Technische Universität München
Institut für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr
Jan Griebsch, Stefan Pfingstl

SS 2005
Aufgabenblatt 3
25. April 2005

Praktikum Diskrete Optimierung

(Abgabetermin: Montag, den 2.05.2005, $14.^{00}$ Uhr)

Kürzeste Pfade

Aufgabe 1 Algorithmus von Dijkstra

Gegeben sei ein gerichteter Graph $G = (V, E)$ mit positiven Kantengewichten, bei dem jeder Knoten von jedem anderen aus erreichbar ist. Implementieren und animieren Sie den Algorithmus von Dijkstra so, dass die kürzesten Pfade von einem vom Benutzer gewählten Startknoten zu allen anderen Knoten berechnet und dargestellt werden. Bei jedem Knoten soll die Länge des berechneten Pfades zu diesem Knoten als Knotenlabel angezeigt werden.

Während der Algorithmus abläuft, sollen zu jeder Zeit die schon erledigten Knoten und die in der Priority-Queue befindlichen Knoten farblich gekennzeichnet sein. Ebenso sollen die bereits gefundenen kürzesten Pfade gut erkennbar sein. Für jeden Knoten in der Priority-Queue soll zusätzlich diejenige Kante markiert werden, über die er momentan am günstigsten von einem schon erledigten Knoten aus erreicht werden kann.

Aufgabe 2 Algorithmus von Bellman und Ford

Gegeben sei ein gerichteter Graph $G = (V, E)$ mit Kantengewichten $c : E \to \mathbb{R}$, die auch negativ sein dürfen. Implementieren Sie den Algorithmus von Bellman und Ford, der die kürzesten Pfade von einem benutzergewählten Startknoten zu allen von dort aus erreichbaren Knoten in Zeit $O(|V| \cdot |E|)$ berechnet. Wenn die Queue nach $|V|$ Phasen noch nicht leer ist, soll der Algorithmus melden, dass der Graph einen negativen Zyklus enthält, und einen solchen Zyklus effizient in Zeit $O(|V|)$ berechnen und am Bildschirm markieren.

Animieren Sie den Algorithmus so, dass der Benutzer den Ablauf gut mitverfolgen kann und dass zu jeder Zeit erkennbar ist, welche vorläufigen kürzesten Pfade der Algorithmus schon berechnet hat und welche Knoten sich gerade in der Queue befinden. Es soll auch angezeigt werden, in der wie vielten Phase sich der Algorithmus gerade befindet. Am Ende des Algorithmus soll der vollständige Kürzeste-Pfade-Baum sichtbar sein.

Hinweise

Sie können den Algorithmus aus Aufgabe 1 mit den Graphen `pos1.gw` und `pos2.gw` testen. Als Eingabe für Aufgabe 2 stehen die Graphen `neg1.gw` bis `neg5.gw` zur Verfügung. Die

Graphen `neg3` bis `neg5` enthalten negative Zyklen. Die Gewichte der Kanten sind wieder als Strings im User-Label gespeichert und sollten in ein `edge_array<double>` eingelesen werden.

Beachten Sie, dass die bearbeiteten Graphen dieses Mal gerichtet sind. Ihre Programme sollten daher ein `gw.set_directed(true)` und *kein* `g.make_undirected()` enthalten.

Aufgabe 3 Wettbewerb [optional]

Lösen Sie in möglichst kurzer Zeit das Single Source Shortest Path Problem für Graphen mit 1000 Knoten, die von dem gegebenen Graphgenerator erzeugt wurden. Die erzeugten Graphen haben beschränkte Kantengewichte zwischen 11 und 19. Versuchen Sie diese Tatsache algorithmisch auszunutzen.

- Animationen sind nicht notwendig, allerdings soll der kürzeste Wege Baum durch Erhöhung der Kantendicke um 1 kenntlich gemacht werden.
- Die Loesungen sollten mit dem Standard-Makefile kompilieren, und keinen Low-Level Code enthalten.
- Laden Sie den Graphen, berechnen Sie die Wege, änderen Sie die Baumkanten, und speichern Sie dann den geänderten Graphen. Gemessen wird die Laufzeit mit dem Shell Kommando `time`.