Komplexität

Wie kann man schnell zur Lösung eines lösbaren Problems kommen?

Was bedeutet Komplexitätsklasse?

Eine Komplexitätsklasse ist eine Gruppe von Problemen (Algorithmen), die sich bezüglich Laufzeit oder Platzbedarf ähnlich verhalten. Sie wird durch eine obere Schranke für den Bedarf einer bestimmten Ressource (z.B. Anzahl der notwendigen Berechnungsschritte zur Lösung eines Problems) gebildet. Man betrachtet die Schritte, die ein Algorithmus ausführt. Wenn n die Größe der Eingabe ist, dann lässt sich die Schrittzahl des Algorithmus beispielsweise mithilfe einer Funktion T(n) beschreiben. Es interessiert aber meist nicht die genaue Schrittzahl, sondern eine ungefähre Abschätzung.

Was bedeutet eigentlich polynomial?

Ein Polynom ist eine (endliche) Summe von Vielfachen von Potenzen mit natürlichzahligen Exponenten einer Variablen, die meist mit x bezeichnet wird, zum Beispiel das Trinom $9x^3 + x^2 - 7^x$ oder das Binom $x^3 + y^3$ oder das Monom a.

In der Komplexitätstheorie bezeichnet man ein Problem als in **Polynomialzeit** lösbar, wenn die benötigte Rechenzeit einer deterministischen Rechenmaschine mit der Problemgröße nicht stärker als mit einer Polynomfunktion wächst. Die besondere Bedeutung der Polynomialzeit besteht darin, dass man sie als eine Grenze zwischen *praktisch lösbaren* und *praktisch nicht lösbaren* Problemen betrachtet. Der Aufwand für Probleme, die nicht in Polynomialzeit lösbar sind, wächst im Allgemeinen so schnell, dass schon relativ geringe Problemgrößen mit verfügbaren Rechnern nicht in überschaubaren Zeiträumen gelöst werden können. Hat ein Algorithmus exponentielle Laufzeit, ist er nicht-polynomial.

Was bedeutet deterministisch?

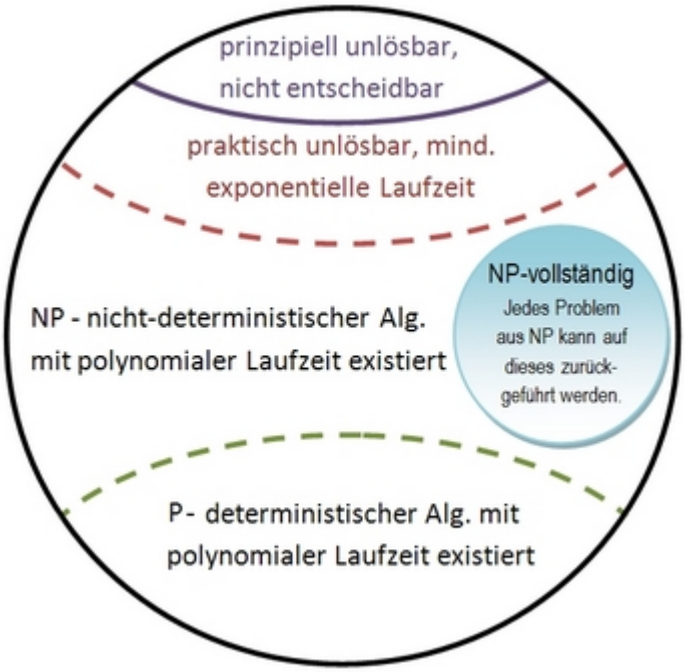
Ist die Rechnung auf einer Turingmaschine durch die Eingabe eindeutig festgelegt, so spricht man von einer deterministischen Maschine (DTM). Eine deterministische Turingmaschine befindet sich zu jedem gegebenen Zeitpunkt in genau einem Zustand. Eine nichtdeterministische Turingmaschine (NTM) kann sich in unendlich vielen Zuständen gleichzeitig befinden. Bei nichtdeterministisch rät man geschickt Lösungen.

Klasse P

(polynomial beschränkte Laufzeit)

Def. 1: Die Klasse P umfasst alle Entscheidungsprobleme, die sich in polynomial beschränkter Laufzeit (in Größenordnung der Eingabe), also effizient, mit deterministischen Turingmaschinen (DTM) lösen lassen.

Def. 2: Die Klasse P enthält alle (Entscheidungs-)Probleme mit einer deterministischen polynomialen Lösung. Beispiele: *Insertion-Sort, Quick-Sort, Merge-Sort, Finden des kürzesten Weges (zum Bsp. mit Dijkstra), Finden des minimalen Spannbaums in einem gewichteten Graphen*



Klasse NP (nicht-deterministisch, polynomial beschränkte Laufzeit)

Def. 1: Die Klasse NP enthält alle (Entscheidungs-)Probleme mit einer nicht-deterministischen polynomialen Lösung. Beispiele: *Problem des Handelsreisenden, Hamiltonkreisproblem, Rucksackproblem, Stundenplanproblem, Sudoku, …*

Def. 2: Die Klasse NP umfasst alle Entscheidungsprobleme, die sich in polynomial beschränkter Laufzeitmit nicht-deterministischen Turingmaschinen (NTM) lösen lassen.

Eine nicht-deterministische Turing-Maschine (NTM)hat die Möglichkeit, zu "raten", wie sie weiter vorgeht, d.h. welchen Befehl sie ausführt. Wenn man also auf den Determinismus verzichtet, hat die Turing-Maschine beim nächsten Rechenschritt eine Auswahl von Befehlen zur Verfügung.

Probleme, die das n im Exponenten haben, und auch die Probleme, die von der "ratenden" TM gelöst werden, gehören also zur Klasse NP.

Beispiel: Ein Passwort setzt sich aus Zeichen aus einem 26-Zeichen-Alphabet zusammen. Im worst case wären 26^n mögliche Folgen zu testen. Gäbe es einen Algorithmus für diesen Test, würde dieser ein exponentielles Laufzeitverhalten aufweisen. Der Glückliche findet die Lösung vielleicht beim ersten, zweiten oder zehnten Versuch. Dann könnte man eben nur sagen - nach linearer Laufzeit - dass man das Passwort mit diesem Ergebnis knacken kann.

Entscheidungsproblem: Ist die Berechnung der Lösung in polynomialer Zeit möglich? Das ist unklar.
Ob die gefundene Lösung funktioniert, kann man in polynomialer Laufzeit herausfinden.

NP-schwere Probleme

Ein Problem ist NP-schwer, wenn es so schwierig zu lösen ist, dass sich mit seiner Hilfe jedes Problem in NP lösen lässt. (Entscheidungsfrage mit Ja oder Nein als Antwort)

Probleme, deren Lösungen deterministisch in polynomialer Zeit überprüft werden können, können auf das Problem derart zurückgeführt werden, dass diese Rückführung auf einem deterministischen Rechner höchstens polynomiale Zeit in Anspruch nimmt. Man spricht von einer Polynomialzeitreduktion.

Ein NP-schweres Problem ist also mindestens so schwer wie das schwerste Problem in NP.

NP-vollständige Probleme

Ein NP-vollständiges Problem ist NP-schwer und liegt selbst in NP.

Liegt ein NP-schweres Problem selbst in NP, gehört es selbst zu den schwersten Problemen in NP, man spricht man von einem NP-vollständigen Problem.

NP-Vollständigkeit besagt, dass für das betrachtete Problem (bisher) kein polynomialer Algorithmus gefunden wurde. Die Lösung dieser Aufgabe erfordert also einen nicht vertretbaren Aufwand (unter Annahme, dass $P \neq NP$, wird das Problem nur deterministisch exponentiell lösbar sein.

Dazu sind mehr als 3000 Probleme aus der Informatik und aus anderen Gebieten bekannt. Alle Probleme in NP-vollständig lassen sich auf eines reduzieren.

Beispiele:
- *Cliquenproblem (Gibt es in einem Graphen eine Clique mit mehr als k-Knoten?)*
- *Knotenüberdeckungsproblem(Existiert in einem gegebenen einfachen Graphen und einer natürlichen Zahl k eine Knotenüberdeckung der Größe von höchstens k?)*
- *Hamiltonkreisproblem (Gibt es in einem Graphen einen Hamiltonischen Kreis?)*
- *Färbeproblem (Kann man die Knoten eines Graphen so einfärben, dass adjazente Knoten nicht die gleiche Farbe haben?)*
- *Rucksackproblem (Kann man Gegenstände von einem Wert größer als w in seinen Rucksack packen, ohne das Gewicht zu überschreiten?)*
- *Problem des Handelsreisenden (Gibt es die kürzeste Route durch alle Städte, wobei jede Stadt nur einmal besucht werden darf.)*

Problem P vs. PN

Unklar ist, ob die beiden Klassen P und NP identisch sind, und damit auch, ob die schwersten Probleme der Klasse NP ebenso effizient wie die der Komplexitätsklasse P gelöst werden können. Um den Begriff des „schwersten Problems in NP“ formal zu fassen, wurde der Begriff der NP-Schwere eingeführt. Ein Problem ist NP-schwer, wenn seine Lösung (in Polynomialzeit) die Lösung jedes anderen Problems in NP in polynomialer Zeit ermöglichen würde (unter Verwendung einer deterministischen Maschine; Polynomialzeitreduktion). Ein Problem, das in NP liegt und NP-schwer ist, heißt NP-vollständig.
Ein anschauliches Problem aus NP, für das nicht klar ist, ob es in P enthalten ist, ist das Rucksackproblem. Bei diesem Problem soll ein Behälter einer bestimmten Größe so mit vorgegebenen Gegenständen gefüllt werden, dass der Inhalt so wertvoll wie nur möglich ist.

Viele praxisrelevante Probleme sind NP-vollständig und eine Lösung würde auf vielen Gebieten helfen. Und wenn man weiß, dass es unmöglich ist, für ein Problem einen effizienten Algorithmus zu finden, braucht man keinen zu suchen.