# Algorithmische Mathematik I

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

universitätbonn

## Übungsblatt 5.

Abgabe am **23.11.2011**.

**Aufgabe 1.** (Wege)

Zeigen Sie: In einem Graphen $G = (V, E)$ gibt es einen x-y-Weg genau dann, wenn es einen Kantenzug von x nach y gibt.

(5 Punkte)

**Aufgabe 2.** (Kreisfreie Graphen)

Es sei $G = (V, E)$ ($|V| > 1$) ein maximal kreisfreier (ungerichteter) Graph, d.h. $G' = (V, E \cup e)$ ist nicht kreisfrei $\forall e \notin E$, aber $G = (V, E)$ ist es. Dann gibt es eine Ecke $v$ in $G$ mit $deg(v) = 1$.

(5 Punkte)

**Aufgabe 3.** (Kreise)

Zeigen Sie: Sei $G = (V, E)$ ein einfacher ungerichteter Graph mit $|V| = n$ und $|E| = m$. Dann gilt: Ist $m \geq n \geq 3$, so besitzt $G$ einen Kreis.

(5 Punkte)

**Aufgabe 4.** (Starke Zusammenhangskomponente)

In der Vorlesung wurden bisher nur Zusammenhangskomponenten auf ungerichteten Graphen definiert. Natuerlich ist eine Analoge Definition auf gerichteten Graphen moeglich. Hierbei spricht man von *starken Zusammenhangskomponenten*:

**Definition (Starke Zusammenhangskomponente)**
Sei G=(V,E) ein einfacher gerichteter Graph und $C \subseteq V$. $C$ heißt starke Zusammenhangskomponente, falls folgende Eigenschaften gelten:

1. Je zwei Ecken in $C$ sind voneinander erreichbar, es gibt also einen (gerichteten) Kantenzug von $v_1$ nach $v_2$ für alle $v_1 \neq v_2, \quad v_1, v_2 \in V$.
2. $C$ ist eine maximale Eckenmenge, welche die vorgenannte Eigenschaft besitzt, d.h. falls $C \subseteq C' \subseteq V$, so dass es einen Kantenzug von $v_1$ nach $v_2$ gibt, für alle $v_1 \neq v_2 \in C'$, so gilt $C = C'$.

Zeigen Sie nun folgende Aussage:
Sei $G$ ein einfacher gerichteter Graph und $v_1, v_2$ zwei Ecken in $G$. Dann gilt: $v_1, v_2$ liegen genau dann in der selben starken Zusammenhangskomponente, wenn die Mengen aller Nachfolgeknoten

$$Post^*(v) = \{w \in V | \text{es gibt einen Kantenzug von } v \text{ nach } w\} \cup \{v\}$$

für beide Ecken $v_1, v_2$ gleich ist, also $Post^*(v_1) = Post^*(v_2)$.

(5 Punkte)

**Programmieraufgabe 1.** (Speicherung dünnbesetzter Matrizen)

Für das Abspeichern von Graphen werden häufig sogenannte Adjazenzmatrizen eingesetzt. Diese haben $|V|$ viele Zeilen und Spalten. Für jede Kante existiert dann ein Eintrag in der Matrix (näheres hierzu folgt in der Vorlesung). Das Fehlen einer Kante zwischen zwei Knoten wird häufig durch einen Wert von Null in der Matrix dargestellt. Dies führt dazu, dass Adjazenzmatrizen fast immer viele Null-Einträge aufweisen.

Ein Abspeichern dieser Null-Einträge im Rechner wäre sehr ineffizient. Daher wurden in der Vergangenheit verschiedene Datenformate entwickelt, mit denen man diese sogenannten *dünnbesetzten* Matrizen speichereffizienter ablegen kann. Eines hiervon ist das CSR-Format (compressed sparse row).

Bei diesem Format wird die Matrix mit Hilfe dreier Vektoren `row_offsets`, `column_indices` und `entries` gespeichert. Der letztgenannte Vektor enthält alle nicht-Null-Einträge der Matrix zeilenweise sortiert. Für $i = 0...N-1$ ($N$ Anzahl der Zeilen) gibt `row_offsets[i]` die erste Indexposition in `entries` an, die nicht-Null-Elemente aus der $i$-ten Zeile enthält. Schließlich enthält `column_indices[j]` zum Eintrag `entries[j]` die entsprechende Spalten-Position. Um eine konsistente Abfrage der Anzahl von nicht-Null-Einträgen in der letzten Zeile zu erhalten, wird an `row_offsets[i]` ein weiterer Eintrag `row_offsets[N]` angehängt, welcher die Länge von `entries` beinhaltet.

Für die Matrix

$$\begin{pmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

ergibt sich dann die nachfolgende Darstellung:

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| entries | 1 | -2 | 1 | 1 |
| column_indices | 0 | 2 | 1 | 0 |
| row_offsets | 0 | 2 | 3 | 4 |

**Achtung:** Die Aufgabenteile a) bis einschließlich e) ergeben zusammen 50% der Punkte.

a) Schreiben Sie die Matrix

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 1 \\ 0 & 1 & 5 & 0 & 0 \\ 0 & 1 & 0 & 2 & 2 \\ 0 & 2 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 & 4 \end{pmatrix}$$

im CSR-Matrix-Format auf.

b) Wir erinnern uns, dass der i-te Eintrag des Ergebnis-Vektors aus der Multiplikation einer Matrix $A \in \mathbb{R}^{n \times m}$ mit einem Vektor $x \in \mathbb{R}^m$ durch $(Ax)_i = \sum_{j=1}^{m} A_{ij} x_j$ gegeben ist. Ueberlegen sie sich auf dem Papier einen Algorithmus, mit dem Sie die Matrix-Vektor-Multiplikation einer Matrix im CSR-Format durchführen können.

c) Implementieren Sie nun diese Matrix-Vektor-Multiplikation in einer Methode `void MatVecMultCSR(int* row_offsets, int* column_offsets, double* entries, double* x, double* y, int n, int m)`, wobei der Ausgabewert in das Feld `y` gespeichert wird. Sichern Sie diese Methode so, dass Sie diese später wieder verwenden können.

d) Vervollständigne Sie Ihr Testprogramm derart, dass sie damit die Matrix aus der ersten Teilaufgabe mit dem Vektor $(11111)^T$ multiplizieren.

e) Überprüfen Sie auch hier wieder ihr Programm mit *valgrind*.

f) Implementieren sie nun ebenfalls eine Matrix-Vektor-Multiplikation für eine vollständig abgespeicherte Matrix: `void MatVecMultDense(double* A, double* x, double* y, int n, int m)`. Die Matrix $A$ ist hierbei ein eindimensionales Array, das die Zeilen der Matrix (auch mit Null-Einträgen) hintereinander aufgereiht enthält.

g) Schreiben Sie nun eine Routine, die zu einem gegebenen $n$ randomisiert eine Matrix $A \in \mathbb{R}^{n \times n}$ erzeugt und diese mit allen Einträgen abspeichert und ebenfalls im CSR-Format speichert. Dabei soll die Matrix genau $n$ nicht-Null-Einträge haben.

h) Führen Sie nun eine Laufzeitanalyse (analog zum letzten Zettel) mit Plot-Ausgabe über gnuplot durch, bei der Sie die Laufzeit der beiden implementierten Matrix-Vektor-Multiplikationen für mehrere Matrizengrößen testen. Welches ist die offensichtliche asymptotische Komplexität der beiden Verfahren?

(15 Punkte)

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.

Die Fachschaft der Mathematik veranstaltet am 22.11.2011 eine Party. Mehr dazu erfahren Sie unter http://fsmath.uni-bonn.de/.