

Übungsblatt 2

Abgabe am 29.10.2018

Aufgabe 1: Master Theorem (1+1+1 Punkte)

Die folgenden Terme beschreiben die rekursiven Laufzeiten diverser Divide & Conquer Algorithmen. Wenden Sie das Master Theorem an, um diese Rekurrenzen asymptotisch aufzulösen, wobei e die Eulersche Zahl bezeichnet ($e \approx 2.72$).

(a) $T(n) = 4 \cdot T(n/4) + n$ (b) $T(n) = T(n/42) + \sqrt[3]{n}$ (c) $T(n) = 10^{11} \cdot T(n/1000) + n^e$

Aufgabe 2: Rekursionen (3+2 Punkte)

(a) Zeigen Sie, dass für $c \geq 1$ die folgende Rekursion $T(n) = \Theta(n^{c+1})$ ist.

$$\begin{aligned} T(1) &= 0 \\ T(n) &= T(n-1) + n^c. \end{aligned}$$

(b) Sei $n = \left(\frac{4}{3}\right)^k$ mit $k \in \mathbb{N}$. Folgende Rekursion ist für die Funktion T gegeben:

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2 \cdot T\left(\frac{3}{4}n\right) + 1 \end{aligned}$$

Finden Sie für $T(n)$ eine geschlossene Form ohne das Mastertheorem zu verwenden und beweisen Sie die Korrektheit Ihrer geschlossenen Form mit vollständiger Induktion.

Hinweis: Eine geschlossene Form ist nur noch von k bzw. n abhängig. Dabei sollen auch keine Summenzeichen $\sum$ oder Produktzeichen $\prod$ mehr vorkommen.

Aufgabe 3: k -närer Baum (1+1+1+1 Punkte)

In einem k -nären Baum hat jeder Knoten bis zu k Kinder. Der Wurzelknoten liegt in Level $\ell = 0$. Beantworten Sie mit kurzer Begründung:

- Wie viele Kanten hat ein k -närer Baum der Höhe ℓ mit insgesamt n Knoten?
- Wie viele Knoten liegen maximal in Level $\ell \geq 0$?
- Wie viele Knoten hat ein voller Baum der Höhe ℓ insgesamt?
- Wie viele Knoten hat ein vollständiger Baum der Höhe ℓ mindestens?

Aufgabe 4: Tree-Walk (2+1+3+1+1 Punkte)

Ein vollständiger Binärbaum kann mit der sogenannten "Level-Order" dargestellt werden. Die Knoten eines Baumes werden Level für Level von links nach rechts in ein Array geschrieben. Begonnen wird mit der Wurzel. Das könnte wie folgt aussehen:

6	3	9	2	1	7
---	---	---	---	---	---

. Betrachten Sie nun die folgenden drei alternativen Reihenfolgen. Jede erhält als Eingabe die Wurzel eines binären Baumes der Größe n (d.h. mit n Knoten):

```
function ORDER1(v)
  if v = null then return
  else
    PRINT(v)
    ORDER1(leftChild)
    ORDER1(rightChild)
  end if
end function
```

```
function ORDER2(v)
  if v = null then return
  else
    ORDER2(leftChild)
    PRINT(v)
    ORDER2(rightChild)
  end if
end function
```

```
function ORDER3(v)
  if v = null then return
  else
    ORDER3(leftChild)
    ORDER3(rightChild)
    PRINT(v)
  end if
end function
```

- a) Bestimmen Sie die worst-case Laufzeiten in Abhängigkeit von n . Die PRINT-Operation benötige dabei Zeit $\Theta(1)$. Begründen Sie kurz Ihre Antwort.
- b) Kann die jeweilige Laufzeit im best-case verbessert werden? Begründen Sie kurz Ihre Antwort.
- c) Betrachten Sie den folgenden mit Buchstaben gelabelten Binärbaum, gegeben in “Level-Order”:

S	R	M	I	G	O	K	T	H	I	E	L	W	A	L	E
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

. Die PRINT-Operation gebe den Buchstaben des an sie übergebenen Knotens aus. In welcher Reihenfolge werden die Buchstaben jeweils durch den Aufruf von ORDER1, ORDER2 und ORDER3 ausgegeben?
- d) Bestimmen Sie einen mit Buchstaben gelabelten vollständigen Binärbaum T , für den ein Aufruf von ORDER2 die Ausgabe “UNIVERSITY” erzeugt. Geben Sie Ihr Ergebnis in der üblichen Array-Schreibweise (Level-Order von T) an.
- e) Betrachten Sie nun auf dem Array von Aufgabenteil (c) den zugehörigen *ternären* Baum, also einen 3-nären Baum, in “Level-Order”. Starten Sie darauf eine verzwickte Variante von ORDER3 für ternäre Bäume, welche vor dem PRINT-Aufruf die rekursiven ORDER3-Aufrufe auf *rightChild*, *leftChild*, *middleChild* in dieser Reihenfolge (!) vornimmt. Was ist nun die Ausgabe?

Freiwillige Zusatzaufgabe 1 (3 Punkte) Entwickeln Sie einen Algorithmus, der eine Zahl in einem sortierten (!) Array sucht. Gehen Sie dabei nach dem Divide&Conquer-Prinzip vor. Stellen Sie eine Rekursionsgleichung für die Laufzeit des Algorithmus auf und lösen Sie diese auf. Messen Sie dabei als Laufzeit die Anzahl der benötigten Vergleiche.