

Klausur Grundgebiete der Informatik 1

WS 07/08

(BPO07/DPO04)

Datum: 20.03.2008

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	9				
2	6				
3	8				
4	8				
5	8				
6	8				
7	27				
8	26				
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. Gewertet werden nur Teilaufgaben, bei denen alle Antworten richtig sind. D.h. eine falsche Antwort führt dazu, dass die Aufgabe mit Null Punkten bewertet wird.

- (a) Ordnen Sie die angegebenen Wertebereiche den jeweiligen Datentypen zu. Kennzeichnen Sie die richtige Zuordnung durch ein Kreuz in der Tabelle. Gehen Sie beim Datentyp `short int` bzw. `unsigned short int` von einer Bitbreite von zwei Byte (= 16 Bit) aus. Gehen Sie von einer Maschine mit 2-Komplement Darstellung aus.

	signed char	unsigned char	unsigned short int	short int
0..255		X		
0..65535			X	
-128.. + 127	X			
-32768.. + 32767				X

- (b) Kreuzen Sie an, welche der folgenden Aussagen wahr sind.

$7n^3 + 3n^2 + 50n \in O(n^3)$

$n(1 + \frac{1}{n}) - 5 \in O(n)$

$n \log_2 n \in O(n)$

$\log_2 \log_2 n \in O(\log_2 n)$

- (c) Welcher Sortieralgorithmus bzw. welche Sortieralgorithmen arbeiten nach dem Divide-and-Conquer-Prinzip?

Heapsort

Bubblesort

Quicksort

- (d) Welche Laufzeitkomplexität besitzt der Quicksort-Algorithmus im schlechtesten Fall bei einem Feld der Größe n ?

$O(n \log_2 n)$

$O(n^2)$

$O(\log_2(n^2))$

$O(n)$

Aufgabe 2 – Kontrollstrukturen

- (a) Ersetzen Sie die for-Schleife des folgenden C-Programms durch eine äquivalente while-Schleife:

```

1  int i; char ersteZeile[80];
2  char str[]="Das ist die erste Zeile\nund das die zweite.\n";
3  for (i = 0; str[i] != '\n'; i++) {ersteZeile[i]=str[i];};
4  ersteZeile[i] = '\0';
5  printf ("i = %d, ersteZeile=%s\n", i, ersteZeile);

```

```

i=0;
while (str[i] != '\n') {
    ersteZeile[i]=str[i];
    i++;
}

```

- (b) Das folgende C-Programmfragment soll prüfen, ob die Variable x gleich 10 oder kleiner 10 ist und eine entsprechende Meldung ausgeben. Falls x größer 10 ist, soll keine Meldung erfolgen.

```

1  if (x >= 10)
2
3      if (x == 10) printf("x ist 10\n");
4
5  else
6
7      printf("x ist kleiner 10");
8

```

Leider arbeitet das Programmfragment nicht wie gewünscht. Geben Sie eine korrigierte Version an.

```

if (x <= 10) {
    if (x == 10) printf ("x ist 10\n");
    else printf ("x ist kleiner 10\n");
}

```


Aufgabe 4 – Sichtbarkeit / Gültigkeitsbereich

Geben Sie für jedes ? die Zeilennummer innerhalb des folgenden C-Programmfragments an, in der sich die Deklaration der jeweiligen Variablen bzw. der Funktion befindet.

```
1  int a(int);
2  int b();
3  int c(int);
4  int i;
5
6  int a(int b) {
7      double i;
8      ... i ... // ? = 7
9      ... b ... // ? = 6
10 }
11 int b() {
12     int a;
13     ... i ... // ? = 4
14     ... a ... // ? = 12
15 }
16 int c(int i) {
17     int a;
18     ... i ... // ? = 16
19     while (i>0){
20         ... a .. // ? = 20 17
21     }
22     ... b ... // ? = 2
23     ... i ... // ? = 16
24 }
```

Aufgabe 5 – O-Notation

- (a) Geben Sie die Definition der O-Notation an und zeigen Sie mit Hilfe der Definition die folgende Aussage. Geben Sie ein $c \in \mathbb{R}$ und ein $n_0 \in \mathbb{N}$ an, für die die Aussage gilt.

$$n^2 + 5n \in O(n^2)$$

Sei $f: \mathbb{N} \rightarrow \mathbb{N}$ und $g: \mathbb{N} \rightarrow \mathbb{N}$ dann $\exists n_0 \in \mathbb{N}, \exists c \in \mathbb{R}$, so dass gilt:
 $\forall n \geq n_0: c \cdot f(n) \geq g(n) \Leftrightarrow O(f) \in O(g)$

Sind $f: \mathbb{N} \rightarrow \mathbb{N}$ und $g: \mathbb{N} \rightarrow \mathbb{N}$ zwei Funktionen, so ist $f \in O(g)$, falls: $\exists n_0 \in \mathbb{N}, \exists c \in \mathbb{R}$, so dass $\forall n \geq n_0: f(n) \leq c \cdot g(n)$

Wegen $f = n^2 + 5n, g = n^2$

Wähle $c=2$ und $n_0=5$, dann gilt $f(n) \leq c \cdot g(n) \quad \forall n \geq n_0$
 $\Rightarrow n^2 + 5n \in O(n^2)$

- (b) Berechnen Sie mit Hilfe der Rechenregeln der O-Notation das Laufzeitverhalten der folgenden Funktion.

$$\frac{n^2 - 1}{n + 1} + 8 = \frac{(n-1)(n+1)}{n+1} + 8 = n + 7$$

$$\Rightarrow n + 7 \in O(n)$$

Aufgabe 6 – Binäre Suche

Gegeben sei das Array $A[0..N-1]$ der Größe $N = 8$ mit folgenden Schlüsselwerten:

$$A[] = \{ 1, 3, 7, 9, 15, 33, 67, 100 \}$$

- (a) Führen Sie für dieses Array eine möglichst effiziente binäre Suche nach dem Schlüssel 3 durch. Das mittlere Element wird durch $\lfloor \frac{l+r}{2} \rfloor$ berechnet. Dabei ist l der Index für die linke Grenze und r der Index für die rechte Grenze. Geben Sie für jeden Schritt die linke Grenze, die rechte Grenze und das mittlere Element an und erläutern Sie Ihre Entscheidungen. Zu Beginn ist $l = 0$ und $r = N - 1$.

1. Schritt: $l=0, r=7, m=3$
 $A[m=3] = 9 > 3 \Rightarrow r=2$

2. Schritt: $l=0, r=2, m=1$
 $A[m=1] = 3 \Rightarrow$ gefunden

- (b) Wieviele Suchschritte werden für die in Teil (a) durchgeführte Suche benötigt?

2

- (c) Geben Sie die Komplexität der binären Suche in der O-Notation in Abhängigkeit von der Anzahl der Arrayelemente N für den schlechtesten Fall an. Begründen Sie Ihre Antwort. Geben Sie einen Schlüsselwert aus dem obigen Beispiel an, der den schlechtesten Fall produzieren würde. Begründen Sie Ihre Wahl.

$O(\log_2 N)$, da sich bei jedem Schritt die Liste halbiert.

Beispiel für den schlechtesten Fall wäre 100, da hier 4 Schritte benötigt werden.

- (d) Geben Sie die Komplexität für den besten Fall in der O-Notation an. Welcher der Schlüsselwerte aus dem obigen Beispiel würde den besten Fall produzieren?

$O(1)$

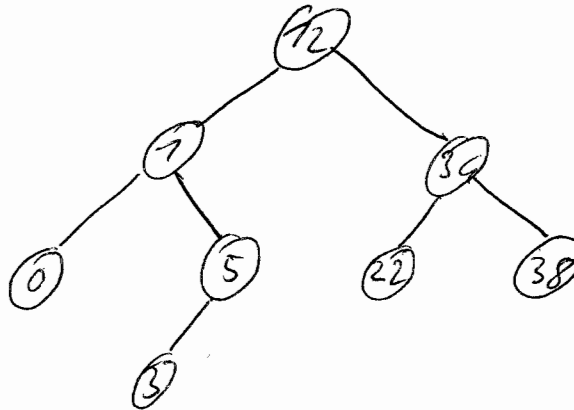
Im obigen Array: 9

Aufgabe 7 – Datenstruktur: Binärbaum

Gegeben sei das Array A der Größe $N = 8$ mit folgenden Schlüsselwerten:

$A[] = \{ 12, 1, 5, 34, 0, 3, 22, 38 \}$

- (a) Zeichnen Sie den Binären Suchbaum, der durch Einfügen der Schlüsselwerte in der dargestellten Reihenfolge in einen anfangs leeren Baum entsteht.



- (b) Geben Sie die Definition einer C-Datenstruktur `btree` an, die einen binären Baum für Schlüsselemente vom Typ `integer` repräsentiert. Definieren Sie den Datentyp `btreeptr` mit Hilfe von `typedef`, der einen Zeiger auf `btree` darstellt.

```

typedef struct btree {
    int data;
    struct btree* left, *right;
} Btree, *btreeptr;
  
```

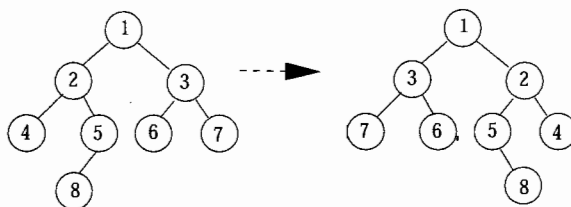
- (c) Implementieren Sie eine C-Prozedur `void PreOrder(btreetptr t)`, die den Baum in Preorder-Reihenfolge durchläuft und die einzelnen Schlüsselwerte ausgibt. Die Ausgabe des Schlüsselwertes ist dabei die für jeden Knoten durchzuführende Aktion.

```
void PreOrder(btreetptr t) {  
    if (t != NULL) {  
        printf("%d\n", t->data);  
        PreOrder(t->left);  
        PreOrder(t->right);  
    }  
}
```

- (d) Welche Reihenfolge der Schlüsselwerte ergibt sich, wenn man den Baum aus Teilaufgabe (a) mit Hilfe der Preorder-Strategie durchläuft?

12, 1, 0, 5, 3, 34, 22, 38

- (e) Schreiben Sie eine C-Funktion `void spiegeln(btreetpr t);`
 Die Funktion soll den übergebenen Binärbaum durch rekursive Vertauschung seiner Äste an der Wurzel spiegeln.
 Beispiel:

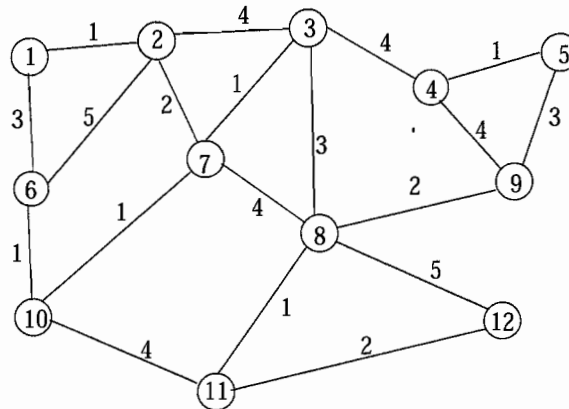


```

void spiegeln (btreetpr t) {
    btreetpr bp;
    if (t != NULL) {
        bp = t->left;
        t->left = t->right;
        t->right = bp;
        spiegeln (t->left);
        spiegeln (t->right);
    }
}
  
```

Aufgabe 8 – Graphen

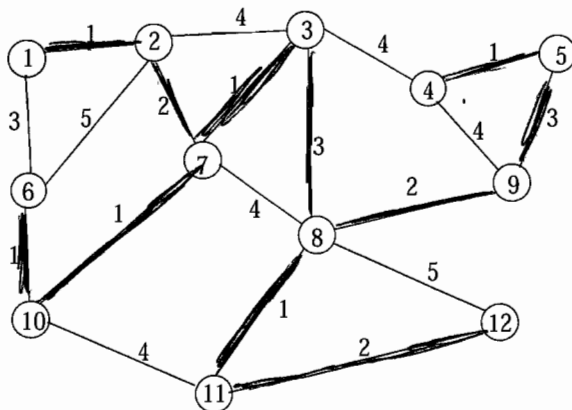
Gegeben ist der folgende ungerichtete, kantengewichtete Graph $G = (V, E)$.



(a) Geben Sie die Adjazenzlisten für alle Knoten des Graphen an.

1: 2,6
 2: 1,3,6,7
 3: 2,4,7,8
 4: 3,5,9
 5: 4,9
 6: 1,2,10
 7: 2,3,8,10
 8: 3,7,9,11,12
 9: 4,5,8
 10: 6,7,11
 11: 8,10,12
 12: 8,11

- (b) Führen Sie für den obigen Graphen den Algorithmus von *Kruskal* zur Berechnung des minimalen Spannbaums durch. Geben Sie dafür die sortierte Kantenliste L und die Liste T der Kanten des Spannbaums an. Geben Sie die Kanten in der Form (A, B) an für eine Kante von Knoten A nach Knoten B . Skizzieren Sie den Spannbaum im Graphen.



$$L = \{ (1,2), (3,7), (4,5), (6,10), (7,10), (8,11), (2,7), (8,9), (11,12), \\ (1,6), (3,8), (5,9), (2,3), (3,4), (9,9), (7,8), (10,11), (2,6), (8,12) \}$$

$$T = \{ (1,2), (3,7), (4,5), (6,10), (7,10), (8,11), (2,7), (8,9), (11,12), (3,8), (5,9) \}$$

- (c) Führen Sie beginnend mit dem Knoten $v = 1$ eine Breitensuche (BFS) durch. Verwenden Sie hierzu eine Queue. Stellen Sie für jeden Zwischenschritt die Queue dar. Geben Sie die Reihenfolge der Knoten an, in der der Graph mit Hilfe der Breitensuche durchlaufen wird. Falls für den nächsten zu verarbeitenden Knoten mehrere Alternativen zur Auswahl stehen, wählen Sie denjenigen mit dem niedrigsten Index.

v	Queue
1	2, 6
2	6, 3, 7
6	3, 7, 10
3	7, 10, 4, 8
7	10, 4, 8
10	4, 8, 11
4	8, 11, 5, 9
8	11, 5, 9, 12
11	5, 9, 12
5	9, 12
9	12
12	

- (d) Führen Sie beginnend mit dem Knoten $v = 1$ eine Tiefensuche (DFS) durch. Geben Sie die Reihenfolge der Knoten an, in der der Graph mit Hilfe der Tiefensuche durchlaufen wird. Falls für den nächsten zu verarbeitenden Knoten mehrere Alternativen zur Auswahl stehen, wählen Sie denjenigen mit dem niedrigsten Index.

1, 2, 3, 4, 5, 9, 8, 7, 10, 6, 11, 12