Prof. Dr. Elmar Schömer, Götz Schwandtner, Tobias Baumann

# 9. Übung (Probeklausur)

**Aufgabe 1:** Sortieren in Linearzeit (2 + 1 Punkte)

Gegeben sei eine Folge $a_1, a_2, ..., a_n$ von $n$ natürlichen Zahlen, wobei für jedes Folgenglied gilt: $a_i \in \{0, 1, 2, ..., 9\}$.

(a) Geben Sie im Pseudocode einen Algorithmus an, der die Folge sortiert und dabei nur $O(n)$ Zeit benötigt.

(b) Begründen Sie die lineare Laufzeit Ihres Algorithmus.

**Aufgabe 2:** Divide & Conquer-Anwendung (3 + 1 Punkte)

Gegeben ist eine Folge $a_1, \ldots, a_n$ von natürlichen Zahlen, für die es ein $j$ mit $1 < j < n$ gibt, so dass $a_1 < a_2 < \cdots < a_j$ und $a_j > a_{j+1} > \cdots > a_n$.

(a) Geben Sie einen sublinearen Algorithmus an (also Laufzeit $o(n)$, asymptotisch schneller als $n$), der zur Eingabe $a_1, \ldots, a_n$ dieses $j$ bestimmt und begründen Sie kurz die Korrektheit.

**Hinweis:** Lokaler Monotonietest für Divide & Conquer.

(b) Analysieren Sie die Laufzeit Ihres Algorithmus.

**Aufgabe 3:** Amortisierte Analyse (1 + 2 Punkte)

Eine Queue mit den Operationen **add** und **del** lässt sich auch mit zwei Stacks $S$ und $T$ implementieren. Die Operationen der Queue werden dann mit den Operationen der beiden Stacks wie folgt implementiert:

```
add(k):
  S.push(k)
```

```
del:
  if ( T.is_empty )
    while ( S.not_empty )
      T.push(S.pop)
  return(T.pop)
```

(a) Verdeutlichen Sie die Funktionsweise dieser Implementierung, indem Sie angeben, wie sich die beiden Stacks bei der Ausführung dieser Operationenfolge verändern:

**add**(1), **add**(2), **del**, **add**(3), **add**(4), **del**, **del**, **add**(5), **del**, **del**

(b) Wir nehmen an, dass **push** und **pop** jeweils den Aufwand 1 haben. Führen Sie mit der **Bankkonto-Methode** eine amortisierte Aufwandsanalyse der Operationen **add** und **del** durch, um zu zeigen, dass sie konstanten amortisierten Aufwand haben.

**Bitte wenden!**

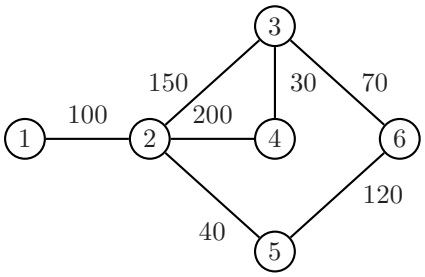
Hinweis: Bearbeitung und Wertung auch als Einzelabgabe möglich. Zeit für Bearbeitung: 180 Minuten

Abgabe der Übung: Montag 07.01.2008, 10h00;
Abgaben in Papierform in Raum 05-230

Johannes Gutenberg-Universität Mainz · Staudingerweg 9 · D-55099 Mainz
Internet: http://www.informatik.uni-mainz.de/lehre/dsea
E-Mail: schoemer@informatik.uni-mainz.de

**Aufgabe 4:** Anwendungsaufgabe: Kaufhausproblem (2 + 1 + 0,5 Punkte)

Der rechts abgebildete Graph repräsentiert das Netz von Filialen einer Kaufhauskette. Die Knoten sind die Filialen und die Kanten sind die Straßen, wobei an den Kanten die Längen der Straßen angegeben sind. In welcher Filiale soll die Kaufhauskette ihr Zentrallager anlegen? Der Ort soll so gewählt werden, dass die weiteste Wegstrecke zwischen Filialen und Zentrallager möglichst kurz ist.




(a) Entwerfen Sie einen Algorithmus zur Lösung dieses Problems und geben Sie eine kurze Begründung der Korrektheit an!

(b) Welche Laufzeit hat Ihr Algorithmus?

(c) Führen Sie den Algorithmus für das angegebene Beispiel aus!

**Aufgabe 5:** Wege in Graphen (1,5 + 1 + 1 Punkte)

Gegeben sei ein gerichteter Graph $G = (V, E)$ sowie zwei Knoten $s, t \in V$. Gesucht sind nun alle Kanten, die sich auf Wegen von $s$ nach $t$ befinden.

(a) Geben Sie einen Algorithmus an, der in Zeit $O(|V| + |E|)$ alle Kanten ausgibt, die sich auf Wegen von $s$ nach $t$ in $G$ befinden!

(b) Begründen Sie kurz die Korrektheit Ihres Algorithmus!

(c) Weisen Sie die Einhaltung der Laufzeitschranke $O(|V| + |E|)$ nach!

**Aufgabe 6:** Topologische Sortierung (1 + 1 + 1 Punkte)

Wir betrachten einen Graphenalgorithmus, der für einen gerichteten, zyklenfreien Graphen (DAG) $G = (V, E)$ wie folgt vorgeht:
Bestimme einen Knoten $v$ mit Eingrad 0, gebe $v$ aus und entferne $v$ aus $G$. Wiederhole das solange, bis alle Knoten ausgegeben wurden.

(a) Zeigen Sie, dass für die Ausgabe $v_1, v_2, \ldots, v_n$ des Algorithmus folgende Eigenschaft gilt:

$$i < j \quad \Rightarrow \quad (v_j, v_i) \notin E$$

(b) Geben Sie eine Pseudocode-Implementierung dieses Algorithmus an, die als Gesamtlaufzeit $O(|V| + |E|)$ hat.

(c) Analysieren Sie die Laufzeit Ihres Algorithmus und weisen Sie die Einhaltung der vorgegebenen Laufzeitschranke nach.