Name	Übungsgruppe	1	2	3	4
Theodor Bajusz	5	x	x		
Valerij Dobler	13	x	x		
Matz Radloff	6	x	x		
Robin Wannags	5	x	x		

Übung 1.

In Algorithmus 1 ist der Pseudocode eines Sortieralgorithmus BubbleSort gezeigt.

Algorithmus 1 : BubbleSort(A[1, ..., n])

```
1 s ← 1
2 i ← 0/* Hilfsvariable für Beweis der Invariante */
3 while s = 1
4     s ← 0
5     i ← i + 1
6     for j ← 1 to n − 1
7         if A[j] > A[j + 1]
8             A[j] ↔ A[j + 1]
9             s ← 1
```

(a) Wenden Sie den Algorithmus am Beispiel des Arrays $A = \langle 6, 7, 3, 1, 9, 5, 2\rangle$ an. Geben Sie dazu den Inhalt des Arrays A sowie die Anzahl der vorgenommenen Vertauschungen v (Zeile 6) nach jedem Durchlauf der äußeren Schleife (Zeilen 2 bis 7) an. (2 Punkte)

Durchlauf n	Array A	Vertauschungen v
0	$\langle 6, 7, 3, 1, 9, 5, 2\rangle$	0
1	$\langle 6, 3, 1, 7, 5, 2, 9\rangle$	4
2	$\langle 3, 1, 6, 5, 2, 7, 9\rangle$	4
3	$\langle 1, 3, 5, 2, 6, 7, 9\rangle$	3
4	$\langle 1, 3, 2, 5, 6, 7, 9\rangle$	1
5	$\langle 1, 2, 3, 5, 6, 7, 9\rangle$	1
6	$\langle 1, 2, 3, 5, 6, 7, 9\rangle$	0
Summe		13

(b) Beweisen Sie die Korrektheit von *BUBBLESORT*. Bestimmen Sie dazu zunächst eine geeignete Invariante für die innere Schleife (Zeilen 4 bis 7) und nutzen Sie diese dann zur Formulierung einer geeigneten Invariante für die äußere Schleife (Zeilen 2 bis 7). (6 Punkte)

Sei das Eingabearray $A = \langle a_1, a_2, \ldots, a_n\rangle$
Behauptung: Wir betrachten die Invariante $I(b) = \forall i \in \mathbb{N}_{<b} : A[b] \geq A[i]$

1) Initialisierung: I(1)
Hat A nur ein Element, und dieses ist nicht kleiner als alle anderen.
$\Longrightarrow I(1) giltauchvordemerstenfor - Schleifendurchlauf$

2) Erhaltung: Angenommen $I(j)$ gilt, und wir kommen nach Zeile 7 in die if-Abfrage, dann gilt:

$$I(j) \wedge A[j] > A[j+1]$$

daher gilt nach dem Tausch in Zeile 8 $I(j+1)$

3) Terminierung: Die for-Schleife läuft bis zum Element $n-1$ und vergleicht in der letzten Iteration die Elemente $n-1$ und n miteinander. Falls wir in die if-Abfrage reingehen, werden beide Elemente getauscht, wenn nicht, dann waren sie bereits korrekt sortiert. In jedem Fall gilt am Ende I(n).

(c) Analysieren Sie die best-case und worst-case Laufzeit von BubbleSort. Hinweis: Die best-case Laufzeit ist eine untere Schranke, die angibt wie lange ein Algorithmus mindestens für die Verarbeitung einer idealen, d.h. am schnellsten zu bearbeiteten, Eingabe benötigt. (3 Punkte)

best-case: Array ist schon sortiert. (lineare Laufzeit)

Zeile	Laufzeit	
1	$\mathcal{O}(1)$	
2	$\mathcal{O}(1)$	
3	$\mathcal{O}(1)$	
4	$\mathcal{O}(1)$	
5	$\mathcal{O}(1)$	
6	$\sum_{i=0}^{n-1} \mathcal{O}(i)$	
7	$\mathcal{O}(1)$	ist nie wahr, weil das Array schon sortiert ist

$\Longrightarrow T(n) = 3 \cdot \mathcal{O}(1) + (n-1) \cdot 2 \cdot \mathcal{O}(1) = (2 \cdot n \cdot +1) \cdot \mathcal{O}(1) = \mathcal{O}(n)$

worst-case: Array muss n-mal durchlaufen werden (n-1 zum tauschen, 1 mal zum überprüfen am Ende) (quadratische Laufzeit)

Zeile	Laufzeit
1	$\mathcal{O}(1)$
2	$\mathcal{O}(1)$
3	$\sum_{i=0}^{n} \mathcal{O}(i)$
4	$\mathcal{O}(1)$
5	$\mathcal{O}(1)$
6	$\sum_{i=0}^{n-1} \mathcal{O}(i)$
7	$\mathcal{O}(1)$
8	$\mathcal{O}(1)$
9	$\mathcal{O}(1)$

$\Longrightarrow T(n) = 2 \cdot \mathcal{O}(1) \; + \; n \cdot (3 + (n-1) \cdot 4 \cdot \mathcal{O}(1))$

$= 4n^2 \cdot \mathcal{O}(1) \; - \; n\mathcal{O}(1) \; + \; 2 \cdot \mathcal{O}(1)$

$= \mathcal{O}(n^2)$

Übung 2.

(a) Bestimmen Sie die Größenordnung der Funktionen wenn möglich mittels Mastertheorem. Falls das Mastertheorem nicht anwendbar sein sollte, begründen Sie dies und verwenden stattdessen die Substitutionsmethode. (4 Punkte)

(i)

$$T_1(n) := \begin{cases} 1, & \text{für } n = 1 \\ 4 \cdot T(\lceil n/4 \rceil) + 8n, & \text{sonst} \end{cases}$$

Rechnung:

$$a = 4, b = 4, f(n) = 8n \Longrightarrow log_b(a) = log_4(4) = 1$$

$$f(n) = 8n \in \Theta(n) \overset{M-Thm.(b)}{\Longrightarrow} T_1(n) = \Theta(n \cdot log\, n) \quad \checkmark$$

(ii)

$$T_2(n) := \begin{cases} 1, & \text{für } n = 0 \\ 2 \cdot T(n-1) + 4, & \text{sonst} \end{cases}$$

Rechnung:

$$\begin{aligned}
\mathrm{T}_2(n) \quad &= 2 \cdot T(n-1) + 4 \\
&= 2 \cdot (2 \cdot T(n-2) + 4) + 4 = 4 \cdot T(n-2) + 2 \cdot 4 + 4 \\
&= 4 \cdot (2 \cdot T(n-3) + 2 \cdot 4 + 4) + 4 = 8 \cdot T(n-3) + 4 \cdot 4 + 2 \cdot 4 + 4 \\
&\vdots \\
&= 2^{n-1} \cdot (2 \cdot T(n-(n-1)) + 4) + \textstyle\sum_{i=0}^{n-2} (2^i \cdot 4) \\
&= 2^{n-1} \cdot (2 \cdot \qquad 1 \qquad + 4) + \textstyle\sum_{i=0}^{n-2} (2^i \cdot 4) \\
&= 2^n + \textstyle\sum_{i=0}^{n-1} (2^i \cdot 4) \\
&= 2^n + 2^{n-1} \cdot 4 + 2^{n-2} \cdot 4 + \ldots + 2 \cdot 4 + 4 \\
&= 2^n + 2^{n+1} + 2^n + \ldots + 2 \cdot 4 + 4 \in \Theta(2^{n+2}) \quad \checkmark
\end{aligned}$$

(iii)

$$T_3(n) := \begin{cases} 1, & \text{für } n = 1 \\ 3 \cdot T(\lfloor n/3 \rfloor) + 2n\, log\, n, & \text{sonst} \end{cases}$$

Rechnung:

$$a = 3, b = 3, f(n) = 2n\, log\, n \implies log_b(a) = log_3(3) = 1$$

Überprüfen der Regularitätsbedingung:

$$a \cdot f(n/b) \leq c \cdot f(n) \quad , \text{für } c < 1 \text{ und große } n$$

$$3 \cdot \frac{2n}{3} \cdot log(\frac{n}{3}) = 2n \cdot log(\frac{n}{3}) \leq c \cdot 2n\, log\, n$$

für $n > 1$, kürze beide Seiten mit $2n$

$$log(\frac{n}{3}) \leq c \cdot log\, n$$

$$log(n) - log(3) \leq c \cdot log\, n$$

für ein beliebiges, aber festes n, bekommt man das gesuchte c derart:

$$1 - \frac{log(3)}{log\, n} \leq c \quad \checkmark$$

$$f(n) = 2n\, log\, n \in \Omega(n^{1+\epsilon}) \overset{M-Thm.(c)}{\Longrightarrow} T_3(n) = \Theta(2n\, log\, n) = \Theta(n\, log\, n)$$

(iv)

$$T_4(n) := \begin{cases} 1, & \text{für } n = 1 \\ 4 \cdot T(\lfloor n/3 \rfloor) + 2n\, log\, n, & \text{sonst} \end{cases}$$

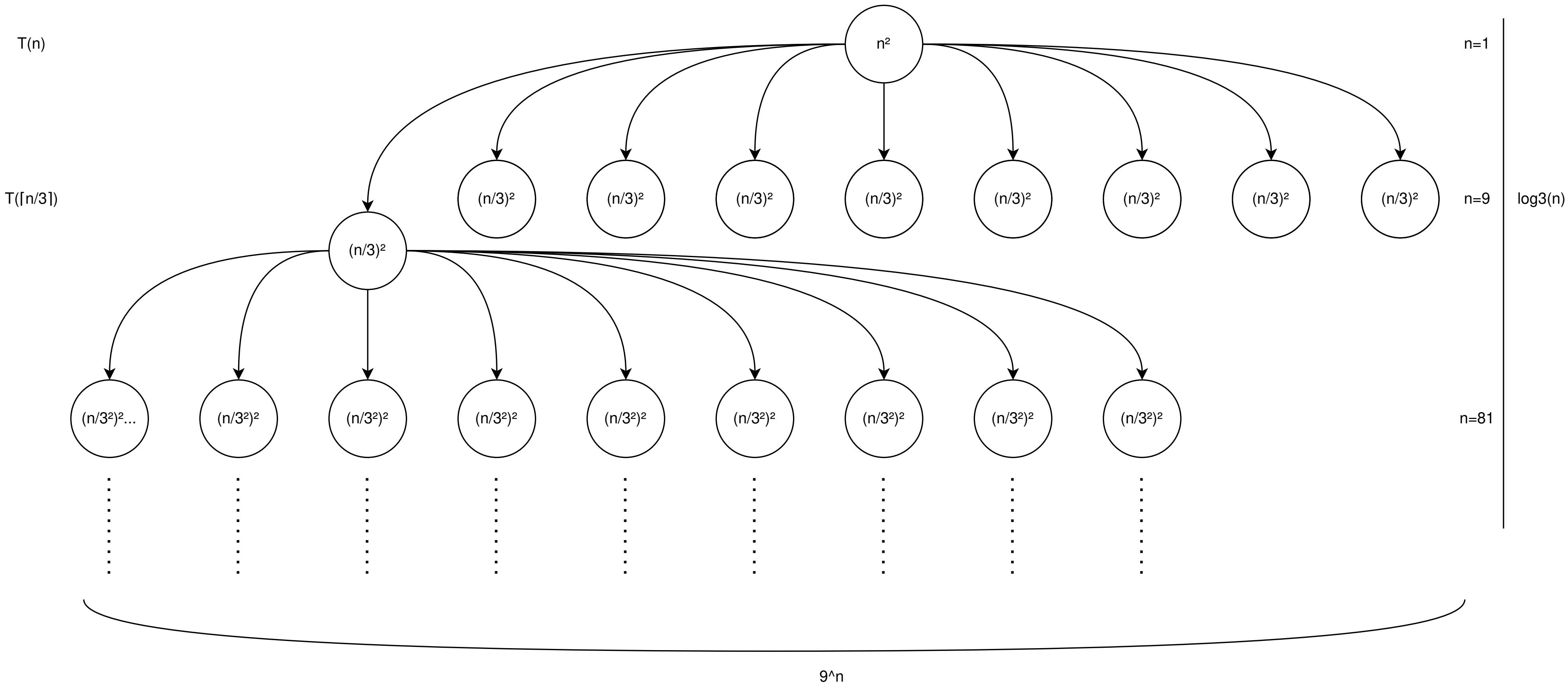
Rechnung:

$$a = 4, b = 3, f(n) = 2n\, log\, n \implies log_b(a) = log_3(4) \approx 1,26$$

$$f(n) = 2n\, log\, n \in O(n^{log_3(4)-\epsilon}) \overset{M-Thm.(a)}{\Longrightarrow} T_4(n) = \Theta(n^{log_3(4)}) \quad \checkmark$$

(b) Bestimmen Sie die Größenordnung der Funktion(en), indem sie die folgende Rekursionsgleichung mithilfe eines Rekursionsbaums lösen.(2 Punkte)

$$T_5(n) := \begin{cases} 1, & \text{für } n = 1 \\ 9 \cdot T(\lceil n/3 \rceil) + n^2, & \text{sonst} \end{cases}$$



$$\implies 9^n \cdot log_3(n) = O(9^n \cdot log(n))$$

(c) Beweisen Sie die Korrektheit ihrer Lösung von T_5 per Induktion. Induktionsbeweis. (4 Punkte)

$$T_5(n) := \begin{cases} 1, & \text{für } n = 1 \\ 9 \cdot T(\lceil n/3 \rceil) + n^2, & \text{sonst} \end{cases}$$

$$\begin{aligned} \mathrm{T}_5(n) \quad &= 9 \cdot T(\lceil \tfrac{n}{3} \rceil) + n^2 \\ &= 9 \cdot (9 \cdot T(\lceil \tfrac{\lceil \frac{n}{3} \rceil}{3} \rceil) + n^2) + n^2 = 9^2 \cdot T(\lceil \tfrac{\lceil \frac{n}{3} \rceil}{3} \rceil) + 9 \cdot n^2 + n^2 \\ &\vdots \\ &\geq 9^n + \textstyle\sum_{i=0}^{n-1} (9^i \cdot n^2) \end{aligned}$$