

Klausur Grundgebiete der Informatik 1

WS 08/09

(BPO07/DPO04)

Datum: 19.03.2009

Keine offizielle Lösung!

Name: <i>(in Druckschrift)</i>
Matr.Nr.:
Unterschrift:

Keinerlei Gewähr auf Richtigkeit

Aufgabe	Max. Punkte	Korrektur		Einsicht	
		Punkte	Kürzel	Punkte	Kürzel
1	12				
2	10				
3	4				
4	18				
5	16				
6	6				
7	18				
8	16				
Σ	100				

Aufgabe 1 – Wissensfragen

Bitte beachten Sie, dass bei den folgenden Teilaufgaben - sofern in der Aufgabenstellung nicht anders verlangt - eine oder mehrere Antworten richtig sein können. Kreuzen Sie bei den folgenden Teilaufgaben jeweils die richtige(n) Aussage(n) an. Das Ankreuzen falscher Antworten führt nicht zu Punktabzügen. Für eine Teilaufgabe wird volle Punktzahl gegeben, wenn alle Kreuze richtig gesetzt sind. Fehlende Kreuze oder falsch gesetzte Kreuze führen dazu, dass die entsprechende Teilaufgabe mit 0 Punkten bewertet wird.

- (a) Gegeben ist die C-Deklaration `unsigned char x`; Welches ist der maximale Wert, der mit der Anweisung `printf("%u\n", x)`; ausgegeben werden kann?
- 127
 - 128
 - 255
 - 256
- (b) Ein Array A ist wie folgt definiert: `int A[10]`;
Was bewirkt dann die folgende Zuweisung: `A[10]=4`;
- Es wird der Speicher hinter dem Array überschrieben.
 - Der Wert wird nach `A[0]` geschrieben (zyklische Adressierung).
 - Das Programm erweitert das Array um ein weiteres Element.
- (c) Welche Laufzeitkomplexität besitzt der Heapsort-Algorithmus im schlechtesten Fall?
- $O(n^2)$
 - $O(n \log n)$
 - $O(n^3)$
- (d) Markieren Sie alle Aussagen, die auf den Heapsort-Algorithmus zutreffen, den Sie in der Vorlesung kennengelernt haben.
- Die sehr gute Zeitkomplexität des Heapsort im worst case rührt daher, dass unabhängig von der Größe des zu sortierenden Feldes genau vier Minimumsbäume aufgebaut werden, die dann in linearer Zeit zum sortierten Datenfeld vereinigt werden.
 - Die Heapsort-Hilfsfunktion `sink` wird sowohl beim initialen Baumaufbau als auch beim eigentlichen Sortieren verwendet.
 - Bei einem Maximums-Heap muss bezüglich eines Knotens `A[i]` gelten: linker Sohn (`A[2i]`) \leq rechter Sohn (`A[2i+1]`).

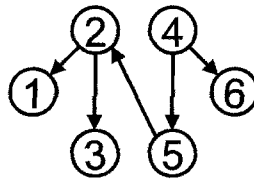
(e) Gegeben ist der C-Code

```
unsigned int fakultaet (unsigned int x)
{
  ????
  return x * fakultaet(x - 1);
}
```

Markieren Sie alle Zeilen, die anstelle der ???? eingesetzt werden können, um den korrekten Funktionswert der Fakultätsfunktion zu berechnen.

- if (x == 0) return 0;
- return 1;
- if (x >= 2) return 2;
- if (x == 0) return 1;
- if (x <= 1) return 1;

(f) Gegeben ist der Graph aus folgender Abbildung. Welche der folgenden Aussagen sind korrekt?



- Die Durchführung einer Depth-First Search beginnend bei Knoten 4 kann folgende Knotenfolge ergeben: 4, 5, 2, 1, 3, 6.
- Die Durchführung einer Breadth-First Search beginnend bei Knoten 4 kann folgende Knotenfolge ergeben: 4, 5, 2, 1, 6, 3.
- Der Graph wird durch folgende Adjazenz-Listen repräsentiert:

1:						
2:	1	3				
3:						
4:	5	6				
5:	2					
6:						

Aufgabe 2 – Fehlersuche

Die unten angegebene Funktion soll die Folge der Fibonacci-Zahlen für $i = 0, \dots, n$ ausgeben. Die Fibonacci-Zahlen sind durch folgende Funktion definiert:

$$Fib(n) = \begin{cases} Fib(n-1) + Fib(n-2), & \text{für } n \geq 2 \\ 1, & \text{für } n = 1 \\ 0, & \text{für } n = 0 \end{cases}$$

Wenn die Fibonacci-Zahl zu groß wird, soll die Funktion eine Fehlermeldung ausgeben und sofort abbrechen. Leider enthält der C-Code einige Fehler. Finden Sie sämtliche Fehler und geben Sie in der rechten Spalte die korrigierte Version an. Enthält eine Zeile keinen Fehler, so lassen Sie die rechte Spalte frei.

Beispiel:

integer i;	int i;
------------	--------

Voraussetzung: Gehen Sie beim Datentyp int von einer Bitbreite von 4 Byte (32 Bit) aus. Hinweis: $Fib(48) = 4.807.526.976$, $2^{32} = 4.294.967.296$

#define <stdio.h>	#include <stdio.h>
#define MAX_N 47	
#define SUM(a;b) (a+b)	#define SUM(a,b) (a+b)
void fibonacci(int n)	
{	
int i;	
unsigned int fib_n[2] = {0,1};	
const unsigned int fib;	unsigned int fib;
for(i=0; i<=n; i++)	
{	
printf("%u, ",fib_n[0]);	
fib=SUM(fib_n[0],fib_n[1]);	
fib_n[0]=fib_n[1];	
fib_n[1]=fib;	
if (i = MAX_N)	if (i == MAX_N)
{	
printf("F%d too large\n",i+1);	
continue;	break;
}	
}	
}	

Aufgabe 3 – Call-by-value / Call-by-reference

Gegeben ist der folgende C-Code:

```
1  #include <stdio.h>
2
3  void f1(int *, int);
4  void f2(int *, int);
5
6  int main() {
7      int a=3, b=5;
8
9      f1(&a, b);
10     printf("%d\t %d\n", a, b);
11
12     a = a*3;
13     b = b+7;
14     f2(&a, b);
15     printf("%d\t %d\n", a, b);
16
17     return 0;
18 }
19
20 void f1 (int *p, int q) {
21     int tmp = *p;
22     *p = q;
23     q = tmp;
24 }
25
26 void f2 (int *q, int p) {
27     int tmp = p;
28     p = *q;
29     *q = tmp;
30 }
```

Welche Ausgabe erzeugt das Programm?

In Zeile 10: 5 5

In Zeile 15: 12 12

Aufgabe 4 – Arrays und Strukturen

Implementieren Sie eine Funktion `void mirrorMatrix (int m[N] [N])` zur Spiegelung einer 2-dimensionalen Matrix der Größe $N \times N$ mit Daten vom Typ `integer` an ihrer Nebendiagonalen. Die Nebendiagonale einer Matrix verläuft von rechts oben nach links unten. Gehen Sie davon aus, dass N als Präprozessorkonstante definiert wurde.

Beispiel: $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ wird zu $\begin{pmatrix} 9 & 6 & 3 \\ 8 & 5 & 2 \\ 7 & 4 & 1 \end{pmatrix}$

```
void mirrorMatrix (int m[N] [N])
```

```
{
```

```
    int i, j, temp;
```

```
    for (i=0; i < N; i++) {
```

```
        for (j=0; j < N-1; j++) {
```

```
            temp = m[i][j];
```

```
            m[i][j] = m[N-1-j][N-1-i];
```

```
            m[N-1-j][N-1-i] = temp;
```

```
        }
```

```
    }
```

```
}
```

~~1/1~~

Aufgabe 5 – O-Notation

(a) Berechnen Sie mit Hilfe der folgenden Rechenregeln der O-Notation

$$(1) O(c) = O(1)$$

$$(2) O(c * f(n)) = O(f(n))$$

$$(3) O(f(n)) + O(f(n)) = O(f(n))$$

$$(4) O(O(f(n))) = O(f(n))$$

$$(5) g(n) = a_k * n^k + a_{k-1} * n^{k-1} + \dots + a_0 \Rightarrow O(g(n)) = O(n^k)$$

$$(6) O(f(n)) * O(g(n)) = O(f(n) * g(n))$$

$$(7) O(f(n)) + O(g(n)) = O(f(n) + g(n)) = O(\max\{f(n), g(n)\})$$

das Laufzeitverhalten der folgenden Funktionen. Geben Sie dabei für jeden Schritt die verwendete Rechenregel an.

$$(i) (n + 2)(n - 1) = n^2 + n - 2$$

$$O(n^2 + n - 2) \stackrel{(5)}{=} O(n^2)$$

$$(ii) 3n + 2 \log n$$

$$O(3n + 2 \log n) \stackrel{(7)}{=} O(3n) \stackrel{(2)}{=} O(n)$$

$$(iii) 3n^2 + 6n$$

$$O(3n^2 + 6n) \stackrel{(5)}{=} O(n^2)$$

- (b) Gegeben ist der folgende Algorithmus für ein Array x von n integer-Zahlen. Gesucht ist ein Array a von n Zahlen, so dass $a[i]$ das arithmetische Mittel der Zahlen $x[0], \dots, x[i]$ ist. Gehen Sie davon aus, dass N als Präprozessorkonstante definiert wurde.

```
void mittelwert(int a[N], int x[N]) {
    int i, j, s;
    for (i=0; i<N; i++) {
        s = 0;
        for (j=0; j<=i; j++) {
            s += x[j];
        }
        a[i] = s/(i+1);
    }
}
```

- (i) Bestimmen Sie die Laufzeit des Algorithmus in Form der O-Notation. Begründen Sie Ihre Antwort.

äußere Schleife N durchläuft, innere im Mittel $\frac{N}{2}$

$$\Rightarrow O(N \cdot \frac{N}{2}) = O(N^2)$$

- (ii) Das Problem kann auch effizienter gelöst werden. Geben Sie den Algorithmus für die effizientere Variante als C-Code an. Geben Sie zu Ihrem C-Code die passende Laufzeit in O-Notation an. Eine Begründung ist nicht erforderlich.

```
void mittelwert mittelwert(int a[N], int x[N]) {
    int i, s;
    s = 0;
    for (i=0; i<N; i++) {
        s += x[i];
        a[i] = s / (i+1);
    }
}
```

Laufzeit: $O(n)$

Aufgabe 6 – Sortieralgorithmen

- (a) Welche Zeitkomplexität besitzt der Insertion Sort-Algorithmus, den Sie in der Vorlesung kennengelernt haben, im worst case? Begründen Sie Ihre Antwort und geben Sie ein Beispiel für den worst case.

$O(n^2)$, da für jedes Element das vollständige sortierte Array verschoben werden muss
 Bsp: $\{5, 4, 3, 2, 1\}$

- (b) Wenden Sie den Insertion Sort-Algorithmus, den Sie in der Vorlesung kennengelernt haben, auf das gegebene Array an. Sortieren Sie aufsteigend (von links nach rechts) nach dem Wert der Zahlen. Stellen Sie alle Veränderungen im Array in einer neuen Tabellenzeile dar, d.h. immer wenn sich ein Wert im Array ändert, muss eine neue Tabellenzeile notiert werden. Interne Zustandsvariablen des Algorithmus müssen nicht dargestellt werden.

9	3	5	8	7
9	9	5	8	7
3	9	5	8	7
3	9	9	8	7
3	5	9	8	7
3	5	9	9	7
3	5	8	9	7
3	5	8	9	9
3	5	8	8	9
3	5	7	8	9

Aufgabe 7 – Bäume

Die folgende Struktur definiert den Datentyp für einen Knoten eines binären Suchbaums für Ganzzahlen:

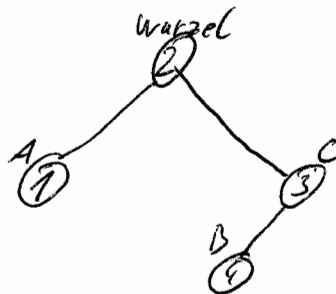
```
typedef struct knoten {
    int zahl;
    struct knoten *pLinks;
    struct knoten *pRechts;
} Knoten;
```

Der Suchbaum ist dabei entsprechend der Vorlesung so aufgebaut, dass für einen Knoten i gilt:

1. sein linker Teilbaum enthält nur kleinere Zahlen und
 2. sein rechter Teilbaum enthält nur größere Zahlen im Vergleich zum Knoten i selbst.
- Außerdem kann eine Zahl nicht mehrfach im Baum vorkommen. Existiert ein Teilbaum unter einem Knoten nicht, so ist der entsprechende Zeiger NULL.

- (a) Skizzieren Sie den binären Baum, der durch folgenden Code im Speicher angelegt wird. Beschriften Sie jeden Knoten mit seinem Variablenamen und der in ihm gespeicherten Zahl.

```
Knoten wurzel, A, B, C;
A.zahl = 1; A.pLinks = NULL; A.pRechts = NULL;
B.zahl = 4; B.pLinks = NULL; B.pRechts = NULL;
C.zahl = 3; C.pLinks = &B; C.pRechts = NULL;
wurzel.zahl = 2; wurzel.pLinks = &A; wurzel.pRechts = &C;
```



- (b) Begründen Sie, warum es sich bei dem Baum aus Teil (a) nicht um einen binären Suchbaum handelt.

Der Wert von B ist größer als der Wert von C, jedoch links von ihm angeordnet.

- (c) Der Baum aus Teil (a) kann durch *Umhängen* eines Blattes in einen binären Suchbaum überführt werden. Bezeichnen Sie die Knoten in ihrer Lösung mit dem Variablenamen, nicht mit den darin enthaltenen Zahlen.
- (i) Geben Sie das Blatt an, welches an einer anderen Stelle im Baum platziert werden muss.

Blatt B ~~links~~ der rechte Sohn von C werden

- (ii) Nennen Sie die Stelle, an der das Blatt platziert werden muss (z.B. als linker Sohn von Knoten A).

rechter Sohn von Knoten C

- (d) Geben Sie eine kurze(!) Code-Sequenz an, die die Zeiger im Baum so verändert, dass die im Aufgabenteil (c) angesprochene Re-Platzierung des Blattes im Baum durchgeführt wird.

$C.pLinks = NULL;$

$C.pRechts = \& B;$

- (e) Die Funktion `int suchen(Knoten *pWurzel, int zahl)`; ist gegeben (d.h. kann benutzt werden und muss hier nicht implementiert werden). Diese Funktion sucht eine Zahl im gegebenen Suchbaum und gibt 1 zurück, falls die Zahl gefunden wurde und 0, falls die Zahl nicht gefunden wurde.

Außerdem ist die Funktion `void einfuegen(Knoten *pWurzel, int zahl)`; gegeben, die eine Zahl an der richtigen Stelle in den Baum einfügt. Diese Funktion darf jedoch nicht mit einer Zahl aufgerufen werden, die im Baum bereits enthalten ist, da die Funktion in diesem Fall nicht korrekt funktioniert.

Vervollständigen Sie die Funktion `void zusammen(Knoten *pZiel, Knoten *pQuelle)`, die zwei Bäume zusammenführt. Diese Funktion soll alle Knoten aus dem Baum `pQuelle` in den Baum `pZiel` einfügen und dabei den Baum `pQuelle` nicht verändern. Zur Realisierung müssen die beiden gegebenen Funktionen `suchen` und `einfuegen` benutzt werden (d.h. es sollen keine Knoten von der Funktion `zusammen` selbst angelegt werden).

```

void zusammen(Knoten *pZiel, Knoten *pQuelle)
{
    if (pQuelle != NULL) {
        if (!suchen(pZiel, pQuelle->Zahl))
            einfuegen(pZiel, pQuelle->Zahl);
        zusammen(pZiel, pQuelle->pLinks);
        zusammen(pZiel, pQuelle->pRechts);
    }
}

```

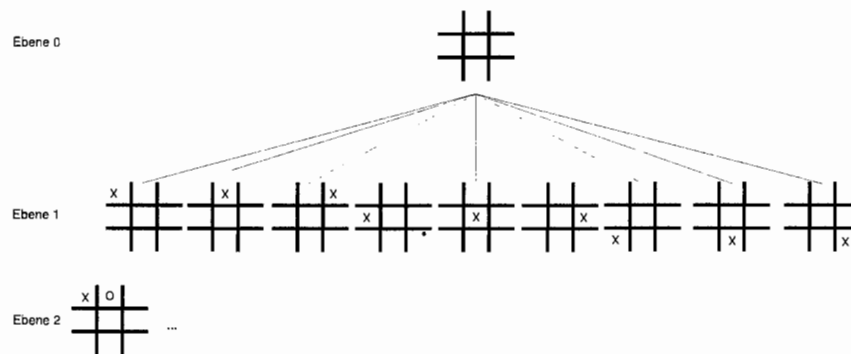


Aufgabe 8 – Techniken zum Algorithmenentwurf

Untersuchen Sie die Eigenschaften des Brettspiels Tic-Tac-Toe. Es gelten die folgenden Regeln:

- Zwei Spieler ziehen abwechselnd. In jedem Zug darf ein Spieler genau einen Stein setzen. In den folgenden Darstellungen verwendet der Spieler 1 das Symbol *X* und der Spieler 2 das Symbol *O*.
- Jedes noch freie Feld darf besetzt werden.
- Gewonnen hat, wer als erster eine horizontale, vertikale oder diagonale Reihe von drei gleichen Steinen gesetzt hat.

Im folgenden sehen Sie die Ebenen 0 und 1 des Spielbaumes für das Spiel.



Das Spiel kann von einem Computer perfekt gespielt werden, indem der gesamte Baum bis zur untersten Ebene aufgebaut wird.

(a) Wieviele Nachfolger haben die Knoten der Ebene 1?

jeder Knoten 8, somit bei allen 9 Knoten der Ebene 1 zusammen 72

(b) Zeichnen Sie alle Nachfolger des ersten Knotens der Ebene 1.



(c) Die Höhe eines Baumes ist die Länge des längsten Pfades von der Wurzel des Baumes zu einem Blatt. Gezählt werden hierbei die Kanten auf dem längstem Pfad. Welches ist die maximale Höhe des Baumes?

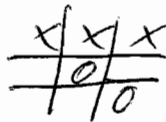
9

- (d) Viele der Knoten im Baum können bei der Suche nach optimalen Zugfolgen ignoriert werden.
- (i) Beim Betrachten der Ebene 1 sehen Sie, dass viele der Knoten durch Rotation oder Spiegelung in einen der anderen Knoten überführt werden können und somit eine Menge äquivalenter Knoten bilden. (Beispiel: durch Rotation des 1. Knotens der Ebene 1 lässt sich dieser u.a. in den 3. Knoten dieser Ebene überführen). Zeichnen Sie eine neue Ebene 1, die nur noch jeweils einen Repräsentanten aus jeder Menge äquivalenter Knoten enthält.



- (ii) Es existieren im gesamten Baum Knoten, deren Kinder für die Suche nach optimalen Zugfolgen nicht betrachtet werden müssen. Nennen Sie **eine** mögliche Bedingung dafür, dass die Kinder eines Knotens abgeschnitten werden können und geben Sie dazu ein Beispiel an.

Es ist bereits eine ~~vollst~~ Reihe vollständig mit Steinen eines Spielers gefüllt:



- (e) Geben Sie ein C-Struct an, das als Datenstruktur für einen Knoten des obigen Baumes geeignet ist. Gehen Sie dabei folgendermaßen vor:
- (i) Erstellen sie einen `enum` Datentyp, der den Zustand eines Feldes (frei, besetzt von Spieler 1, besetzt von Spieler 2) repräsentiert.

```
typedef enum feldzustand { frei, Spieler1, Spieler2 } feldzustand;
```

- (ii) Erstellen Sie nun ein `struct`, welches ein zweidimensionales Array des eben erstellten Typs enthält. Falls Sie den Aufgabenteil (i) nicht bearbeitet haben, erstellen sie ein zweidimensionales Array von chars.

```
typedef struct feld {
    feldzustand zustand[3][3];
} feld;
```

- (iii) Deklarieren Sie nun eine neue Struktur, die zusätzlich zu den Anforderung aus Aufgabenteil (ii) noch ein Array enthält, welches die Nachfolger des Knotens speichert. Dimensionieren Sie dieses Array groß genug, um garantiert alle Nachfolger speichern zu können. (Falls ein Knoten weniger Nachfolger hat, werden die unbenutzten Einträge des Arrays mit NULL gefüllt.)

```
typedef struct feld2 {
    feldzustand zustand[3][3];
    struct feld *nachfolger[9];
} feld2;
```