Algorithmen für Fortgeschrittene
Dozent: Prof. Dr. Helmut Alt

Vorlesung vom 1. Juni 2004

PSPACE

- umfasst NP (unbekannt, ob echte oder unechte Obermenge)
- Algorithmen dieser Klasse verbrauchen polynomiell viel Platz in Abhänigkeit von der Länge einer logarithmisch kodierten Eingabe

PSPACE -vollständig

- Definition analog zu NP-vollständig:

  $L \in$ PSPACE-vollständig, wenn

  1. $L \in$ PSPACE
  2. L ist PSPACE-schwer

  L ist PSPACE-schwer, wenn: $\forall\, L' \in \text{PSPACE} : L' \leq_P L$

- Beispiele für PSPACE-vollständig:

  ▶ Quantifizierte Bool'sche Formeln QBF
  - z.B.: $\forall x_1 \exists x_2 \forall x_3\ (x_1 \vee x_2) \wedge x_3$
  - QBF sind diejenigen quantifizierten Bool'schen Formeln, die wahr sind
  - QBF $\in$ PSPACE
  - QBF ist PSPACE-schwer
    (Beweis ähnlich dem der NP-Schwere von SAT)

  ▶ Verallgemeinertes Geographiespiel GEO
  - Ablauf Geographiespiel:
    · der Erste nennt einen geographischen Begriff

- der Nächste nennt einen noch nicht erwähnten geographischen Begriff, der mit dem Buchstaben beginnt, mit dem der vorherige Begriff endet
- verallgemeinertes GEO: Begriffe allen Genres sind erlaubt

– mathematischs Modell:

geg: gerichteter Graph $G = (V, E)$

Spiel:
- Spieler *A* setzt einen Stein auf einen Knoten
- Spieler *B* setzt auf einen Nachfolgeknoten

  ⋮
- verloren hat, wer nicht mehr setzen kann

Frage: hat der Spieler, der anfängt, eine *Gewinnstrategie*?

Gewinnstrategie:

$\exists$ Zug für Spieler *A*, so dass
$\forall$ Folgezüge von Spieler *B*
$\exists$ Zug für Spieler *A*, so dass

⋮

B verliert

– es gilt:

a) QBF $\leq_P$ GEO damit ist GEO PSPACE-schwer
b) GEO $\in$ PSPACE

Satz: Ist die Länge eines Spiels polynomiell durch die Größe der Eingabe beschränkt, so liegt das Spiel in PSPACE.

Dieses ist der Fall für: GEO, GOBANG
und nicht der Fall für: GO, Schach$_-$ (normales Schach)

Weil die folgenden Brettspiele in ihrer Spielfeldgröße beschränkt sind, wird die Abhänigkeit der Gewinnstrategie von einer (beliebig großen) Spielfeldgröße betrachtet:

► GOBANG

Variante des GO-Spiels, bei der die umkreisten Steine nicht entfernt, sondern in Steine eigener Farbe umgewandelt werden (von GO ist es unbekannt, ob es in PSPACE liegt)

► Schach$_+$

= Schach mit Zusatzregeln (von Schach$_-$ ist nicht bekannt, ob es in PSPACE liegt) :

wenn m die Größe der Eingabe ist, dann muss nach $p(m)$ Schritten ein Bauer gezogen oder eine Figur geschlagen werden

$\Rightarrow$ Länge des Spiels wird polynomiell

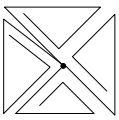
► Entfalten von Gestängen in der Ebene GEST

mit Drehgelenken verbundene Stangen (Stangen dürfen sich nicht kreuzen)

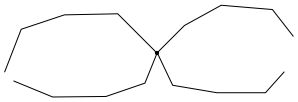
Frage: gibt es eine *stetige überkreuzungsfreie* Bewegung, die das Gestänge von einer Position in eine gegebene andere Position überführt?

Gestänge: Graph $G = (V,E)$, jeder Kante ist eine feste Länge zugeordnet

Position: Einbettung in die Ebene, die die Längen der Kanten respektiert und bei der sich keine zwei Kanten überschneiden



zusammengefaltet



entfaltet

Problem:

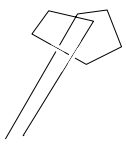
geg.: 2 Positionen P_1, P_2 eines Gestänges

Frage: gibt es eine Bewegung, die P_1 in P_2 überführt?

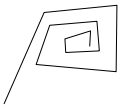
GEST ist PSPACE-vollständig; Beweisidee (direkt):

- benutze Turingmaschine, die Gestänge als Rechenwerk benutzt
- Position $\widehat{=}$ Konfiguration

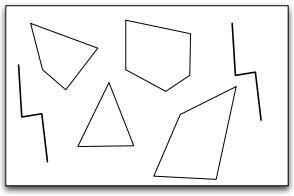
auch: „Bäume in der Ebene“ und „Kantenzüge im Dreidimensionalen“ sind PSPACE-vollständig



nicht: „Kantenzüge in der Ebene“(Zollstockproblem)
gehen immer aufzurollen:



- Beispiele für PSPACE-schwer:
  - ► kürzeste Wege in 3d
  - ► Kann sich das Gestänge an den Hindernissen vorbei in eine andere gegebene Position bewegen?



Jenseits von PSPACE Betrachten wir nun folgende Klassen jenseits von PSPACE:

EXPTIME

Die Klasse EXPTIME ist definiert als

$$EXPTIME := \bigcup_{c,P} TIME(c^{P(m)})$$

wobel $c > 1$ Konstante und Polynome $P(n)$.
Diese Klasse beschreibt alle Probleme, die in exponentieller Zeit lösbar sind.

NEXPTIME

Entsprechend zu der Klasse EXPTIME ist NEXPTIME definiert als die Menge aller *nichtdeterministisch* in exponentieller Zeit lösbarer Probleme.

Satz: Ist ein Problem mit $s(n)$ Platz lösbar, so auch mit $c^{s(n)}$ Zeit für eine Konstante c.

Beweis: Eine Turingmaschine M deren Arbeitsband in der Größe beschränkt ist durch $s(n)$ hat für ein Bandalphabet Σ genau $|\Sigma|^{s(n)}$ unterschiedliche Bandbeschriftungen sowie $s(n)$ unterschiedliche Kopfstellungen des Lesebandes auf dem Arbeitsband. Hat die endliche Kontrolle der Turingmaschine $|Q|$ viele Zustände. Daraus ergibt sich, dass es

$$|\Sigma|^{s(n)} s(n) |Q| \leq c^{s(n)}$$

viele Konfigurationen gibt. Sollte die Turingmaschine länger benötigen, gerät sie zwingend in eine Schleife, da sie in dem nächsten Schritt ein eine Konfiguration wechseln würde, in der sie schon einmal gewesen ist (und von der aus sie in die Konfiguration gekommen ist, in der sie sich jetzt gerade befindet).

Daraus folgt, dass

$$PSPACE(s(n)) \subset_{c>1} TIME(c^{s(n)})$$

also

$$PSPACE(s(n)) \subset EXPTIME$$

Fügen wir nun diese neue Erkenntnis in die Komplexitätsklassenlandschaft ein, so ergibt sich

$$P \subset NP \subset PSPACE \subset EXPTIME$$

Für die Mengen in dieser Kette ist (noch) nicht bekannt, ob die jeweiligen Untermengen echte Untermengen sind oder ob diese Mengen vielleicht gleich sind.
Betrachten wir nun folgende Hierarchiesätze (ohne Beweis):

Satz:
1. Ist $S_1 = o(S_2)$, so ist $SPACE(S_1) \subset_{\neq} SPACE(S_2)$
2. Ist $T_1 \log T_1 = o(T_2)$, so ist $TIME(T_1) \subset_{\neq} TIME(T_2)$

Hieraus folgt aber, dass P echte Untermenge von $EXPTIME$ ist. Wenn P echte Teilmenge von $EXPTIME$ ist, muss entweder P echte Untermenge von NP oder NP echte Untermenge von *PSPACE* oder *PSPACE* echte Untermenge von $EXPTIME$ sein. Es können natürlich von links nach rechts alle Mengen echte Untermengen ihrer Übermengen sein. Es wird vermutet, dass alle Mengen echte Untermengen ihrer Übermenge sind, also auch, dass $P \neq NP$.

nachweisbar schwere Probleme

Alle Probleme, die wir bis jetzt betrachtet haben könnten noch in P liegen, betrachten wir nun Probleme, die *nicht* in P liegen können, also mindestens in EXPTIME liegen.

- Äquivalenz regulärer Ausdrücke mit $\cup, \cap, ^*, ^-, \circ$ als Operatoren

  zum Beispiel $\overline{a^*ba^* \cup ba^{*^*}} = a^*$?

  ÄRA:

  Problem:

  geg.: 2 reguläre Ausdrücke a_1 und a_2

  Frage: beschreiben a_1 und a_2 die gleiche Sprache?

  Wir wissen, dass es für jeden regulären Ausdruck a einen NFA N gibt mit $L(a) = L(N)$, diesen Automat kann man in einen DFA überführen, diesen wiederum in den Äquivalenzklassenautomaten überführen. Ein möglicher Algorithmus erstellt für beide Ausdrücke a_1 und a_2 die Äquivalenzklassenautomaten und testet diese auf Isomorphie. Somit ist ARA $\in$ EXPTIME.
  Ferner kann man zeigen, dass jeder Algorithmus, der dieses problem lösst eine Laufzeit von $\Omega(c^{\sqrt{\frac{n}{\log n}}})$ für eine Konstante $c > 1$ hat.
  Daraus folgt, dass ÄRA $\notin P$.