Universität Innsbruck
Zweite Prüfung

Institut für Informatik
16. Oktober 2008

Algorithmen und Datenstrukturen

Name:	
Matrikelnr:	

Die Prüfung besteht aus 8 Aufgaben. Die verfügbaren Punkte für jede Aufgabe sind am Rand angegeben. Um die Prüfung erfolgreich abzuschließen sind 40 von 80 Punkten ausreichend.

Aufgabe	Erreichte Punkte
1	8
2	12
3	8
4	15
5	16
6	9
7	8
8	4
Summe	80

1 Auswertung von Ausdrücken in C

Bestimmen Sie für die folgenden Codefragmente, ob das Ergebnis der Ausgabe compilerabhängig ist. Geben Sie die möglichen Ausgaben an.

[4] (a)

```
int z = 4;
printf("%i + 1 = %i\n", z, ++z);
```

Lösung.
Compilerabhängig: ☒ ja ☐ nein
Mögliche Ausgaben:
$4 + 1 = 5$ oder $5 + 1 = 5$

[4] (b)

```
int c = 0;
do {
  int c = 0;
  ++c;
  printf("c = %d ", c );
} while( c++ < 1 );
```

Lösung.
Compilerabhängig: ☐ ja ☒ nein
Mögliche Ausgaben:
$c = 1 \quad c = 1$

2 **Datentypen in C** Gegeben sind die folgenden Beschreibungen von Datentypen. Definieren Sie jeweils eine Datenstruktur in C, mit der dieser Datentyp repräsentiert werden kann.

[4] (a) Geben Sie eine Datenstruktur für eine verkettete Liste an, die zu speichernden Daten sind Ganzzahlen.

Lösung.

```
struct linked_list_struct {
  int data;
  struct linked_list_struct* next;
};
```

[4] (b) Finden Sie eine Datenstruktur für einen AVL-Baum. Der Schlüssel soll von einer Ganzzahl repräsentiert werden, die zu speichernden Daten sind Zeichenketten.

Lösung.

```
struct avl_tree_struct {
int balance;
int key;
char* data;
struct avl_tree_struct* left;
struct avl_tree_struct* right;
};
```

[4] (c) Definieren Sie eine Datenstruktur für eine Hashtabelle, die 10 Ganzzahlen mittels offener Adressierung speichern kann.

Lösung.
int hashTable[10];

3 **O Notation**

Bestimmen Sie für die folgenden Aussagen, ob sie wahr oder falsch sind! Kreuzen Sie T für wahr, F für falsch! Für jede richtige Antwort erhalten Sie einen Punkt, für jede falsche Antwort wird ein Punkt abgezogen. Die Aufgabe wird jedoch nicht insgesamt negativ bewertet.

[1] (a) $f(n) \in \Omega(g(n)) \iff g(n) \in O(f(n))$ T F

[1] (b) $f(n) \in \Theta(g(n)) \iff f(n) \in O(g(n))$ und $f(n) \in \Omega(g(n))$ T F

[1] (c) $f(n) \in \Theta(g(n)) \iff$ es gibt $c_1, c_2 > 0$ und N sodass für alle $n > N$ gilt: $c_1 g(n) \leq f(n) \leq c_2 g(n)$ T F

[1] (d) $(\log(n))^{10} \in \Omega(n)$ T F

[1] (e) $\sum_{i=0}^{n} a_i x^i,\ a_n \neq 0$ is in $\Theta(x^n)$ T F

[1] (f) $n \log(n) \in \Omega(n^2)$ T F

[1] (g) $(\log(n))^{\log(n)} \in O(\frac{n}{\log(n)})$ T F

[1] (h) $n! \in \Theta(n^n)$ T F

Lösung.

(a) true

(b) true

(c) true

(d) false

(e) true

(f) false

(g) false

(h) false

4 **Entwurf eines Suchalgorithmus**

[15] Gegeben sei ein Feld $x[1, \ldots, n]$ von verschiedenen, aufsteigend sortierten *int*-Werten und ein *int*-Wert z. Geben Sie Pseudocode für einen nicht-rekursiven Algorithmus an, der einen Index i bestimmt sodass $z = x[i]$, falls ein derartiger Index existiert. Die Laufzeit des Algorithmus soll in $O(\log n)$ liegen.

Lösung.

Algorithmus 1 Binäre Suche

Eingabe: Feld von ganzen Zahlen $x[1 \ldots n]$ und eine Zahl z
Ausgabe: Index i sodass $x[i] = z$ falls ein derartiger Index existiert, ansonsten $i = -1$

```
begin
  lower := 1;
  upper := n;
  i := 1;
  while lower ≤ upper do
    mid := lower + ⌊(upper − lower)/2⌋;
    if x[mid] > z then
      upper := mid − 1;
    else if x[mid] < z then
      lower := mid + 1;
    else
      i := mid;
      break;
  if x[i] ≠ z then
    i := −1;
end
```

5 Binäre Suchbäume

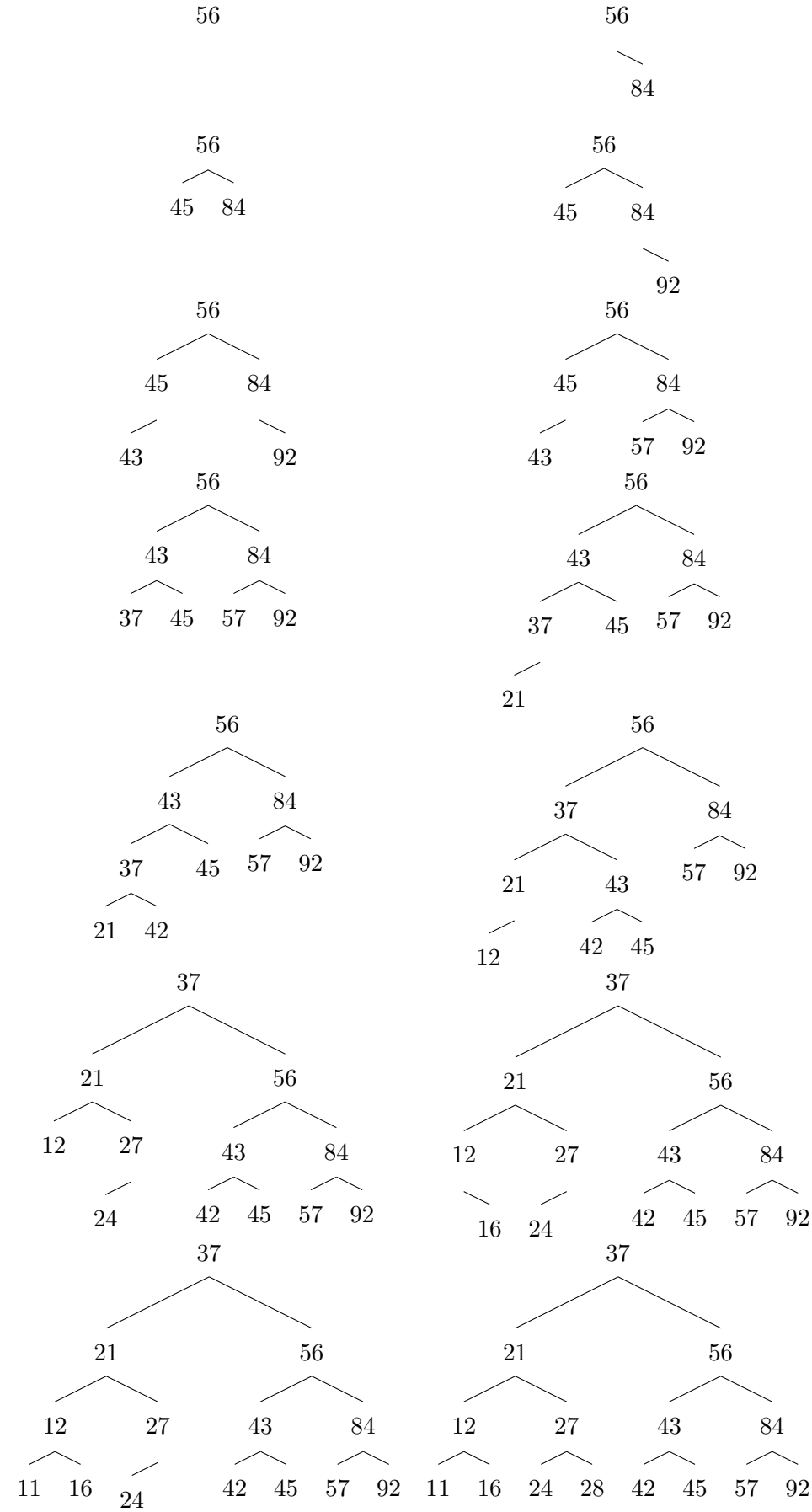
[16]

Erzeugen Sie einen AVL-Baum, in den Sie die folgenden Werte in der gegebenen Reihenfolge einfügen:

$$[56, 84, 45, 92, 43, 57, 37, 21, 42, 12, 27, 24, 16, 11, 28].$$

Geben Sie bei jeder Rotation den kritischen Knoten an.

Lösung.


56
56
45 84
56
45 84
43 92
56
43 84
37 45 57 92
56
43 84
37 45 57 92
21 42
37
21 56
12 27 43 84
24 42 45 57 92
37
21 56
12 27 43 84
11 16 24 42 45 57 92
56
84
56
45 84
92
56
45 84
43 57 92
56
43 84
37 45 57 92
21
56
37 84
21 43 57 92
12 42 45
37
21 56
12 27 43 84
16 24 42 45 57 92
37
21 56
12 27 43 84
11 16 24 28 42 45 57 92

6 **Hashing mit offener Adressierung** Gegeben sind die folgenden Werte:

$$[4, 24, 28, 60, 71, 21, 68, 73, 16].$$

[9]

Fügen Sie die oben angegebenen Werte in eine anfangs leere Hashtabelle der Größe 13 in der gegebenen Reihenfolge ein. Verwenden Sie dafür doppeltes Hashing mit der Hashfunktion $h(k, j) = ((h_1(k) + j * h_2(k)) \mod 13)$, wobei $h_1(k) = k \mod 13$ und $h_2(k) = k \mod 12$. Fügen Sie in jeder Zeile der mittleren Tabelle den angegebenen Schlüssel an der korrekten Position ein und tragen Sie die Anzahl der dabei auftretenden Kollisionen sowie deren Positionen in der rechten Tabelle ein.

Lösung.

K	0	1	2	3	4	5	6	7	8	9	10	11	12	Koll.	Pos.
4					4									0	
24					4							24		0	
28			28		4							24		0	
60			28		4				60			24		0	
71			28		4		71		60			24		0	
21			28		4	21	71		60			24		1	8
68			28	68	4	21	71		60			24		0	
73			28	68	4	21	71		60		73	24		1	8
16	16		28	68	4	21	71		60		73	24		2	3,8

7 Analyse von Algorithmen

[5] (a) Betrachten Sie folgendes Code-Fragment und stellen Sie die exakte Rekursionsgleichung für die Anzahl $T(n)$ der ausgeführten Multiplikationen in Abhängigkeit vom Eingabeparameter n auf. Nehmen Sie dazu an, dass n eine Zweierpotenz ist, d.h., $n = 2^k$ für $k \in \mathbb{N}$.

[3] (b) Geben Sie ausgehend von dieser Rekursionsgleichung eine asymptotische Schranke für die Anzahl der Multiplikationen an.

Algorithmus 2 *Power*

Input: Zahlen x und $n \in \mathbb{N}$
Output: x^n

```
begin
if n = 0 then
  return 1;
if n = 1 then
  return x;
temp := Power(x, n/2);
if n mod 2 = 0 then
  return temp * temp;
else
  return temp * temp * x;
end
```

Lösung.
Falls n eine Zweierpotenz ist, gilt:

$$T(n) = T(\frac{n}{2}) + 1$$

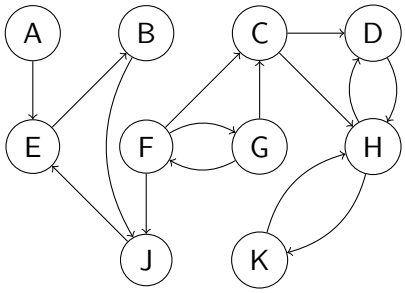
mit Anfangsbedingungen $T(0) = T(1) = 0$. Diese Rekursion ist einfach zu lösen, und die explizite Lösung ist:

$$T(n) = \log_2(n)$$

Also ist $T(n) \in \Theta(\log(n))$, was man auch einfach mit Hilfe des Master Theorems ermitteln hätte können.

8 **Graphen**

[4] Eine *starke Zusammenhangskomponente* G' eines gerichteten Graphen G ist ein größter Teilgraph von G, so dass G' stark zusammenhängend ist. Bestimmen Sie alle starken Zusammenhangskomponenten des gegebenen Graphen.



Lösung.
Es gibt 5 starke Zusammenhangskomponenten, die sich jeweils durch Projektion der Kanten von G auf die Knotenmengen $\{\mathsf{A}\}$, $\{\mathsf{B}, \mathsf{E}, \mathsf{J}\}$, $\{\mathsf{C}\}$, $\{\mathsf{F}, \mathsf{G}\}$ und $\{\mathsf{D}, \mathsf{H}, \mathsf{K}\}$ ergeben.