Prof. Dr.-Ing. Dr. h. c. T. Härder
Fachbereich Informatik
Arbeitsgruppe Datenbanken und Informationssysteme
Universität Kaiserslautern

Übungsblatt 7 – Lösungsvorschläge

Unterlagen zur Vorlesung: „wwwdvs.informatik.uni-kl.de/courses/DBSREAL/“

Aufgabe 1: Externes Hashing mit Separatoren

192

Lösung:

Gegeben seien 5 Buckets mit einer Kapazität von 3 Sätzen. In der folgenden Tabelle seine Zuordnungen von Schlüsseln zu Signaturfolgen bzw. Sondierungen gegeben:

Schlüssel	h	s
a	(1 3 5)	(1110 0100 0001)
b	(2 4 1)	(1011 1100 0101)
c	(4)	(0000)
d	(3)	(0001)
e	(3 5)	(0011 0011)
f	(3)	(0010)
g	(3 2 4)	(0011 1011 1000)
h	(5 3)	(1010 0001)
i	(2 4 5)	(1011 0111 0100)
j	(3 2 1)	(1110 1011 0001)
k	(5)	(0101)
l	(3 1)	(0011 1010)

Nach der Initialisierung seien alle Buckets leer. Anschließend werden folgende Operationen ausgeführt:

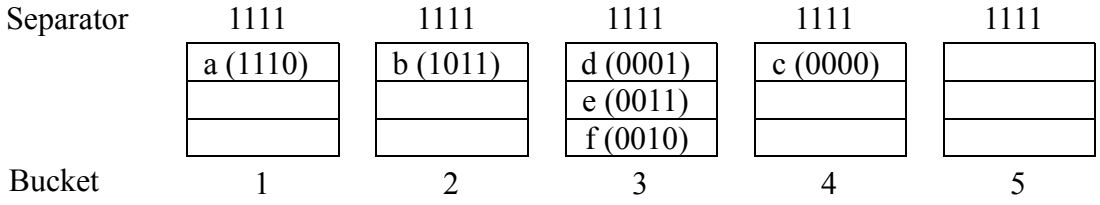
a) Einfügen der Sätze a, b, c, d, e, f, g, h, i, j

b) Löschen der Sätze d, e, f

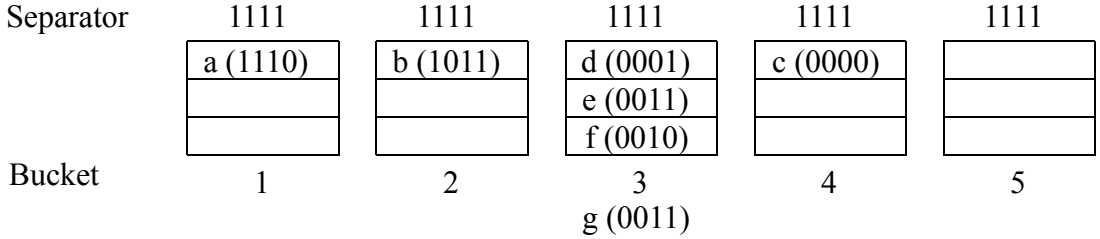
c) Einfügen der Sätze k, l

Machen Sie sich anhand dieser Operationen die Funktionsweise des Externen Hashing mit Separatoren klar.

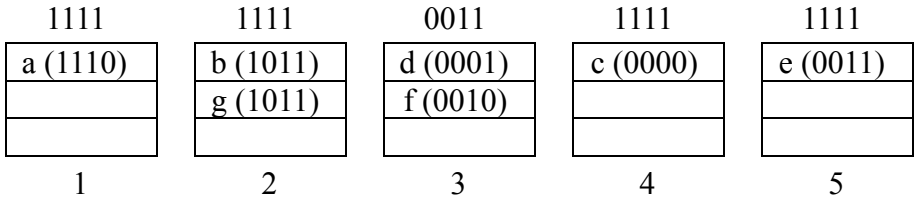
Einfügen von „a“ bis „f“



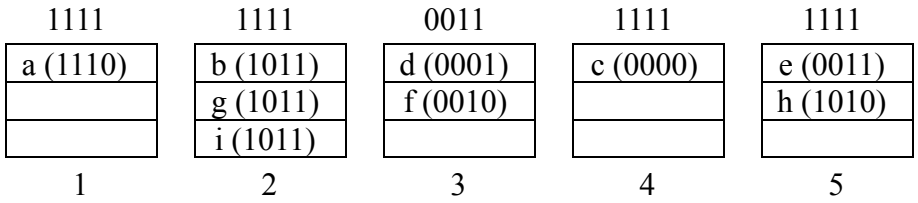
Einfügen von „g“ führt zu Kollision in Bucket 3



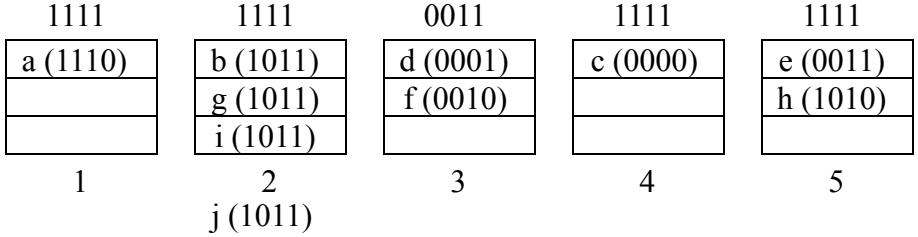
Neuverteilung von Bucket 3



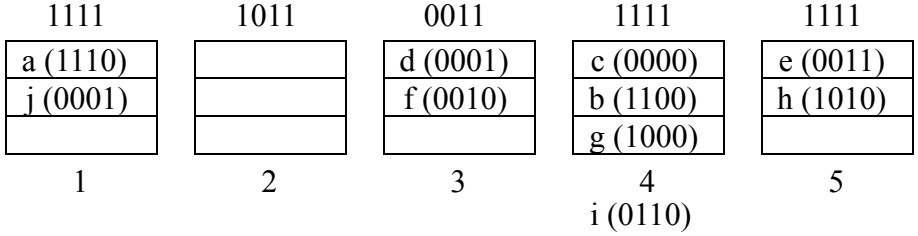
Einfügen von „h“ bis „i“



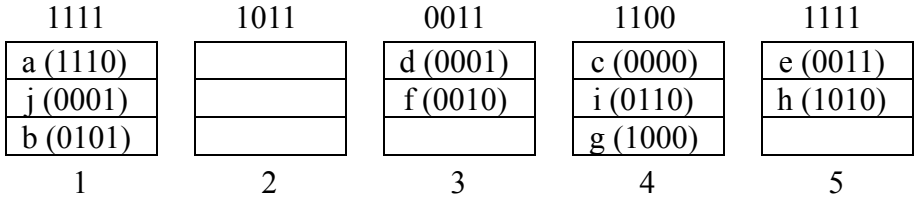
Einfügen von „j“ führt zu Kollision in Bucket 2



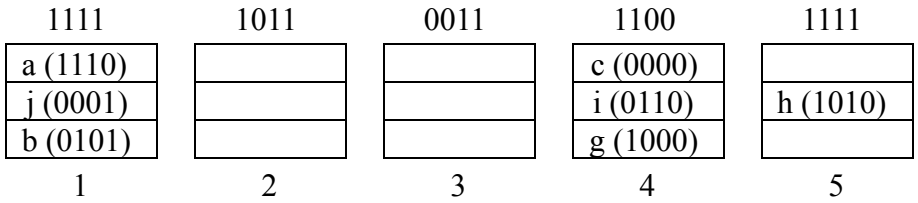
Neuverteilung von Bucket 2 führt zu Kollision in Bucket 4



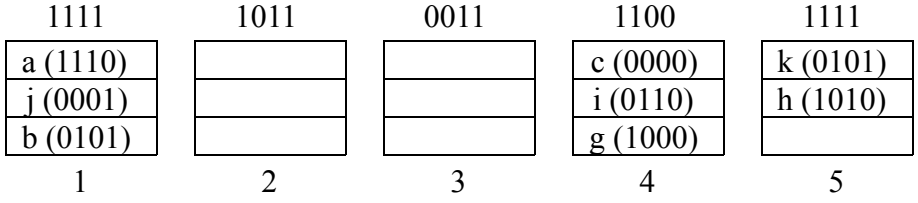
Neuverteilung von Bucket 4: „b“ nach Bucket 1.



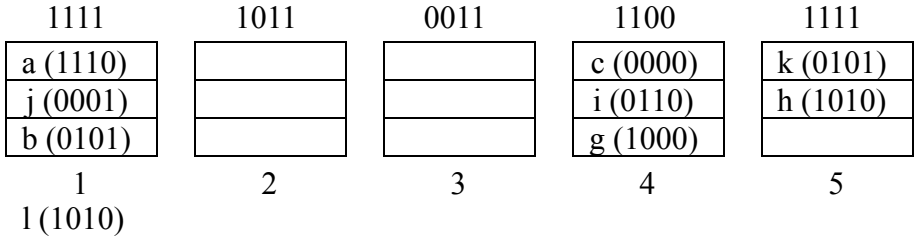
Löschen von „d“, „e“, „f“



Einfügen von „k“



Einfügen von „l“ führt zu Kollision in Bucket 1



Einfügen von „l“ führt zu Kollision in Bucket 1

1110	1011	0011	1100	1111
l (1010)			c (0000)	k (0101)
j (0001)			i (0110)	h (1010)
b (0101)			g (1000)	a (0001)
1	2	3	4	5

Welche Probleme treten auf, wenn bei einer gleichmäßigen Verteilung der Separatoren sehr viele Einfüge- und Löschoperationen ausgeführt werden?

Da Separatoren nie wieder auf einen größeren Wert gesetzt werden, kann es nach Überläufen passieren, dass ein Bucket aufgrund des niedrigen Separatorwertes nicht oder nur noch gering belegt wird.

Aufgabe 2: Lineares Hashing

Gegeben sei eine Datei mit 3 Buckets mit einer Kapazität von je 3 Datensätzen. Für die Folge von Hashfunktionen gelten folgende Funktionen:

$h_i(K) = K \bmod (3 * 2^i)$ mit $i \in \{0, 1, 2, ..., n\}$

Zu Beginn seien alle Buckets leer. Fügen Sie als Schlüssel folgende Zahlen in der gegebenen Reihenfolge mittels linearem Hashing ein:

7, 3, 2, 6, 10, 12, 11, 8, 14, 9, 23, 19, 24, 20, 25, 30

Zur Kontrolle der Speicherplatzbelegung verwenden Sie folgende Verfahren:

a) Unkontrolliertes Splitting

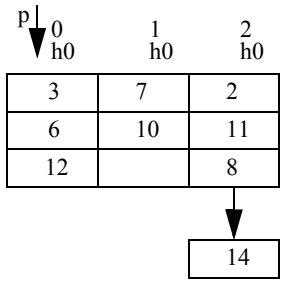
b) Kontrolliertes Splitting mit $\beta = 0,7$

Für beide Verfahren geben Sie jeweils den Zustand der Datei bzw. Buckets vor und nach jedem Split-Vorgang eines Buckets an.

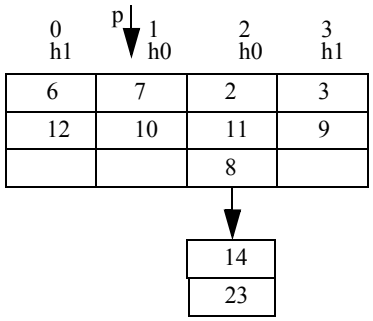
Lösung:

a) Unkontrolliertes Splitting

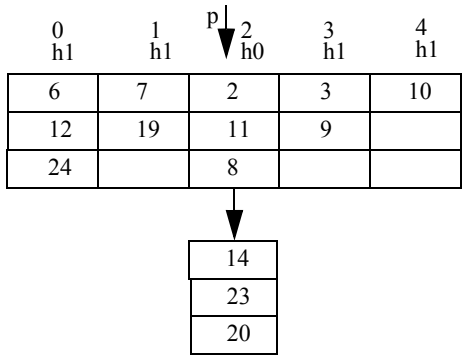
Einfügen: 7, 3, 2, 6, 10, 12, 11, 8, 14



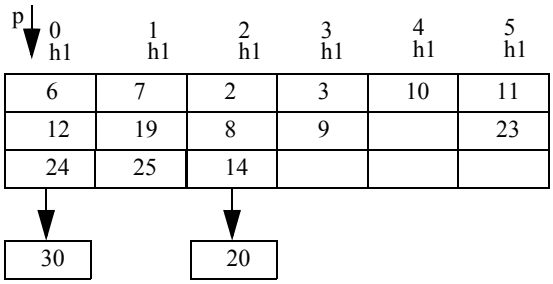
Einfügen: 9, 23



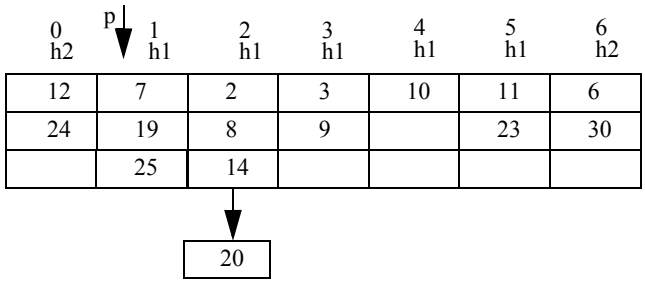
Einfügen: 19, 24, 20



Einfügen: 25, 30



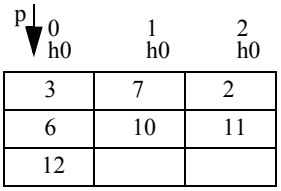
Nach dem Einfügen von 30:



b) Kontrolliertes Splitting mit β = 0,7

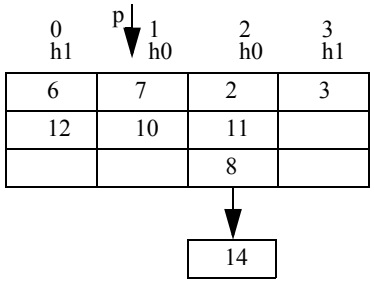
Splitting wird nach dem Einfügen vom 7., 9., 11., 13. und 15. Schlüssel durchgeführt (11, 14, 23, 24, 25)

Einfügen: 7, 3, 2, 6, 10, 12, 11



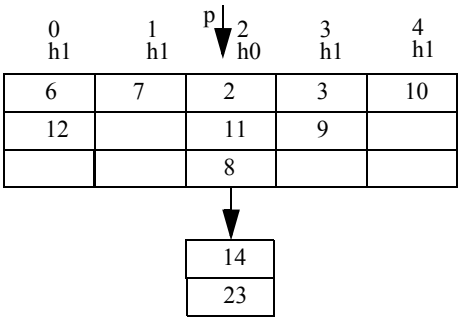
Nach dem Einfügen von 11:
$\beta = 7/(3*2^0+0)*3 = 7/9 > 0,7$

Einfügen: 8, 14



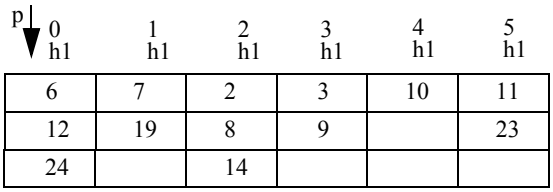
Nach dem Einfügen von 14:
$\beta = 9/(3*2^0+1)*3 = 9/12 > 0,7$

Einfügen: 9, 23



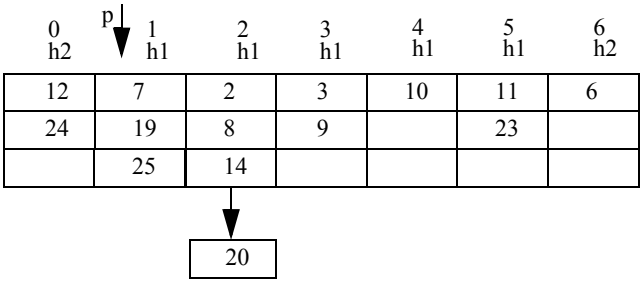
Nach dem Einfügen von 23:
$\beta = 11/(3*2^0+2)*3 = 11/15 > 0,7$

Einfügen: 19, 24



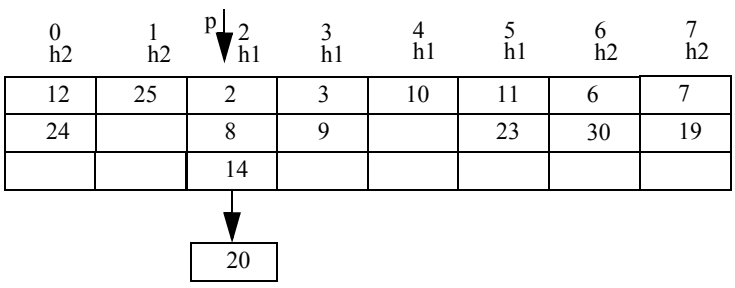
Nach dem Einfügen von 24:
$\beta = 13/(3*2^1+0)*3 = 13/18 > 0,7$

Einfügen von: 20, 25



Nach dem Einfügen von 25:
$\beta = 15/(3*2^1+1)*3 = 15/21 > 0,7$

Nach dem Einfügen von 30:



Aufgabe 3: Verweislisten und Bitlisten

Gegeben seien folgende Datensätze von Bestellungsinformationen, welche die Adressen Zn mit $n \in \{1, 2, .., 10\}$ besitzen:

Adressen	Bestellnummer	Bestelldatum	Kundennummer
Z1	711	01.02.1999	100
Z2	600	03.01.1999	102
Z3	801	01.02.1999	141
Z4	505	02.01.1999	100
Z5	475	02.01.1999	102
Z6	730	01.02.1999	102
Z7	621	05.01.1999	141
Z8	699	09.01.1999	102
Z9	670	09.01.1999	100
Z10	515	02.01.1999	180

Invertieren Sie die oben vorgegebenen Datensätze jeweils nach den Attributen Bestelldatum und Kundennummer mit folgenden Verfahren:

a) Verweislisten

b) Bitlisten

a) Verweislisten

Invertierung nach Bestelldatum:

02.01.1999	Z4	Z5	Z10
03.01.1999	Z2		
05.01.1999	Z7		
09.01.1999	Z8	Z9	
01.02.1999	Z1	Z3	Z6

Invertierung nach Kundennummer:

100	Z1	Z4	Z9	
102	Z2	Z5	Z6	Z8
141	Z3	Z7		
180	Z10			

b) Bitlisten

Invertierung nach Bestelldatum:

02.01.1999	0 0 0 1 1 0 0 0 0 1
03.01.1999	0 1 0 0 0 0 0 0 0 0
05.01.1999	0 0 0 0 0 0 1 0 0 0
09.01.1999	0 0 0 0 0 0 0 1 1 0
01.02.1999	1 0 1 0 0 1 0 0 0 0

Invertierung nach Kundennummer:

100	1 0 0 1 0 0 0 0 1 0
101	0 1 0 0 1 1 0 1 0 0
141	0 0 1 0 0 0 1 0 0 0
180	0 0 0 0 0 0 0 0 0 1

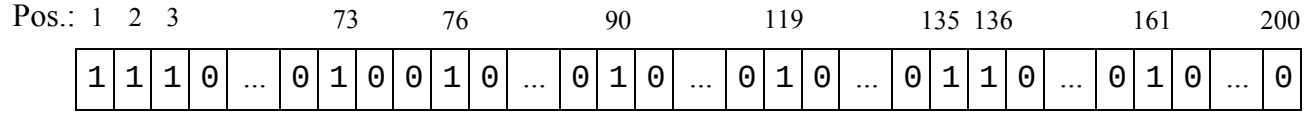
Aufgabe 4: Bitlistenkomprimierung

Gegeben sei eine Bitliste der Länge 200 mit folgenden Eigenschaften:

An den Positionen 1, 2, 3, 73, 76, 90, 119, 135, 136, 161 tritt der Wert ‘1’ und sonst ‘0’ auf.

Führen Sie für die oben angegebene Bitliste eine Komprimierung nach jedem der folgenden Verfahren:

a) *Laufkomprimierung* mit k = 6
b) *Nullfolgenkomprimierung* mit Codiereinheiten fester Länge mit k = 6
c) *Nullfolgenkomprimierung* mit Codiereinheiten variabler Länge mit einer festen Länge des Längen-
feldes *l* = 3
d) *Nullfolgenkomprimierung* unter der Anwendung der *Golomb-Codierung* mit m = 4

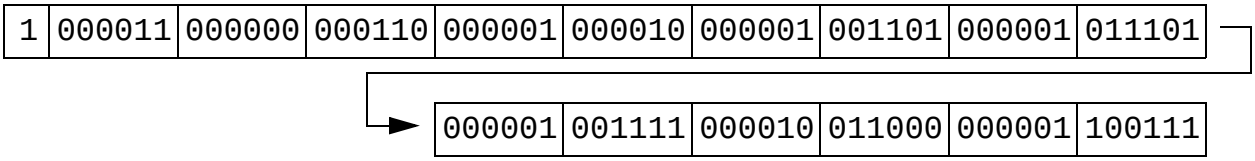


a) Laufkomprimierung mit k = 6:

Man hat folgende Eins- und Null-Folgen:

3 Einsen, 69 Nullen, 1, 2 Nullen, 1, 13 Nullen, 1, 28 Nullen, 1, 15 Nullen, 2 Einsen, 24 Nullen, 1 und

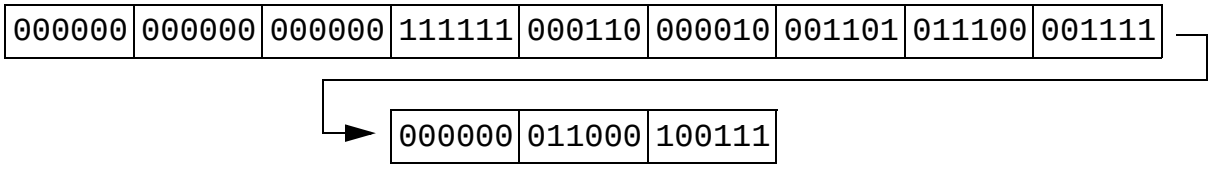
39 Nullen



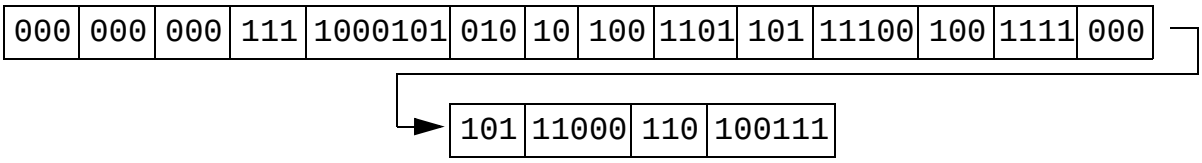
b) Nullfolgenkomprimierung mit Codiereinheiten fester Länge mit k =6:

Man hat folgende Nullfolgen:

Drei Nullfolgen der Länge 0, 69 Nullen, 2 Nullen, 13 Nullen, 28 Nullen, 15 Nullen, eine Null-foge der Länge 0, 24 Nullen und 39 Nullen



c) Nullfolgenkomprimierung mit Codiereinheiten variabler Länge mit $l = 3$:



d) Nullfolgenkomprimierung unter der Anwendung der *Golomb-Codierung* mit m = 4: