Teil III

Komplexitätstheorie

Übersicht

Nichdeterministische Turingmaschine: Definition

Ein nichtdeterministische Turing-Maschine (NTM) ist definiert wie eine DTM, nur mit

- Übergangsfunktion $\delta : Q \times \Gamma \longrightarrow 2^{Q \times \Gamma \times \{L,R\}}$

Genau wie bei der DTM werden definiert:

- Konfigurationen
- die Übergangsrelation $\vdash_M$

Nachdem im vorherigen Kapitel das Thema war, welche Probleme sich überhaupt mit dem Rechner lösen lassen, ist die Frage in diesem letzten Kapitel, welche Probleme *effizient* lösbar sind. Leider gibt es hier keine definitiven negativen Antworten wie in der Berechnenbarkeitstheorie.

Die Komplexitätstheorie teilt Probleme in Klassen vergleichbarer Komplexität ein. Die zwei wichtigsten Komplexitätskassen P und NP werden hier vorgestellt.

Zur Definition der Komplexitätsklasse NP brauchen wir ein weiteres Maschinenmodell, die NTM. Diese verhält sich zur DTM genauso wie der NEA zum DEA, d.h. zu gegebenem Zustand q und gelesenem Bandsymbol a kann es mehrere Tripel aus Folgezustand, geschriebenem Symbol und Kopfbewegung geben. Die Übergangsfunktion liefert die Menge aller möglichen solchen Tripel.

Für eine Konfiguration α einer solchen NTM kann es deshalb mehrere Folgekonfiguration $\alpha_1, \ldots, \alpha_k$ mit $\alpha \vdash_M \alpha_i$ geben.

Berechnungsbaum

Der Berechnungsbaum einer NTM M bei Eingabe w ist ein Baum, beschriftet mit Konfigurationen von M, so dass:

- Wurzel ist $q_0 w$
- ist ν ein Knoten mit α, und sind $\alpha_1, \ldots, \alpha_k$ die Konfigurationen mit $\alpha \vdash_M \alpha_i$ für $i = 1, \ldots k$,

  dann hat ν genau k Kinder,

  die mit $\alpha_1, \ldots, \alpha_k$ beschriftet sind.

Jeder Pfad im Berechnungsbaum ist eine mögliche Berechnung von M, die entweder hält, oder unendlich ist.

Wie dir Berechnung einer DTM ist der Berechnungsbaum einer NTM M bei Eingabe w eindeutig bestimmt: jede Konfiguration hat genau so viele Nachfolger, wie es mögliche Folgekonfigurationen gibt, es wird also jede mögliche Berechnung ausgeführt.

Entscheidung durch eine NTM

Eine NTM M entscheidet die Sprache $L \subseteq \Sigma^*$, wenn

- für alle $w \in \Sigma^*$ hält jeder Pfad im Berechnungsbaum von M bei Eingabe w
- es gibt einen Pfad mit einer Endkonfiguration vqv' mit $q \in F$ genau dann, wenn $w \in L$ ist.

Theorem

Jede Sprache, die durch eine NTM entschieden wird, ist auch durch eine DTM entscheidbar.

Der Berechnungsbaum einer NTM bei einer Eingabe w kann endlich oder unendlich sein. Damit eine NTM eine Sprache entscheidet, wird aber verlangt, dass er für jede Eingabe endlich ist.

Das Kriterium für die Entscheidung ist dann analog wie beim NEA definiert: für ein Wort $w \in L$ muss es eine Berechnung, d.h. einen Pfad im Berechnungsbaum geben, der in einem Endzustand endet. Für ein Wort $w \notin L$ dagegen muss jede Berechnung in einer Konfiguration enden, deren Zustand kein Endzustand ist.

Zum Beweis des Satzes: eine NTM M, die eine Sprache L entscheidet, kann durch eine DTM simuliert werden. Da der Berechnungsbaum von M stets endlich ist, kann er von einer DTM (z.B. mittels depth-first-search) durchsucht werden.

Die Klasse P

Ein Problem A ist in der Klasse P, wenn

- es eine DTM M gibt, die A entscheidet,
- die Berechnung bei Eingabe w hält nach $p(|w|)$ Schritten, für ein Polynom $p()$.

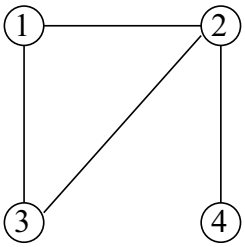
Seit den Arbeiten von *Jack Edmonds* (1965) wird P mit der Klasse der effizient lösbaren Probleme identifiziert.

Da eine DTM jedes andere vernünftige Berechnungsmodell simulieren kann, ist die Klasse P zu verstehen als die Klasse der Probleme, für die es Algorithmen gibt, deren Laufzeit durch ein Polynom in der Eingabegröße beschränkt ist.

Jack Edmonds war der erste, der Algorithmen als effizient bezeichnet hat, wenn ihre aufzeit polynomiell von der Eingabegröße abhängt, in den berühmten Arbeiten, in denen er die bekannten Algorithmen für das Matching- und das Maximalflussproblem entwickelt hat.

Codierung von Graphen

Ein Graph $G = (V, E)$ wird codiert als Adjazenzmatrix:



$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Repräsentation als String: 0110#1011#1100#0100

Andere mögliche Codierungen von Graphen – etwa durch Adjazenzlisten – führen zu leicht verschiedenen Laufzeiten, die sich aber nie um mehr als einen polynomiellen Faktor unterscheiden und daher für die Frage, ob ein Problem in P ist, irrelevant sind.

Probleme in P

Problem MINIMUM PATH

Instanz:	Graph $G = (V, E)$, Knoten s, t, $k \in \mathbb{N}$
Frage:	Gibt es Weg von s nach t der Länge $\leq k$?

Problem MAXIMUM MATCHING

Instanz:	Graph $G = (V, E)$, $k \in \mathbb{N}$
Frage:	Gibt es Matching M mit $\|M\| \geq k$?

Matching: $M \subseteq E$ mit $e_1 \cap e2 = \emptyset$ für alle $e_1, e_2 \in E$

Das Problem MINIMUM PATH, das etwa für Routenplanung relevant ist, wird z.B. durch den Algorithmus von Dijkstra, den Sie in *Algorithmen und Datenstrukturen* kennengelernt haben, in polynomieller Zeit gelöst.

Das Probem MAXIMUM MATCHING, für das Edmonds einen Algorithmus entwickelt hat, war eines der ersten Probleme, für die ein Verfahren bekannt war, dass es explizit in polynomieller Zeit löst.

Weitere Probleme in P sind alle solchen, für die Sie Algorithmen kennengelernt haben, deren Laufzeit polynomiell ist, wie etwa das Problem, einen minimalen Spannbaum in einem gewichteten Graphen zu finden.

Die Klasse NP

Ein Problem A ist in der Klasse NP, wenn

- es eine NTM M gibt, die A entscheidet,
- jede Berechnung bei Eingabe w hält nach $p(|w|)$ Schritten, für ein Polynom $p()$.

Ist P = NP ?

- Das zentrale offene Problem der theoretischen Informatik!
- Die Clay Foundation gibt einen Preis von $ 1.000.000 für die Lösung (Millenium Prize).

Die Laufzeit einer NTM wird also als die *Tiefe* des Berechnungsbaumes definiert, entsprechend der Idealvorstellung, dass die Maschine nur eine – nichtdeterministisch gewählte – Berechnung ausführt.

Dies ist natürlich kein realistisches Modell, erfasst aber genau die Kompexität zahlreicher wichtiger Probleme aus allen Bereichen der Informatik und anderer Fachgebiete.

Ein NP-Verfahren lässt sich in offensichtlicher Weise nur in exponentieller Zeit deterministisch simulieren, da das Durchsuchen eines Baumes der Tiefe d Zeit exponentiell in d benötigt.

Probleme in NP

Problem INDEPENDENT SET

Instanz:	Graph $G = (V, E)$, $k \in \mathbb{N}$
Frage:	Gibt es unabhängige Menge I mit $\lvert I \rvert \geq k$?

unabhängige Menge: $I \subseteq V$ mit $\{x, y\} \notin E$ für alle $x, y \in I$

Problem VERTEX COVER

Instanz:	Graph $G = (V, E)$, $k \in \mathbb{N}$
Frage:	Gibt es vertex cover U mit $\lvert U \rvert \leq k$?

Vertex cover: $U \subseteq V$ mit $e \cap U \neq \emptyset$ für alle $e \in E$

Auf dieser Folie sind zwei typische Beispiele von Probemen in NP.

Zur Motivation des Problems INDEPENDENT SET stellen Sie sich vor, dass die Knoten des Graphen irgendwelche Anforderungen repräsentieren, die erfüllt werden sollen, und eine Kante zwischen zwei Knoten besteht, wenn diese beiden Anforderungen im Konflikt stehen, also nicht gleichzeitig erfüllt werden können.

Eine unabhängige Menge ist dann entsprechend eine Menge von Anforderungen, die ale gleichzeitig erfüllt werden können.

Zur Motivation des Problems VERTEX COVER ist die Vorstellung nützlich, das der Graph ein Netzwerk repräsentiert, mit den Knoten als Verbindungsstellen und den Kanten als Leitungen. Ein vertex cover ist also eine Menge von Knoten, die mindestens ein Ende jeder Leitung enthält, so dass z.B. Monitore an jedem dieser Knoten das gesamte Netzwerk überwachen können.

Charakterisierung von NP

Ein Problem A ist in NP genau dann wenn es

- eine Relation R_A in P und
- ein Polynom $p()$ gibt, so dass

$$x \in A \iff \exists z : |z| \leq p(|x|) \ \& \ (x, z) \in R_A$$

Diese Charakterisierung der Klasse NP ist viel wichtiger als die eigentliche Definition: Probleme in NP sind die, für die es für die positiven Instanzen kurze und leicht zu überprüfende Zertifikate gibt.

Dies ist z.B: für das Problem der Fall, ob eine natürliche Zahl zusammengesetzt, also keine Primzahl ist. Diese Tatsache kann leicht durch Angabe der Primfaktoren zertifiziert werden, ist aber ohne Kenntnis dieser Faktoren schwierig zu beantworten.

Der Idee des Beweises der Charakterisierung ist einfach:

- Ist ein Problem a von der gegeben Art, kann eine NTM bei Eingabe x das Zertifikat z in $p(|x|)$ nichtdeterministischen Schritten raten, und dann in polynomieller Zeit überprüfen, ob $R_A(x, z)$ gilt.
- Ist umgekehrt ein Problem A in NP, dann wird es von einer NTM M in polynomieller Zeit gelöst. Ein Zertifikat für $x \in A$ ist dann eine Berechnung von M bei Eingabe x, die im Endzustand endet.