Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Dr. Hanjo Täubig
Tobias Lieber

Sommersemester 2011
Übungsblatt 6
19. September 2011

Grundlagen: Algorithmen und Datenstrukturen

Abgabetermin: 19. September 2011

Hausaufgabe 1

Implementieren Sie in der Klasse `SSSP` den Algorithmus von Dijkstra zur Berechnung der kürzesten Wege von einem Knoten s zu allen anderen Knoten in dem ungerichteten Graphen G.
Verwenden Sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die Klasse `SSSP` aber auf keinen Fall deren Interface `ISSSP`.
Hinweis: Verwenden Sie grundlegende Datenstrukturen, wie sie Java bereitstellt, und implementieren Sie diese *nicht* neu.

Hausaufgabe 2

Implementieren Sie in der Klasse `MST` den Algorithmus von Jarnik und Prim zur Berechnung eines Minimum Spanning Tree beginnend von einem Knoten s von einem ungerichteten Graphen G.
Der Algorithmus von Jarnik und Prim wählt zu einer Menge V' den Knoten der über eine Kante mit dem geringsten Gewicht erreichbar ist und fügt Ihn der Menge V' hinzu. Gestartet wird der Algorithmus mit $V' = s$.
Verwenden Sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die Klasse `MST` aber auf keinen Fall deren Interface `IMST`.
Hinweis: Verwenden Sie grundlegende Datenstrukturen, wie sie Java bereitstellt, und implementieren Sie diese *nicht* neu.

Aufgabe 1

Bestimmen Sie die Laufzeit des BFS-Algorithmus. Betrachten Sie zwei Implementierungen: Zum einen basierend auf einer Adjazenzmatrix und zum anderen basierend auf einer Adjazenzliste.

Aufgabe 2

Gegeben Sei die Adjazenzmatrix $A = (a_{ij})_{1 \leq i,j \leq n}$ eines gerichteten, einfachen Graphen G mit n Knoten. Geben Sie einen Algorithmus an, der eine Senke in $\mathcal{O}(n)$ Schritten findet. Eine Senke sei (abweichend von der üblichen Definition) ein Knoten v, der Eingrad $n-1$ und keine ausgehenen Kanten hat.

Aufgabe 3

In dieser Aufgabe wollen wir ein alternatives Verfahren zur topologischen Sortierung auf einem DAG G betrachten. Dazu nehmen wir an, dass uns weder die üblichen Graphoperationen zum Traversieren einer Kante oder Abfragen der Nachbarschaft eines Knotens, noch irgendwelche „höheren Datenstrukuren“ wie z.B. Queues, Listen, PriorityQueues oder Stacks zur Verfügung stehen.
Die einzige Operation, die wir benutzen können ist eine Tiefensuche (DFS), die Arrays `dfs_num` und `finish_num` zurückgibt, die zu den jeweiligen Knoten die entsprechenden Werte beeinhalten. Außerdem stehen uns `for`-Schleifen und die übliche Addition bzw. Subtraktion zur Verfügung.

a) Geben Sie einen Algorithmus (Pseudocode reicht) an, der dieselbe asymptotische Laufzeit erreicht, wie der in der Vorlesung vorgestellte Algorithmus.

b) Begründen Sie die Laufzeit Ihres Algorithmus kurz.

c) Beweisen Sie die Korrektheit Ihres Algorithmus.

Aufgabe 4

Argumentieren Sie, warum es unklug ist, bei der Berechnung eines kürzesten Weges von s nach t in einem Graphen mit echt positiven Gewichten alle möglichen (einfachen) Wege aufzuzählen und deren Distanz zu berechnen.
Benutzen Sie dazu exemplarisch ein Gitter der Größe n. Die n^2 Knoten werden durch die Tupel (x, y) mit $x, y \in [n]$ identifiziert.
Die Kanten $((x, y), (x + 1, y))$ seien gerichtet während die Kanten $((x, y), (x, y + 1))$ bidirektional verlaufen. Berechnen Sie die Anzahl aller Pfade von $s = (1, 1)$ zu $t = (n, n)$.

Aufgabe 5

Geben Sie einen möglichst effizienten Algorithmus an, der die *Anzahl* der kürzesten Wege von einem Knoten s zu allen anderen Knoten in einem ungerichteten Graphen G mit echt positiven Gewichten berechnet.