Grundzüge von Algorithmen und Datenstrukturen, WS 15/16: Lösungshinweise zum 11. Übungsblatt

Markus Bläser

Aufgabe 11.1 Der Algorithmus führt Tiefensuche auf dem gerichteten Graph durch (Algorithmus DFS_explore aus der Vorlesung). Immer, wenn die Untersuchung eines Knotens vollständig beendet ist (Status wird auf "finished" gesetzt), wird dieser Knoten an eine anfangs leere doppelt verkettete Liste angehängt. Diese Liste in umgekehrter Reihenfolge enthält die Knoten topologisch sortiert (wird noch gezeigt).

Laufzeit: Laufzeit von Tiefensuche: $O(|V| + |E|)$ plus $O(|V|)$ für das Umkehren der Liste.

Wir müssen nun für jede beliebige gerichtete Kante (u, v) zeigen, dass bei der Tiefensuche, die Untersuchung von v *vor* der von u abgeschlossen wird.

1. *Fall: wenn v entdeckt wird, ist u noch nicht entdeckt worden.* Wird die Untersuchung von u vor der von v abgeschlossen, so muss es einen gerichteten Weg von v nach u geben. Zusammen mit der Kante (u, v) ist dies aber ein Kreis. Dies widerspricht der Voraussetzung, dass der Graph kreisfrei ist.

2. *Fall: wenn v entdeckt wird, ist u schon entdeckt worden.* Dann ist entweder die Untersuchung von u zum Entdeckungszeitpunkt von v bereits abgeschlossen, woraus die Behauptung folgt. Oder aber die Untersuchung von u ist noch nicht abgeschlossen. Daraus folgt, dass der Aufruf von Depth_first_search(v) ein Unteraufruf von Depth_first_search(u) ist. Daher wird die Untersuchung von v vor der von u abgeschlossen.

Aufgabe 11.2 a) Beweis per Induktion in k. Klar für $k = 0$, denn dann ist A^k die Einheitsmatrix. Für den Induktionsschritt $k \to k + 1$ können wir annehmen, dass $a_{i,j}^{(k)}$ die Anzahl der Wege von i nach j der Länge k sind. Es gilt

$$a_{i,j}^{(k+1)} = \sum_{h=1}^{n} a_{i,h}^{(k)} a_{h,j}.$$

Die Anzahl der Wege der Länge $k + 1$ von i nach j ist die Anzahl aller Wege der Länge k zu irgendeinem Knoten h, von dem es eine Kante nach i gibt. Dies berechnet gerade der Ausdruck.

b) Es empfiehlt sich, statt über den ganzen Zahlen in der Booleschen Algebra zu rechnen (d.h. $+$ wird $\vee$ und $\cdot$ wird $\wedge$. Matrixmultiplikation ist dann immer noch

assoziativ, insbesondere dann auch A^k wohldefiniert. (Hintergrund: So werden die Zahlen nicht so groß.)

Jetzt muss man nur A, A^2, $\ldots$, A^{n-1} berechnen und schauen, ob es einen Pfad der Länge i für ein $1 \leq i \leq n-1$ gibt.Denn wenn es einen Weg gibt, so gibt es einen der Länge $\leq n-1$. Laufzeit $O(n^4)$ naiv. Hängt man an jeden Knoten eine Schleife (dies entspricht "verodern" von A mit der Einheitsmatrix), so gibt es einen Weg der Länge $\leq n-1$ genau dann wenn es einen Weg der Länge $= n-1$ gibt. Dann reicht es A^{n-1} auszurechnen. Dies geht mit $O(n^3 \log n)$ mittels Square-and-Multiply.

Will man noch einen Pfad konstuieren, so muss man zu jeder 1 in A^k einen Pfad als "Zeugen" speichern. Ist im Berechnen des Matrixproduktes $b_{i,h} \wedge c_{h,j} = 1$, so ist die Konkatenation der beiden Zeugen ein Zeuge für $a_{i,j} = \bigvee b_{i,h} \wedge c_{h,j} = 1$. Verwendet man verkettete Listen, so geht die Berechnung der Zeugen ohne zusätzlichen Aufwand.