

Inf Progtech Übung: Topologisches Sortieren

Klaus Kusche

Du findest bei der Angabe ein Rahmenprogramm, in dem die Funktion für das topologische Sortieren fehlt.

Schreib diese Funktion. Sie sollte wie folgt arbeiten:

- Du brauchst intern eine Liste von Knoten. Sie enthält alle jene Knoten, die keinen unsortierten Vorgänger haben, d.h. die sofort in das Ergebnis der topologischen Sortierung übernommen werden können.
- Im ersten Schritt wird die Anzahl der Vorgänger aller Knoten auf 0 gesetzt.
- Im zweiten Schritt wird die Adjazenzmatrix durchlaufen und für jede Kante die Anzahl der Vorgänger im Zielknoten um eins erhöht.
- Im dritten Schritt werden nochmals alle Knoten durchlaufen und dabei alle jene Knoten in die Liste gehängt, bei denen die Anzahl der Vorgänger 0 ist.
- Der vierte Schritt ist der eigentliche Sortier-Schritt: Es wird immer wieder der vorderste Knoten der Liste aus der Liste entnommen und in das Ergebnis übernommen, bis die Liste leer ist.

In unserem Fall bekommt der gewählte Knoten einfach die nächste Positionsnummer im Ergebnis (beginnend bei 0), siehe unten.

Weiters muss für alle Kanten, die vom gewählten Knoten ausgehen, die Vorgängeranzahl im Zielknoten der Kante um eins erniedrigt werden. Sinkt sie dabei in einem Knoten auf 0, so wird dieser Knoten frisch in die Liste gehängt.

- Ist die Liste leer, so wird geprüft, ob alle Knoten in das Ergebnis übernommen worden sind.

Wenn ja war die Sortierung erfolgreich, wenn nein enthält der Graph Zyklen.

Hinweise:

- Der Graph ist bereits in globalen Variablen als Array von Knoten und Adjazenzmatrix gespeichert, siehe Rahmenprogramm.
- Beachte, dass die Verkettung der vorgängerlosen Knoten zu einer Liste diesmal mit Indices und nicht mit Pointern erfolgt (siehe Deklaration der Knoten-Struktur).

Einfacher ist es, neue Knoten ohne Vorgänger vorne an die Liste anzuhängen. Die etwas "natürlicheren" Sortierungen erhält man allerdings, wenn man neue Knoten ohne Vorgänger hinten an die Liste anhängt.

- Zur graphischen Darstellung wird wieder die SDL verwendet: Compiliere dein Programm wie in der SDL-Installationsanleitung angegeben.

In horizontaler Richtung ist die Position der Knoten fix (durch deren Index im Array bestimmt).

Die Position eines Knotens in der topologischen Sortierung ist im Member **pos** seiner Knoten-Struktur angegeben: Dieses Member bestimmt die Höhe des Knotens in der graphischen Darstellung, sie wird durch die Sortierung verändert.

Am Anfang entspricht **pos** dem Index des Knotens. Wenn deine Sortier-Funktion einem Knoten die nächste Positionsnummer im sortierten Ergebnis geben will, muss sie die Funktion **animate** mit der alten und der neuen Position des Knotens aufrufen, wobei die neue Position kleinergleich der alten Position sein muss.

Das bewirkt Folgendes:

- Der Knoten mit der angegebenen alten Positionsnummer steigt in der Grafik langsam an seine neue Position auf.
 - Seine Positionsnummer ändert sich dabei von der alten auf die neue.
 - Die Nummer aller anderen Knoten zwischen alter und neuer Nummer wird um eins erhöht, d.h. diese Knoten rutschen in der Sortierung einen Platz nach unten.
- Programmaufruf zum Testen: Das Programm wird mit der Anzahl der Knoten und der "Kantendichte" aufgerufen. Es erzeugt einen zufälligen Graphen, wobei die Kantendichte regelt, wie viel Prozent aller möglichen Kanten im Graph zufällig erzeugt werden (100 würde einen vollständigen Graph ergeben, 0 einen Graph ohne Kanten).

Je höher die Kantendichte ist, umso größer ist die Wahrscheinlichkeit von Zyklen und damit eines Versagens der topologischen Sortierung. Für wenige Knoten (5-10) hat man mit einer Kantendichte von 10-20 % gute Chancen auf einen zyklensfreien Graphen, bei vielen Knoten (30 und mehr) sind 1-5 % empfohlen.