Der Algorithmenbegriff

Was ist eigentlich ein Algorithmus?

Diese Frage haben wir schon in der Theo.Inf. II Vorlesung zu beantworten versucht. Dabei haben wir gelernt, dass die These von Church allgemein anerkannt ist, nach der man davon ausgehen kann, dass wir wahlweise mit Turingmaschinen oder mit beliebigen (allgemeinen) Programmiersprachen arbeiten können.

Allen sogenannten „Konkretisierungen des Algorithmenbegriffs“ gemeinsam ist:

Ein Algorithmus ist eine schrittweise auszuführende Vorschrift, bei der jeder auszuführende Schritt tatsächlich in endlicher Zeit und auf eindeutig definierte Weise ausgeführt werden kann.

Zum Algorithmenbegriff

Wir gehen grundsätzlich davon aus, dass vor Beginn eines *Ablaufs* des gegebenen Algorithmus eine *Eingabe* bereitgestellt wird, auf die die ausführende *Maschine* während ihrer Arbeit zugreifen kann.

Die Eingabe wird häufig in Form eines Wortes über einem Alphabet angegeben. Es kann sich aber ebensogut um eine Zahl, einen Graph, eine Matrix, eine Kombination hiervon oder andere ähnliche Dinge handeln.

Grundsätzlich sollte eine Eingabe eine *Länge* $n \in \mathbb{N}$ haben.

Mit $|x|$ bezeichnen wir die Länge der Eingabe x.

Worst-case Analyse

Die worst-case Analyse haben wir bereits in der Vorlesung Theoretische Informatik II kennengelernt.

Wir rekapitulieren nochmal:

$$time_M(x) = \text{Anzahl der Schritte von } M \text{ bei Input } x$$

$$\text{d.h. } time_M : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$$

$$time_M(n) = \max_{|x| \leq n} \{time_M(x)\}$$

Analog werden Funktionen für nichtdeterministische und/oder platzbeschränkte Maschinen definiert:

$$ntime_M \qquad dspace_M \qquad nspace_M$$

Wir werden meist über *Algorithmen* sprechen, daher entsprechend $time_A, \ldots, nspace_A$.

Average-case Analyse

Um über *Durchschnittskomplexität* – oder *Komplexität im Mittel* – also über eine sogenannte Average-case Rechenzeit sprechen zu können, gehen wir üblicherweise davon aus, dass alle Eingaben der Länge n mit gleicher Wahrscheinlichkeit auftreten.

Sie sind also im mathematischen Sinn gleichverteilt.

Dann ist $av\text{-}time_A(n) = E[T_A(n)]$, wobei $T_A(n)$ bei gegebenem Algorithmus A die Zufallsvariable ist, die beim Zufallsexperiment

„Führe A auf zufälliger Eingabe der Länge n aus“

durch die Laufzeit gegeben ist. Das heißt:

$$av\text{-}time_A(n) = \frac{1}{N} \cdot \sum_{|x|=n} time_A(x)$$

mit $N = |\{x : \ |x| = n\}|$.

Beispiel: Maximumberechnung

Wir betrachten den folgenden Algorithmus, der das Maximum von n Eingaben (in einem Array $a[\,]$) ermittelt:

```
max := a[1];                          Aufwand c1
FOR i := 2 TO n DO
    IF a[i] > max THEN                Aufwand c2
        max := a[i];                  Aufwand c3
```

Wir haben vier Zeilen mit Aufwand c_1 für Zeile 1, Aufwand c_2 für Zeilen 2 und 3, sowie Aufwand c_3 für die letzte Zeile.

Im worst-case ergibt sich als Rechenzeit $c_1 + (n-1)(c_2 + c_3)$.

Average case? Die W'keit, Zeile 3 ausführen zu müssen, ist abhängig von i.
Man kann sich leicht überzeugen, dass sie genau $1/i$ ist. (Bitte überprüfen!)
Dadurch ergibt sich über die harmonische Reihe als erwartete Rechenzeit
für große n ungefähr $c_1 + (n-1)c_2 + ((\ln n) - 0.42)c_3$.