Überblick Blatt 3: Abstrakte Datentypen

Aufgabe 3.1 Bei Laufzeitabschätzung von Algorithmen geht man davon aus, dass alle "eingebauten" Konstrukte (wie Addition, Zuweisung usw.) eine konstante Zeit benötigen. Die Konstante selbst kennen wir nicht, sie ist typischerweise auch stark von der Hardware abhängig. Die Laufzeit von Prozeduren hängt dagegen i.a. von den Argumenten ab.

1. Bestimmen Sie Anzahl der Operationen, die der folgenden Algorithmus ausführt:

Algorithm 1 Quersumme von $A[1..n]$

```
1: x = 0
2: for i = 1 to n do
3:     x = x + A[i]
4: end for
5: return x
```

2. Bestimmen Sie Anzahl der Operationen, die der folgenden Algorithmus ausführt:

Algorithm 2 Alg. 1

```
1: for i = 1 to n do
2:     A[i] = i
3: end for
4: for i = 1 to n do
5:     C[i] = 0
6:     for j = n downto 1 do
7:         if A[j] > C[i] then
8:             C[i] = A[j]
9:         end if
10:    end for
11: end for
12: return C
```

3. Bestimmen Sie Anzahl der Operationen, die der folgenden Algorithmus ausführt:

Algorithm 3 Matrixmultiplikation

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         C[i][j] = 0
4:         for k = 1 to n do
5:             C[i][j] = A[i][k] * B[k][j]
6:         end for
7:     end for
8: end for
9: return C
```

4. Bestimmen Sie Anzahl der Operationen, die der folgenden Algorithmus ausführt:

Algorithm 4 Alg. Beispiel

```
1: for i = 1 to n do
2:     for j = i downto 1 do
3:         x = x + A[i][j]
4:     end for
5: end for
6: return x
```

Aufgabe 3.2 Die O-Notation.

1. Zeigen Sie: $15n^2 \in O(n^3)$.
2. Zeigen Sie: $\frac{1}{2}n^3 \notin O(n^2)$.
3. Betrachte $g(n) = 2n^2 + 3$.

   Geben Sie eine Funktion $f_1 : \mathbb{N} \to \mathbb{N}$ an, für die $f_1 \in O(g)$ und $f_1(n) < g(n)$ für alle n ab einem n_0 gilt.

   Geben Sie eine Funktion $f_2 : \mathbb{N} \to \mathbb{N}$ an, für die $f_2 \in O(g)$ und $f_2(n) > g(n)$ für alle n ab einem n_0 gilt.

   (Beide Funktionen f_1 und f_2 liegen also in $O(g)$, aber $f_1(n)$ ist stets kleiner und $f_1(n)$ ist stets größer als $g(n)$.)
4. Wir betrachten Polynome mit natürlichzahligen Koeffizienten, d.h. Funktionen der Form $f(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$ mit $a_i \in \mathbb{N}$ und wenn $a_k \neq 0$. Das Polynom hat dann den Grad k.

   Zeigen Sie: Für zwei Polynome f und g mit gleichem Grad gilt $f \in \Theta(g)$.
5. Zeigen Sie: Sei $f(n) := \sum_{i=0}^{n} 2^i$. Es gilt $f \in O(2^n)$

Aufgabe 3.3 Daten(de)kompression (Teil 1)

Wir wollen Binärdaten komprimieren. Primäres Ziel ist eine hohe Kompressionsrate.

Prinzipiell wollen wir einen Algorithmus entwerfen, der mit Datenströmen operieren kann, d.h. wir können nicht in einem ersten Durchlauf alle Daten analysieren und in einem zweiten Durchlauf komprimieren. Stattdessen steht uns nur ein kurzes Initialisierungsfenster zur Verfügung. In erster Näherung soll Ihr Algorithmus also wie folgt vorgehen:

- In einem Initialisierungsfenster (einstellbarer, beschränkter Größe, z.B. 128 Bit) liest der Algorithmus zunächst Daten in einen Puffer.
- Aus den Daten in diesem Fenster werden dann die Kompressionsparameter für Ihren Algorithmus berechnet.
- Das Fenster und der weitere Strom wird dann mit diesen Parametern komprimiert.
- Luxus: Es gibt eine Rückkopplung der weiteren Daten auf die Parameter. Aber Achtung: Dies erschwert vermutlich die Dekomprimierung.

Beachten Sie, dass die zu komprimieren Daten (u.a. auch die Testdaten in EMIL) nicht zufällige Daten sind. Es gibt eine innere Struktur. Evtl. kann Ihr Algorithmus diese nutzen...

1. Entwerfen Sie einen (De)Kompressionsalgorithmus und beschreiben Sie seine Arbeitsweise in Informatik-Prosa (also nicht durch Code).

   Sie sollen u.a. folgende Leitfragen beantworten:

   - Wie funktioniert das Verfahren im Prinzip?
   - Wie greifen Sie die Erzeugungsstruktur der Daten auf?
   - Welche Datenstrukturen sind nötig?
   - Welches Datenformat besitzt Ihre komprimierte Datei?
   - Wie spezifizieren Sie die Korrektheit?
   - Welche Testfälle sind sinnvoll?
   - Mit welcher Laufzeit ist zu rechnen?
   - ....

2. Erläutern Sie zudem, warum Sie vermuten, dass Ihr Algorithmus eine hohe Kompressionsrate erzeugt.

Eine Implementation ist (noch) nicht gefordert.