TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einführung in die Informatik 2 für Ingenieure (MSE)*

Christoph Anneser (anneser@in.tum.de)
http://db.in.tum.de/teaching/ss21/ei2/

Blatt Nr. 8
Dieses Blatt wird am Montag, den 14. Juni 2021 besprochen.

Aufgabe 1: Hashing mit Linear Probing

Veranschaulichen Sie Hashing mit Linear Probing.

Die Größe der Hashtabelle ist dabei jeweils $m = 13$. Führen Sie die folgenden Operationen aus:

`insert`	$16, 3, 12, 17, 29, 10, 24$
`delete`	16
`insert`	$5, 1, 15$
`delete`	10
`insert`	14
`delete`	1

Verwenden Sie die Hashfunktion

$$h(x) = 3x \bmod 13.$$

Bei dieser Aufgabe sind die Schlüssel der Elemente die Elemente selbst.

Beim **Löschen** soll die Wiederherstellung der folgenden Invariante erfolgen: *Für jedes Element e in der Hashtabelle mit Schlüssel $k(e)$, aktueller Position j und optimaler Position $i = h(k(e))$ sind alle Positionen $i, (i+1) \bmod m, (i+2) \bmod m, \ldots, j$ der Hashtabelle belegt.* Überlegen Sie sich eine effiziente Strategie, die diese Invariante widerherstellt. Bei dieser Aufgabe soll keine dynamische Größenanpassung der Hashtabelle stattfinden.

1. Operation: `insert`(16) mit opt. Position:

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

2. Operation: `insert`(3) mit opt. Position:

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

3. Operation: `insert`(12) mit opt. Position:

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

4. Operation: `insert`(17) mit opt. Position:

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

5. Operation: `insert`(29) mit opt. Position:

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

6. Operation: `insert`(10) mit opt. Position:

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

7. Operation: `insert`(24) mit opt. Position:

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

8. Operation: `delete`(16) mit opt. Position:

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

9. Operation: `insert`(5) mit opt. Position:

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

10. Operation: `insert`(1) mit opt. Position:

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

11. Operation: `insert`(15) mit opt. Position:

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

12. Operation: `delete`(10) mit opt. Position:

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

13. Operation: `insert`(14) mit opt. Position:

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

14. Operation: `delete`(1) mit opt. Position:

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

Aufgabe 2: 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.

Verwenden Sie zunächst die Hashfunktion

$$g(x) = 5x \mod m.$$

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

1. Operation: `insert`(3):

0	1	2	3	4	5	6	7	8	9	10

2. Operation: `insert`(11):

0	1	2	3	4	5	6	7	8	9	10

3. Operation: `insert`(9):

0	1	2	3	4	5	6	7	8	9	10

4. Operation: `insert`(7):

0	1	2	3	4	5	6	7	8	9	10

5. Operation: `insert`(14):

0	1	2	3	4	5	6	7	8	9	10

6. Operation: `insert`(56):

0	1	2	3	4	5	6	7	8	9	10

7. Operation: `insert`(4):

0	1	2	3	4	5	6	7	8	9	10

8. Operation: `insert`(12):

0	1	2	3	4	5	6	7	8	9	10

9. Operation: `insert`(15):

0	1	2	3	4	5	6	7	8	9	10

10. Operation: `insert`(8):

0	1	2	3	4	5	6	7	8	9	10

11. Operation: `insert`(1):

0	1	2	3	4	5	6	7	8	9	10

12. Operation: `delete`(56):

0	1	2	3	4	5	6	7	8	9	10

13. Operation: `insert`(25):

0	1	2	3	4	5	6	7	8	9	10

Aufgabe 3: Hashing mit Linear Chaining - Implementierung

Implementieren Sie in Java eine Hashtabelle die "linear chaining" zur Kollisionsbehandlung verwendet: Jede Position in der Hashtabelle speichert eine Liste von Elementen mit diesem Hashwert. Bonus: Implementieren Sie einen Mechanismus um die Hashtabelle wachsen zu lassen.