

9. Übungsblatt

Abgabe: Freitag, 22.6.2007, vor der Übung

Aufgabe 33

5 Punkte

In Aufgabe 30 habt Ihr gezeigt, wie man durch *capacity scaling* einen maximalen Fluss in einem Flussnetzwerk durch mehrere Berechnungen von maximalen Flüssen in dem Flussnetzwerk mit 0-1-Kapazitäten berechnen kann.

- Welche Laufzeit hat ein in dieser Hinsicht guter Max-Flow-Algorithmus auf einem Flussnetzwerk mit 0-1-Kapazitäten?
- Auf welchen Flussnetzwerken stellt *capacity scaling* die effizientere Technik zur Berechnung eines maximalen Flusses dar?

Die Aufgabe ist so gestellt nicht ganz richtig; siehe z.B. Ahuja, Magnanti, Orlin: Network Flows, pp. 210-212, für die richtige Formulierung.

Aufgabe 34

5 Punkte

Sei ein Flussnetzwerk (G, u, s, t) gegeben. Gebt einen auf dem Preflow-Algorithmus basierenden Algorithmus an, der einen minimalen s - t -Schnitt berechnet und beweist seine Korrektheit.

Wichtig: Die Laufzeit dieses Algorithmus soll kürzer sein als die eines Preflow-Algorithmus zur Berechnung eines maximalen Flusses. Gemeint ist hier ausnahmsweise nicht die asymptotische, sondern die tatsächliche Laufzeit.

Aufgabe 35

5 Punkte

In der Vorlesung habt ihr *maximum adjacency orderings* kennengelernt.

- Gebt einen Algorithmus mit Laufzeit $\mathcal{O}(m \log m)$ an, der zu gegebenem Graphen ein maximum adjacency ordering seiner Knoten berechnet. Beweist seine Korrektheit und Laufzeit.
- Welche Ähnlichkeiten seht ihr zum Algorithmus von Prim und zum Dijkstra-Algorithmus?

Aufgabe 36

5 Punkte

In einem Wettbewerb treten n Mannschaften an. Jede Mannschaft spielt gegen jede andere genau einmal und es gibt kein Unentschieden. Der Vektor $w = (w_i)_{1 \leq i \leq n}$, in dem w_i die Anzahl der Siege von Mannschaft i bezeichnet, heißt der *Gewinnvektor* des Wettbewerbs.

Entwickelt einen Algorithmus, der zu einem gegebenen Vektor $w \in \mathbb{N}^n$ entscheidet, ob er der Gewinnvektor eines solchen Wettbewerbs ist und zeigt seine Korrektheit und Laufzeit.

Programmieraufgabe 7 (Abnahme während einer der RBs am 22.6.2007):

Implementiert den Algorithmus von Nagamochi und Ibaraki zur Bestimmung eines minimalen Schnitts aus der Vorlesung. Gebt sowohl die den Schnitt definierende Knotenmenge als auch die Menge der Kanten im Schnitt mit der Summe ihrer Kapazitäten aus. Achtet wie immer auch auf die Einhaltung der bewiesenen asymptotischen Laufzeit des Algorithmus (hier $\mathcal{O}(nm \log m)$) durch Eure Implementation.