Prof. Dr. Leif Kobbelt
Stefan Dollase, Ira Fesefeldt, Alexandra Heuschling, Gregor Kobsik

# Lösung - Übung 4

## Aufgabe 4 (Ordnungen und Sortieren): 3 + 4 + 4 = 11 Punkte

**a)** Die Quersumme einer Zahl $n = \sum_{i=0}^{k} a_i \cdot 10^i$ ist $q(n) = \sum_{i=0}^{k} a_i$. Wir definieren die Relation $\sqsubseteq$ als

$$n \sqsubseteq m \qquad \text{genau dann wenn} \qquad q(n) \leq q(m) \ .$$

Zeigen oder widerlegen Sie, dass $\sqsubseteq$ eine totale Ordnung ist.

**b)** Gegeben ein Graph $G = (V, E)$. Wir definieren die Relation $\sqsubseteq$ auf $V$ als $v \sqsubseteq w$ genau dann wenn $v = w$ oder ein Pfad von $v$ nach $w$ existiert.

- Wenn $G$ ein Baum ist, welche Form hat dann die Relation $\sqsubseteq$? Begründen Sie ihre Antwort!
- Wenn $G$ eine Liste ist, welche Form hat dann die Relation $\sqsubseteq$? Begründen Sie ihre Antwort!

**c)** Gegeben ein nicht stabilen Sortieralgorithmus $S$. Geben Sie ein einfaches Verfahren $S'$ an, welches $S$ beinhaltet, sodass $S'$ stabil ist. Welche Eigenschaften verliert $S'$ aufgrund ihres Verfahrens, die $S$ noch hatte?

### Lösung

**a)** Die Relation $\sqsubseteq$ ist nicht Antisymmetrisch, da z.b. $q(10) = 1 = q(1)$, also ist $10 \sqsubseteq 1$ und $10 \sqsupseteq 1$, aber nicht $10 = 1$.

**b)**

- Wenn $G$ ein Baum ist, dann gilt:
  - Reflexivität da mit $v = v$ auch $v \sqsubseteq v$ gilt.
  - Antisymmetrie da für $v \sqsubseteq w$ und $w \sqsubseteq v$ entweder $v = w$ oder es gibt ein Pfad von $v$ nach $w$ und ein Pfad von $w$ nach $v$. Im zweiten Fall existiert aber auch ein Pfad von $v$ nach $v$ und damit würde $G$ ein Zyklus enthalten. Das widerspricht aber der Baumeigenschaft. Also muss $v = w$.
  - Transitivität da für $v \sqsubseteq w$ und $w \sqsubseteq x$ gibt es vier Optionen:
    - $v = w$ und $w = x$, dann auch $v = x$ und $v \sqsubseteq x$,
    - $v = w$ und es gibt ein Pfad von $w$ nach $x$, also auch einen von $v$ nach $x$ und damit $v \sqsubseteq x$,
    - es gibt ein Pfad von $v$ nach $w$ und $w = x$, also auch einen von $v$ nach $x$ und damit $v \sqsubseteq x$,
    - es gibt ein Pfad von $v$ nach $w$ und einen von $w$ nach $x$, also auch einen von $v$ nach $x$ und damit $v \sqsubseteq x$.

  Somit ist $\sqsubseteq$ für $G$ als Baum eine partielle Ordnung.
- Wenn $G$ eine (einfach verkettete) Liste ist, dann gilt für $\sqsubseteq$ Reflexivität, Antisymmetrie und Transitivität da $G$ als Liste auch ein Baum ist. Es gilt aber weiterhin, dass $\sqsubseteq$ trichotom ist, da für beliebige Knoten $v$ und $w$ stets entweder ein Pfad von $v$ nach $w$, von $w$ nach $v$ oder $v = w$ gilt. Somit ist auch $w \sqsubset v$, $v \sqsubset w$ oder $v = w$.

  Somit ist $\sqsubseteq$ eine totale Ordnung.

  **Hinweis:**
  - Man könnte $G$ auch als doppelt verkettete Liste nehmen. Dabei käme man dann auf eine Äquivalenzrelation für $\sqsubseteq$.

**c)** Tatsächlich können wir jeden Sortieralgorithmus stabil machen, wenn wir dafür weitere Speicherkomplexität erlauben. Dafür speichern wir neben dem Schlüssel key[i] eines jeden Objekts i seine derzeitige Position pos[i] im Array. Zum Sortieren nutzen wir nun nicht die Ordnung die auf key definiert wurde, sondern die lexikographische Ordnung, d.h. i $\sqsubseteq$ j genau dann wenn key[i] $<$ key[j] $\vee$ (key[i] $=$ key[j] $\wedge$ pos[i] $\leq$ pos[j].

Dies garantiert wieder die Stabilität, da zwei Objekte i und j mit gleichem Schlüssel dann nach ihrer ehemaligen Position geordnet werden müssen, um sortiert zu bleiben.

Da wir jedoch weiteren Speicherbedarf haben, würde ein Algorithmus die in-place Eigenschaft verlieren, sowie eine Steigerung der Laufzeit zum Initialisieren und Auslesen des Arrays pos.

## Aufgabe 5 (Sortieralgorithmen): 3 + 3 + 3 = 9 Punkte

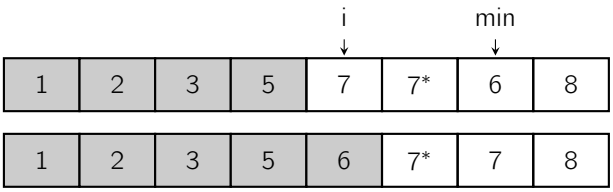
Welche der folgenden Sortieralgorithmen sind stabil? Begründen Sie Ihre Antwort für jeden Sortieralgorithmus.

- Bubblesort
- Selectionsort
- Insertionsort

### Lösung

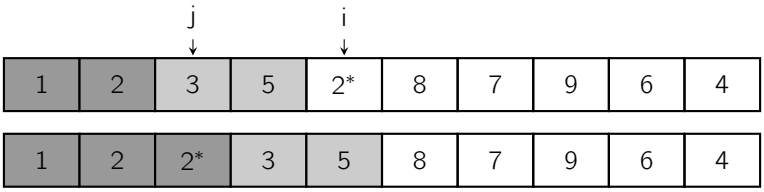
**Bubblesort** ist stabil. Bubblesort vertauscht immer nur direkt benachbarte Elemente, jedoch nur dann, wenn das linke Element **echt größer** ist, also insbesondere dann nicht, wenn beide Elemente gleich sind. Es wird also nie die Reihenfolge gleicher Elemente vertauscht, was Stabilität bedeutet. Würde Bubblesort benachbarte Elemente $n$ und $m$ tauschen, wenn bereits $n \geq m$ statt nur $n > m$ gilt, dann wäre er nicht stabil.

**Selectionsort** ist nicht stabil. Selectionsort arbeitet mit einem sortierten Bereich (unten grau) und einem unsortierten Bereich (unten weiß). In jedem Schritt wird das kleinste Element im unsortierten Bereich (unten min) mit dem ersten Element im unsortierten Bereich (unten i) vertauscht. Falls das Element an der Stelle i mehrfach im Array vorkommt, so wird möglicherweise bei dieser Vertauschung die Reihenfolge gleicher Elemente verändert (siehe 7 und 7*).




**Insertionsort** ist stabil. Insertionsort zieht das erste Element (unten i) aus dem unsortierten Bereich (unten weiß) soweit nach vorne in den sortierten Bereich (unten hell/dunkel grau), dass es im sortierten Bereich an der richtigen Stelle steht (unten j). Dazu wird zunächst die Stelle j gesucht, indem von i ausgehen solange j dekrementiert wird bis das Element direkt links von j nicht mehr **echt kleiner** als das Element an der Stelle i ist. Anschließend wird das Element an der Stelle i an die Stelle j getauscht und der hellgraue Bereich um eins nach rechts verschoben.

Auch hier ist wieder das $<$ für die Stabilität wichtig. Wäre es $\leq$ statt $<$ dann wäre Insertionsort nicht stabil.




## Aufgabe 6 (Programmierung in Python - Dynamische Programmierung + Sortieren):

**10 + 10 = 20 Punkte**

Bearbeiten Sie die Python Programmieraufgaben. In dieser praktischen Aufgabe werden Sie sich mit dem Konzept der dynamischen Programmierung näher vertraut machen. Außerdem werden Sie einfache Sortierverfahren implementieren. Diese Aufgabe dient dazu einige Konzepte der Vorlesung zu wiederholen.
Zum Bearbeiten der Programmieraufgabe können Sie einfach den Anweisungen des Notebooks *blatt04-python.ipynb* folgen. Das Notebook steht in der .zip-Datei zum Übungsblatt im Lernraum zur Vergügung.
Ihre Implementierung soll immer nach dem # *YOUR CODE HERE* Statement kommen. Ändern Sie keine weiteren Zellen.
Laden Sie spätestens bis zur Deadline dieses Übungsblatts auch Ihre Lösung der Programmieraufgabe im Lernraum hoch. Die Lösung des Übungsblatts und die Lösung der Programmieraufgabe muss im Lernraum an derselben Stelle hochgeladen werden. Die Lösung des Übungsblatts muss dazu als .pdf-Datei hochgeladen werden. Die Lösung der Programmieraufgabe muss als .ipynb-Datei hochgeladen werden.

**Übersicht der zu bearbeitenden Aufgaben:**

**a)** Dynamische Programmierung

- Matrix-Multiplikation

**b)** Sortierverfahren

- Selection-Sort
- Insertion-Sort
- Vergleich von Operationen

**Lösung**

Die Lösung der Programmieraufgaben finden Sie im Lernraum. Die Datei trägt den Namen *blatt02-python-solution.ipynb*.