# Komplexitätstheorie

Yağmur Şentürk

23. Juni 2014

## Inhaltsverzeichnis



## 1 Einführung in die Komplexitätstheorie

Dieses Dokument gibt einen groben Überblick über die Komplexitätstheorie und wozu sie verwendet wird. Zuerst wird die O-Notation erklärt, um die genaue Funktionalität dieser Theorie zu verstehen. Anschließend werden die Komplexitätsklassen eingeführt.

Für detailliertere Erklärungen wird auf [1] und [2] verwiesen.

# 2 O-Kalkül

Um herauszufinden, wie viel Zeit ein Algorithmus[1] braucht, wird in der Informatik der O-Kalkül verwendet. Diese Zeit hängt von der Größe des Eingabewertes für den Algorithmus ab. Da sich bei jedem Rechner verschieden lange Laufzeiten herausstellen würden, schaut man sich die Anzahl der Schritte an, die ein Algorithmus ausführt.

Man nehme an, dass $n$ die Größe der Eingabe ist. Die Schrittzahl des Algorithmus beschreibt sich als $f(n)$ für die Funktion $f$. Damit man das Algorithmus in eine Schwierigkeitsklasse einteilen kann, braucht man nicht die genaue Anzahl der Schritte, sondern eine ungefähre Abschätzung. Dafür sucht man sich eine einfache Funktion $g(n)$, damit $f(n)$ maximal so schnell wächst wie $g(n)$. Typische Funktionen für $g(n)$ sind $g(n) = 2^n$, $g(n) = n^2$, $g(n) = n$, $g(n) = 1$ und $g(n) = log\ n$. Falls $f$ höchstens so schnell wächst wie $g$, dann sagt man, $f = O(g)$.

## 2.1 Definition

*Seien $e, f : N \to R$ Funktionen.*

$$e = O(f) :\Longleftrightarrow \exists c \in R_+\ \exists n_0 \in N\ \forall n \geq n_0\ \ \mid h(n) \mid \leq c \mid f(n) \mid$$

$e = O(f)$ ist keine Gleichung, nur eine Abkürzung für „h befindet sich in der Klasse O(f)“.

**Beispiel**: $2n + 1 = O(n)$

## 2.2 Rechenregeln für $O$

Folgende Rechenregeln gelten:

- $f = O(f)$
- $k \cdot O(f) = O(f)$ für Konstante k
- $O(O(f)) = O(f)$
- $O(f) \cdot O(g) = O(f \cdot g)$
- $O(f \cdot g) = \mid f \mid \cdot O(g)$
- $\mid f \mid \leq \mid g \mid \longrightarrow O(f) \subseteq O(g)$

[1] Rechenvorgang nach einem bestimmten Schema

# 3 Komplexitätsklassen

Zur Einordnung des Ressourcenbedarfs von Algorithmen werden Komplexitätsklassen verwendet. Die Laufzeit- und Speicherplatzkomplexität werden meist hierfür betrachtet. Mehr unter [3] und [4].

## 3.1 Laufzeitkomplexität

Die Laufzeitkomplexität ist die Anzahl der Schritte, die ein Algorithmus braucht. Interessant ist dabei nicht der Zeitaufwand eines Programms, sondern wie der Zeitbedarf bei mehreren Datenverarbeitungen wächst.

### 3.1.1 Definition

Sei $M$ eine Turingmaschine[2]. Die Abbildung $T : N \to N$ ist die Laufzeitkomplexität von $M$. $T(n)$ beschreibt die maximale Schrittanzahl von $M$ auf allen Eingaben der Länge $n$, die wie folgt aussieht:

$$T(n) := \max\{k \mid M \text{ hält bei Eingabe } x \text{ nach } k \text{ Schritten und } l(x) = n.\}$$

### 3.1.2 Beispiel

Eine Turingmaschine $M = (\{s, p, t, r\}, \{0, 1\}, \{\vdash, \sqcup, 0, 1\}, \vdash, \sqcup, \delta, s, t, r)$ wird durch ein Zustandsdiagramm[3] dargestellt (*Beispiel aus [1]*):

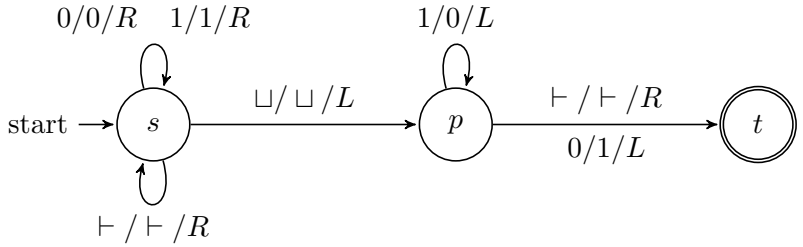


Abbildung 1: Zustandsdiagramm der Turingmaschine $M$

Für $T(1)$ ergibt sich 5, da die Turingmaschine 5 Schritte bis zur Terminierung benötigt.

[2]Die Turingmaschine ist ein sehr einfaches Modell eines Computers. Mehrere Informationen unter [1] und [2]

[3]Eine grafische Darstellung der Zustände und Übergänge

### 3.2 Speicherkomplexität

Die Speicherkomplexität ist der minimaler Speicherplatzbedarf eines Algorithmus. Das Interessante ist nicht der Speicherbedarf eines Programms, sondern wie der Speicheraufwand bei mehreren Datenverarbeitungen wächst.

#### 3.2.1 Definition

Sei $M$ eine k-bändige Turingmaschine. Die Abbildung $S : N \to N$ ist die Speicherplatzkomplexität von $M$. $S(n)$ beschreibt die maximale Bandfelderanzahl von $M$, die auf allen Eingaben der Länge $n$ von $M$ gelesen werden. Währenddessen werden nur die Zeichen auf den Arbeitsbändern angesehen.

## 4 Zusammenfassung

Die Komplexitätstheorie beschäftigt sich mit Zeit- und Platzbedarf von Algorithmen. Dabei wird eine ungefähre Abschätzung gesucht, mit der man den Ressourcenverbrauch der Berechnung abhängig von der Größe der Eingabe vorhersagen kann. Durch diese Abschätzung werden Algorithmen, die ähnlichen Aufwand bezüglich Laufzeit- oder Platzbedarf haben, zu Klassen eingeteilt. Für eine engmaschige Gruppierung der Algorithmen wird die O-Notation verwendet. Es gibt jedoch neben der O-Notation viele verschiedene Hierarchien von Aufwandsklassen.

## 5 Literaturverzeichnis


[1] Harald Zankl. “Diskrete Mathematik für Informatiker”, 2. Auflage, 2013. Skriptum zur Vorlesung, Universität Innsbruck, SS 2013.

[2] Erk Priese. “Theoretische Informatik”, 3. Auflage. Springer, 2001.

[3] Wikipedia. “Zeitkomplexität”, 2014. Zugriff: 17. April 2014

[4] Wikipedia. “Platzkomplexität”, 2014. Zugriff: 17. April 2014