UNIVERSITÄT KARLSRUHE (TH)
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders, Dr. Rob van Stee
G. Veit Batz, Johannes Singler

4. Übungsblatt zur Algorithmentechnik WS 2007/08

http://algo2.iti.uni-karlsruhe.de/algotech.php
{sanders|vanstee|batz|singler}@ira.uka.de

Aufgabe 1 (*Zertifikate für starke Zusammenhangskomponenten, Schwierigkeit* 2)
Sei G ein stark zusammenhängender gerichteter Graph, und sei a ein Knoten von G. Zeigen Sie auf, wie man zwei Bäume in G mit Wurzel a konstruiert, die folgende Eigenschaften haben: Der erste Baum beweist, dass alle Knoten von a erreicht werden können, der zweite Baum beweist, dass a von allen Knoten erreicht werden kann. Ein Algorithmus zur Berechnung der starken Zusammenhangskomponenten könnte diese Bäume für jede Komponente zurückgeben, um die korrekte Lösung anzuzeigen.

Aufgabe 2 (*Routenplanung für öffentlichen Verkehr, Schwierigkeit* $1 + 1 + 2 + 2 + 1$)
Das Finden von schnellsten Verbindungen im öffentlichen Verkehr (d. h. nach Fahrplänen) kann als Kürzeste-Wege-Problem in einem azyklischen Graphen modelliert werden. Betrachten wir einen Bus oder Zug, der den Halt p zum Zeitpunkt t verlässt, und seinen nächsten Halt p' zum Zeitpunkt $t' > t$ erreicht. Diese Verbindung kann als Kante mit Gewicht $t' - t$ angesehen werden, die die Knoten (p, t) und (p', t') verbindet. Außerdem muss für jede Haltestelle p und jede Ankunft oder Abfahrt bei p, z. B. zu den Zeiten t und $t' > t$, eine Wartekante mit Gewicht $t' - t$ von (p, t) nach (p, t') eingefügt werden.

a) Zeigen Sie, dass der resultierende (gerichtete) Graph azyklisch ist.

b) Ein zusätzlicher Knoten muss für jede Anfrage die Position in Raum und Zeit repräsentieren. Wie wird dieser mit dem Graphen verbunden?

c) Nehmen wir an, der Baum der kürzesten Wege vom Startknoten zu allen erreichbaren Knoten wurde in dem konstruierten Graph berechnet. Wie findet man nun die eigentliche Verbindung heraus?

d) Konstruieren Sie den Graphen für die folgenden zwei Linien von 14.00 bis 15.30 Uhr:

Weihnachtsmarkt-Straßenbahn	Ankunft	Abfahrt
Universität		14.00
Marktplatz	14.03	14.04
Bahnhof	14.14	14.16
Universität	14.29	
. . . entsprechend alle 10 Minuten		

KIT-Shuttle	Ankunft	Abfahrt
Universität		14.00
Neureut	14.10	14.11
FZK	14.25	14.30
Neureut	14.44	14.45
Universität	14.55	
. . . entsprechend alle 15 Minuten		

e) Finden Sie im Graphen aus d) aus die schnellste Verbindung vom Bahnhof zum Forschungzentrum ab 14.18 Uhr.

Aufgabe 3 (*Kürzeste Wege, Schwierigkeit* $1 + 3 + 2$)
Sei $G = (V, E)$ ein gerichteter Graph und $c : E \to \mathbb{R}$ eine allgemeine reelle Kantengewichtung (d. h. es kann auch $c(e) < 0$ gelten für $e \in E$).

a) Zeigen Sie: Alle Teilwege eines kürzesten Weges sind ebenfalls kürzeste Wege, d.h wenn pqr kürzester Weg in G ist für drei Wege p, q und r, dann ist auch q kürzester Weg.

b) Nehmen Sie nun an, dass in G alle Kantengewichte aus $\mathbb{N}_{>0}$ stammen und die Addition stets $\mathcal{O}(1)$ Zeit benötigt. Verändern Sie den Dijkstra-Algorithmus aus der Vorlesung dahingehend, dass er zu jedem Knoten $v \in V$ statt eines kürzesten Weges vom Startknoten s nach v die *Anzahl* der kürzesten Wege von s nach v liefert. Ihr Algorithmus soll dabei die gleiche asymptotische Laufzeit wie der Dijkstra-Algorithmus haben, nämlich

$$T = \mathcal{O}\Big(m \cdot T_{decreaseKey}(n) + n \cdot \big(T_{deleteMin}(n) + T_{insert}(n)\big)\Big)$$

für $n := |V|$, $m := |E|$ und eine beliebige Implementierung der NodePQ. Argumentieren Sie, warum Ihr Algorithmus tatsächlich das gewünschte asymptotische Laufzeitverhalten hat.

c) Geben Sie ein Familie von Graphen an, so dass die Gesamtzahl der kürzesten Wege exponentiell in der Anzahl der Knoten ist.

Aufgabe 4 (*Bidirektionale Dijkstra-Suche, Schwierigkeit* $2+3+2+2+2$)
Gegeben sei ein gerichteter Graph $G = (V, E)$ mit nicht-negativen Kantengewichten, sowie zwei Knoten $s, t \in V$. Gesucht sei ein kürzester Weg in G von s nacht t. Wie in der Vorlesung erklärt, kann diese Frage auch mit Hilfe einer *bidirektionalen* Dijkstra-Suche beantwortet werden.

a) Nach Vorlesung kann man die bidirektionale Suche beenden, sobald man irgend einen Knoten u aus *beiden* NodePQs entnommen hat. Allerdings muss es keinen kürzesten Weg von s nach t geben, der über dieses u führt (was ja einen weiteren Verarbeitungsschritt erforderlich macht). Geben Sie ein Beispiel für eine solche Situation an (also einen gewichteten Graphen samt den Knoten s und t, sowie eine Strategie zum Umschalten zwischen Vorwärts- und Rückwärtssuche).

b) Nun habe G die Form eines 2D-Gitters wobei alle Kanten das Gewicht 1 haben (siehe Abbildung 1). Um welchen Faktor ist die bidirektionale Suche in dieser Situation schneller als die unidirektionale Suche? Mit welcher Strategie muss dabei zwischen Vorwärts- und Rückwärtssuche umgeschaltet werden? Nehmen Sie vereinfachend an, dass sich G nach allen Seiten unendlich weit ausdehnt.

c) Das Gitter G sei nun nicht mehr unendlich sondern „quadratrisch“ mit $\ell \times \ell$ Knoten. Welchen Einfluss hat das auf die Verbesserung der Laufzeit?

d) Verallgemeinern Sie das unendliche 2D-Gitter aus Teilaufgabe b) auf die Dimension $k \in \mathbb{N}$. Um welchen Faktor ist die bidirektionale Suche schneller für die jeweilige Dimesion k?

e) Stellen Sie eine Strategie auf, wie während der bidirektionalen Suche zwischen Vorwärts- und Rückwärtssuche hin und her geschaltet werden soll. Ziel soll dabei sein, dass die bidirektionale Suche *niemals* mehr als doppelt soviele Knoten betrachtet wie die unidirektionale Suche.

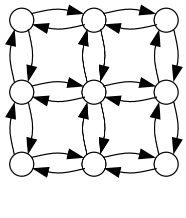

Abbildung 1: Gerichteter Graph von der Form eines 2D-Gitters mit 3×3 Knoten.