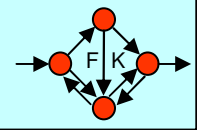
Hashfunktionen

3.3.2.1 Aufgabe:
Elemente einer Menge B sollen in einem array (0..p-1) of ... gesucht und dort in irgendeiner Reihenfolge eingefügt und gelöscht werden können. Es sei |B| > p (sonst ist die Aufgabe ohne Konflikte durch irgendeine injektive Zuordnung zu lösen).

Benutze hierfür eine Funktion f: B → {0, 1, ..., p-1}, genannt *Hashfunktion*, die **surjektiv** ist (d.h., jede Zahl von 0 bis p-1 tritt als Bild auf), die die Elemente von B möglichst gleichmäßig über die Zahlen von 0 bis p-1 verteilt und die schnell berechnet werden kann.



Kollisionsstrategien

Wenn auf Platz j eine Kollision stattfindet, so versuche man, den Schlüssel b auf dem Platz G(...) einzufügen. Es sei i die Zahl der versuchten Zugriffe. c ist eine fest gewählte Konstante; man kann hier stets c=1 wählen; wichtig ist ggT(c,p)=1.

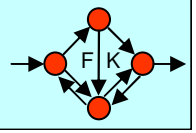
$G(j) = (j+c) \underline{\text{mod}}\ p$ heißt "lineare Fortschaltung" oder "lineares Sondieren" oder "Lineares Hashing".

$G(b,i) = (f(b)+i^2) \underline{\text{mod}}\ p$ heißt "quadratische Fortschaltung" oder "quadratisches Sondieren".

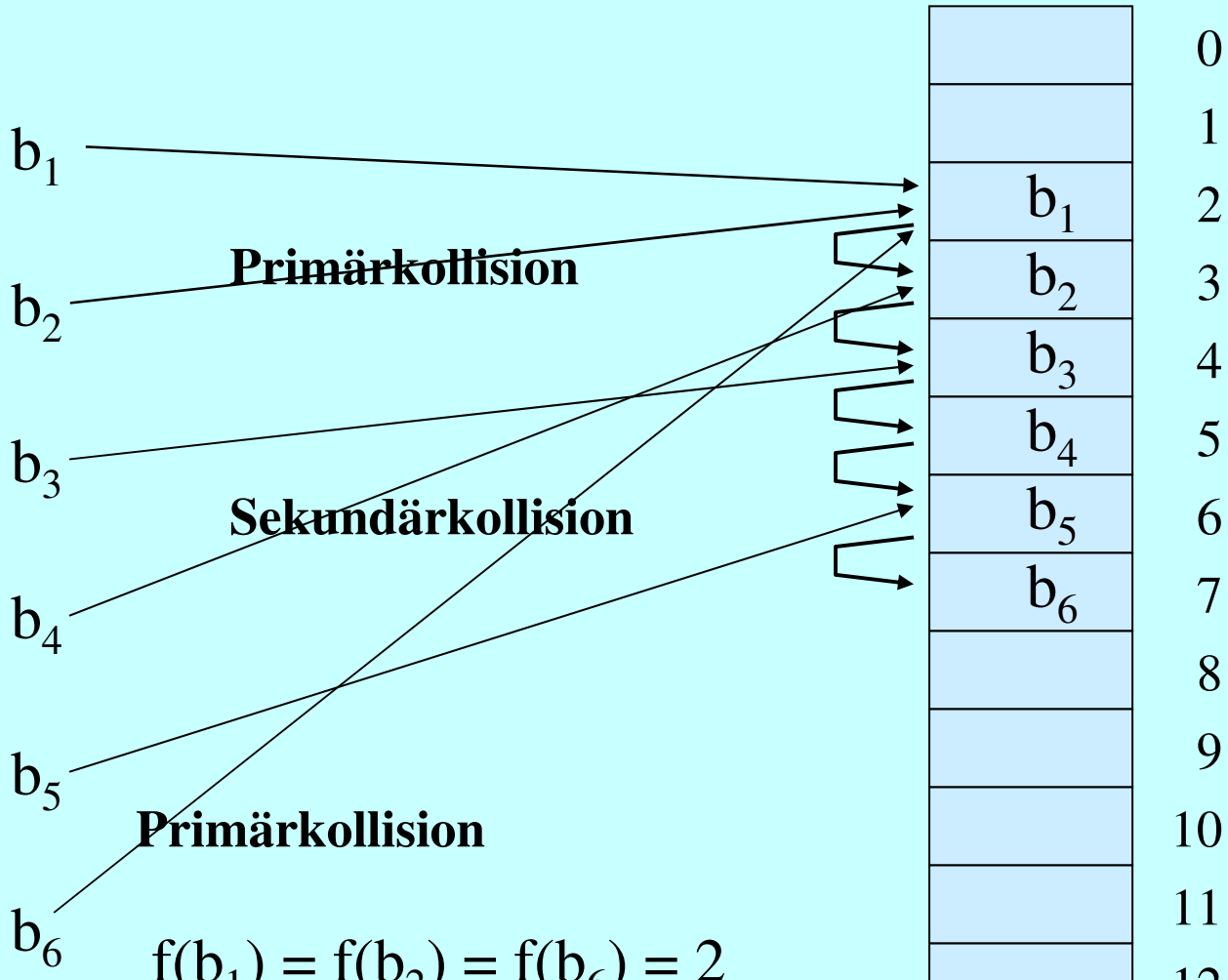
Nachteile:

Lineares Sondieren: Clusterbildung durch Sekundärkollissionen

Quadratisches Sondieren: Aufwand des Löschens ist groß



Beispielskizze: Lineares Sondieren mit c=1 p=13



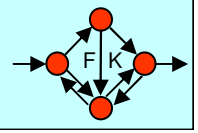
$f(b_1) = f(b_2) = f(b_6) = 2$

$f(b_3) = 4, f(b_4) = 3, f(b_5) = 6$

Löschen in Hashtabellen (DELETE)

Der einfachste Weg ist es, das Löschen durch Setzen eines Booleschen Wertes zu realisieren: Wenn der Eintrag mit dem Schlüssel b gelöscht werden soll, so suche man seine Position A(j) auf und setze A(j).geloescht := true.
Beim Einfügen behandelt man dieses Feld A(j) dann wie einen freien Platz.

Hat man sich jedoch für das lineare Sondieren entschieden, dann kann man das Löschen korrekt durchführen: Man sucht den Eintrag A(j) mit dem zu löschenden Element auf und geht dann die Einträge A(j+c), A(j+2c), A(j+3c) solange durch, bis man auf einen freien Platz stößt. In dieser Kette kopiert man alle Einträge um c, 2c, 3c usw. Plätze zurück, aber niemals über den Platz k hinaus mit f(b)=k.



Aufgabe 1 (Lineares und Quadratisches Sondieren – offenes Hashing)

Geben Sie die Belegung einer Hashtabelle der Länge 13 an, wenn die Schlüssel

5, 1, 19, 23, 14, 17, 32, 30, 2

in eine anfangs leere Tabelle eingefügt werden und offenes Hashing mit der Hashfunktion $f(x) = x \bmod 13$ und

a) lineares Sondieren
b) quadratisches Sondieren

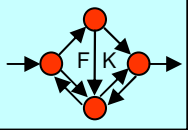
verwendet wird. Vergleichen Sie außerdem die Anzahl der beim Einfügen betrachteten Hashtabellenplätze für die beiden Sondierungsverfahren. Welche Kosten sind jeweils für eine erfolgreiche Suche zu erwarten, wenn nach jedem vorhandenen Schlüssel mit gleicher Wahrscheinlichkeit gesucht wird?

Annahmen des Hashing

Hash Strategien arbeiten nur im Mittel schnell. Die Operationen Insert, Find und Delete haben im Worst case einen Aufwand von O(n).

Die folgenden Abschätzungen verlangen die Gleichverteilung der Schlüssel und die Unabhängigkeit der Ereignisse und folgende Annahmen:

1. Die Hashfunktion f ist gleichverteilt über die Schlüsselmenge, sie bevorzugt oder benachteiligt dort keine Bereiche.
2. Jeder Schlüssel ist bei beim Suchen und beim Einfügen gleichwahrscheinlich.
3. Erfolgt beim Einfügen eines Schlüssels eine Kollision, so werden bis zu dessen Eintrag auf einen freien Platz nur paarweise verschiedene Plätze besucht.



Aufwand des Hashing (wichtige Daten)

Um den (k+1)-ten Schlüssel in eine Hashtabelle der Größe p einzufügen, werden im Mittel $\frac{p+1}{p+1-k} = \frac{1}{1-\lambda}$ Vergleiche (mit $\lambda = k/(p+1)$) benötigt.

Wahrscheinlichkeit beim Suchen auf ein belegtes Feld zu treffen

$$(1-1/p)^k \approx e^{-\frac{k}{p}} = e^{-\lambda}$$

mit $\lambda = k/p$ "Auslastungsgrad" der Tabelle.

Für die erfolgreiche Suche nach einem Schlüssel in einer Hashtabelle der Größe p werden im Mittel $S_k \approx \frac{1}{\lambda} \cdot \ln(\frac{1}{1-\lambda})$ Vergleiche (mit $\lambda = k/(p+1)$) benötigt.

Doppel-Hash-Verfahren

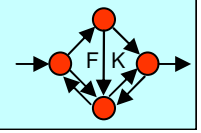
Es seien f und g zwei unterschiedliche Hashfunktionen. Sei i die Zahl der Zugriffe. Die Kollisionsstrategie

$G(i,b) = (f(b) + i \cdot g(b)) \underline{\text{mod}}\ p$ heißt "Doppel-Hash-Verfahren".

Es seien f_1, f_2, f_3, f_4, ... eine Folge von möglichst unterschiedlichen Hashfunktionen. Die Kollisionsstrategie

$G(i,b) = f_i(b)$ heißt "Multi-Hash-Verfahren".

Hinweis: In der Praxis hat man mit Doppel-Hash-Strategien gute Erfahrungen gemacht.



Aufgabe 2 (Löschen beim Hashing)

Gegeben sei eine Hashtabelle der Größe m = 11. Der Zugriff soll mittels linearem Sondieren mit der folgenden Hashfunktion erfolgen: $f(x) = x \bmod 11$. Tragen Sie die folgenden Zahlen in dieser Reihenfolge in die Tabelle ein und zeichnen Sie jeweils den Zustand der Tabelle nach dem Eintragen jeder einzelnen Zahl:

5, 21, 18, 3, 39, 9, 32, 14.

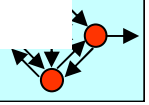
Danach löschen Sie die Zahlen 21, 39, 3.
Schließlich fügen Sie noch die Zahlen 16, 7, 8 ein.
Zeichnen Sie wieder den Zustand der Tabelle nach jeder Operation.

Aufgabe 3 (Löschen 2. Teil)

Wie sieht eine Hashtabelle der Größe 11 aus, wenn nacheinander die Werte

15, 38, 21, 26, 39, 37, 16, 6

eingefügt und danach die Werte 26 und 6 gelöscht werden? Als Hashfunktion soll $f(x) = x \bmod 11$ verwendet werden.



Aufgabe 4 (Doppel Hashing)
Gegeben sei eine anfangs leere Hashtabelle mit 11 Elementen, in die der Reihe nach die Schlüssel 12, 21, 1, 19, 12, 16, 4, 11 mittels Doppel Hashing eingefügt werden. Die zu verwendenden Hashfunktionen sind $f(x) = x \bmod 11$ und $g(x) = 1 + (x \bmod 7)$. Geben Sie die Belegung der Hashtabelle an, wenn die Schlüssel in

a) gegebener Reihenfolge

b) aufsteigend sortiert

eingefügt werden.

Aufgabe 5 (Doppel Hashing 2. Teil)
Gegeben sei eine anfangs leere Hashtabelle mit 13 Elementen, in die der Reihe nach die Schlüssel 14, 21, 27, 28, 8, 18, 15, 36, 5 und 2 mit Doppel Hashing eingefügt werden sollen. Die zu verwendenden Hashfunktionen seien $f(x)=x \bmod 13$ und $g(x)=1+ (x \bmod 11)$. Geben Sie die Belegung der Hashtabelle an.

Aufgabe 6 (Doppel Hashing 3. Teil)

Gegeben ist eine Hashtabelle der Größe N = 13 und die Hashfunktion $f(x) = x \bmod 13$. Fügen Sie in der gegebenen Reihenfolge die Schlüssel

$$E_1,\ A,\ S_1,\ Y,\ Q,\ U,\ E_2,\ S_2,\ T,\ I,\ O,\ N$$

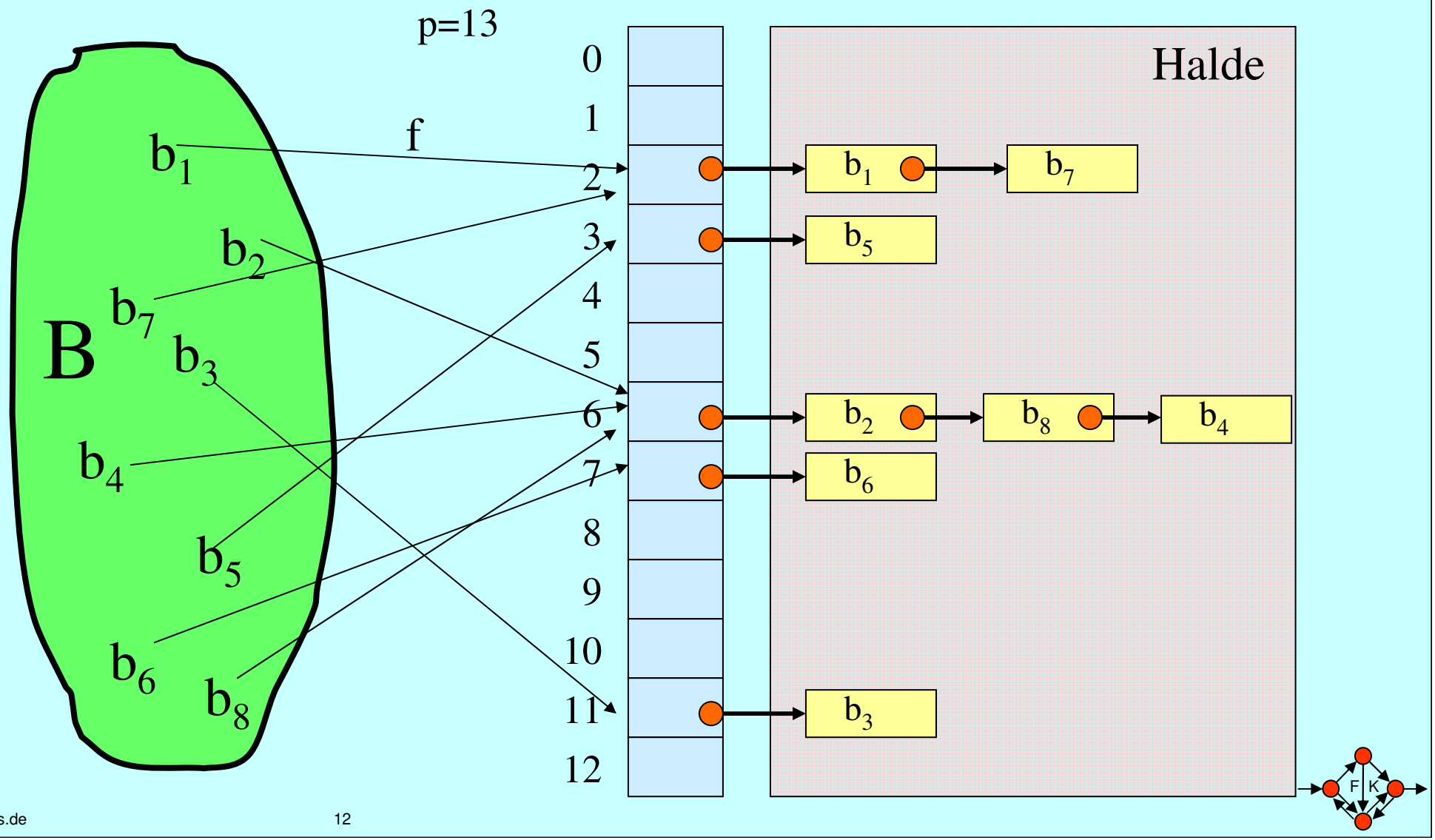
ein, wobei A, B, C, .. durch 0, 1, 2, .. codiert werden und Indizes nichts am Code eines Schlüssels ändern. Benutzen Sie dabei

(a) lineares Sondieren,
(b) quadratisches Sondieren mit $G(x, i) = (f(x) + i + i^2) \bmod 13$ und
(c) doppeltes Hashing mit f und g, wobei $g(x) = 1 + (x \bmod 11)$.

Geben Sie für jeden Schlüssel an, wie oft Sie vergeblich sondiert haben, bis Sie ihn erfolgreich platzieren konnten. Geben Sie außerdem für jede Hashvariante die Gesamtzahl der erfolglosen Sondierversuche an. Ist die Sondierfolge $G(x, i)$ für $i = 0, 1, ..12$ bei (b) tatsächlich eine Permutation von 0, 1, .. 12?

Externes Hashing

Skizze: "Externe" Hashtabelle

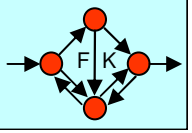


Aufgabe 7 (externes Hashing)

a) In eine anfangs leere Hashtabelle der Größe 13 werden in der angegebenen Reihenfolge die Schlüssel

5, 1, 19, 23, 27, 17, 32, 9, 2

eingefügt. Es wird externes Hashing mit Verketten verwendet, und zwar mit der Hashfunktion $f(x)=x \bmod 13$. Geben Sie die Belegung der Hashtabelle an!



Aufgabe 8 (Hashing) - ehemalige Klausuraufgabe

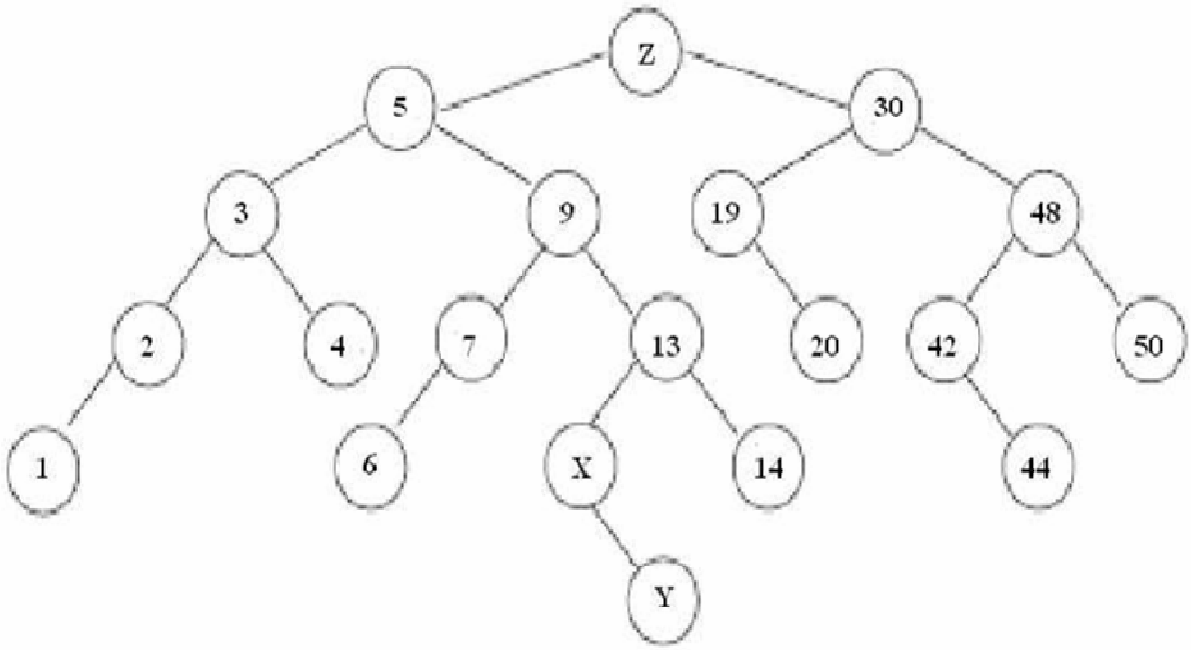
Gegeben sei für m = 11 die Hashfunktion $h(k) = k \bmod m$ und die Kollisionsstrategie Quadratisches Sondieren. Bisher wurden nur Elemente mit dem gleichen Hashwert eingefügt. Die Belegung der Tabelle sieht nach dem Einfügen von sechs Elementen so aus:

Speicherzelle *i*	0	1	2	3	4	5	6	7	8	9	10
Belegung		X		X	X		X	X	X		

a) (2 P) Welchen Hashwert X haben die Elemente?
b) (1 P) Fügen Sie jetzt ein Element mit dem Wert 6 ein. In welcher Speicherzelle wird das Element gespeichert?
c) (1 P) Warum können Sie kein weiteres Element mit dem Hashwert X in die Tabelle einfügen?

Aufgabe 9: (AVL-Bäume) - ehemalige Klausuraufgabe

Gegeben sei folgender AVL-Baum bei welchem alle Elemente vom Typ integer sind und verschieden sein sollen.



a) (2 P) Welche Werte können die Knoten X, Y und Z annehmen? Achten Sie insbesondere auf die Beziehung der Werte untereinander, z.B. wenn X=7 ist, dann kann Z nur 13 oder 15 sein.

b) (2 P) Welche Blätter kann man in diesem Baum löschen, ohne das Ausgleichsoperationen notwendig sind?

c) (3 P) Löschen Sie nun den Knoten Z und rebalancieren Sie den Baum. Beim Ersetzen von Knoten verwenden Sie dabei immer den Inordernachfolger. Beschreiben Sie die einzelnen Schritte.

www.lern-plus.de

Aufgabe 15 (Hashing – ehemalige Klausuraufgabe)

Gegeben sei für m = 13 die Hashfunktion $h(k) = k \bmod m$ und die Kollisionsstrategie Doppel-Hashing mit $h_i(k) = (h(k) + i \cdot (1 + k \bmod 7)) \bmod m$. In welcher Reihenfolge sind Schlüssel in eine zunächst leere Hashtabelle eingefügt worden, wenn die Hashtabelle wie folgt gefüllt ist:

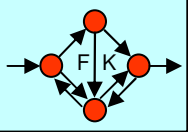
Speicherzelle *i*	0	1	2	3	4	5	6	7	8	9	10	11	12
Belegung	23					60		33	7		8		

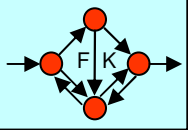
Beschreiben Sie die Kollisionstypen des Elementes 60.

Gegeben sei für $m = 7$ die Hashfunktion $h(k) = k \bmod m$ und die Kollisionsstrategie ‚Lineares Sondieren' mit $h_i(k) = (h(k) + i) \bmod m$. Die Hashtabelle ist wie folgt gefüllt:

Speicherzelle *i*	0	1	2	3	4	5	6
Belegung	13	7	0	3	20		6

Löschen Sie das Element ‚7' (an der Stelle 1) und reorganisieren Sie die Hashtabelle nach dem in der Vorlesung vorgestellten Verfahren ohne Verwendung eines ‚gelöscht Bit' und ohne Rehashing anzuwenden.




F
K