
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	
2	
3	
4	
5	
6	
7	
8	
Summe	

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);
```

Compilerabhängig: ja nein
Mögliche Ausgaben:

[4] (b)

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

Compilerabhängig: ja nein
Mögliche Ausgaben:

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.

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

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

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) | $f(n) \in \Omega(g(n)) \iff g(n) \in O(f(n))$ | <input type="checkbox"/> T | <input type="checkbox"/> F |
| [1] | (b) | $f(n) \in \Theta(g(n)) \iff f(n) \in O(g(n))$ und $f(n) \in \Omega(g(n))$ | <input type="checkbox"/> T | <input type="checkbox"/> 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: $c_1g(n) \leq f(n) \leq c_2g(n)$ | <input type="checkbox"/> T | <input type="checkbox"/> F |
| [1] | (d) | $(\log(n))^{10} \in \Omega(n)$ | <input type="checkbox"/> T | <input type="checkbox"/> F |
| [1] | (e) | $\sum_{i=0}^n a_i x^i, a_n \neq 0$ is in $\Theta(x^n)$ | <input type="checkbox"/> T | <input type="checkbox"/> F |
| [1] | (f) | $n \log(n) \in \Omega(n^2)$ | <input type="checkbox"/> T | <input type="checkbox"/> F |
| [1] | (g) | $(\log(n))^{\log(n)} \in O(\frac{n}{\log(n)})$ | <input type="checkbox"/> T | <input type="checkbox"/> F |
| [1] | (h) | $n! \in \Theta(n^n)$ | <input type="checkbox"/> T | <input type="checkbox"/> F |

4 Entwurf eines Suchalgorithmus

[15] Gegeben sei ein Feld $x[1, \dots, n]$ 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.

Algorithmus 1 Suche im sortierten Feld

Eingabe: Feld von ganzen Zahlen $x[1 \dots n]$ und eine Zahl z

Ausgabe: Index i , sodass $z = x[i]$ falls ein derartiger Index existiert; andernfalls $i = -1$

begin

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.

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  $n \bmod 2 = 0$  then  
    return  $temp * temp$ ;  
else  
    return  $temp * temp * x$ ;  
end
```

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.

