Tutorium 10

Florian Weber

25. Juni 1014

Tiefensuche

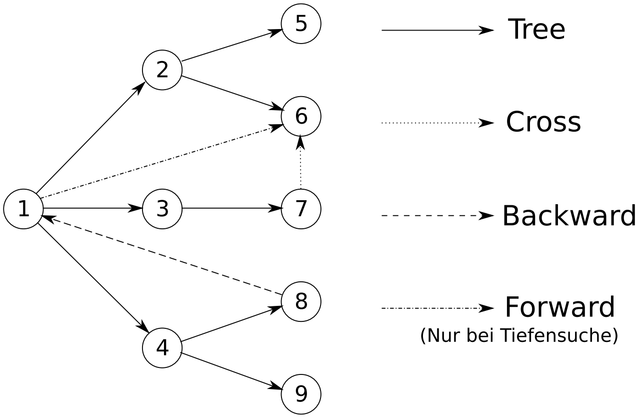
Prinzip

- Sei N ein Knoten eines Graphen V
- Verwende N
- Für alle Nachbarknoten M:
  - Führe eine Tiefensuche auf M durch

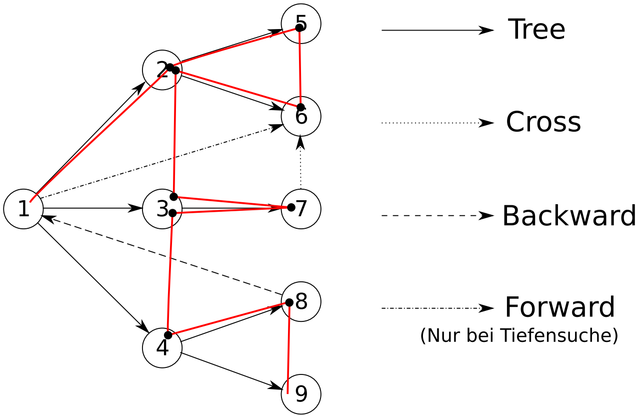
Prinzip

- Sei N ein Knoten eines Graphen V
- Verwende N
- Für alle Nachbarknoten M:
  - Führe eine Tiefensuche auf M durch
- Laufzeit: $O(|E|)$

Kantenarten



Kantenarten



Kürzeste Wege

Algorithmus von Dijkstra

- Problemstellung:
  - gewichteter Graph ohne negative Kanten
  - Finde kürzesten Weg zwischen zwei Knoten

Algorithmus von Dijkstra

- Problemstellung:
  - gewichteter Graph ohne negative Kanten
  - Finde kürzesten Weg zwischen zwei Knoten
- Prinzip:
  - Sei R die Menge bereits erreichter Knoten (anfänglich: nur Startknoten S)
  - Wähle die leichteste Kante e die von einem Knoten $a \in R$ zu einem Knoten $b \in \overline{R}$ geht.
  - der Pfad von S zu a gefolgt von e ist nun der kürzeste Pfad zu b
  - Nun: $b \in R$, führe fort, bis Zielknoten einsortiert.

Algorithmus von Dijkstra

- Problemstellung:
  - gewichteter Graph ohne negative Kanten
  - Finde kürzesten Weg zwischen zwei Knoten
- Prinzip:
  - Sei R die Menge bereits erreichter Knoten (anfänglich: nur Startknoten S)
  - Wähle die leichteste Kante e die von einem Knoten $a \in R$ zu einem Knoten $b \in \overline{R}$ geht.
  - der Pfad von S zu a gefolgt von e ist nun der kürzeste Pfad zu b
  - Nun: $b \in R$, führe fort, bis Zielknoten einsortiert.
- Anschaulich:
  - Netz $=$ Graph
  - ziehe Start und Zielknoten auseinander
  - gespannte Verbindung $=$ kürzester Weg.

Bellman-Ford (Prinzip)

- Speichere für alle Knoten eine Distanz von ∞ und `NIL` als Vorgänger
- n-mal ($n = |V|$): für jede Kante $e = (u, v)$:
  - Falls $\mathrm{dist}(u) + \mathrm{weight}(e) \leq \mathrm{dist}(v)$:
    - $\mathrm{dist}(v) := \mathrm{dist}(u) + \mathrm{weight}(e)$
    - $\mathrm{predecessor}(v) := u$
- $\exists (u, v) \in E : \mathrm{dist}(u) + \mathrm{weight}(e) \leq \mathrm{dist}(v) \Leftrightarrow$ Graph enthält negativen Kreis

Bellman-Ford (Eigenschaften)

- Brute-Force, sehr langsam: $O(|V| \times |E|)$
- kann mit negativen Kanten umgehen
- negative Kreise können erkannt werden

Kreativaufgaben

isDAG auf Graphen mit vielen kleinen Kreisen

- Gegeben: Ein großer Graph der vermutlich viele Kleine Kreise enthält
- Gesucht: Ein Algorithmus der in $O(|E|)$ herausfindet, ob der Graph ein DAG ist
- Problem: Der Algorithmus soll bei Graphen wie dem obigen erwartet sehr viel schneller sein.

Breitensuche mit $O(1)$ Speicher

- auf Tutblatt