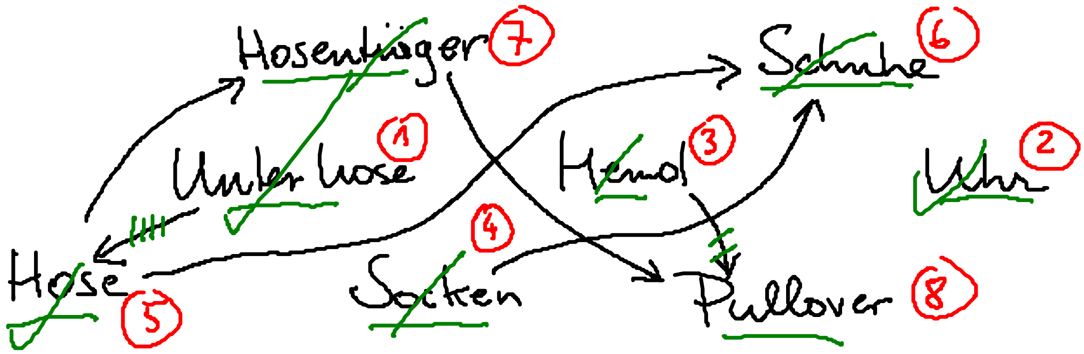
ALP3 Algorithmen Datenstrukturen Datenabstraktion

Freie Universität Berlin

Topologisches Sortieren



GEGEBEN: n Aufgaben

m Abhängigkeiten (vorher-nachher-Beziehungen)

(a,b) „a muss vor b kommen"

(geordnete Paare von Aufgaben)

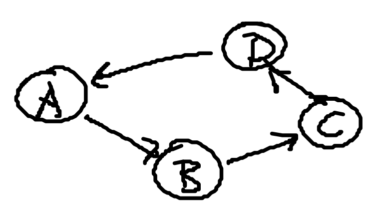
GESUCHT: Eine Reihenfolge $(a_1, a_2, \ldots, a_n)$ der Aufgaben, die die Abhängigkeiten berücksichtigt.

ALGORITHMUS

1. Wähle eine Aufgabe ohne Vorgänger, und setze sie an die nächste Stelle. *
2. Streiche alle Abhängigkeiten, in denen diese Aufgabe vorkommt. [als erste Komponente]
3. Wiederhole ab 1, bis alle Aufgaben erledigt sind.

---

*1a). Falls es keine solche Aufgabe gibt: ABBRUCH



SATZ: Eine topologische Sortierung existiert genau dann, wenn es keine zyklische Abhängigkeit $\boxed{(a_1,a_2),(a_2,a_3),\dots(a_{k-1},a_k),(a_k,a_1)}$ gibt.

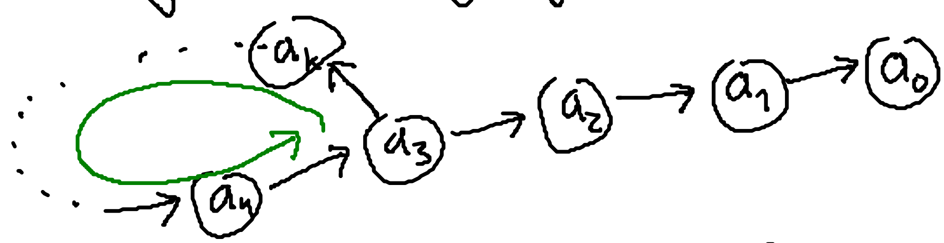
BEWEIS: „⇒" ✓

„⇐" Wenn der Algorithmus im Fall 1a steckenbleibt, dann gibt es eine zyklische Abhängigkeit.

Annahme: Jede Aufgabe a hat eine Vorgängeraufgabe $v(a)$.

Beginne mit einer beliebigen Aufgabe a_0 und gehe zur Vorgängeraufgabe.

$a_1 := v(a_0)$
$a_2 := v(a_1)$
⋮



Weil es nur endlich viele Aufgaben gibt, muss irgendwann (zum ersten Mal) eine Aufgabe doppelt vorkommt ($a_k = a_\ell$, $\ell < k$)

Dann ist $(a_\ell, a_k),(a_k, a_{k-1}) \dots (a_{\ell+1}, a_\ell)$ eine zyklische Abhängigkeit. □

GEGEBEN: m Abhängigkeiten (a,b) „a vor b", $a,b \in \{1,\dots,n\}$

ALGORITHMUS

1. Wähle eine Aufgabe ohne Vorgänger, und setze sie an die nächste Stelle. * $\Big\} O(n \cdot m)$
2. Streiche alle Abhängigkeiten, in denen diese Aufgabe vorkommt. [als erste Komponente] $\Big\} O(m)$

$\Big] \times n$

3. Wiederhole n-mal.

IMPLEMENTIERUNG: Ganz naiv: $O(n^2 m)$

1.) Durchlaufe alle Abhängigkeiten (a,b) und markiere jedes b. $\Big\} O(m)$
Suche eine unmarkierte Aufgabe $\Big\} O(n)$

$\rightarrow O((m+n) \cdot n)$ insgesamt

2.) Speichere am Anfang zu jeder Aufgabe a, die Liste der Abhängigkeiten (a,b) $O(m)$

Schritt 2: $O(m)$ insgesamt statt $O(m \cdot n)$

1.) ~~Durchlaufe alle Abhängigkeiten (a,b) und zähle für jedes b die Anzahl der Vorgänger.~~ $\Big\} \cancel{O(m)}$
Suche eine ~~unmarkierte~~ Aufgabe mit 0 Vorgängern. $\Big\} O(n)$

$\rightarrow O((m+n) \cdot n)$ insgesamt

2.) Für jede gelöschte Abhängigkeit (a,b), dekrementiere den Zähler von b *

$\rightarrow O(n) \cdot n$ insgesamt für (1)

Führe eine Liste F der ~~(Knoten)~~ Aufgabe, die keine Vorgänger haben.

Wenn der Vorgängerzähler auf 0 sinkt, füge b zur Liste F hinzu.

[1.) $O(1) \times n = O(n)$
2.) $O(m)$ insgesamt.

a) Für jede Aufgabe a, die Liste der Abhängigkeiten (a,b)

b) Für jede Aufgabe a, die Anzahl V(a) der Vorgänger (Abhängigkeiten (b,a))

c) Liste F aller Aufgaben a mit V(a) = 0.

1.) Wähle $a \in F$ und streiche sie aus F. $O(1)$ $\times n$

2.) Durchlaufe die Liste der Abhängigkeiten (a,b) von a und dekrementiere V(b)
Falls V(b) = 0, füge b zu F hinzu. } $O(m)$ insgesamt.

$O(m+n)$
Die Listen a), b), c) müssen zu Beginn initialisiert werden.

Welche Datenstrukturen? ⇐ Welche Operationen?

c) Einfügen, beliebiges Element herausnehmen
→ F ... Stapel oder Schlange.
maximale Größe oder Längen ... Stapel als Feld.

b) Aufgaben = $\{1, \dots, n\}$ → V(b) als Feld.
→ V(b) als Attribut des Knotens b.

c) Operationen: • Einfügen (beim Einlesen)
• Durchlaufen (1x)
Länge unbekannt. → verkettete Listen.
[Adjazenzliste]

SATZ. $\boxed{\text{Topologisches Sortieren kann in } O(m+n) \text{ Zeit und Speicher gelöst werden.}}$

GENERELLE FRAGE:

? → neu ausrechnen

? → speichern (und bei Änderung auf den neuesten Stand bringen)

Eigentlich ist das Problem ein <u>Graphenproblem</u>!

Knoten $\hat{=}$ Aufgaben

gerichtete Kanten (a,b) $\hat{=}$ Abhängigkeiten

Innengrad von a $(d^-(a))$ $\hat{=}$ $V(a)$

gerichteter Kreis $\hat{=}$ zyklische Abhängigkeit.