Grundlagen der theoretischen Informatik

Vorlesung 5:
Tiefensuche, ungünstigste Wege

Tiefensuche

Tiefensuche

Input: gerichteter oder ungerichteter Graph G und Anfangsknoten $s \in V$.
Output: Menge der aus s erreichbaren Knoten.
Ablauf: Zuerst wird s besucht, dann ein Knoten v_1 mit $sv_1 \in E$. In einem allgemeinem Schritt wird nach Knoten v_i ein Knoten v_j mit $v_i v_j \in E$ besucht falls v_j früher nicht besucht wurde. Falls es keinen solchen v_j gibt, geht der Algorithmus zurück zum Knoten v_k aus den v_i durch Kante $v_k v_i$ erreicht wurde.

DFS-Nummerierung: Bei der Tiefensuche werden die Knoten in der Reihenfolge nummeriert, in sie besucht werden. (Notation: $m(v)$)
DFS-End-Nummerierung: Die Nummerierung der Knoten in der Reihenfolge, in der ihre auslaufenden Kanten vollständig abgearbeitet wurden. (Notation: $b(v)$)

Satz

Die Tiefensuche liefert genau die Knoten, die von s aus erreichbar sind.
Der Zeitbedarf des Algorithmus ist $konst \cdot (n + m)$.

Beweis. Es ist trivial, dass der Algorithmus nur Knoten besucht, die von s aus erreichbar sind.
Sei $T \subseteq V$ die Menge der aus s erreichbaren Knoten, $F \subseteq T$ die Menge der vom Algorithmus gefundenen Knoten, und nehmen wir indirekt an: $\exists v \in T \setminus F$. Da $v \in T$, gibt es einen $s \sim v$ Weg W. Es gibt zwei nacheinander kommende Knoten in W, x und y, so dass $x \in F$ und $y \notin F$. Das heißt aber, dass der Algorithmus für x aufgerufen wurde. Da $xy \in E$, hat der Algorithmus bei diesem Aufruf auch den Knoten y untersucht und dafür gesorgt, dass y auch besucht wird, also muss auch $y \in F$ gelten.

Wiederholung

Die Kanten des Bäumes werden als *Baumkanten (fa-élek)* bezeichnet.
Diejenigen Kanten, die nicht benutzt werden und von einem Knoten zu einem anderen Knoten im selben Teilbaum führen, der bei der Suche später besucht wird, heißen *Vorwärtskanten (előre-élek)*.
Diejenigen Kanten, die nicht benutzt werden und von einem Knoten zu einem anderen Knoten im selben Teilbaum führen, der bei der Suche bereits vorher besucht wurde, heißen *Rückwärtskanten (vissza-élek)*.
Diejenigen Kanten, die „quer“ von einem Teilbaum zu einem anderen Teilbaum verlaufen, heißen *Querkanten (kereszt-élek)*.

In ungerichteten Graphen gibt es nur Baumkanten, Querkanten und Vorwärtskanten.

Satz

Sei G ein gerichteter Graph und uv eine Kante in G. Wenn der Algorithmus Kante uv benutzt die folgende Eigenschaften gelten:

- falls $m(v)$ existiert nicht: uv ist eine Baumkante.
- falls $m(v) > m(u)$: uv ist eine Vorwärtskante.
- falls $m(v) < m(u)$ und $b(v)$ existiert nicht: uv ist eine Rückwärtskante.
- falls $m(v) < m(u)$ und $b(v)$ existiert: uv ist eine Querkante.

Bemerkung: In ungerichteten Graphen gibt es keine Querkanten nach dem DFS.

Gerichtete azyklische Graphen

Definition

Ein gerichteter Graph ist ein *DAG (directed acyclic graph)*, wenn er keinen gerichteten Kreis enthält.

Definition

Sei G ein gerichteter Graph. Eine Reihenfolge $v_1, v_2, \ldots, v_n$ der Knoten ist eine *topologische Ordnung (topologikus sorrend)*, wenn für alle Kanten $v_i v_j$ gilt, dass $i < j$.

Quelle (forrás): ein Knoten ohne eingehenden Kanten.
Senke (nyelő): ein Knoten ohne ausgehenden Kanten.

Lemma

Ein DAG immer enthält mindestens eine Senke und mindestens eine Quelle.

Beweis. Existenz von Senke:
Nehmen wir indirekt an, dass G ein DAG ist und er hat keine Senke. Betrachten wir einen Weg P mit maximaler Länge. Sei die Knoten von P $v_1, v_2, \ldots, v_k$. v_k ist keine Senke, dann gibt es eine Kante $v_k v_{k+1}$ in G. Aber v_{k+1} soll in P sein weil er maximale Länge hat, $v_{k+1} = v_j$ für $1 \leq j \leq k$.
Aber dann ist $v_j, v_{j+1}, \ldots, v_k, v_j$ ein Kreis, das ist ein Widerspruch.

Existenz von Quelle ist ähnlich.

Satz

Ein gerichteter Graph hat genau dann eine topologische Ordnung, wenn er ein DAG ist.

Beweis. Es ist klar, dass es nicht möglich ist, dass ein Graph sowohl eine topologische Ordnung als auch einen gerichteten Kreis hat.
Um eine topologische Ordnung in einem DAG G zu finden soll man zuerst eine Quelle v_1 nehmen. $G - v_1$ ist auch ein DAG, sei v_2 eine Quelle in diesem Graphen, usw. Diese Methode liefert eine topologische Ordnung.

Bemerkung

Mit einer einfachen Ergänzung der Tiefensuche kann in $konst \cdot (n + m)$ Schritten entweder eine topologische Ordnung oder ein gerichteter Kreis gefunden werden.
Falls eine Rückwärtskante gefunden wird, ist der Graph kein DAG.
Sonst gibt die umgekehrte DFS-End-Nummerierung eine topologische Ordnung.

Ungünstigster Weg (leghosszabb út): ein $u \sim v$ Weg mit maximalen Kosten unter $u \sim v$-Wege.

Bestimmung der ungünstigsten Wege in einem DAG

Input: $G = (V, E)$ DAG, $k : E \to \mathbb{R}$, $s \in V$.
Output: Kosten der ungünstigsten Wege von s zu allen anderen, von s aus erreichbaren Knoten.
Ablauf:

1. Bestimmung einer topologischen Ordnung $v_1, v_2, \ldots, v_n$.
2. Sei $s = v_i$. Für alle $j < i$ ist v_j aus s nicht erreichbar.
3. Sei $b(s) = 0$
4. für $j = i + 1, \ldots, n$

$$b(v_j) = \max\{b(v_l) + k(v_l v_j) : v_l v_j \in E \text{ und } l \geq i\}$$

5. Am Ende enthält b(v) die Kosten des ungünstigsten $s \sim v$ Weges.

Falls G nicht ein DAG ist, keinen solchen Algorithmus ist bekannt.

Satz

Der Algorithmus liefert die Kosten der ungünstigsten Wege von s zu allen anderen, von s aus erreichbaren Knoten. Die Anzahl der Schritte beträgt $konst \cdot (n + m)$.

Netzplan

Definition

Ein *Netzplan* ist ein DAG, in dem die Kanten Vorgänge (Teilaufgaben eines größeren Vorhabens), die Knoten Ereignisse (Meilensteine, Teilergebnisse) darstellen. Für jede Kante ist die Dauer des Vorgangs als eine reelle Zahl gegeben. Es gibt zwei ausgezeichnete Knoten: den *Start (S)* und den *Abschluss (A)*, die in jeder topologischen Ordnung des Graphen an der ersten bzw. letzten Stelle sind. Jeder Knoten hat einen *frühstmöglichen (ASAP)* und einen *letztmöglichen (ALAP)* Zeitpunkt (diese sind nicht gegeben, sondern müssen berechnet werden). Die Differenz zwischen diesen Zeitpunkten ist die *Pufferzeit* des Knotens. Die Pufferzeit einer Kante xy ist ALAP(y)-ASAP(x)-Dauer(xy). Wenn die Pufferzeit eines Knotens / einer Kante 0 ist, ist es ein *kritischer Knoten* bzw. eine *kritische Kante*. Die kritischen Knoten und Kanten bilden den kritischen Teilgraphen, der oft ein Weg ist (der kritische Weg)

PERT

PERT Methode (Project Evaluation and Review Technique)

Input: Netzplan.
Output: ASAP- und ALAP-Zeitpunkte der Knoten, kleinstmögliche Dauer.
Ablauf:
1. Bestimmung einer topologischen Ordnung $v_1, v_2, \ldots, v_n$.
2. Bestimmung der ungünstigsten Wege von S zu allen anderen Knoten. Die Kosten dieser Wege definieren die ASAP-Zeitpunkte. Speziell der ASAP-Zeitpunkt für A ist die kleinstmögliche Dauer des gesamten Vorhabens. Sei ALAP(A) = erlaubte Dauer.
3. Die Kanten und die topologische Ordnung werden in umgekehrter Reihenfolge betrachtet und die ungünstigsten Wege von A zu allen anderen Knoten werden berechnet. Diese definieren die ALAP-Zeitpunkte:
(erlaubte Dauer) - (Länge des ungünstigsten Weges von A aus).

Satz

Dieser Algorithmus liefert in $konst \cdot (n + m)$ Schritten die ASAP- und ALAP-Zeitpunkte aller Knoten (und damit auch die Pufferzeiten aller Knoten und Kanten bzw. die kritischen Knoten und Kanten).