{\log_b(n)-1} - 1}{a - 1}\right) + cn + d \\
&= a^{\log_b(n)}e + cn \cdot \frac{\left(\frac{a}{b}\right)^{\log_b(n)} - \frac{a}{b}}{\frac{a}{b} - 1} + cn \cdot \frac{\frac{a}{b} - 1}{\frac{a}{b} - 1} + d \cdot \frac{a^{\log_b(n)} - a}{a - 1} + d \cdot \frac{a - 1}{a - 1} \\
&= a^{\log_b(n)}e + cn \cdot \frac{\left(\frac{a}{b}\right)^{\log_b(n)} - 1}{\frac{a}{b} - 1} + d \cdot \frac{a^{\log_b(n)} - 1}{a - 1}.
\end{aligned}$$

Hinweis: Für die Annahme $a \neq b, a = 1$ ändert sich unsere Vermutung zu

$$T(n) = e + cn \cdot \frac{\left(\frac{1}{b}\right)^{\log_b(n)} - 1}{\frac{1}{b} - 1} + d\log_b(n) = d\log_b(n) + cb \cdot \frac{1 - n}{1 - b} + e.$$

Falls $a = b$ und $a \neq 1$ gilt, gelangen wir zur Vermutung

$$T(n) = ne + cn\log_b(n) + d \cdot \frac{n - 1}{a - 1}.$$

Lösung 2.3 *Offene Hashverfahren.*

a) (i) Lineares Sondieren:

10: $h(10) = 3$

			10			

18: $h(18) = 4$

			10	18		

2: $h(2) = 2$

		2	10	18		

4: $h(4) = 4 \to h(4) - 1 \equiv 3 \to h(4) - 2 \equiv 2 \to h(4) - 3 \equiv 1$

	4	2	10	18		

17: $h(17) = 3 \to h(17) - 1 \equiv 2 \to h(17) - 2 \equiv 1 \to h(17) - 3 \equiv 0$

17	4	2	10	18		

Die Anzahl der Kollisionen beträgt 6.

(ii) Quadratisches Sondieren:

10: $h(10) = 3$

			10			

18: $h(18) = 4$

			10	18		

2: $h(2) = 2$

		2	10	18		

4: $h(4) = 4 \to h(4) - 1 \equiv 3 \to h(4) + 1 \equiv 5$

		2	10	18	4	

17: $h(17) = 3 \to h(17) - 1 \equiv 2 \to h(17) + 1 \equiv 4 \to h(17) - 4 \equiv 6$

		2	10	18	4	17

Die Anzahl der Kollisionen beträgt 5.

(iii) Double Hashing:

10: $h(10) = 3$

			10			

18: $h(18) = 4$

			10	18		

2: $h(2) = 2$

		2	10	18		

4: $h(4) = 4 \to h(4) - h'(4) \equiv 6$

		2	10	18		4

17: $h(17) = 3 \to h(17) - h'(17) \equiv 0$

17		2	10	18		4

Die Anzahl der Kollisionen beträgt 2. Damit ist Double Hashing für diese Situation optimal.

b) Die Löschung eines Schlüssels k ist problematisch, wenn ein weiterer Schlüssel k' mit $h(k) = h(k')$ nach Schlüssel k eingefügt wurde. Würde k einfach gelöscht werden (z.B. indem die Position als frei markiert wird), dann könnte der Schlüssel k' nicht mehr gefunden werden, da das Sondieren endet, sobald eine freie Position gefunden wird. Im gegebenen Beispiel könnte der Schlüssel 17 nicht mehr gefunden werden. Aber auch der Schlüssel 4 könnte (zumindest bei linearem und quadratischem Sondieren) nicht mehr gefunden werden, denn die Sondierung bräche vorzeitig ab, sobald sie die (leere) Position 3 besucht. Folglich muss die entsprechende Position explizit als gelöscht markiert werden, und bei der Suche nach einem Schlüssel muss die Sondierung fortgesetzt werden, sobald eine solche Position gefunden wird. Selbstverständlich darf die Markierung beim späteren Einfügen eines anderen Schlüssels überschrieben werden, sofern dies nötig ist.

Werden nun viele Schlüssel gelöscht, dann kann es passieren, dass die Suche nach einem Schlüssel sehr ineffizient wird (denn die Sondierung besucht u.U. viele als gelöscht markierte Positionen). Hashing ist also besonders dann geeignet, wenn Schlüssel grösstenteils eingefügt und gesucht, jedoch nur selten gelöscht werden.

c) Auch hier nehmen wir an, dass beide Sondierverfahren zuerst nach links sondieren. Wir fügen in die Hashtabelle die 4-Tupel $6j+3, 6j+4, 6j+5, 6j+m+4$ für $j = 0, ..., \lfloor n/6 \rfloor - 1$ ein. Quadratisches Sondieren erzeugt für jedes 4-Tupel genau drei Kollisionen und fügt die Elemente schliesslich an den Stellen $6j+3, 6j+4, 6j+5$ und $6j$ ein. Lineares Sondieren erzeugt nur jeweils zwei Kollision und fügt das letzte Element des Tupels an der Stelle $6j+2$ ein.

Sondiert man beim quadratischen Sondieren hingegen zuerst nach rechts, so reicht es, wenn man Tripel der Form $3j+1, 3j+2, 3j+1+m$ für $j = 0, ..., \lfloor n/3 \rfloor - 1$ einfügt. Quadratisches Sondieren erzeugt für jedes solche Tripel genau zwei Kollisionen und fügt die drei Elemente schliesslich an den Stellen $3j+1, 3j+2, 3j$ ein. Lineares Sondieren erzeugt nur jeweils eine Kollision und fügt die Elemente an die gleichen Positionen ein.

In diesem Fall gibt es sogar eine einfachere Sequenz, bei der quadratisches Sondieren $n-1$ Kollisionen erzeugt und lineares nur eine, nämlich $1, 1, 2, 3, 4, ..., n-1$. Bei quadratischem Sondieren kollidiert jedes Element mit seinem Vorgänger in der Einfügereihenfolge. Bei linearem Sondieren kollidiert lediglich die zweite 1 mit der ersten und wird dann an Position 0 eingefügt.

Lösung 2.4 *Kuckucks-Hashing (Cuckoo hashing).*

a) Im folgenden wird Tabelle T_1 (links) und T_2 (rechts) nach jeder Einfügeoperation dargestellt:

			3	
0	1	2	3	4

0	1	2	3	4

	16		3	
0	1	2	3	4

0	1	2	3	4

	16		18	
0	1	2	3	4

3				
0	1	2	3	4

	16		23	
0	1	2	3	4

3			18	
0	1	2	3	4

	1		18	
0	1	2	3	4

3			16	23
0	1	2	3	4

b) Beim Einfügen des Schlüssels 48 terminiert das Verfahren nicht. In der folgenden Tabelle werden alle Verdrängungsoperationen aufgeführt bis dieselbe Tabellenkonfiguration wie zu Beginn auftritt.

Schlüssel	Eintrag in T_1		Eintrag in T_2	
	vorher	nachher	vorher	nachher
48	18	48	–	–
18	–	–	16	18
16	1	16	–	–
1	–	–	3	1
3	48	3	–	–
48	–	–	23	48
23	3	23	–	–
3	–	–	1	3
1	16	1	–	–
16	–	–	18	16
18	23	18	–	–
23	–	–	48	23
48	18	48	–	–
...	...	...	...	...