FernUniversität in Hagen

Lösungsvorschläge zur Hauptklausur 1661 „Datenstrukturen I“

15.08.2015

Aufgabe 1 Festival-Algebra

(a)

algebra	*festival*		
sorts	*person*, *camp*, *festival*, *stringset*		
ops	*createPerson*:	$string \times int^+ \times int^+ \times int^+$	$\rightarrow$ *person*
	createFestival:		$\rightarrow$ *festival*
	insertPerson:	$festival \times person \times string$	$\rightarrow$ *festival*
	name:	$camp \cup person$	$\rightarrow$ *string*
	names:	$camp \cup festival$	$\rightarrow$ *stringset*
	cnames:	*festival*	$\rightarrow$ *stringset*
	who	$string \times festival \cup camp$	$\rightarrow$ *person*
	position:	$festival \times string$	$\rightarrow$ *camp*
	visit:	$festival \times string \times string$	$\rightarrow$ *festival*
	drink	$festival \times string$	$\rightarrow$ *festival*

Alternativ zu den Vereinigungsoperationen können auch mehrere Signaturen angegeben werden, z.B.

name:	*person*	$\rightarrow$ *string*
name:	*camp*	$\rightarrow$ *string*

(b)

Die einzige Sorte, die weder vorausgesetzt ist, noch in der Aufgabenstellung definiert ist, ist *stringset*, deren Trägermenge sich wie folgt defnieren lässt.

sets $stringset = \{s | s \subseteq string\}$

(c)

functions

$createPerson(n, r, b, m) = (n, \text{„home“}, r, b, m)$

$createFestival() = \{(\text{„stage“}, \emptyset)\}$

$$insertPerson(f, (n, h, r, b, m), c) = \begin{cases} f & \text{falls } n \in names(f) \vee h \neq \text{„home“} \vee c = \text{„stage“} \\ f \cup \{nc\} & \text{falls } n \notin names(f) \wedge c \neq \text{„stage“} \wedge c \notin cnames(f) \\ f' & \text{otherwise} \end{cases}$$

mit:

$nc = (c, \{(n, c, r, b, m)\})$

$f' = (f \setminus \{c_{old}\}) \cup \{c_{new}\}$

$c_{old} = searchCamp(f, c)$

$c_{new} = add(c_{old}, (n, c, r, b, m))$

$name((n, p)) = n$

$name((n, h, r, b, m)) = n$

$names((n,p)) = \{n' | (n',h,r,b,m) \in p\}$
$names(f) = \bigcup_{c \in f} names(c)$

$cnames(f) = \{n | (n,p) \in f\}$

Sei $nobody$ = („nobody“, „home“, 0, 0, 0)

$$who(n,(nc,p)) = \begin{cases} (n,c,r,b,m) & \text{falls } (n,c,r,b,m) \in p \\ nobody & \text{sonst} \end{cases}$$
$who(n,f) = who(n, position(n,f))$

Sei $nowhere$ = („home“, $\emptyset$)

$$position(f,n) = \begin{cases} c & \text{falls } \exists c \in f : contains(c,n) \\ nowhere & \text{sonst} \end{cases}$$

$$visit(f,n,cn) = \begin{cases} f & \text{falls } name(position(f,n)) = cn \\ & \vee n \notin names(f) \vee cn \notin cnames(f) \\ (f \setminus \{c_1,c_2\}) \cup \{c_3,c_4\} & \text{sonst} \end{cases}$$

mit:
$c_1 = (n_1,p_1) = position(n,f)$, aktueller Aufenthaltsort der Person
$c_2 = (n_2,p_2) = searchCamp(f,cn)$, Zielort des Besuchs
$c_3 = (n_1, p_1 \setminus \{who(n,f)\})$, Ausgangscamp ohne die Person
$c_4 = (n_2, p_2 \cup \{p\})$, Zielcamp mit Person

$$p = \begin{cases} who(n,f) & \text{falls } n_1 \neq getHome(who(n,f)) \\ decBeer(who(n,f)) & \text{sonst} \end{cases}$$

$$drink(f,n) = \begin{cases} f_1 & \text{falls } name(position(f,n)) = \text{„stage“} \\ f_2 & \text{falls } name(position(f,n)) = getHome(who(n,f)) \wedge n \in names(f) \\ f & \text{falls } position(f,n) = nowhere \\ f_3 & \text{sonst} \end{cases}$$

mit:
$f_1 = (f \setminus \{c_{old}\}) \cup \{s_{new}\}$
$c_{old} = (n_o, p_o) = position(f,n)$
$s_{new} = (n_o, (p_o \setminus \{who(n,f)\}) \cup \{decMoney(who(n,f),3)\})$

$f_2 = (f \setminus \{c_{old}\}) \cup \{h_{new}\}$
$h_{new} = (n_o, (p_o \setminus \{who(n,f)\}) \cup \{decBeer(who(n,f))\})$

$f_3 = (f \setminus \{c_{old}\}) \cup \{c_{new}\}$
$c_{new} = (n_o, p_{new})$

$$p_{new} = \begin{cases} p_o & \text{falls } getDonor(c_{old}) = nobody \\ (p_o \setminus \{getDonor(c_{old})\}) \cup \{decBeer(getDonor(c_{old}))\} & \text{sonst} \end{cases}$$

Aufgabe 2

Der initiale Hashwert sowie die Folge der Kollisionen können der folgenden Tabelle entnommen werden:

String	k	k^2	$h_0(k)$	$h_1(k)$	$h_2(k)$	$h_3(k)$	$h_4(k)$	$h_5(k)$
Berlin	25	625	2					
Bochum	20	400	0					
Frankfurt	25	625	2	3				
Hamburg	22	484	8					
Karlsruhe	30	900	0	1				
Dortmund	37	1369	6					
Bregenz	25	625	2	3	6	1	8	7
Wien	37	1369	6	7	0	5		

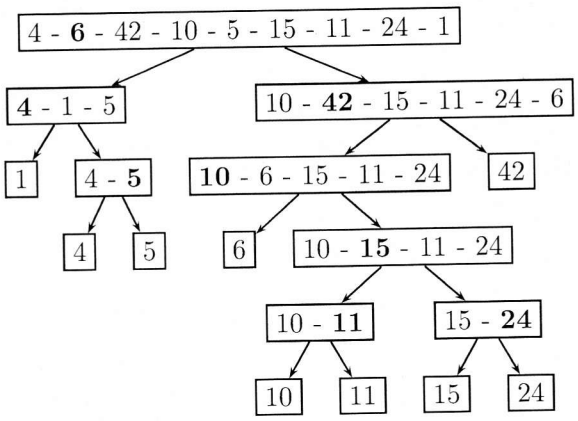
Damit ergibt sich die folgende Hashtabelle:

0	1	2	3	4	5	6	7	8	9
Bochum	Karlsruhe	Berlin	Frankfurt	-	Wien	Dortmund	Bregenz	Hamburg	-

Aufgabe 3 Sortieren

(a)

Der Baum der rekursiven Aufrufe von Quicksort für die in der Aufgabe angegebene Folge sieht wie folgt aus:



(b)

Betrachte die Folge 2 - 6 - 1 - 1. Im ersten Schritt ist 6 das Pivotelement. Im Partition-Schritt wird die letzte 1 mit der 6 vertauscht und dabei vor die erste 1 geschoben.

(c)

Um die Wörter alphabetisch durch Radixsort zu sortieren, gehen wir die Worte von hinten nach vorn durch und verwenden für jeden Buchstaben einen Behälter. Um unterschiedliche Stringlängen behandeln zu können, verwenden wir den Behälter $B_\varnothing$, in den alle Strings einsortiert werden, deren Länge für die aktuell betrachtete Stringposition noch nicht ausreichend ist.

Während der Initialisierung wird nur das Wort Wetter außerhalb von $B_\varnothing$ platziert, da dieses das längste Wort ist und kein weiteres die Länge 6 aufweist. Dementsprechend erhalten wir die Verteilung für die Phase 0. Die folgenden Phasen ergeben sich nun dadurch, indem wir, wie oben beschrieben, das Vorgehen mit der jeweiligen nächsten Stelle von hinten nach vorne iterieren. Aus dem Ergebnis der letzten Phase lässt sich dann das sortierte Ergebnis ablesen.

Initiale Verteilung (Phase 0):

$B_\varnothing$ = Hund, Teil, Ast, Mund, Hai, Wut, Kahn, Fahne, Kai, Wolf, Backe
B_r = Wetter

Phase 1:

$B_\varnothing$ = Hund, Teil, Ast, Mund, Hai, Wut, Kahn, Kai, Wolf
B_e = Fahne, Backe, Wetter

Phase 2:

$B_\varnothing$ = Ast, Hai, Wut, Kai
B_d = Hund, Mund
B_f = Wolf
B_k = Backe
B_l = Teil
B_n = Kahn, Fahne
B_t = Wetter

Phase 3:

B_c = Backe
B_h = Kahn, Fahne
B_i = Hai, Kai, Teil
B_l = Wolf
B_n = Hund, Mund
B_t = Ast, Wut, Wetter

Phase 4:

B_a = Backe, Kahn, Fahne, Hai, Kai
B_e = Teil, Wetter
B_o = Wolf
B_s = Ast
B_u = Hund, Mund, Wut

Phase 5:

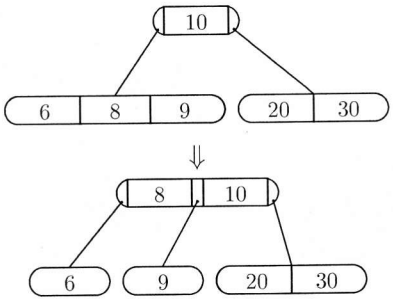
B_a = Ast
B_b = Backe
B_f = Fahne
B_h = Hai, Hund
B_k = Kahn, Kai
B_m = Mund
B_t = Teil
B_w = Wetter, Wolf, Wut

Daraus ergibt sich die sortierte Folge:
Ast, Backe, Fahne, Hai, Hund, Kahn, Kai, Mund, Teil, Wetter, Wolf, Wut

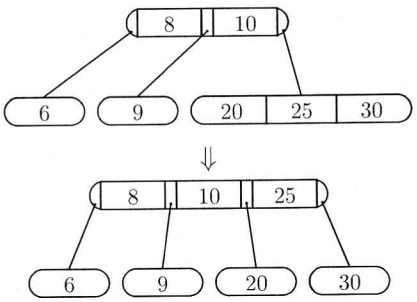
Aufgabe 4 B-Bäume

(a)

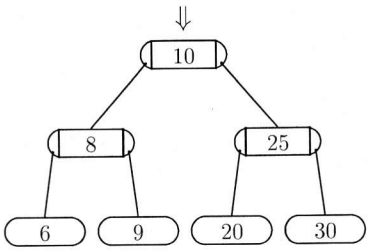
Beim Einfügen des Schlüssels 9 ensteht ein Überlauf und es ist ein Split erforderlich:



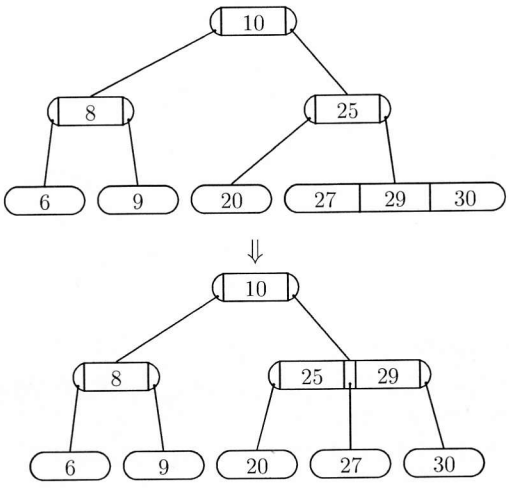
Durch das Einfügen des Elements 25 ensteht wiederum ein Überlauf und es wird eine Split-Operation ausgeführt:



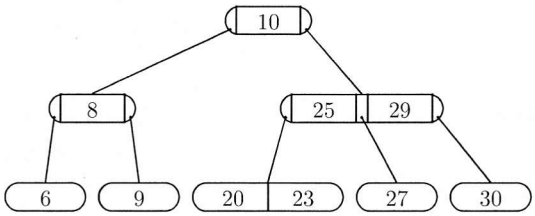
Durch den Split entsteht ein Überlauf in der Wurzel, der beseitigt wird:



Das Einfügen der 27 verursacht keine Strukturänderung. Nach dem Einfügen des Elements 29 kommt es erneut zu einem Überlauf, der durch einen Split-Operation aufgelöst wird.

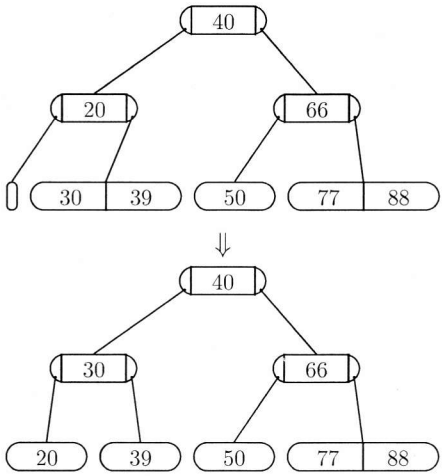


Das Einfügen des Elements 23 birgt keine Probleme. Der endgültige Baum ist:

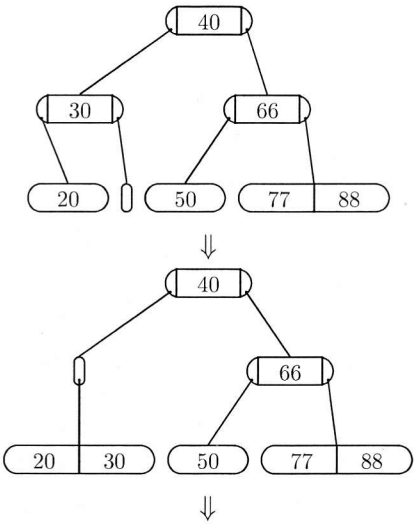


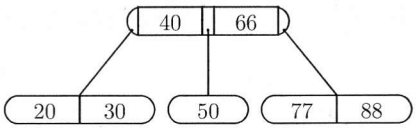
(b)

Beim Löschen der 12 gibt es einen Unterlauf im betroffenen Knoten. Da der rechte Bruder genügend Elemente besitzt, wird dieser Unterlauf durch eine Balance-Operation ausgeglichen:

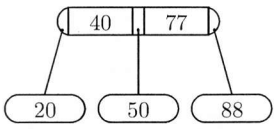


Beim Löschen des Elements 39 tritt ebenfalls ein Unterlauf im betroffenen Knoten ein. Kein Nachbar hat genügend Einträge für eine Balance-Operation. Daher wird der Unterlauf durch ein Merge ausgeglichen. Im Vater des betroffenden Knotens ist nun erneut ein Unterlauf entstanden. Da der einzige Nachbar wieder nicht über genügend Einträge verfügt, muss erneut Merge angewandt werden. Weil die Wurzel infolgedessen keine Schlüssel mehr hat, wird der neu geschaffene Knoten die neue Wurzel des Baumes:





Beim Löschen der 66 aus der Wurzel muss der nächstgrößere Schlüssel im Baum gesucht werden. Er liegt im rechten Blatt und ist die 77. Die 77 wird mit der 66 getauscht und die 66 aus dem Baum entfernt. Das Löschen der 30 ist unproblematisch, sie wird einfach aus dem entsprechenden Blatt entfernt. Es ergibt sich folgender Baum als Endergebnis aller Löschungen:



Aufgabe 5 Deckblatt

Hier erhalten Sie den Punkt, wenn Sie beide Klausurdeckblätter korrekt und vollständig ausgefüllt haben, also Namen, Matrikelnummer und Adresse korrekt eingetragen und genau diejenigen Aufgaben markiert haben, die Sie auch bearbeitet haben.