Prof. Dr. Ir. Joost-Pieter Katoen

Jonathan Heinen, Sabrina von Styp

Hinweise:

- Die Übungsblätter sollen in Gruppen von je 3 Studierenden aus der gleichen Kleingruppenübung bearbeitet werden.
- Die Lösungen müssen bis Montag, den 10. Mai um 11:00 Uhr in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55).
- Namen und Matrikelnummern der Studenten sowie die Nummer der Übungsgruppe sind auf jedes Blatt der Abgabe zu schreiben. Heften bzw. tackern Sie die Blätter!
- Bitte beachten Sie, dass am 4. Mai wegen der Fachschaftsvollversammlung von 10:00 - 14:00 Uhr keine Übungen stattfinden. Die Ausweichtermine wurden per Email den betroffenen Studierenden mitgeteilt.

Aufgabe 1 (Maximale Laufzeit): (8 Punkte)

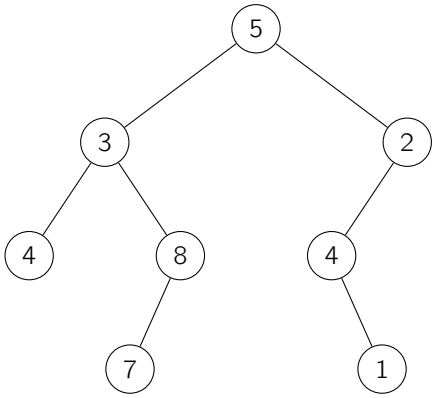
Geben Sie einen Algorithmus an, der in $\mathcal{O}(n)$ für eine Folge von n ganzen Zahlen (gegeben als Array) eine maximale Teilfolge findet. Eine Teilfolgen wird hierbei von beliebig vielen (maximal n) *aufeinanderfolgender* Zahlen gebildet. Sie ist maximal, wenn die Summe ihrer Elemente maximal ist, d.h. wir suchen aus allen möglichen Teilfolgen eine mit maximaler Summe.
Die Teilfolge soll dabei als *Startindex*, *Endindex* sowie *Summe* der Folge ausgegeben werden. Die Eingangsfolge

$$12,\ -34,\ 56,\ -5,\ -6,\ 78,\ -32,\ 8$$

liefert beispielsweise die Indizes 3 und 6 sowie die Summe $56 - 5 - 6 + 78 = 123$.

Aufgabe 2 (Baumtraversierung): (3+4+3 Punkte)

a) Geben Sie jeweils das Ergebnis der in-, pre- und post-order Traversierung des folgenden Baumes an:



b) Bestimmen Sie zu den folgenden Paaren von Linearisierungen den jeweils zugehörigen Baum:

(i) in-order: 8 4 5 7 1 2 6 9 3
pre-order: 1 4 8 7 5 3 6 2 9

(ii) in-order: 3 2 6 1 5 4 7 9 8
post-order: 3 6 1 2 4 8 9 7 5

c) Geben Sie ein minimales Beispiel für zwei unterschiedliche Bäume an, die sowohl die gleicher pre- als auch post-order Linearisierung besitzen. Minimal bedeutet hier mit der kleinstmöglichen Anzahl von Elementen wobei jedes Element maximal einmal pro Baum enthalten sein darf.

Aufgabe 3 (Analyse rekursiven Codes): (3+3+3 Punkte)

Geben Sie für die folgenden Programme die Laufzeitkomplexität als Rekursionsgleichung in Abhängigkeit des Eingabeparameters n an:

a)

```
int berechne1(int n){

    int sum = 0;

    for(int i = 0; i < n/2; i++){
        sum += 2*i - 1;
    }
    if(n <= 0)
        return sum;
    else
        return sum + 4 * berechne1(n-1) + 5;
}
```

b)

```
int berechne2(int n){

    if(n <= 0)
        return n*n;

    int wert = berechne2(n-3) * (berechne2(n-3) + 2);

    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            sum += i - j;
        }
    }
    return wert * berechne2(n-3);
}
```

c)

```
int berechne3(int n){

    if(n <= 0)
        return 5;
    else
        return berechne3(n-1) * berechne3(n-2) * berechne3(n-4);
}
```

Aufgabe 4 (Rekursionsgleichungen): (3+3+3+3 Punkte)

a) Zeigen Sie mit Hilfe der Substitutionsmethode, dass für $T(n) = 2 \cdot T(\sqrt{n}) + log_2 n$ mit $T(1) = 1$ gilt, dass $T(n) = log_2 n \cdot log_2 log_2 n$

b) Raten Sie die Komplexitätsklasse von $T(n) = 2 \cdot T(n/2) + 4 \cdot T(n/4) + n$ mit $T(1) = 1$ und zeigen Sie mit Hilfe der Substitutionsmethode die Korrektheit Ihrer geratenen Lösung.

c) Raten Sie mit Hilfe des entsprechenden Rekursionsbaum die Komplexitätsklasse zu $T(n) = 3 \cdot T(n-1) + 3n + 5$ mit $T(1) = 1$ und zeigen Sie mit Hilfe der Substitutionsmethode die Korrektheit Ihrer geratenen Lösung.

d) Nutzen sie die Methode der Variablentransformation um die exakte Lösung der Rekursionsgleichung $T(n) = T(\sqrt{n}) + 1$ mit $T(1) = 1$ zu bestimmen.