Grundlagen: Algorithmen und Datenstrukturen

Woche 10

Tobias Eppacher

School of Computation, Information and Technology

30. Juni 2025

Inhalt

Aufgaben

E-Aufgaben

Hausaufgaben

Aufgabe 10.1 - Hashing mit Chaining

Veranschaulichen Sie Hashing mit Chaining. Die Größe m der Hash-Tabelle ist in den folgenden Beispielen jeweils die Primzahl 11. Die folgenden Operationen sollen nacheinander ausgeführt werden:

- insert $3, 11, 9, 7, 14, 56, 4, 12, 15, 8, 1$
- delete 56
- insert 25

Der Einfachheit halber sollen die Schlüssel der Elemente die Elemente selbst sein.

a) Verwenden Sie zunächst die Hashfunktion

$$g(x) = 5x \mod 11$$

b) Berechnen Sie die Hashwerte unter Verwendung der Hashfunktion

$$h(x) = \mathbf{a} \cdot \mathbf{x} \mod m$$

nach dem aus der Vorlesung bekannten Verfahren füreinfache universelle Hashfunktionen, wobei $\mathbf{a} = (7, 5)$ und $\mathbf{x} = (\lfloor \frac{x}{2^w} \rfloor \mod 2^w, x \mod 2^w)$ für $w = \lfloor \log_2 m \rfloor = \lfloor 3.45\ldots \rfloor = 3$ gilt und der Ausdruck $\mathbf{a} \cdot \mathbf{x}$ ein Skalarprodukt bezeichnet.

Aufgabe 10.1 - Hashing mit Chaining (a)

$$g(x) = 5x \mod 11$$

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$												

Aufgabe 10.1 - Hashing mit Chaining (a)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

Aufgabe 10.1 - Hashing mit Chaining (a)

1. Operation: insert(3)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10

Aufgabe 10.1 - Hashing mit Chaining (a)

2. Operation: insert(11)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10
				3						

Aufgabe 10.1 - Hashing mit Chaining (a)

3. Operation: insert(9)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10
11				3						

Aufgabe 10.1 - Hashing mit Chaining (a)

4. Operation: insert(7)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10
11	9			3						

Aufgabe 10.1 - Hashing mit Chaining (a)

5. Operation: insert(14)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3						

Aufgabe 10.1 - Hashing mit Chaining (a)

6. Operation: insert(56)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3 14						

Aufgabe 10.1 - Hashing mit Chaining (a)

7. Operation: insert(4)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3 14	56					

Aufgabe 10.1 - Hashing mit Chaining (a)

8. Operation: insert(12)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3 14	56				4	

Aufgabe 10.1 - Hashing mit Chaining (a)

9. Operation: insert(15)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3 14	56 12				4	

Aufgabe 10.1 - Hashing mit Chaining (a)

10. Operation: insert(8)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3	56				4	
				14	12				15	

Aufgabe 10.1 - Hashing mit Chaining (a)

11. Operation: insert(1)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3	56		8		4	
				14	12				15	

Aufgabe 10.1 - Hashing mit Chaining (a)

12. Operation: delete(56)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3	56		8		4	
				14	12				15	
					1					

Aufgabe 10.1 - Hashing mit Chaining (a)

13. Operation: insert(25)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3	12		8		4	
				14	1				15	

Aufgabe 10.1 - Hashing mit Chaining (a)

13. Operation: insert(25)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3	12		8		4	
				14	1				15	
				25						

Aufgabe 10.1 - Hashing mit Chaining (b)

b) Berechnen Sie die Hashwerte unter Verwendung der Hashfunktion

$$h(x) = \mathbf{a} \cdot \mathbf{x} \mod m$$

nach dem aus der Vorlesung bekannten Verfahren für einfache universelle Hashfunktionen, wobei $\mathbf{a} = (7, 5)$ und $\mathbf{x} = (\lfloor \frac{x}{2^w} \rfloor \mod 2^w, x \mod 2^w)$ für $w = \lfloor \log_2 m \rfloor = \lfloor 3.45 \ldots \rfloor = 3$ gilt und der Ausdruck $\mathbf{a} \cdot \mathbf{x}$ ein Skalarprodukt bezeichnet.

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$h(k(e))$												

Aufgabe 10.1 - Hashing mit Chaining (b)

$$h(x) = \mathbf{a} \cdot \mathbf{x} \mod m$$

$$\mathbf{a} = (7, 5) \qquad \mathbf{x} = \left(\left\lfloor \frac{x}{2^w} \right\rfloor \mod 2^w, x \mod 2^w \right)$$

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$h(k(e))$												

Aufgabe 10.1 - Hashing mit Chaining (b)

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$h(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

Aufgabe 10.2 - B-Baum Analyse (Viel Text)

Der Aufwand nach dem Einfügen oder Entfernen einen perfekt balancierten Binärbaum wiederherzustellen ist $\mathcal{O}(n)$ im Worstcase. Daher versuchen AVL-Bäume keinen perfekt balancierten Binärbaum zu erreichen sondern stellen lediglich einen möglichst gut balancierten Binärbaum sicher.

Die Tiefe eines AVL-Baums ist dabei im Worstcase $1.44 * \log_2(n)$ und bei einem Rot-Schwarz-Baum $2 * \log_2(n)$. Diese Tiefe ist bei Binärbäumen auch die maximale Anzahl an notwendigen Vergleichen z.B. beim Suchen eines Werts.

Wir wollen nun so eine Analyse für einen B-Baum machen. B-Bäume sind eng verwandt mit den AB-Bäumen. Ein k-Baum ist dasselbe wie ein k,$2k$-Baum, bei einem B-Baum ist also die Obergrenze fur die Anzahl der Kinder eines Knoten genau das Doppelte.

Aufgabe 10.2 - B-Baum Analyse (a)

Wie ist ein Baum aufgebaut, der die größt mögliche Tiefe mit der kleinst möglichen Anzahl an Elementen erreicht?

Schätzen Sie die notwendigen Vergleiche im Worstcase für eine Suche nach einem Wert in diesem Baum ab.

Hinweis: Es müssen dabei nicht alle Werte aus einem Knoten betrachtet werden.

Aufgabe 10.2 - B-Baum Analyse (a)

Aufgabe 10.2 - B-Baum Analyse (b)

Wie sieht der Baum aus, sodass möglichst viele Vergleiche im Worstcase im Verhältnis zu der Anzahl von Elementen im Baum notwendig sind bei der Suche nach einem Wert?

Wie viele Vergleiche sind nun im Worstcase schätzungsweise nötig?

Aufgabe 10.2 - B-Baum Analyse (b)

Aufgabe 10.2 - B-Baum Analyse (c)

Wie verhält sich die notwendige Anzahl an Vergleichen bei wachsenden k?

Erkläre warum sich die Anzahl der Vergleiche so verhält.

Aufgabe 10.2 - B-Baum Analyse (c)

Aufgabe 10.3 - Klausurvorbereitung: Transfer

a. Wie viele verschiedene Baumstrukturen sind mit 11 Elementen bei einem 3,5-Baum möglich?

b. Wie viele verschiedene Baumstrukturen sind bei einem 3-Baum (3,6-Baum) mit 3 Ebenen möglich?

Aufgabe 10.3 - Klausurvorbereitung: Transfer (a)

Wie viele verschiedene Baumstrukturen sind mit 11 Elementen bei einem 3,5-Baum möglich?

Aufgabe 10.3 - Klausurvorbereitung: Transfer (a)

Wie viele verschiedene Baumstrukturen sind mit 11 Elementen bei einem 3,5-Baum möglich?

Aufgabe 10.3 - Klausurvorbereitung: Transfer (b)

Wie viele verschiedene Baumstrukturen sind bei einem 3-Baum (3,6-Baum) mit 3 Ebenen möglich?

Aufgabe 10.3 - Klausurvorbereitung: Transfer (b)

Wie viele verschiedene Baumstrukturen sind bei einem 3-Baum (3,6-Baum) mit 3 Ebenen möglich?

Aufgabe 10.4 - Klausurvorbereitung: Fehlersuche

Schauen Sie sich folgende drei Methoden an, die einen Algorithmus aus der Vorlesung implementiert haben. Beschreiben Sie welcher Algorithmus implementiert wurde und was dessen Laufzeit sein sollte. Geben Sie zudem den oder die Fehler in den gegebenen Methoden an und beschreiben Sie welche Auswirkungen die Fehler auf die Laufzeit und/oder das Ergebnis der Methoden haben. Beschreiben sie zudem, wie der Code verändert werden muss, um den Fehler zu beheben. Sind diese Methoden genau so performant wie die Implementierung aus der Vorlesung?

Aufgabe 10.4 - Klausurvorbereitung: Fehlersuche

a) Methode 1:

```
1  public static void foo(int[] arr, int l, int r) {
2      if (l >= r - 1) {
3          return;
4      }
5
6      int p = arr[r - 1];
7      int i = l;
8      for (int j = l; j < r - 1; j++) {
9          if (arr[j] < p) {
10             int temp = arr[i];
11             arr[i] = arr[j];
12             arr[j] = temp;
13             i++;
14         }
15     }
16     int temp = arr[i];
17     arr[i] = arr[r - 1];
18     arr[r - 1] = temp;
19
20     foo(arr, l, i);
21     foo(arr, i, r);
22 }
```

Aufgabe 10.4 - Klausurvorbereitung: Fehlersuche

b) Methode 2:

```
1  public static int bar(int[] arr, int v) {
2      int l = 0;
3      int r = arr.length;
4      while (l < r) {
5          int m = l + (r - l) / 2;
6          if (arr[m] - v < 0) {
7              l = m + 1;
8          } else if (arr[m] - v > 0) {
9              r = m - 1;
10         } else {
11             return m;
12         }
13     }
14     return -1;
15 }
```

Aufgabe 10.4 - Klausurvorbereitung: Fehlersuche

c) Methode 3:

stream erzeugt dabei einen Stream aus dem übergebenen Arrray, *of* gibt einen Stream zurück, der nur den angegebenen Wert enthält, und *concat* konkateniert zwei Streams zu einem längeren.

```
1  public static int[] baz(int[] arr) {
2      if (arr.length == 0) {
3          return arr;
4      }
5
6      int pivot = arr[arr.length - 1];
7      int[] left = stream(arr).filter(x -> x < pivot).toArray();
8      int[] right = stream(arr).filter(x -> x > pivot).toArray();
9      return concat(
10         concat(stream(baz(left)), of(pivot)),
11         stream(baz(right))
12     ).toArray();
13 }
```

E-Aufgaben

- Aufgabe 10.5 - Chaining 2
  - Hashing mit Chaining Wiederholung
- Aufgabe 10.6 - Klausurvorbereitung: Transfer 2
  - Theoriefragen zu Bäumen

Hausaufgaben

- Hausaufgabe 8 - AVL Baum
  (Deadline: 02.07.2025)
- Hausaufgabe 9 - AB-Baum
  (Deadline: 09.07.2025)
- Hausaufgabe 10 - Simple Hashing with Chaining
  (Deadline: 16.07.2025)

Fragen?

- Nach Übung gerne bei mir melden
- Tutoriumschannel oder DM an mich auf Zulip
- Vorlesungschannels von GAD auf Zulip (insbesondere bei Hausaufgaben)

Feedback oder Verbesserungsvorschläge?
Gerne nach dem Tutorium mit mir quatschen oder DM auf Zulip

Bis nächste Woche!