Technische Universität Braunschweig

Theoretische Informatik 2

Arne Schmidt

Kapitel 9.2 – PATH ist NL-vollständig

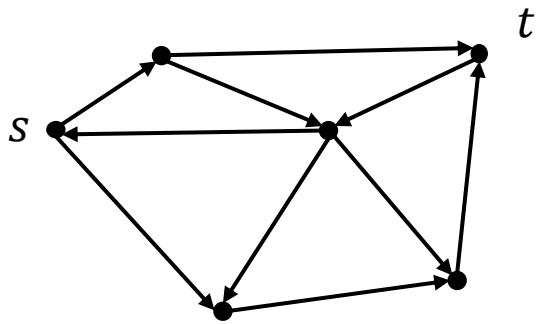
PATH-Problem

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.

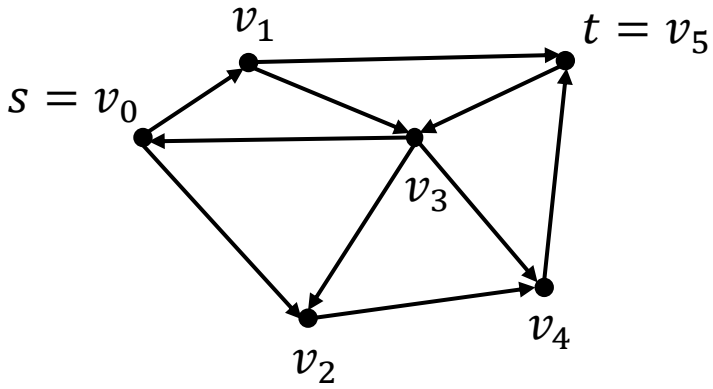
Wie können wir mit logspace testen, ob ein Pfad existiert?
Wie codieren wir überhaupt den Graphen?



Graphcodierung

Adjazenzliste:
Liste zu jedem Knoten die Nachbarn auf!
Für die TM reichen uns dafür Zahlen.

$0 \to 1, 2$
$1 \to 3, 5$
$2 \to 4$
$3 \to 0, 2, 4$
$4 \to 5$
$5 \to 3$



Das können wir einfach binär codieren!

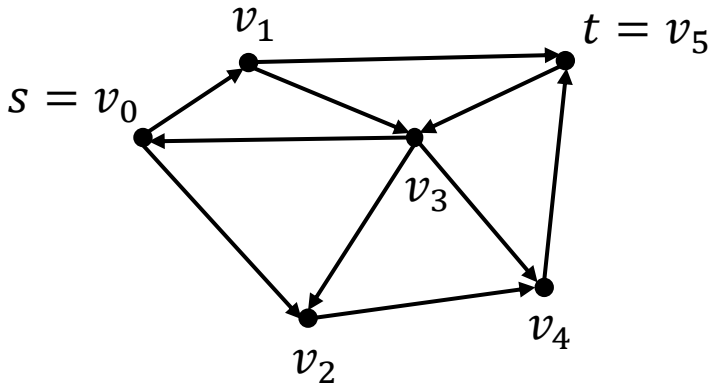
Damit ist PATH = $\{ G\#s\#t \mid G$ codiert einen Graphen, in dem es einen Pfad von s nach t gibt. $\}$

PATH in NL

Lemma 9.2
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 $new \in \{1, \ldots, n\}$

  if new ist Nachfolger von $current$ **then**

    if $new = t$ **then**

      return true

    end if $current = new$

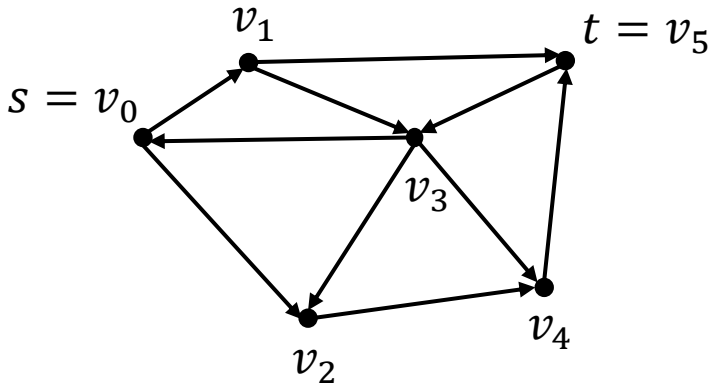
  else

    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, ..., n codiert sind.
  → $2 \log n$ Speicherbedarf

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

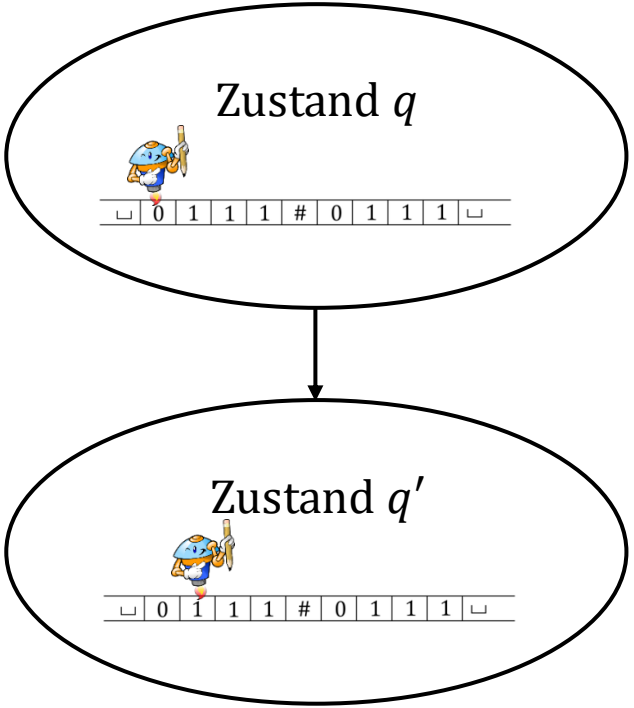
Korrektheit ist recht klar:
Gibt es einen Pfad, dann gibt es eine Berechnung, die genau diesen Pfad rät und damit true zurückgibt.

PATH ist NL-Vollständig

Lemma 9.2
PATH ist in NL-Vollständig (bzgl. Logspace-Reduktionen).

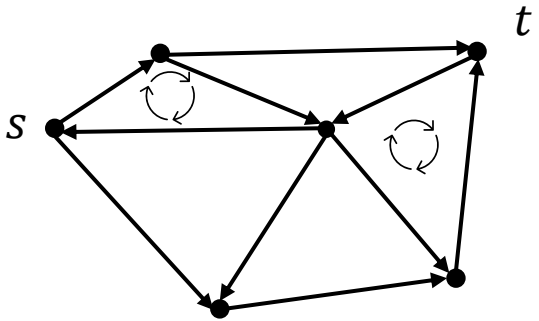
Zur Erinnerung, der Konfigurationsgraph einer TM ist wie folgt:

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

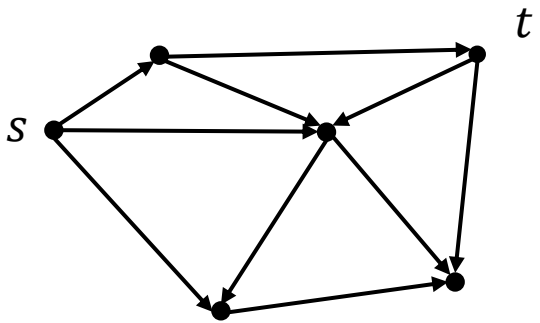


Azyklische Graphen

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?



Nicht azyklisch



Azyklisch

Lemma 9.2
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
NL = coNL.

Theorem 9.17
Es sei $f: \mathbb{N} \to \mathbb{N}$ mit $f(n) \geq \log n\,,\ \forall\, n$. Dann gilt

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

Kapitel 9.4 – 2SAT
(siehe Notizen)

Nächstes Kapitel