Relationen und DAGs, starker Zusammenhang

Anmerkung:

- Sei $D = (V, E)$. Dann ist $A \subseteq V \times V$ eine Relation auf V.
- Sei andererseits $R \subseteq S \times S$ eine Relation auf S.
- Dann definiert $D = (S, R)$ einen DAG.
- D.h. DAGs sind Darstellungen von Relationen.

Definition starker und schwacher Zusammenhang

Sei $D = (V, E)$ ein DAG. Der ungerichtete $G = (V, E')$ mit $E' = \{\{u, v\} \in V^2 \mid (u, v) \in E \text{ oder } (v, u) \in E\}$ heißt *zugrundeliegender Graph* von D.

1. D ist *stark zusammenhängend*, falls für alle $u, v \in V$ ein u-v Pfad in D existiert.
2. D ist *schwach zusammenhängend*, falls der zugrundeliegende Graph G von D zusammenhängend ist.

Starker Zusammenhang algorithmisch

Zwei Algorithmen zum Testen von starkem Zusammenhang:

- Führe TIEFENSUCHE für alle Startknoten $s \in V$ durch.
- Falls stets alle Knoten erreicht werden, dann ist G stark zusammenhängend.
- Laufzeit dieses Algorithmus: $\mathcal{O}(|V| \cdot (|V| + |E|))$.

- Verbesserter Algorithmus: Verwendung von Ruckwärtsgraph $\bar{D}$.
- Digraph $\bar{D} = (V, \bar{E})$ besitzt $\bar{E} = \{(u, v) \in V^2 \mid (v, u) \in E\}$.
- Man kann zeigen, dass eine Anwendung von Tiefensuche auf D und auf $\bar{D}$ genügt. (Beweis ist nicht-trivial)
- Führt zu einem Algorithmus mit Laufzeit $\mathcal{O}(|V| + |E|)$.

Berechnung der transitiven Hülle

Algorithmus SIMPLE-TRANSITIV

EINGABE: Relation $R \subseteq S \times S$ in Adjazenzmatrix-Darstellung

1. While ($\exists x, y, z$ mit $(x, y), (y, z) \in R$ und $(x, z) \notin R$)
   1. $R \leftarrow R \cup \{(x, z)\}$

AUSGABE: Transitive Hülle $R^+ = R$

- **Korrektheit:** Solange transitive Beziehungen fehlen, werden diese zur Relation R hinzugefügt.
- **Laufzeit:** Die Menge R^+ enthält höchstens $|S|^2$ Tupel aus $S \times S$.
- In jeder Iteration wird ein Tupel (x, z) hinzugefügt.
- D.h. die Schleife durchläuft maximal $\mathcal{O}(|S|^2)$ Iterationen.
- Pro Iteration: Prüfe für alle $x, y, z \in S$, ob für die Einträge in der Adjazenzmatrix $(x, y) = (y, z) = 1$ und $(x, z) = 0$ gilt.
- Die Anzahl aller möglichen $x, y, z \in S$ ist $\mathcal{O}(|S|^3)$.
- Damit ist die Gesamtlaufzeit von SIMPLE-TRANSITIV $\mathcal{O}(|S|^5)$.

Mit Dynamischer Programmierung

- Sei $D = (V, E)$. Die transitive Hülle $D^+ = (V, E^+)$ besitzt die Kantenmenge $E^+ = \{(u, v) \in V^2 \mid \exists\ u\text{-}v \text{ Pfad in } D.\}$.

Definition Wege mit eingeschränkten inneren Knoten

Sei $D = ([n], E)$ Wir definieren für $k, u, v \in [n]$

$$W_k[u, v] = \begin{cases} 1 & \exists\ u\text{-}v \text{ Pfad in } D \text{ mit inneren Knoten aus } [k] \\ 0 & \text{sonst} \end{cases}.$$

- Für $k = 0$ gilt $\{(u, v) \in V^2 \mid W_0[u, v] = 1\} = E$.
- Für $k = n$ gilt $\{(u, v) \in V^2 \mid W_n[u, v] = 1\} = E^+$.
- Berechnen $W_k[u, v]$ rekursiv aus W_{k-1}. $W_k[u, v] = 1$, falls
  - Es gibt einen u-v Pfad mit inneren Knoten aus $[k-1]$.
  - Es gibt einen u-k Pfad und k-v Pfad mit inneren Knoten aus $[k-1]$.
- D.h. $W_k[u, v] = \max\{W_{k-1}[u, v], W_{k-1}[u, k] \cdot W_{k-1}[k, v]\}$.

Algorithmus von Warschall

Algorithmus WARSCHALL

EINGABE: $D = ([n], E)$ in Adjazenzmatrix-Darstellung

1. Für alle $u, v \in V$
   1. If $(u, v) \in E$ then $W[u, v] \leftarrow 1$ else $W[u, v] \leftarrow 0$;
2. For $k \leftarrow 1$ to n
   1. Für alle $u, v \in V$
      1. $W[u, v] \leftarrow \max\{W[u, v], W[u, k] \cdot W[k, v]\}$

AUSGABE: $D^+ = (V, E^+)$ mit $E^+ = \{(u, v) \mid W[u, v] = 1\}$

Transitive Hülle in kubischer Laufzeit

Satz Transitive Hülle

Sei $D = (V, E)$. Dann berechnet WARSCHALL die transitive Hülle $D^+ = (V, E^+)$ in Zeit $\mathcal{O}(|V|^3)$ und mit Speicherbedarf $\mathcal{O}(|V|^2)$.

Korrektheit:

- Wissen $W_k[u, v] = \max\{W_{k-1}[u, v], W_{k-1}[u, k] \cdot W_{k-1}[k, v]\}$.
- WARSCHALL verwendet nur ein zweidimensionales Array für W.
- D.h. die benötigten Zwischenwerte $W_{k-1}[u, k]$ bzw. $W_{k-1}[k, v]$ werden durch $W_k[u, k]$ bzw. $W_k[k, v]$ überschrieben.
- Zeigen: $W_{k-1}[u, k] = 1$ gdw $W_k[u, k] = 1$. (analog für $W_{k-1}[k, v]$)
  - "$\Rightarrow$": Jeder u-k Pfad mit inneren Knoten aus $[k-1]$ ist auch ein u-k Pfad mit inneren Knoten aus $[k]$.
  - "$\Leftarrow$": Sei $p = (u, v_1, \ldots, v_\ell, k)$ ein u-k Pfad. Aufgrund der Pfadeigenschaft sind die inneren Knoten $v_1, \ldots, v_\ell$ aus $[k-1]$.

Laufzeit und Speicherplatz

- Laufzeit Schritt 1: $\mathcal{O}(|V|^2)$, Schritt 2: $\mathcal{O}(|V|^3)$.
- Speicherbedarf für Array W: $\mathcal{O}(|V|^2)$.

Wurzelbäume

Definition Wurzelbaum

Sei $T = (V, E)$ ein Baum und $v \in V$. Wir definieren den in v gewurzelten Baum T_v wie folgt.

1. v heißt *Wurzel* des Baums.
2. Alle zu v adjazenten Knoten u heißen *Kinder* des *Vater-/bzw. Elternknotens* v.
3. Kinder mit gleichem Elternknoten heißen *Geschwister*.
4. Sei u ein Kind von v. Dann ist u Wurzel eines Teil-Baums, dessen Kinder die Elemente der Menge $\Gamma(u) \setminus \{v\}$ sind.
5. Sei u ein Knoten mit u-v Pfad der Länge k. Dann gilt $\mathrm{Tiefe}[u] = k$.
6. Die *Höhe* des Baums T_v ist $h(T_v) = \max_{u \in V}\{\mathrm{Tiefe}[u]\}$.

Binärbäume

Definition Binärbaum

Sei $T_v = (V, E)$ ein in v gewurzelter Baum.

- T_v heißt *Binärbaum*, falls jeder Knoten höchstens zwei Kinder besitzt.
- Ein Binärbaum heißt *vollständig*, falls jedes Nicht-Blatt genau zwei Kinder besitzt und alle Blätter gleiche Tiefe haben.

Bemerkungen:

- Vollständige Binärbäume können als Array realisiert werden.
- Dabei erhalten Knoten in Tiefe t Indizes $2^t, \ldots, 2^{t+1} - 1$.
- Die Kinder eines Nicht-Blatts i besitzen Indizes $2i$ und $2i + 1$.
- Der Vaterknoten eines Knotens i ist $\lfloor \frac{i}{2} \rfloor$.
- Vollständige Bäume mit n Knoten besitzen Höhe $\Theta(\log n)$.

Binäre Suchbäume

Definition Binärer Suchbaum

Sei $T_v = (V, E)$ ein Binärbaum mit $V \subseteq \mathbb{Z}$. T_v heißt *Suchbaum*, falls für alle $u \in V$ gilt

1. Für alle u_l im linken Teilbaum von u gilt $u_l \leq u$.
2. Für alle u_r im rechten Teilbaum von u gilt $u_r \geq u$.

Suche in Binärbäumen

Algorithmus SUCHE-ELEMENT

EINGABE: Suchbaum $T_v = (V, E)$, u

1. While ($w \neq u$ und w ist kein Blatt)
   1. If ($u \leq w$) then $w \leftarrow$ linkes Blatt von w.
   2. Else $w \leftarrow$ rechtes Kind von w.

AUSGABE: $\begin{cases} u \text{ gefunden} & , \text{falls } w = u \\ u \text{ nicht gefunden} & , \text{sonst} \end{cases}$

Laufzeit:

- Die Laufzeit ist $\mathcal{O}(h(T_v))$.
- Verschiedene Strategien, um $h(T_v) = \mathcal{O}(\log |V|)$ zu erzwingen: höhen-/gewichtsbalancierte Bäume, Rot-Schwarz-Bäume, etc.

Definition Teiler, ggT, kgV, Primzahlen

Seien $a, b \in \mathbb{Z}$.

1. *a teilt b*, geschrieben $a|b$, falls ein $k \in \mathbb{Z}$ existiert mit $ak = b$.
2. Falls a die Zahl b nicht teilt, schreiben wir $a \nmid b$.
3. Der *größte gemeinsame Teiler* von a und b ist
$$\mathrm{ggT}(a, b) = \max_{k \in \mathbb{N}} \{k | a \text{ und } k | b\}.$$
4. Das *kleinste gemeinsame Vielfache* von a und b ist
$$\mathrm{kgV}(a, b) = \min_{k \in \mathbb{N}} \{a | k \text{ und } b | k\}.$$
5. $a > 1$ heißt *Primzahl*, falls die einzigen Teiler $k \in \mathbb{N}$ von a die Zahlen 1 und a sind.
6. a, b heißen *teilerfremd*, falls $\mathrm{ggT}(a, b) = 1$.

Modulare Arithmetik

Definition Modulare Arithmetik

Seien $a, b \in \mathbb{Z}$ und $m \in \mathbb{N}$. Wir nennen a und b *kongruent modulo* m, falls $m|a-b$. Wir schreiben $a \equiv b \bmod m$ oder auch $a = b \bmod m$.

Satz Äquivalenzklassen der modularen Arithmetik

Sei $m \in \mathbb{N}$. $R = \{(a,b) \in \mathbb{Z}^2 \mid a = b \bmod m\}$ ist eine Äquivalenzrelation auf $\mathbb{Z}$.

Beweis:

- **Reflexivität:** $a = a \bmod m$, denn $m|(a-a)$.
- **Symmetrie:** $a = b \bmod m$, d.h. $a - b = km$ für ein $k \in \mathbb{Z}$.
- Damit gilt $b - a = (-k)m$ bzw. $b = a \bmod m$.
- **Transitivität:** Seien $a = b \bmod m$ und $b = c \bmod m$.
- Damit gilt $a - b = k_1 m$ und $b - c = k_2 m$. Es folgt $a - c = (a-b) + (b-c) = (k_1 + k_2)m$ und daher $a = c \bmod m$.

Form der Äquivalenzklassen

Anmerkung:

- Es gilt $a = a \pm m = a \pm 2m = \ldots = a + km \bmod m$ für alle $k \in \mathbb{Z}$.
- Wir schreiben auch $\{x \in \mathbb{Z} \mid x = a + mk, k \in \mathbb{Z}\} = a + m\mathbb{Z}$.
- Es gibt m verschiedene Äquivalenzklassen modulo m:

$$0 + m\mathbb{Z}, 1 + m\mathbb{Z}, \ldots, (m-1) + m\mathbb{Z}.$$

- Sei $a = \lfloor \frac{a}{m} \rfloor m + r$ mit $r \in \mathbb{Z}_m$. Es gilt $r = a \bmod m$.
- Wir repräsentieren $a + m\mathbb{Z}$ durch eindeutiges $r \in \mathbb{Z}_m$.

Anwendung: modularer Arithmetik

- Pseudozufallszahlen mittels Linearem Kongruenzgenerator.
- Beginne mit zufälligem Startwert $x_0 \in \mathbb{Z}_m$.
- Berechne iterativ für festes $a, b \in \mathbb{Z}_m$: $x_i = ax_{i-1} + b \bmod m$.
- $x_1, x_2, x_3, \ldots$ definieren eine Pseudozufallsfolge.