Algorithmische Mathematik I

Wintersemester 2011 / 2012
Prof. Dr. Sven Beuchler
Peter Zaspel

universitätbonn

Übungsblatt 6.

Abgabe am **30.11.2011**.

Aufgabe 1. (Binär Heaps)

Es sei die Zahlenfolge

$$7, 25, 172, 11, 23, 1, 43, 9, 58, 34, 12, 87, 45$$

gegeben.

a) Bauen sie einen Heap minimaler Tiefe, der die angegebenen Zahlen enthält

b) Fügen Sie die Werte 23 und 77 ein. Geben Sie hierbei die Teilschritte im Baum an.

c) Entfernen Sie die Einträge 25 und 43 aus dem Baum von Teilaufgabe b) und geben Sie erneut die Teilschritte an.

(5 Punkte)

Aufgabe 2. (Graphentraversierung)

Betrachten Sie den ungerichteten Graphen auf Seite 8 der PDF-Datei mit Graphen von der Webseite. Verwenden Sie die dort angegebenen Bennenungen der Knoten `K, H, BN, FU,...`. Ignorieren Sie jedoch die Kantengewichte.

a) Geben Sie die Adjazenz-Matrix für diesen Graphen an.

b) Führend Sie beginnend mit dem mit `K` bezeichneten Knoten eine Breiten-Suche durch. Folgen Sie dabei dem Beispiel aus der Vorlesung zu Algorithmus 3.1 mit der entsprechenden Wahl des Knoten v, so dass eine Breiten-Suche erfolgt.

c) Wiederholen Sie dieses Vorgehen fuer die Tiefensuche.

(5 Punkte)

Aufgabe 3. (Laufzeitkomplexität)

Beweisen Sie Lemma 3.9: Falls die Auswahl von v und w in $O(1)$ möglich ist, dann besitzt der Algorithmus die Komplexität $O(|V| + |E|)$.

(5 Punkte)

Aufgabe 4. (Starke Zusammenhangskomponenten)

Geben Sie ein Beispiel für einen gerichteten Graphen an, der drei starke Zusammenhangskomponenten hat, aber bei Uminterpretierung der Kanten als ungerichtete Kanten zusammenhängend ist.

(5 Punkte)

Programmieraufgabe 1. (Graphen-Durchmusterung)

Ziel dieser Aufgabe ist es, Tiefen- und Breiten-Suche in *ungerichteten* Graphen zu implementieren. Hierbei wollen wir uns erneut die Vorteile von effizienten Datenstrukturen in Erinnerung rufen. Gehen Sie nun wiefolgt vor:

(Die Teilaufgaben a) - c) ergeben zusammen 50% der Punkte.)

a) Implementieren Sie, basieren auf einer vollständig abgespeicherten Adjazenzmatrix das auf Algorithmus 3.1 beruhende Graphdurchmusterungsverfahren. Dieses soll sowohl als Tiefen- als auch als Breiten-Suche nutzbar sein. (Hierfür können verschiedene Methoden implementiert werden.)

b) Testen Sie das Verfahren auf Korrektheit mit dem Graphen aus Theorie-Aufgabe 2 (mit dem gleichen Startknoten) und führen Sie wie immer den *valgrind*-Test durch.

c) Erzeugen Sie folgenden Graphentypen:

- Einen vollständigen Binärbaum mit n Ebenen (die also alle vollständig besetzt sind).
- Ein 2D-Gitter mit $n \times m$ Knoten, d.h. n mal m Knoten in einer Rechtecks-Gitterstruktur, die jeweils mit ihren Knotennachbarn oben, unten, links und rechts (sofern vorhanden) verbunden sind.
- Einen vollständigen Graphen mit n Knoten, d.h. alle n Knoten sind mit allen anderen $n - 1$ Knoten verbunden.

Messen Sie für diese mit verschiedenen (ansteigenden) n, m die Laufzeit des Algorithmus und plotten Sie die Ergebnisse mit gnuplot. Starten Sie beim Baum in der Wurzel, im Gitter links oben und im vollständigen Graphen an einem beliebigen Knoten für die Traversierung.

d) Implementieren Sie nun die Algorithmen auf Basis von Adjazenzmatrizen im CSR-Format.

e) Testen Sie sie analog zu Aufgabenteil b).

f) Führen Sie analog Teil c) durch und interpretieren Sie (im Gespraech mit dem Tutor) die Ergebnisse vor dem Hintergrund der bekannten Aussagen über die Laufzeitkomplexität der Algorithmen.

Die Abgabe der Programmieraufgaben erfolgt in den CIP-Pools in der Woche vom 28.11. bis 02.12.2011. Die Listen für die Anmeldung zu den Abgabe-Terminen hängt in der Woche vom 21.11. bis 25.11.2011 aus.