Übungen zu Algorithmen
WSI für Informatik
Kaufmann/Kuckuk/Förster

Wintersemester 2020/21
Universität Tübingen
18. November 2021

Aufgabensammlung Teil 4

Thema: Hashing

Aufgabe 1: Hashing mit Verkettung

Dies ist eine alte Klausuraufgabe!

Betrachten Sie Hashing mit Verkettung.

a) Fügen Sie die Elemente 6,4,1,5,2,9,7,11 nacheinander in eine zu Beginn leere Hashtabelle der Größe 5 ein. Verwenden Sie dabei die Hashfunktion $h(x) = (2x + 1) \bmod 5$ und zeigen Sie alle Schritte.

b) Betrachten Sie nun die Hashfunktion $h_{a,b}(x) = (ax + b) \bmod 9$. Gibt es Werte für a and b, so dass es für die Elemente aus a) zu keiner Kollision kommt, wenn sie in eine zu Beginn leere Hashtabelle der Größe 9 eingefügt werden? Begründung!

c) Welche worst-case Laufzeiten ergeben sich jeweils für Suche, Einfügen und Löschen, wenn die Listen bei der Verkettung unsortiert sind?

c) und d) setzen die Themen "Suchen" und "Suchbäume" voraus.

d) Welche worst-case Laufzeiten ergeben sich jeweils für Suche, Einfügen und Löschen, wenn die Listen bei der Verkettung sortiert sind bzw. werden?

Aufgabe 2: Rehashing

Dies ist eine alte Klausuraufgabe!

Betrachten Sie ein Hashing-Verfahren, das folgendermaßen funktioniert: Wir verwenden eine Folge von Hashtafeln $T_0, T_1, T_2, \ldots$. Die Mindestgröße von jeder Hashtafel T_i ist 4 und T_0 hat die Größe 4 und ist leer. Der *Belegungsfaktor* β_i von T_i ist

$$\beta_i = \frac{\text{Anzahl der Elemente in } T_i}{\text{Größe } |T_i| \text{ von } T_i}.$$

Wenn nach **Einfügen** eines Elementes $\beta_i = 1$ gilt, dann wird eine neue Hashtafel T_{i+1} erzeugt, die die doppelte Größe von T_i hat. Andererseits wird eine Tafel T_{i+1} der halben Größe von T_i erzeugt, wenn nach **Löschen** eines Elementes $\beta_i \leq \frac{1}{4}$ und $|T_i| > 4$ gilt. Bei jeder **Verdopplung** und **Halbierung** werden alle Elemente aus T_i in die Tafel T_{i+1} eingefügt unter Verwendung der entsprechenden Hashfunktion für T_{i+1}.

Es wird Hashing mit Verkettung verwendet. Die Hashfunktion für Tabelle T_i ist definiert durch:

$$h_i : \mathbb{N} \to \{0, \ldots, |T_i| - 1\},\ h_i(x) = x \bmod |T_i|.$$

a) Fügen Sie die Elemente $6, 1, 2, 8, 24, 7, 11, 19$ in eine leere Hashtafel T_0 ein. Geben Sie die Zwischenschritte an!

b) Löschen Sie die Elemente $8, 1, 19, 24$ aus der aktuellen Hashtafel.

c) Nun sollen im Allgemeinen n Einfüge- bzw. Löschoperationen in einer zu Beginn leeren Hashtafel vorgenommen werden. Dabei sollen wieder die obigen Regeln beachtet werden. Nehmen Sie an, dass es keine Kollisionen gibt (d. h. die verketteten Listen haben höchstens die Größe 1.) Zeigen Sie dass die n Einfüge- bzw. Löschoperationen amortisierte Zeit $\mathcal{O}(n)$ brauchen.

Aufgabe 3: Hashing mit offener Adressierung

Betrachten Sie die folgende Hashfunktion

$$\begin{aligned} h_{a,b,m}\colon \mathbb{N} &\to \{0,\dots,m-1\} \\ x &\mapsto (ax+b) \bmod m \end{aligned}$$

und die Eingabewerte

$$S = \{13, 45, 64, 78, 116\} \ .$$

a) Geben Sie Werte für a, b und m an, sodass

   i. alle Werte aus S auf den selben Hashwert abgebildet werden.

   ii. alle Werte aus S auf unterschiedliche Hashwerte abgebildet werden.

b) Sei $g_i(x) := (h_{1,0,5}(x) + i) \bmod 5$. Geben Sie jeweils die Hashtabelle an, die entsteht, wenn S in aufsteigender Reihenfolge in eine zuvor leere Hashtabelle eingefügt wird. Verwenden Sie dabei Hashing mit offener Adressierung mit $(g_i)_{i\geq 0}$.

c) Zeigen Sie, dass es für alle Werte von a und b bei $m = 5$ zu Kollisionen kommt.

d) Geben Sie Werte für a, b und m an, sodass m minimal ist und alle Werte aus S auf unterschiedliche Hashwerte abgebildet werden.

Lösungen:

1. a) Ergebnis:

Hash-Feld	Wert(e)
0	2,7
1	5
2	
3	6,1,11
4	4,9

   b) Nein, da 2 mod 9 = 11 mod 9.

   c) Einfügen $\mathcal{O}(1)$ (Einfügen nach letztem Element). Suchen und Löschen jeweils $\mathcal{O}(n)$ (alle Werte können in gleicher Zelle stehen und man muss alle betrachten).

   d) Jeweils $\mathcal{O}(\log n)$ (mittels binärer Suche, dazu muss die Datenstruktur geeignet abgeändert werden, z.B. als AVL-Baum).

2. a) T_0:

Hash-Feld	Wert(e)
0	
1	1
2	6,2
3	

   T_1:

Hash-Feld	Wert(e)
0	8, 24
1	1
2	2
3	11
4	
5	
6	6
7	7

T_2:

Hash-Feld	Wert(e)
0	
1	1
2	2
3	19
4	
5	
6	6
7	7
8	8,24
9	
10	
11	11
12	
13	
14	
15	

b) T_3:

Hash-Feld	Wert(e)
0	
1	
2	2
3	11
4	
5	
6	6
7	7

c) Nutze Buchführungsmethode. Zwischen Erstellen von T_i und dem Erstellen von T_{i+1} finden mindestens $|T_i|/4$ Operationen statt. Jede Operation führt 4 Credits ein, diese können für die $|T_i|$ Kopierungen beim Erstellen von T_{i+1} genutzt werden. Damit kostet dann jede Operation $\mathcal{O}(1)$ plus die 4 Credits (kosten je $\mathcal{O}(1)$ Zeit).

3. a) Beispielwerte für a, b und m:
   i. $a = b = 0$, $m = 2$
   ii. $a = 1$, $b = 0$ und $m = 117$

b) Ergebnis:

Element	g_0	g_1	g_2	g_3	0	1	2	3	4
13	3	-	-	-	-	-	-	13	-
45	0	-	-	-	45	-	-	13	-
64	4	-	-	-	45	-	-	13	64
78	3	4	0	1	45	78	-	13	64
116	1	2	-	-	45	78	116	13	64

c) $h_{a,b,5}(13) = h_{a,b,5}(78)$ kollidieren immer, da 13 mod 5 = 78 mod 5.

d) S hat Größe 5, also $m \geq 5$. Mit $m = 5$ gibt es immer Kollisionen (siehe c)). Beispielsweise geht $a = 1$, $b = 0$ and $m = 6$.