Prof. Dr. André Schulz

Kurs 01686

Komplexitätstheorie

LESEPROBE

Fakultät für
Mathematik und Informatik

FernUniversität in Hagen

Kurseinheit 1

Grundlegende Komplexitätsklassen

1.1 Übersicht

Am Anfang diese Kurses lernen wir grundlegende Komplexitätsklassen kennen. Um überhaupt eine Komplexitätsklasse definieren zu können, müssen wir uns Gedanken über das zu Grunde liegende Berechnungsmodell machen. Des Weiteren sollten diese Definitionen auch robust und sinnvoll sein (Komplexitätsklassen machen nur Sinn wenn eine genügend große Menge an Problemen in Ihnen enthalten sind). Diese Vorarbeiten werden in dieser Kurseinheit durchgeführt. Dies geschieht ohne zu großen Formalismus, aber intuitiv sehr anschaulich. Die Anschaulichkeit birgt natürlich immer auch das Risiko, technische Details zu übergehen. Mit den Anmerkungen zu jeder Kurseinheit versuchen wir den Lesern des Textes eine Hilfestellung zum besseren Verständnis des Textes zu geben. Dies gilt generell für den gesamten Kurs.
Zum Ende der Lerneinheit werden dann einige wichtige Klassen vorgestellt, und deren Beziehung zueinander (soweit bekannt) beschrieben.

Lesen Sie im Basistext die Seiten 1–46.

1.2 Anmerkungen

S. 14: Nehmen wir für das Manhattan Beispiel der Einfachheit halber an, dass $m = n$. Ein kürzester Weg von $(0, 0)$ nach (n, n) besteht dann aus n Schritten nach rechts und n Schritten nach oben. In welcher Reihenfolge diese Schritte gemischt werden, ist dabei unerheblich. Wir können also sagen, dass wir die Positionen der n „nach-oben-Schritte“ innerhalb der $2n$ Schritte frei wählen können. Das ergibt also $\binom{2n}{n} = \frac{(2n)!}{n!n!}$ Möglichkeiten. Nutzt man die Stirlingformel $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$, so kann man die asymptotische Anzahl der kürzesten Wege mit $\Theta(4^n)$ beschränken. Hier ist die Abschätzung des Autors also sehr defensiv.

S. 16: Der Autor erwähnt viele andere Versionen von TSP. Hier wäre vor allen Dingen noch das *graphische TSP* zu ergänzen. Im graphischen TSP wird als Metrik die kürzeste-Pfad-Metrik eines ungerichteten Graphen benutzt. Seit 2011 wurden Fortschritte zur Approximierbarkeit von TSP vor allen Dingen in der Variante des graphischen TSP erzielt.

S. 19, unten Konjunktive Normalform wird häufig mit CNF abgekürzt (entsprechend DNF bei disjunktiver Normalform). Bei CNF mit 3 Literalen pro Klausel sagt man dann auch 3CNF.

S. 21: Die Registermaschine ist in der Tat das Referenzmodell für die meisten aktuellen (!) Betrachtungen in der Komplexitätstheorie. Sehr viele klassische Ergebnisse beruhen aber auf dem Modell der Turingmaschine, welches ebenfalls im Basistext erklärt wird. Ich möchte aber nicht verschweigen, dass es durchaus markante Unterschiede zu aktuellen Prozessoren gibt. So kann in einer Registermaschine in jedem Register eine beliebig große natürliche Zahl stehen. Diese unnatürliche Annahme wird durch das logarithmische Kostenmaß (definiert auf der nächsten Seite) wieder relativiert. Trotzdem gibt es auch Modelle, die die Wortbreite des Prozessors (und damit auch die maximale Größe einer Zahl im Register) berücksichtigen. Ein solches Modell ist die *word-RAM*. Mit ihr kann man auch ein kleines Maß an Parallelität in den Rechenschritten realisieren und somit einige Probleme (wie das Sortieren) schneller lösen.

S. 23, unten: Bei der Laufzeit vom Dijkstra-Algorithmus bezieht sich das n auf die Anzahl der Knoten und m auf die Anzahl der Kanten des Graphen. Diese Konvention ist für Graphenprobleme standard.

S. 26, oben: Beachten Sie, dass die groß-O Notation *Mengen* von Funktionen definiert. Man sollte also eigentlich schreiben $g(n) \in O(f(n))$ und nicht $g(n) = O(f(n))$. Die zweite Schreibweise hat sich aber in der Theoretischen Informatik eingebürgert und ist (trotz nicht korrekter Benutzung des Formalismus) etabliert und zum Standard geworden.

S. 29, unten: Vergegenwärtigen sie sich, dass das Polynom eines Polynoms (also die Komposition zweier Polynome) wieder ein Polynom ist.

S. 30, Def. 3.1.1: Die Klasse P und alle anderen in diesem Abschnitt definierten Klassen enthalten nur Entscheidungsprobleme. Dies wurde in der Definition unterschlagen. (Die abgeleitenen Optimierungs- und Auswertungsprobleme der Probleme aus P sind aber äquivalent, dazu in Kapitel 4.2 mehr.)

S. 30, mitte Das kleine Omega ω bezeichnet eine asymptotisch scharfe untere Schranke. Schlagen Sie im Anhang A.1. nach und lesen sie die Ausführungen zur Groß-O Notation.

S. 31, mitte: Startet der Algorithmus neu, bekommt er eine neue Sequenz von Zufallsbits. Dies wird zum Teil in der Praxis anders umgesetzt. Hier liefert der Pseudozufallszahlengenerator bei einigen Programmiersprachen immer die gleich Sequenz an Zufallsbits, wenn man vorher nicht den *seed* ändert.

S. 31, unten: Der Autor spricht hier und im weiterem Verlauf von der *durchschnittlichen* Rechenzeit. Konkret meint er damit den *Erwartungswert* der Rechenzeit, wenn man als Zufallsvariable die Zufallsbits betrachtet. Da sich das durch das ganze Buch zieht, behalten Sie das bitte stets im Hinterkopf.

S. 33, unten: Die co- Konvention ist nur für Klassen von Entscheidungsproblemen definiert. In diesem Fall wird die Komplexitätsklasse formal als eine Sprachfamilie (Kodierung der Ja-Instanzen der Probleme) beschrieben. Wenn $\mathcal{L}$ also eine Sprachklasse ist, dann ist co-$\mathcal{L}$ definiert als $\{\overline{L} \mid L \in \mathcal{L}\}$. Auf der Problemebene heißt dies nichts anderes, als das die Ja- und die Nein-Antworten vertauscht sind.

Beachten Sie, dass in der Regel co-$\mathcal{L} \neq \overline{\mathcal{L}}$. Zum Beispiel sei $\mathcal{L} = \{\{\mathsf{a}\}\}$, dann ist co-$\mathcal{L} = \{\{w \mid w \neq \mathsf{a}\}\}$, jedoch $\overline{\mathcal{L}} = \{L \mid L \neq \{\mathsf{a}\}\}$.

S. 36, unten: Der Autor spricht von „nicht schnell gegen 1 wachsenden“ Funktionen $\varepsilon(n)$, damit meint er Funktionen die durch $1 - 1/p(n)$, für $p(n)$ Polynom, beschränkt sind (siehe Thm. 3.3.2). Aus den ähnlichen Gründen sollte man aber auch statt $\varepsilon(n) = 0$ lieber $\varepsilon(n) \geq 2^{-p(n)}$ fordern.

S. 38, Zeile 4: Hier wurde nach dem Einsetzen von $t(n)$ für die weiteren Umformungen eine 1 addiert und subtrahiert.

S. 39, Thm. 3.3.8: Achtung! Beim Diagramm handelt es sich nicht notwendiger Weise um echte Teilmengen. In den meisten Fällen ist nicht bekannt, ob es sich um echte Teilmengen handelt.

S. 39, Beweis Thm. 3.3.8: Wir können in einem Algorithmus für ein Problem aus BPP einfach statt einen ? ein beliebiges Ergebnis ausgeben.

Lernziele

Nach dem Studium dieser Lerneinheit sollten Sie in der Lage sein

- den Begriff **Algorithmus** definieren zu können,
- zu erklären, wie man die **Rechenzeit** eines Algorithmus misst,
- die **Erweiterte Churche These** wiederzugeben,
- die Rechenmodelle **Registermaschine** und **Turingmaschine** zu beschreiben,
- wichtige Probleme in der Komplexitätstheorie aufzuzählen (auf Seite 15 gelistet),
- die Komplexitätsklassen im Allgemeinen und die Klassen P, ZPP, BPP, RP, coRP, NP, co-NP, PP im Besonderen zu definieren,
- Beziehungen zwischen den oben genannten Komplexitätsklassen zu beweisen.