



Technische
Universität
Braunschweig



Theoretische Informatik 2

Arne Schmidt

Kapitel 9.2 – PATH ist NL-vollständig

PATH-Problem

Pfadexistenz (PATH)

Gegeben: Ein gerichteter Graph $G = (V, E)$, Knoten $s, t \in V$.

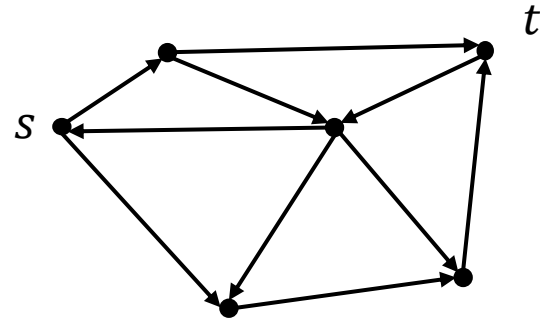
Frage: Gibt es einen Pfad von s nach t in G ?

Aus „Algorithmen und Datenstrukturen“:

- Pfad ist eine Kantenfolge von s nach t ohne Knotenwiederholungen.
- Ein Pfad besitzt maximal $|V|$ Knoten.

Wie können wir mit logspace testen, ob ein Pfad existiert?

Wie codieren wir überhaupt den Graphen?



Graphcodierung

Adjazenzliste:

Liste zu jedem Knoten die Nachbarn auf!

Für die TM reichen uns dafür Zahlen.

$0 \rightarrow 1, 2$

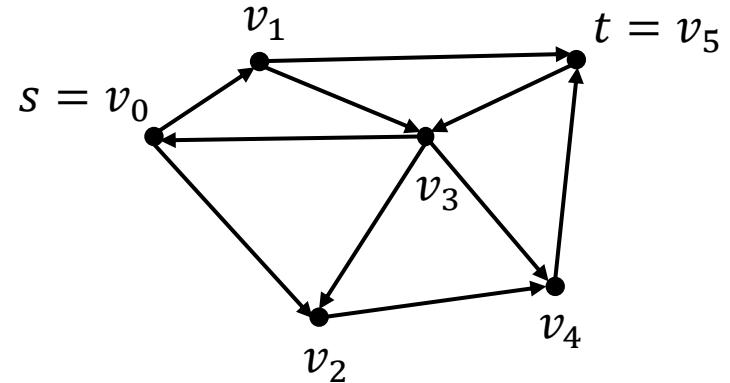
$1 \rightarrow 3, 5$

$2 \rightarrow 4$

$3 \rightarrow 0, 2, 4$

$4 \rightarrow 5$

$5 \rightarrow 3$



Das können wir einfach binär codieren!

Damit ist $\text{PATH} = \{ G\#s\#t \mid G \text{ codiert einen Graphen, in dem es einen Pfad von } s \text{ nach } t \text{ gibt.} \}$

PATH in NL

Lemma 9.12

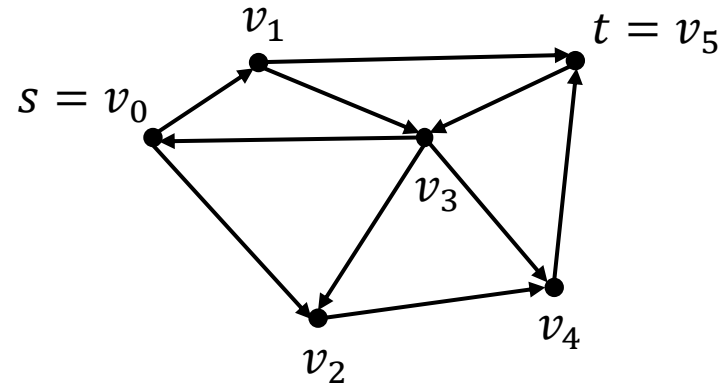
PATH ist in NL.

Betrachte Graphenscan aus AuD:

- Breitensuche: Muss die Level speichern!
→ Benötigt zu viel Speicher
- Tiefensuche: Wenn wir irgendwo nicht weiterkommen, müssen wir zurück.
→ Wir müssen den gesamten Pfad speichern!
→ Benötigt zu viel Speicher!

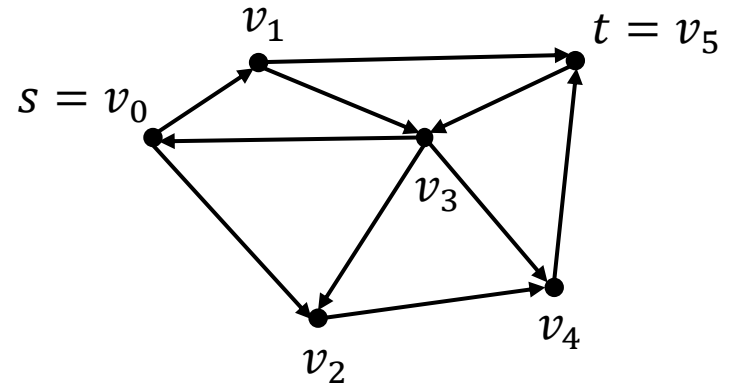
Beachte: Wir sind nicht-deterministisch!

→ In Tiefensuche: Wenn wir nicht weiterkommen, ist dieser Pfad einfach nicht der Gesuchte!



Random Walk

```
count = 0
current = s
while count < n do
  count ++
  Rate Knoten  $new \in \{1, \dots, n\}$ 
  if new ist Nachfolger von current then
    if new = t then
      return true
    end if current = new
  else
    return false
  end if
end while
return false
```



Beachte: Es kann passieren, dass wir teilweise im Kreis laufen. Das ist aber egal.
Existiert ein Weg, existiert auch ein Pfad!

Random Walk – Analyse

```
count = 0
current = s
while count < n do
  count ++
  Rate Knoten new ∈ {1, ..., n}
  if new ist Nachfolger von current then
    if new = t then
      return true
    end if current = new
  else
    return false
  end if
end while
return false
```

Zu jedem Zeitpunkt müssen nur folgende Daten **gespeichert** werden:

- Zähler count, der durch n beschränkt ist.
→ $\log n$ Speicherbedarf
- Zwei Knoten, die durch eine Zahl $1, \dots, n$ codiert sind.
→ $2 \log n$ Speicherbedarf

Wir benötigen also nur $O(\log n)$ Speicherbedarf.

Korrektheit ist recht klar:

Gibt es einen Pfad, dann gibt es eine Berechnung, die genau diesen Pfad rät und damit true zurückgibt.

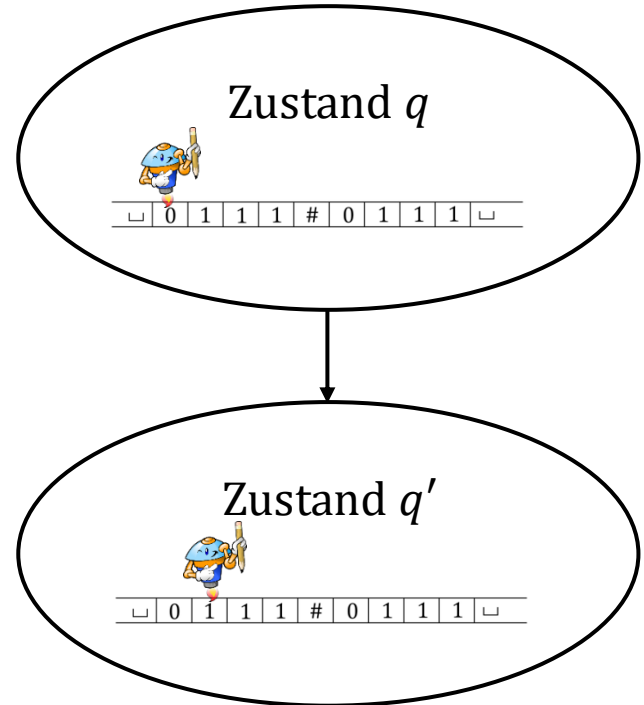
PATH ist NL-Vollständig

Theorem 9.13

PATH ist in NL-Vollständig (bzgl. Logspace-Reduktionen).

Zur Erinnerung, der Konfigurationsgraph einer TM ist wie folgt:

- Jede Konfiguration $c = v q w \times$ Arbeitsband entspricht einem Knoten.
- Zwei Konfigurationen c_1, c_2 sind verbunden, wenn $c_1 \rightarrow c_2$ gilt.

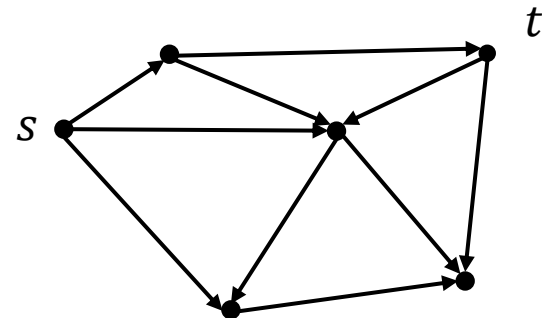
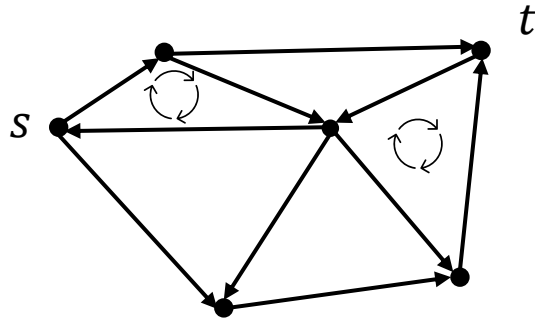


Azyklische Graphen

Pfadexistenz in azyklischen Graphen (ACYCLICPATH)

Gegeben: Ein gerichteter, azyklischer Graph $G = (V, E)$, Knoten $s, t \in V$.

Frage: Gibt es einen Pfad von s nach t in G ?



Lemma 9.14

ACYCLICPATH ist NL-Vollständig (bzgl. Logspace-Reduktionen).

Kapitel 9.3 – coNL

Unerreichbarkeit

Unerreichbarkeit ($\overline{\text{PATH}}$)

Gegeben: Ein gerichteter Graph $G = (V, E)$, Knoten $s, t \in V$.

Frage: Gibt es keinen Pfad von s nach t in G ?

Von Theorem 2.1 (Satz von Immerman & Szelepcsényi) wissen wir bereits:

Proposition 9.16

$\overline{\text{PATH}} \in \text{NL}$.

Theorem 9.15

$\text{NL} = \text{coNL}$.

Theorem 9.17

Es sei $f: \mathbb{N} \rightarrow \mathbb{N}$ mit $f(n) \geq \log n$, $\forall n$. Dann gilt

$$\text{coNSPACE}(O(f)) = \text{NSPACE}(O(f))$$

Kapitel 9.4 – 2SAT (siehe Notizen)

Nächstes Kapitel

