DAP2 – Heimübung 4

Ausgabedatum: 5. 5. 17 — Abgabedatum: Fr. 12. 5. 17 (Mo. 15. 5. für Gruppen 27–32) 12 Uhr

Schreiben Sie unbedingt immer Ihren **vollständigen Namen**, **Ihre Matrikelnummer** und **Ihre Gruppennummer** auf Ihre Abgaben!

Aufgabe 4.1 (6 Punkte): (Rekursionsgleichungen)

Gegeben seien die Rekursionsgleichungen:

a)
$$T(n) = \begin{cases} 1 & \text{für } n = 1 \\ 4 \cdot T\left(\frac{n}{2}\right) + n^2 & \text{für } n > 1 \end{cases}$$

b)
$$T(n) = \begin{cases} 1 & \text{für } n = 1 \\ 9 \cdot T\left(\frac{n}{3}\right) + n\sqrt{n} & \text{für } n > 1 \end{cases}$$

c)
$$T(n) = \begin{cases} 1 & \text{für } n = 1 \\ 2 \cdot T\left(\frac{n}{4}\right) + 8n & \text{für } n > 1 \end{cases}$$

Bestimmen Sie eine asymptotische obere Schranke für $T(n)$ und beweisen Sie sie mittels Induktion. Sie dürfen annehmen, dass n von der Form 2^k, 3^k bzw. 4^k für ein $k \in \mathbb{N}$ ist.

Lösung:

a) Aus dem Master-Theorem (aus der Vorlesung) würden wir haben, dass $f(n) = n^2$, $\gamma \cdot n^2 = 4 \cdot \left(\frac{n}{2}\right)^2 \Leftrightarrow \gamma = 1$ ist, wovon folgt $T(n) \in O(n^2 \log n)$. Da wir aber keine Konstante für diese Behauptung kennen, sollten wir genaueren Ausdrück herleiten.

Behauptung: $T(n) = n^2 \log n + n^2 = O(n^2 \log n)$ (wir nehmen an, dass n eine 2er Potenz ist).

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

für alle k, so dass $n > 2^k$. Wenn $k = \log_2 n$, gilt weiter:

$$\begin{aligned} T(n) &= 4^{\log_2 n} \cdot T(1) + n^2 \cdot \log_2 n = n^2 \cdot 1 + n^2 \cdot \log_2 n \\ &= n^2 \cdot \log_2 n + n^2 = O(n^2 \cdot \log n) \end{aligned}$$

Die bisherige Analyse ist eher ein intuitives, schlaues Erraten der Lösung. Die Behauptung ist mittels Induktion zu beweisen:

Beweis:

(I.A.) Für $n = 1$ ist $T(n) = 1 = 1^2 \cdot \log_2 1 + 1^2 = 1 \cdot 0 + 1 = 1$.

(I.V.) Die Behauptung gelte für ein beliebiges, aber festes $n_0 = 2^{k_0}$ (n_0 ist eine Zweierpotenz).

(I.S.) Betrachte $n = 2 \cdot n_0$ (die nächste 2er Potenz). Dann gilt

$$\begin{aligned} T(n) &= 4 \cdot T\left(\frac{2 \cdot n_0}{2}\right) + n^2 = 4 \cdot T(n_0) + n^2 \\ &\overset{\text{(IV)}}{=} 4 \cdot \left(n_0^2 \cdot \log_2 n_0 + n_0^2\right) + n^2 = 4 \cdot n_0^2 \cdot \log_2 n_0 + 4n_0^2 + n^2 \\ &= = n^2 \cdot \log_2 n_0 + n^2 + n^2 = n^2 \cdot \log_2 n_0 + n^2 \cdot \log_2 2 + n^2 \\ &= = n^2 \cdot (\log_2 n_0 + \log_2 2) + n^2 = n^2 \cdot \log_2 n + n^2 \end{aligned}$$

Damit folgt die Behauptung für alle natürliche Zahlen n.

b) Aus dem Master-Theorem würden wir haben, dass $f(n) = n\sqrt{n}$, $\gamma \cdot n\sqrt{n} = 9 \cdot \frac{n}{3} \cdot \sqrt{\frac{n}{3}} \Leftrightarrow \gamma = \sqrt{3}$ ist, wovon folgt $T(n) \in O(n^{\log_3 9}) = O(n^2)$.

Behauptung: $T(n) = \frac{3+\sqrt{3}}{2} \cdot n^2 - \frac{1+\sqrt{3}}{2} \cdot n\sqrt{n} = O(n^2)$ (wir nehmen an, dass n eine 3er Potenz ist).

$$\begin{aligned} T(n) &= 9 \cdot T\left(\frac{n}{3}\right) + n\sqrt{n} \\ &= 9 \cdot \left(9 \cdot T\left(\frac{n}{3^2}\right) + \frac{n}{3} \cdot \sqrt{\frac{n}{3}}\right) + n\sqrt{n} = 9^2 \cdot T\left(\frac{n}{3^2}\right) + \sqrt{3} \cdot n\sqrt{n} + n\sqrt{n} \\ &= 9^2 \cdot \left(9 \cdot T\left(\frac{n}{3^3}\right) + \frac{n}{3^2} \cdot \sqrt{\frac{n}{3^2}}\right) + \sqrt{3} \cdot n\sqrt{n} + n\sqrt{n} \\ &= 9^3 \cdot T\left(\frac{n}{3^3}\right) + 3 \cdot n\sqrt{n} + \sqrt{3} \cdot n\sqrt{n} + n\sqrt{n} \\ &= \ldots = 9^k \cdot T\left(\frac{n}{2^k}\right) + n\sqrt{n} \sum_{i=0}^{k-1} \sqrt{3}^i \end{aligned}$$

für alle k, so dass $n > 3^k$. Wenn $k = \log_3 n$, gilt weiter:

$$\begin{aligned} T(n) &= 9^{\log_3 n} \cdot T(1) + n\sqrt{n} \sum_{i=0}^{\log_3 n - 1} \sqrt{3}^i = n^2 \cdot 1 + n\sqrt{n} \cdot \frac{\sqrt{3}^{\log_3 n} - 1}{\sqrt{3} - 1} \\ &= n^2 + n\sqrt{n} \cdot \frac{\sqrt{n} - 1}{\sqrt{3} - 1} = n^2 + n\sqrt{n} \cdot \frac{\sqrt{n} - 1}{\sqrt{3} - 1} \cdot \frac{\sqrt{3} + 1}{\sqrt{3} + 1} \\ &= \frac{3 + \sqrt{3}}{2} \cdot n^2 - \frac{1 + \sqrt{3}}{2} \cdot n\sqrt{n} = O(n^2) \end{aligned}$$

Die bisherige Analyse ist eher ein intuitives, schlaues Erraten der Lösung. Die Behauptung ist mittels Induktion zu beweisen:

Beweis:

(I.A.) Für $n = 1$ ist $T(n) = 1 = \frac{3+\sqrt{3}}{2} \cdot 1^2 - \frac{1+\sqrt{3}}{2} \cdot 1\sqrt{1} = \frac{3}{2} - \frac{1}{2} = 1$.

(I.V.) Die Behauptung gelte für ein beliebiges, aber festes $n_0 = 3^{k_0}$ (n_0 ist eine Dreierpotenz).

(I.S.) Betrachte $n = 3n_0$ (die nächste 3er Potenz). Dann gilt

$$\begin{aligned}
T(n) &= 9 \cdot T\left(\frac{3n_0}{3}\right) + n\sqrt{n} = 9 \cdot T(n_0) + n\sqrt{n} \\
&\overset{\text{(IV)}}{=} 9 \cdot \left(\frac{3+\sqrt{3}}{2} \cdot n_0^2 - \frac{1+\sqrt{3}}{2} \cdot n_0\sqrt{n_0}\right) + n\sqrt{n} \\
&= \frac{3+\sqrt{3}}{2} \cdot (3 \cdot n_0)^2 - \sqrt{3} \cdot \frac{1+\sqrt{3}}{2} \cdot 3n_0\sqrt{3n_0} + n\sqrt{n} \\
&= \frac{3+\sqrt{3}}{2} \cdot n^2 - \frac{3+\sqrt{3}}{2} \cdot n\sqrt{n} + n\sqrt{n} \\
&= \frac{3+\sqrt{3}}{2} \cdot n^2 - \frac{1+\sqrt{3}}{2} \cdot n\sqrt{n} \in O(n^2).
\end{aligned}$$

Damit folgt die Behauptung für alle natürliche Zahlen n.

□

c) Aus dem Master-Theorem würden wir haben, dass $f(n) = 8n$, $\gamma \cdot 8n = 2 \cdot 8 \cdot \frac{n}{4} \Leftrightarrow \gamma = \frac{1}{2}$ ist, wovon folgt $T(n) \in O(8n) = O(n)$.

Behauptung: $T(n) = 16n - 15\sqrt{n} = O(n)$ (wir nehmen an, dass n eine 4er Potenz ist).

$$\begin{aligned}
T(n) &= 2 \cdot T\left(\frac{n}{4}\right) + 8n \\
&= 2 \cdot \left(2 \cdot T\left(\frac{n}{4^2}\right) + 8 \cdot \frac{n}{4}\right) + 8n = 2^2 \cdot T\left(\frac{n}{4^2}\right) + 8n \cdot \frac{1}{2} + 8n \\
&= 2^2 \cdot \left(2 \cdot T\left(\frac{n}{4^3}\right) + 8 \cdot \frac{n}{4^2}\right) + 8n \cdot \frac{1}{2} + 8n \\
&= 2^3 \cdot T\left(\frac{n}{4^3}\right) + 8n \cdot \frac{1}{2^2} + 8n \cdot \frac{1}{2} + 8n \\
&= \ldots = 2^k \cdot T\left(\frac{n}{4^k}\right) + 8n \cdot \sum_{i=0}^{k-1} \frac{1}{2^i}
\end{aligned}$$

für alle k, so dass $n > 4^k$. Wenn $k = \log_4 n$, gilt weiter:

$$\begin{aligned}
T(n) &= 2^{\log_4 n} \cdot T(1) + 8n \cdot \sum_{i=0}^{\log_4 n - 1} \frac{1}{2^i} = \sqrt{n} \cdot 1 + 8n \cdot \frac{1 - \frac{1}{2^{\log_4 n}}}{1 - \frac{1}{2}} \\
&= \sqrt{n} + 8n \cdot \frac{1 - \frac{1}{\sqrt{n}}}{\frac{1}{2}} = \sqrt{n} + 16n \cdot \left(1 - \frac{1}{\sqrt{n}}\right) = 16n - 15\sqrt{n} = O(n)
\end{aligned}$$

Die bisherige Analyse ist eher ein intuitives, schlaues Erraten der Lösung. Die Behauptung ist mittels Induktion zu beweisen:

Beweis:

(I.A.) Für $n = 1$ ist $T(n) = 1 = 16 \cdot 1 - 15\sqrt{1} = 16 - 15 = 1$.

(I.V.) Die Behauptung gelte für ein beliebiges, aber festes $n_0 = 4^{k_0}$ (n_0 ist eine Vierer-potenz).

(I.S.) Betrachte $n = 4 \cdot n_0$ (die nächste 4er Potenz). Dann gilt

$$\begin{aligned} T(n) &= 2 \cdot T\left(\frac{4 \cdot n_0}{4}\right) + 8n = 2 \cdot T(n_0) + 8n \\ &\overset{\text{(IV)}}{=} 2 \cdot (16n_0 - 15\sqrt{n_0}) + 8n = 2 \cdot 8 \cdot 2 \cdot n_0 - 2 \cdot 15\sqrt{n_0} + 8n \\ &= 8 \cdot 4n_0 - 15\sqrt{4n_0} + 8n = 8n - 15\sqrt{n} + 8n \\ &= 16n - 15\sqrt{n} \end{aligned}$$

Damit folgt die Behauptung für alle natürliche Zahlen n.

□

Hinweise: Für Fälle, wo die Aufteilung genau gleich viele Kosten verursacht wie die Kosten beim Zusammenfügen (Teilaufgabe 1), lässt sich der Induktionsbeweis meist ohne größeren Aufwand formulieren. Lediglich in den Fällen (Teilaufgaben 2 und 3), wo die Kosten des Zusammenfügens echt geringer sind als die Kosten der Aufteilung, muss man auf die Wahl der Konstanten achten. Hier ist es allerdings häufig möglich, eine gleiche Umformung der rekursiven Form anzugeben, wie das hier auch bei allen Teilaufgaben geschehen ist. Wenn die Umformung keine Abschätzung enthält, so lässt sich mit dem Ergebnis direkt der Induktionsbeweis führen.

Aufgabe 4.2 (4 Punkte): (Merge-Operation)

Wir betrachten die Funktion Merge(A, p, q, r) aus der Vorlesung. Diese Funktion fügt die sortierten Teilarrays $A[p \ldots q]$ und $A[q+1 \ldots r]$ $(1 \leq p \leq q < r \leq \text{length}[A])$ von A zum sortierten Teilarray $A[p \ldots r]$ zusammen.

a) Spezifizieren Sie die Merge-Funktion in Pseudo-Code und **kommentieren Sie diesen**. Verwenden Sie dabei zwei Indexvariablen i und j, die auf den Teilarrays $A[p \ldots q]$ und $A[q+1 \ldots r]$ nach dem *Reißverschlussprinzip* fortschreiten. Das Resultat soll in einem (lokalen) Hilfsarray B im Teilbereich $B[p \ldots r]$ aufgebaut werden. Beachten Sie die Randfälle, in denen eines der Teilarrays vollständig abgearbeitet ist.

b) Führen Sie eine exakte Worst-Case-Laufzeitanalyse für Ihr Merge-Programm bei Eingabe von zwei Arrays der Länge n durch.

Lösung:

a) Die Merge-Funktion als Pseudo-Code:

```
Merge(A(p,q,r)):                          /* Voraussetzung: p ≤ q und q+1 ≤ r */
1.  Array B                          /* Resultatsfeld B mit length[B] = length[A] */
                                /* Bemerkung: Es reicht auch ein globales Hilfsarray */
2.  i ← p                                          /* Indexvariable i in A[p..q] */
3.  j ← q+1                                   /* Indexvariable j in A[q+1..r] */
4.  while (i ≤ q+1 and j ≤ r) or (i ≤ q and j ≤ r+1) do
        /* Solange mindestens einer der Indizes i und j im erlaubten Bereich ist
        */
5.      if i = q+1 or (j ≤ r and A[j] < A[i]) then
            /* Man wählt aus zweitem Teil falls Erster ausgeschöpft ist oder das
            aktuelle Element im Zweiten kleiner ist */
6.          B[i+j-q-1] ← A[j]
7.          j ← j+1                  /* Zeiger im zweiten Teil wird aktualisiert */
8.      else
9.          B[i+j-q-1] ← A[i]
10.         i ← i+1              /* Sonst wird Zeiger im ersten Teil aktualisiert */
11. for k ← p to r do
                                                  /* Übernahme des Hilfsarrays zu A */
12.     A[k] ← B[k]
```

b) Die Worst-Case-Laufzeitanalyse für das in 1) angegebene Programm für Merge.

Nach Aufgabenstellung kann angenommen werden: $q - p + 1 = n$ sowie $r - q = n$.

Aufwand für Zeile:

1: $2n$

2 und 3: 2

4: $2n + 1$

5: $2n$

(6 und 7) oder (9 und 10): 2 mal $2n$

11: $2n + 1$

12: $2n$

Insgesamt: $14n + 4 \in O(n)$.

c) **Zusatzaufgabe: Beweis der Korrektheit der Merge-Funktion.**

Es ist vorausgesetzt, dass $A[p \dots q]$ und $A[q+1 \dots r]$ sortiert sind (wir setzen hier immer aufsteigende Sortierung voraus) und ferner $p \leq q$ sowie $q + 1 \leq r$ gilt.

In dem folgenden Lemma formulieren wir eine Invariante für die **while**-Schleife in Merge. Wir vereinbaren dazu die folgende Notation:

$$C \in B[s \dots t] \text{ genau dann, wenn } C = B[j] \text{ für ein } j \text{ mit } s \leq j \leq t$$

Lemma 1: Nach jedem Schleifendurchlauf durch die **while**-Schleife gilt die Invariante $F(i, j)$, die sich aus den beiden Bedingungen B1 und B2 zusammensetzt:

B1: $B[p \ldots i+j-q-2]$ enthält die Elemente aus $A[p \ldots i-1] \cup A[q+1 \ldots j-1]$ und ist sortiert.

B2 Für alle $C \in B[p \ldots i+j-q-2]$, für alle $D \in A[i \ldots q] \cup A[j \ldots r] : C \leq D$.

Durch vollständige Induktion über die Anzahl n der Schleifendurchläufe:

(I.A.) Für $n = 0$, also vor dem ersten Schleifendurchlauf, ist $i = p$ und $j = q+1$. Es gilt: $B[p \ldots i+j-q-2]$, $A[p \ldots i-1]$ und $A[q+1 \ldots j-1]$ sind leer. Da ein leeres Array sortiert ist, ist $F(i,j)$ erfüllt.

(I.V.) Für ein beliebiges, aber festes $n \geq 0$ gelte $F(i,j)$ nach dem n-ten Schleifendurchlauf.

(I.S.) Es gelte nach dem n-ten Schleifendurchlauf $i \leq q$ oder $j \leq r$ (andernfalls ist nichts zu zeigen), so dass ein weiterer Schleifendurchlauf erfolgt .

Nach (I.V.) enthält nach dem n-ten Schleifendurchlauf $B[p \ldots i+j-q-2]$ die Elemente aus $A[p \ldots i-1] \cup A[q+1 \ldots j-1]$ und ist sortiert. Wir betrachten den $(n+1)$-ten Durchlauf durch die Schleife. Wir bezeichnen die Werte von i und j nach dem Schleifendurchlauf mit i' und j'. Zu zeigen ist, dass nach dem Schleifendurchlauf $F(i',j')$ gilt.

Fall 1: $i = q+1$ oder ($j \leq r$ und $A[j] < A[i]$).

Es wird $B[i+j-q-1]$ gleich $A[j]$ gesetzt. Ferner ist $j' = j+1$ und $i' = i$.

Nach I.V. ist $B[p \ldots i+j-q-2]$ sortiert und enthält die Elemente aus $A[p \ldots i-1] \cup A[q+1 \ldots j-1]$.

Ebenfalls nach I.V. gilt:

$$\forall C \in B[p \ldots i+j-q-2] : C \leq A[j].$$

Damit ist dann $B[p \ldots i+j-q-1]$ sortiert und enthält die Elemente aus $A[p \ldots i-1] \cup A[q+1 \ldots j]$. Wegen $i' = i$ und $j' = j+1$ bedeutet dies, dass $B[p \ldots i'+j'-q-2]$ sortiert ist und die Elemente aus $A[p \ldots i'-1] \cup A[q+1 \ldots j'-1]$ enthält, so dass Bed.1 für $F(i',j')$ erfüllt ist.

Da ferner $A[j] \leq D$ für alle $D \in A[i \ldots q] \cup A[j+1 \ldots r]$ (im Falle $i > q$ ist $A[i \ldots q] = [\,]$), und $A[j]$ zugleich aber auch das größte Element in $B[p \ldots i'+j'-q-2]$ ist, ist auch Bed. 2 für $F(i',j')$ erfüllt.

Damit gilt in diesem Fall die Aussage $F(i',j')$.

Fall 2: Fall 1 gilt nicht, d.h. es ist $i \leq q$ und ($j = r+1$ oder $A[i] \leq A[j]$).

Die Argumentation verläuft analog zu Fall 1.

Nach I.V. gilt

$$\forall C \in B[p \ldots i+j-q-2] : C \leq A[i].$$

Damit ist dann $B[p \ldots i+j-q-1]$ sortiert und enthält die Elemente aus $A[p \ldots i] \cup A[q+1 \ldots j-1]$. Wegen $i' = i+1$ und $j' = j$ bedeutet dies, dass $B[p \ldots i'+j'-q-2]$ sortiert ist und die Elemente aus $A[p \ldots i'-1] \cup A[q+1 \ldots j'-1]$ enthält. Somit ist in diesem Fall Bed. 1 für $F(i',j')$ erfüllt.

Da $A[i] \leq D$ für alle $D \in A[i+1 \ldots q] \cup A[j \ldots r]$, zugleich $A[i]$ aber auch das größte Element in $B[p \ldots i'+j'-q-2]$ ist, ist Bed. 2 für $F(i',j')$ erfüllt.

Damit ist $F(i,j)$ als Schleifeninvariante nachgewiesen.

Da in jedem Schleifendurchlauf entweder i oder j um 1 erhöht wird, andererseits bei Erreichen von $i = q+1$ und $j = r+1$ kein weiterer Schleifendurchlauf erfolgt, terminiert die **while**-Schleife. Da $B[p \dots q+1+r+1-q-2] = B[p \dots r]$, erhalten wir aus Lemma 1):

Satz 1: $B[p \dots r]$ enthält die Elemente aus $A[p \dots r]$ in sortierter Reihenfolge.

Aufbauend auf Satz 1 formulieren wir als ächstes eine Invariante für die **for**-Schleife in Merge, deren Beweis offensichtlich ist:

Lemma 2: Nach s Schleifendurchläufen der **for**-Schleife, $0 \leq s \leq r-p+1$, gilt: $A[p \dots p+s-1] = B[p \dots p+s-1]$

Als Folgerung von Lemma 2) erhalten wir:

Satz 2: Die **for**-Schleife terminiert, so dass in $A[p \dots r]$ die Elemente $A[p \dots r]$ aus dem ursprünglichen Array A in aufsteigend sortierter Reihenfolge enthalten sind.