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

Übung 4

Tutoraufgabe 1 (Ordnungen und Sortieren):

a) Wir definieren die Relationen $\sqsubset_2$, $\equiv_2$ und $\sqsubseteq$ auf natürlichen Zahlen als

$$\begin{aligned} n \sqsubset_2 m \quad &\text{genau dann wenn} \quad n \bmod 2 < m \bmod 2 \,, \\ n \equiv_2 m \quad &\text{genau dann wenn} \quad n \bmod 2 = m \bmod 2 \,, \\ n \sqsubseteq m \quad &\text{genau dann wenn} \quad n \sqsubset_2 m \quad \text{oder} \quad n \equiv_2 m \text{ und } n \leq m \end{aligned}$$

Wobei $n \bmod 2$ den Rest nach Division von n durch 2 angibt.

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

b) Gegeben folgender Sortieralgorithmus der das Array a sortiert:

```python
from random import shuffle

def is_sorted(a):
    i = 1
    while i < len(a):
        if a[i - 1] > a[i]:
            return False
        i += 1
    return True

def sort(data) -> list:
    """Shuffle data until sorted."""
    while not is_sorted(data):
        shuffle(data)
    return data
```

Ist dieser Sortieralgorithmus sinnvoll?

c) Die Laufzeiten der Sortieralgorithmen Bubblesort, Insertionsort und Selectionsort sind bereits im Bestcase sehr unterschiedlich.

Ist es möglich, die Bestcase Laufzeiten aller drei auf einfache Weise zu optimieren?

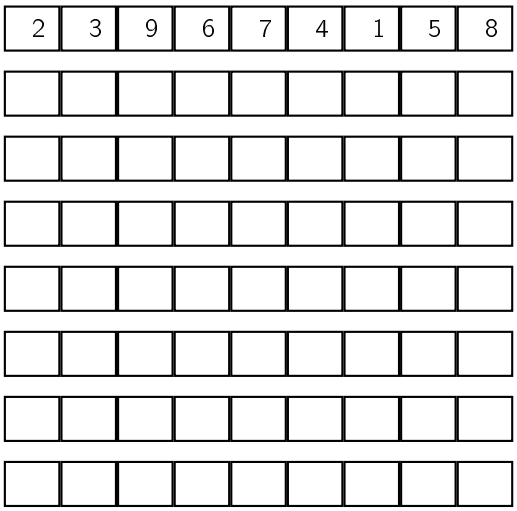
Falls ja, welche Konsequenzen hat dies für die Averagecase und Worstcase Laufzeit?

Tutoraufgabe 2 (Memoization):

Für eine Biologie Präsentation wollen wir eine Menge von Affen als Beispiel heran nehmen, die zwar möglichst berühmt sind, aber auch weit über das Spektrum an Affen verstreut ist.
Wir nehmen dabei an, dass Affen in einem Baum $T = (A, E)$ strukturiert sind, wobei die Affenart a' direkter Nachfahre von a ist, falls $(a, a') \in E$ gilt. Wir suchen nun eine Teilmenge $B \subseteq A$ sodass keine Affenart in B direkter Nachfahre einer anderen Affenart von B ist, oder formal für alle $a, a' \in B$ gilt weder $(a, a') \in E$ noch $(a', a) \in E$. Jede Affenart a hat einen bestimmten Berühmtheitswert $b(a)$. Der Berühmtheitswert einer Teilmenge $B \subseteq A$ ist gegeben als die Summe der Berühmtheitswerte seiner Elemente, also $b(B) = \sum_{a \in B} b(a)$.
Nutzen Sie Memoization um einen Algorithmus zu entwerfen, der eine Menge $B \subseteq A$ findet, sodass keine Affenart in B direkter Nachfahre einer anderen Affenart von B ist und sodass $b(B)$ maximiert wird.

Tutoraufgabe 3 (Selectionsort):

Sortieren Sie das folgende Array mithilfe von Selectionsort. Geben Sie dazu das Array nach jeder Swap-Operation an.



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?

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

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

- Bubblesort
- Selectionsort
- Insertionsort

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 *blatt02-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