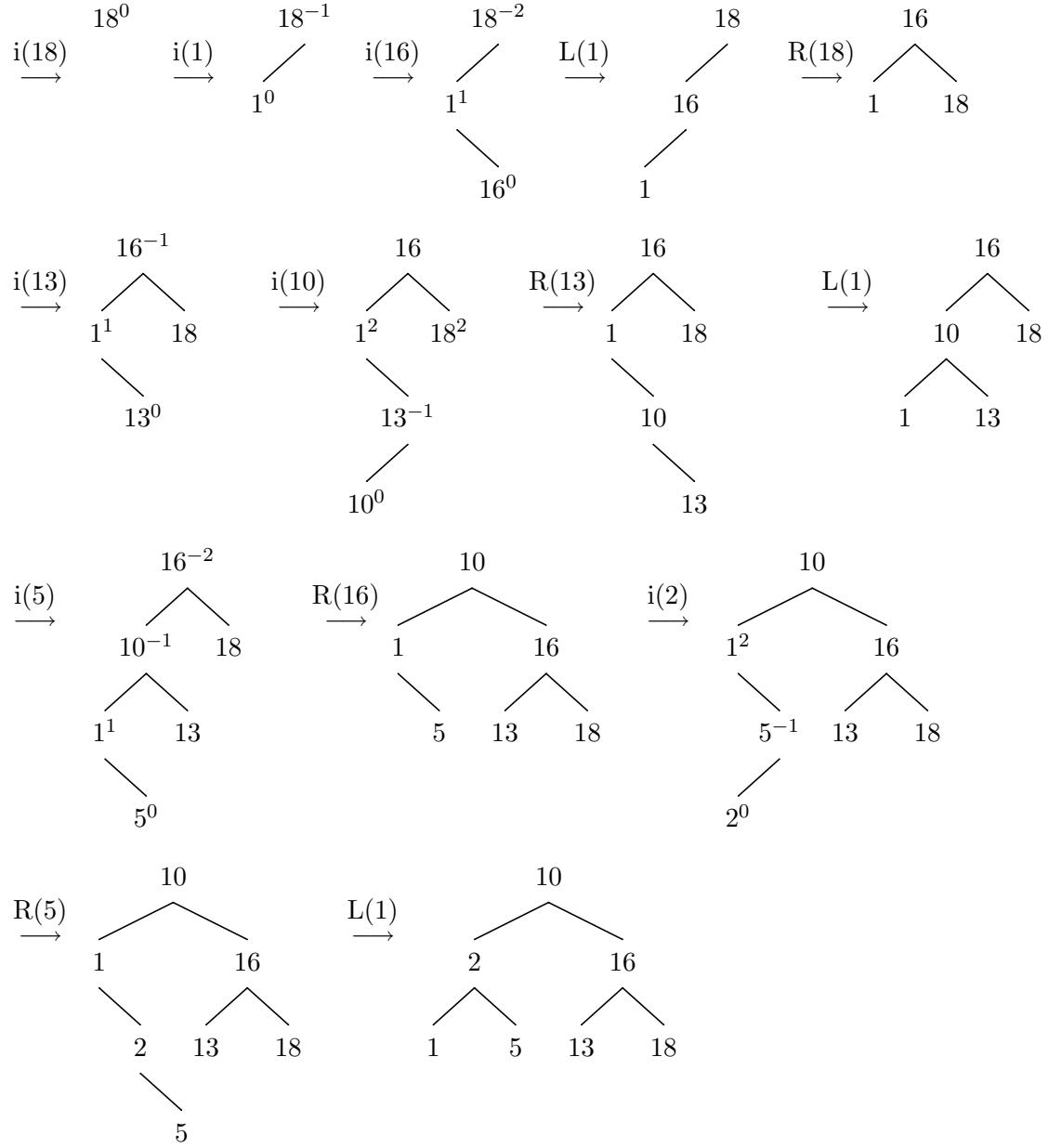
Lösungen Probeklausur „Algorithmen und Datenstrukturen“ vom Januar/Februar 2010

1. Aufgabe (AVL-Baum): (5 Punkte)



2. Aufgabe (Prozedurkonzept - pulsierender Speicher): (2 + 4 = 6 Punkte) AGS 4.16 (geändert)

(a)

e	: 2-36,	a	: 31-36,	g	: 4-36,	f	: 6-36	h	: 15-36,
x, y, z in g	: 20-28,	u in g	: 21-28,	y, z in f	: 6-13,	u in f	: 7-13,		
x in h	: 15-18	u in h	: 16-18	main	: 30-36				

(b)

Haltepunkt	RM	Umgebung								
		1	2	3	4	5	6	7	8	9
label6	-	e	a							
		2	?							
label1	1	e	*z	y	z	u				
		2	?	2	#2	?				
label1	2:1	e				*z	y	z	u	
		2	?	2	#2	?	1	#5	?	
label2	1	e	*z	y	z	u				
		2	?	2	#2	4				
label4	3:1	e	*x				y	z	x	u
		2	?	2	#2	4	1	4	#2	?
label3	1	e	*z	y	z	u				
		2	5	2	#2	4				
label7	-	e	a							
		2	5							

3. Aufgabe (Rek. Funktionen u. dyn. Datenstrukturen in C): (1 + 3 +2 = 6 Punkte)

(a)

```
1  typedef struct ele *Tree;
2  typedef struct ele {
3      int wert;
4      int balance;
5      Tree left;
6      Tree right;
7  } ele_type;
```

(b)

```
1  void balance(Tree tree, int *pfl)
2  {
3      int l, r;
4      if (tree == NULL) {
5          *pfl = 0;
6          return;
7      }
8      balance(tree->left, &l);
9      balance(tree->right, &r);
10     tree->balance = r - l;
11     if (l>r) *pfl = l + 1;
12     else *pfl = r + 1;
13 }
14 /*Aufruf:
15 int dummy;
16 Tree tree;
17 balance(tree, &dummy);
18 */
19 //oder
20 int balance(Tree tree) {
21     int l, r;
22     if (tree == NULL) return 0;
23     l = balance(tree->left);
24     r = balance(tree->right);
25     tree->balance = r - l;
26     if (l>r) return l + 1; // bzw. return (l>r) ? l+1 : r+1;
27     return r + 1;
28 }
29 /*Aufruf:
30 Tree tree;
31 balance(tree);
```

```
32 */
```

(c)

```
 1 typedef struct elel *LPtr;
 2 typedef struct elel {
 3     int wert;
 4     LPtr next;
 5 } elel_type;
 6
 7 // In-Order-Durchlauf
 8 void tree_to_liste(Tree t, LPtr *l)
 9 {
10     if(t==NULL) return;
11     tree_to_liste(t->right, l);
12     append(l, t->wert);
13     tree_to_liste(t->left, l);
14 }
15
16 //Aufruf
17 Tree t;
18 LPtr l;
19 //...
20 tree_to_liste(t,&l);
21
22 //nicht gefordert, nur der Vollständigkeit halber
23 void append(LPtr *list, int n)
24 {
25     LPtr neu;
26     if(*list == NULL) {
27         neu = (LPtr) malloc(sizeof(elel_type));
28         neu->wert = wert;
29         neu->next = NULL;
30         *list = neu;
31         return;
32     }
33     append(&((*list)->next), wert);
34 }
```

4. Aufgabe (Rekursion und Iteration): (3 + 2 = 5 Punkte)

(a)

$f(n) = iter(n, 3, 2, 2, 1, 1),\ n \geq 0$
$iter(n, a, b, c, d, e) = iter(n-1, e+1+b-2(3c+d-2b), a, 3c-2b+d, c, e+1),\ n \geq 1$
$iter(0, a, b, c, d, e) = b$

(b)

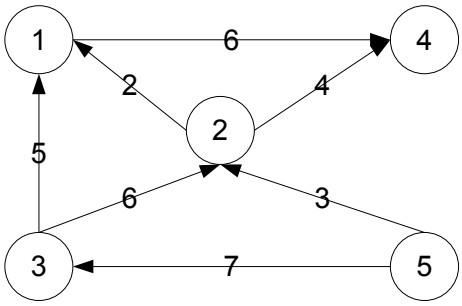
$g(1) = 1$
$g(2) = 2$

$g(n+1) = (g(n-1)+1) * (g(n)-1),\ n \geq 2$

5. Aufgabe (Algebraisches Pfadproblem): (2 + 3 + 2 = 7 Punkte)

(a)

Es handelt sich um das Kapazitätsproblem mit dem Semiring $SR = (\mathbb{N}_\infty, max, min, 0, \infty)$.



(b)

$$D_G^{(0)} = \begin{pmatrix} \infty & 0 & 0 & 6 & 0 \\ 2 & \infty & 0 & 4 & 0 \\ 5 & 6 & \infty & 0 & 0 \\ 0 & 0 & 0 & \infty & 0 \\ 0 & 3 & 7 & 0 & \infty \end{pmatrix}$$

$D_G^{(1)} : (3, 4, 5)$

$D_G^{(2)} : (5, 1, 2), (5, 4, 3)$

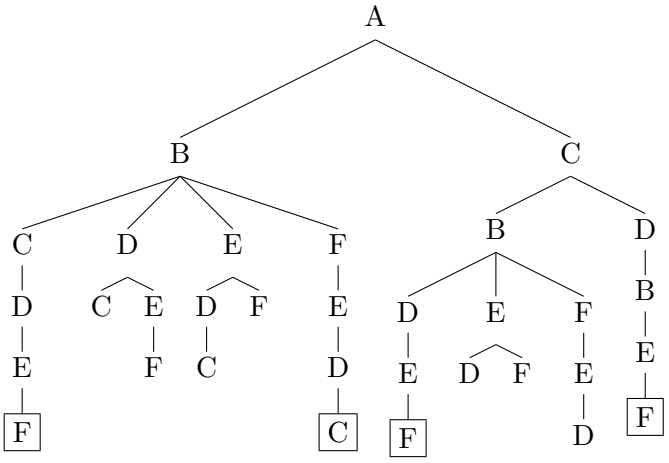
$D_G^{(3)} : (5, 2, 6), (5, 1, 5), (5, 4, 5)$

$D_G = D_G^{(4)} = D_G^{(3)}$, da Knoten 4 eine Senke und Knoten 5 eine Quelle ist.

$$D_G = \begin{pmatrix} \infty & 0 & 0 & 6 & 0 \\ 2 & \infty & 0 & 4 & 0 \\ 5 & 6 & \infty & 5 & 0 \\ 0 & 0 & 0 & \infty & 0 \\ 5 & 6 & 7 & 5 & \infty \end{pmatrix}$$

(c) Es verändern sich von Insel 5 aus die maximalen Fahrzeuglasten: $(5, 1, 2)$ und $(5, 4, 4)$. Fahrzeuge mit 5 Tonnen Gewicht auf dem Weg von 5 nach 4 bleiben, wenn sie während des Einsturzes auf Insel 3 sind, in der Inselkette stecken, da sie weder zu Insel 5 weiterkommen, noch zu Insel 4 zurück.

6. Aufgabe (Backtracking): (5 Punkte)



7. Aufgabe (Fixpunktsemantik): (2 + 2 + 2 = 6 Punkte)

(a) $\begin{pmatrix} W(\mathcal{E}, S) \\ W(\mathcal{E}, A) \end{pmatrix} \rightsquigarrow \begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix} \overset{f}{\mapsto} \begin{pmatrix} \emptyset \\ \{\varepsilon\} \end{pmatrix} \overset{f}{\mapsto} \begin{pmatrix} \{\varepsilon\} \\ \{\varepsilon, c\} \end{pmatrix} \overset{f}{\mapsto} \begin{pmatrix} \{\varepsilon, c, ab\} \\ \{\varepsilon, c, cc\} \end{pmatrix}$
$\overset{f}{\mapsto} \begin{pmatrix} \{\varepsilon, c, cc, ab, acb, aabb\} \\ \{\varepsilon, c, cc, ccc\} \end{pmatrix}$
$\overset{f}{\mapsto} \begin{pmatrix} \{\varepsilon, c, cc, ccc, ab, acb, accb, aabb, aacbb, aaabbb\} \\ \{\varepsilon, c, cc, ccc, cccc\} \end{pmatrix}$

(b) $f^i(\bot) = f^i \begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix} = \left(\begin{array}{c|c} \{a^k c^l b^k & 0 \le k + l \le i - 2\} \\ \{c^k & 0 \le k \le i - 1\} \end{array} \right) \ (i \ge 1)$

(c) für $i \to \infty$:

$W(\mathcal{E}, S) = \{a^k c^l b^k | k, l \ge 0\}$

$W(\mathcal{E}, A) = \{c^n | n \ge 0\}$