# Die Klasse P



Ein Problem $A$ ist in der Klasse P, wenn

- es eine DTM $M$ gibt, die $A$ entscheidet,
- die Berechnung vom $M$ 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 Laufzeit polynomiell von der Eingabegröße abhängt, in den berühmten Arbeiten, in denen er die bekannten Algorithmen für das Matching- und das Maximalflussproblem entwickelt hat.

Die Gleichsetzung von Effizienz mit polynomieller ist in mehrerer Hinsicht diskussionswürdig. Ist ein Algorithmus, der bei einer Eingabe der Größe $n$ eine Zeit der Größenordnung $n^{200}$ braucht, wirklich effizient? Dagegen würde ein anderer Algorithmus mit Laufzeit $1.000001^n$ als nicht effizient betrachtet.

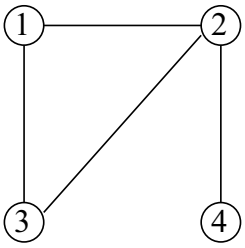
Auch wird nur die *Existenz* eines Algorithmus mit polynomieller Laufzeit gefordert. So gibt es in der Graphentheorie Probleme, von denen man weiß, dass sie in Zeit $O(n^3)$ gelöst werden können, es ist aber überhaupt kein Algorithmus, egal wie ineffizient, dafür bekannt.

Für diese Festlegung spricht aber, dass sie mathematisch robust ist, also z.B. nicht vom verwendeten Maschinenmodell abhängt, und dass die Erfahrung zeigt, dass Probleme in P meist auch tatsächlich effiziente Algorithmen haben.

# 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 $\lvert M \rvert \geq k$ ? |

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

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 NP, wenn es

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

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

Ein $z$ mit $(x, z) \in R_A$ ist Zertifikat für $x \in A$.

# P versus NP



Offenbar ist P $\subseteq$ NP.

Andererseits: Probleme in NP in exponentieller Zeit lösbar:

Ist $R_A$ in Zeit $q(|x|, |z|)$ entscheidbar,
und gilt $|z| \leq p(|x|)$ für Zertifikate $z$ für $x \in A$:

entscheide $A$ durch Testen aller möglichen Zertifikate

Zeit: $2^{p(|x|)} \cdot q(|x|, p(|x|))$

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).

# 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 Problemen 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 alle 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.

# Alternative Definition von NP



Definition durch nichtdeterministische Turing-Maschinen (NTM):

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 $p()$.

Die Laufzeit einer NTM wird also als die maximale Länge aller möglichen Berechnungen definiert, entsprechend der Idealvorstellung, dass die Maschine nur eine – nichtdeterministisch gewählte – Berechnung ausführt.

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

Der Idee des Beweises der Äquivalenz ist einfach:

- Ist ein Problem $A$ in NP, so 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.
- Wird umgekehrt ein Problem $A$ von einer NTM $M$ in polynomieller Zeit gelöst, so ist ein Zertifikat für $x \in A$ eine Berechnung von $M$ bei Eingabe $x$, die im Endzustand endet.

# Polynomielle Reduktion



Seien $A, B \subseteq \Sigma^*$.

Eine polynomielle Reduktion von $A$ auf $B$ ist eine Funktion $f : \Sigma^* \to \Sigma^*$ die in polynomieller Zeit berechenbar ist, mit der Eigenschaft:

$$w \in A \Leftrightarrow f(w) \in B \quad \text{für alle } w \in \Sigma^*$$

Notation: $A \leq_P B$ falls es eine polynomielle Reduktion von $A$ auf $B$ gibt.

Eigenschaft:

- Ist $B$ in P und $A \leq_P B$, dann ist auch $A$ in P.
- Ist $B$ in NP und $A \leq_P B$, dann ist auch $A$ in NP.

Der Begriff der polynomiellen Reduktion ist eine Verschärfung des Reduktionsbegriffs aus dem vorherigen Kapitel. Hier muss die Reduktionsfunktion nicht bloss berechenbar sein, sondern darüber hinaus in polynomieller Laufzeit berechnet werden.

Die Eigenschaften werden wie folgt begründet: sei $f$ eine Reduktion von $A$ auf $B$, die bei Input $w$ in Zeit $p(|w|)$ berechnet wird, und sei $B$ bei Input $v$ in Zeit $q(|v|)$ durch eine DTM $M$ entscheidbar, für Polynome $p()$ und $q()$.

Dann hat für Eingabe $w$ die Ausgabe $f(w)$ höchstens die Größe $|f(w)| \leq p(|w|)$, da die Maschine, die $f$ berechnet, in jedem Schritt höchstens ein Symbol schreibt.

Dann wird $f(w) \in B$ in Zeit $q(|f(w)|) \leq q(p(|w|))$ von $M$ entschieden, also ist die gesamte Zeit zur Entscheidung, ob $w \in A$ ist, $p(|w|) + q(p(|w|))$, ein Polynom in $|w|$.

Das Argument für die zweite Eigenschaft ist ähnlich.

# NP-vollständige Probleme



$A$ ist NP-schwer, falls $B \leq_P A$ für alle $B$ in NP ist.

$A$ ist NP-vollständig, falls $A$ in NP und NP-schwer ist.

**Theorem**

*Ist $A$ NP-vollständig, dann ist $A$ in $P$ genau dann, wenn $P = NP$.*

Ein Problem ist also NP-schwer, wenn es so schwierig zu lösen ist, dass sich mit seiner Hilfe jedes Problem in NP lösen lässt.

Ist es darüber hinaus selbst in NP, nennen wir es NP-vollständig. Die NP-vollständigen Probleme sind also die schwierigsten Probleme in NP.

Wir werden in den nächsten Wochen einige NP-vollständige Probleme kennen lernen. Tatsächlich sind tausende solche Probleme aus allen möglichen Bereichen der Informatik und anderer Gebiete bekannt. Einen effizienten Algorithmus, also einen mit polynomieller Laufzeit, für irgendeines dieser Probeme zu finden würde also ausreichen, das P-NP-Problem zu lösen (und damit 1 Million US-Dollar zu verdienen).