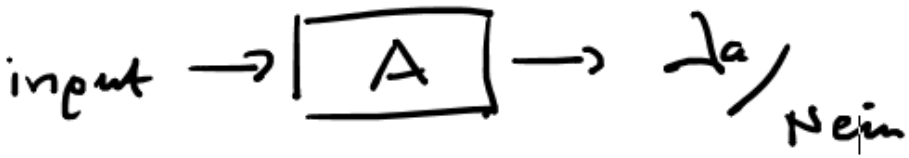
P = NP?

Informatik

Im Jahr 2000 veröffentlichte das Clay Mathematics Institute eine Liste mit sieben Problemen, auf deren Lösung jeweils ein Preisgeld von 1 Million Dollar ausgesetzt ist (Millenium-Probleme).

Die Frage P = NP? ist eines dieser Probleme.

Wir betrachten nur Entscheidungsprobleme, d.h. Probleme mit einer Antwort Ja oder Nein.



Entscheidungsprobleme:

- Gibt es in einem Graphen eine Verbindung von Knoten u nach Knoten v?
- Hat dieses Sudoku eine Lösung?
- Gegeben diese Schachstellung, kann Weiß sicher gewinnen?

P ist die Menge alle Entscheidungsprobleme, die sich in polynomialer Zeit lösen lassen.

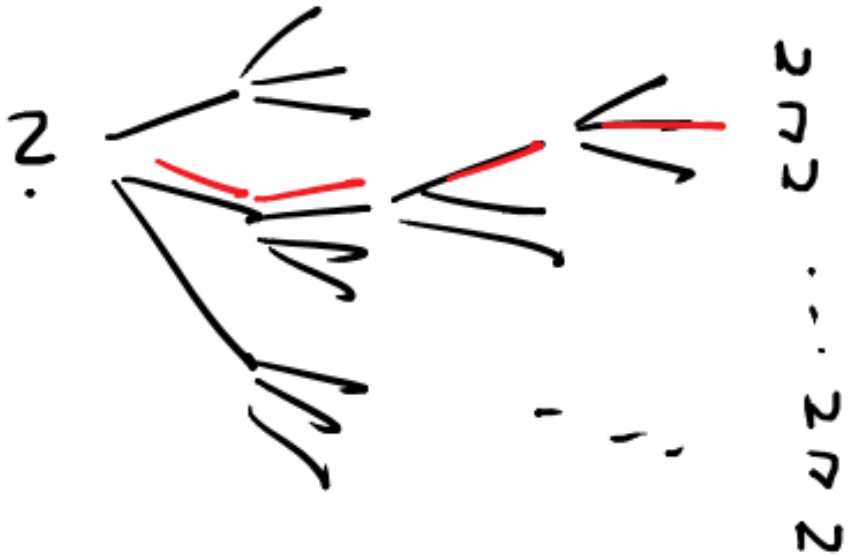
NP umfasst alle Entscheidungsprobleme, die sich mit einem “lucky algorithm” in polynomialer Zeit lösen lassen.

EXP umfasst alle Entscheidungsprobleme, die sich mit einem Algorithmus in exponentieller Zeit lösen lassen.

R umfasst alle Entscheidungsprobleme, die sich in endlicher Zeit lösen lassen.

"lucky algorithm": ein nicht-deterministisches Modell eines Algorithmus, der im Lauf seiner Tätigkeit immer wieder "guesses" machen kann, die ihn zur richtigen Antwort führen.

"lucky algorithm": ein nicht-deterministisches Modell eines Algorithmus, der im Lauf seiner Tätigkeit immer wieder "guesses" machen kann, die ihn zur richtigen Antwort führen.



Wenn es ein Ja gibt, dann findet der "lucky algorithm" garantiert einen Weg dorthin.

Genauer: der Algorithmus darf unter polynomial vielen Optionen wählen, eine Wahl dauert O(1).

Das SAT-Problem: Gegeben ein boolescher Ausdruck. Gibt es eine Belegung seiner Variablen, so dass er sich zu True auswertet?

Das SAT-Problem: Gegeben ein boolescher Ausdruck. Gibt es eine Belegung seiner Variablen, so dass er sich zu True auswertet?

Wir kennen keinen polynomialen Algorithmus, der das SAT-Problem löst. Aber es gibt einen "lucky Algorithm", der dieses Problem in polynomialer Zeit löst: Rate den Wert jeder Variablen, dann werte den booleschen Ausdruck aus.

```
rate x_1 = True oder False
rate x_2 = True oder False
...
if boolescher Ausdruck == True: return JA
else: return NEIN
```

Das SAT-Problem: Gegeben ein boolescher Ausdruck. Gibt es eine Belegung seiner Variablen, so dass er sich zu True auswertet?

Wir kennen keinen polynomialen Algorithmus, der das SAT-Problem löst. Aber es gibt einen "lucky Algorithm", der dieses Problem in polynomialer Zeit löst: Rate den Wert jeder Variablen, dann werte den booleschen Ausdruck aus.

```
rate x_1 = True oder False
rate x_2 = True oder False
...
if boolescher Ausdruck == True: return JA
else: return NEIN
```

Wir stellen uns folgende Reihenfolge vor: Raten am Anfang, dann überprüfen, ob das Geratene zu einem JA führt. Dadurch können wir uns das Ganze auch als Verifikations-Algorithmus vorstellen.

Ein Entscheidungsproblem ist in NP, wenn eine Lösung in polynomialer Zeit überprüft werden kann.

Ein Entscheidungsproblem ist in NP, wenn eine Lösung in polynomialer Zeit überprüft werden kann.

- Eine Zahl faktorisieren ist schwer. Zu überprüfen, ob eine Faktorisierung stimmt, ist leicht.

Ein Entscheidungsproblem ist in NP, wenn eine Lösung in polynomialer Zeit überprüft werden kann.

- Eine Zahl faktorisieren ist schwer. Zu überprüfen, ob eine Faktorisierung stimmt, ist leicht. $\Rightarrow$ Faktorisierung $\in$ NP
- Ein Sudoku lösen ist schwer. Zu überprüfen, ob eine Lösung gültig ist, ist leicht.

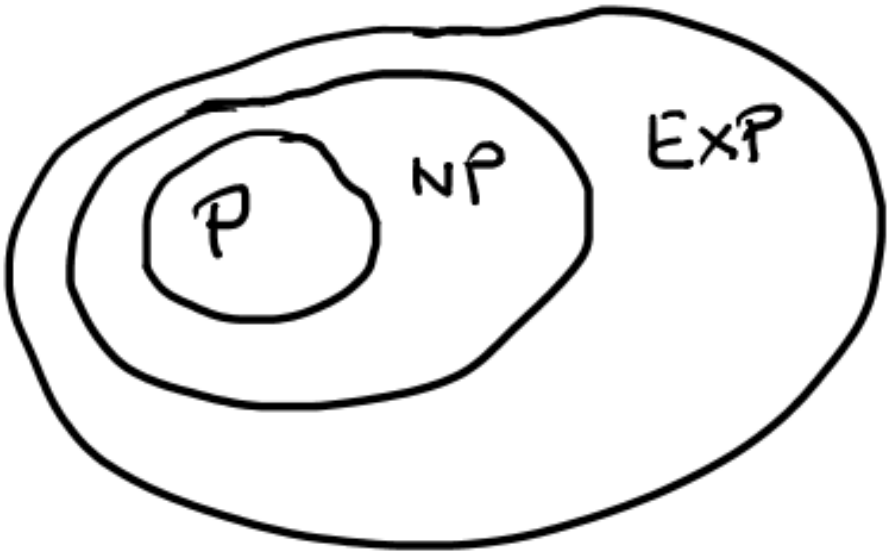
Ein Entscheidungsproblem ist in NP, wenn eine Lösung in polynomialer Zeit überprüft werden kann.

- Eine Zahl faktorisieren ist schwer. Zu überprüfen, ob eine Faktorisierung stimmt, ist leicht. $\Rightarrow$ Faktorisierung $\in$ NP
- Ein Sudoku lösen ist schwer. Zu überprüfen, ob eine Lösung gültig ist, ist leicht. $\Rightarrow$ Sudoku $\in$ NP
- Zu einer gegebenen Stellung einen Gewinnzug für Weiß im Schach zu finden, ist schwer. Zu Überprüfen, ob die gefundene Lösung richtig ist, ist auch schwer.

Ein Entscheidungsproblem ist in NP, wenn eine Lösung in polynomialer Zeit überprüft werden kann.

- Eine Zahl faktorisieren ist schwer. Zu überprüfen, ob eine Faktorisierung stimmt, ist leicht. $\Rightarrow$ Faktorisierung $\in$ NP
- Ein Sudoku lösen ist schwer. Zu überprüfen, ob eine Lösung gültig ist, ist leicht. $\Rightarrow$ Sudoku $\in$ NP
- Zu einer gegebenen Stellung einen Gewinnzug für Weiß im Schach zu finden, ist schwer. Zu Überprüfen, ob die gefundene Lösung richtig ist, ist auch schwer. $\Rightarrow$ Schach $\notin$ NP

Stand heute:



$P \subseteq NP \subseteq EXP$

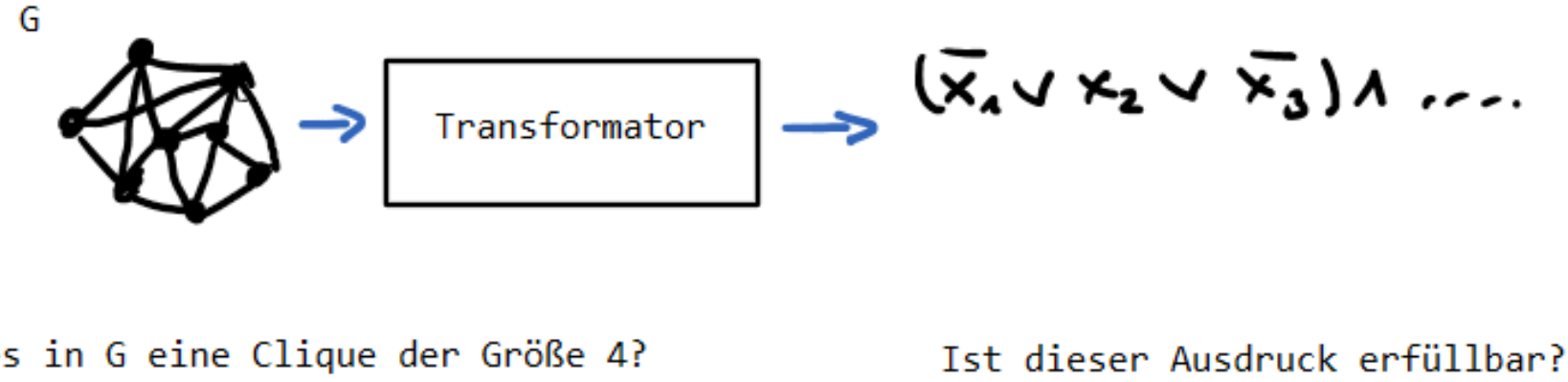
$P \subset EXP$ ✓

$P = NP$? $NP = EXP$?

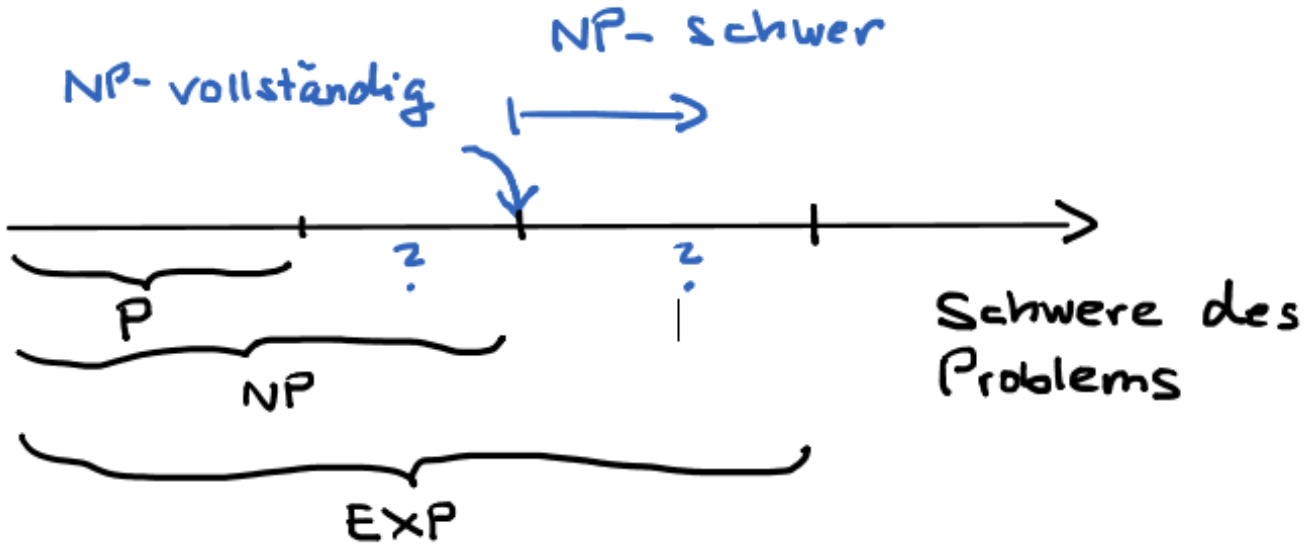
Stephen Cook, 1971: SAT ist NP-vollständig, d.h;

1. SAT ist in NP
2. Jedes andere Problem in NP kann auf SAT reduziert werden (SAT ist NP-schwer).

Beispiel: das Clique-Problem ist in NP (Gibt es in einem Graphen G eine Clique der Größe k?). Jedes Clique-Problem kann transformiert werden in ein SAT-Problem mit äquivalenter Antwort.



Es gibt auch eine Reduktion von SAT auf Clique, also ist Clique auch NP vollständig. Inzwischen wurde für über 1000 Probleme gezeigt, dass sie NP vollständig sind. Wenn für eines dieser Probleme ein polynomialer Algorithmus gefunden wird, sind auch alle anderen in polynomialer Zeit lösbar und es gilt $P = NP$.



Andere Formulierung: P = NP? ist die Frage, ob man Suchprobleme auch ohne Suchen lösen kann.

Kann man die Nadel im Heuhaufen finden, ohne im Heuhaufen zu suchen?

Andere Formulierung: P = NP? ist die Frage, ob man Suchprobleme auch ohne Suchen lösen kann.

Kann man die Nadel im Heuhaufen finden, ohne im Heuhaufen zu suchen? Vielleicht mit einem großen Magnet?

Kann man herausfinden, ob eine Zahl eine Primzahl ist, ohne nach ihren Faktoren zu suchen?

Andere Formulierung: P = NP? ist die Frage, ob man Suchprobleme auch ohne Suchen lösen kann.

Kann man die Nadel im Heuhaufen finden, ohne im Heuhaufen zu suchen? Vielleicht mit einem großen Magnet?

Kann man herausfinden, ob eine Zahl eine Primzahl ist, ohne nach ihren Faktoren zu suchen? Ja.

Andere Formulierung: P = NP? ist die Frage, ob man Suchprobleme auch ohne Suchen lösen kann.

Kann man die Nadel im Heuhaufen finden, ohne im Heuhaufen zu suchen? Vielleicht mit einem großen Magnet?

Kann man herausfinden, ob eine Zahl eine Primzahl ist, ohne nach ihren Faktoren zu suchen? Ja.

Ist p Primzahl und $a < p$, dann gilt: $a^{p-1} \% p = 1$. (Kleiner Fermatscher Satz). Darauf basiert ein Primzahltest, der in P ist (2002).