Institut für Informatik I
Prof. Dr. Heiko Röglin
Tobias Brunsch

Algorithmen und Berechnungskomplexität I
Wintersemester 2010/11
Abgabe: 13.12.2010 in der Vorlesung

Übungsblatt 8

Aufgabe 8.1

3+3 Punkte

Es sei A die Adjazenzmatrix eines (gerichteten oder ungerichteten) Graphen. Das Boolesche Matrixprodukt $Z = X * Y$ zweier $(n \times n)$-Matrizen X und Y über $\{0, 1\}$ ist definiert durch

$$z_{ij} = \bigvee_{l=1}^{n} (x_{il} \wedge y_{lj}) .$$

Dabei werden, wie üblich, 1 als *true* und 0 als *false* interpretiert. Es sei A^k die k-te Potenz von A, d.h. $A^1 = A$ und $A^k = A * A^{k-1}$ für $k \geq 2$.

(a) Welche Bedeutung haben die Einträge von A^k? Beweisen Sie Ihre Vermutung!

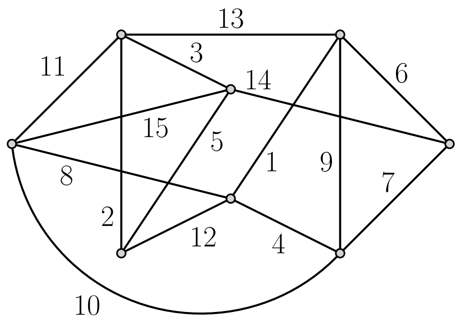
(b) Sei $f(n)$ der Aufwand, zwei gegebene $(n \times n)$-Matrizen X und Y über $\{0, 1\}$ zu multiplizieren. Beschreiben Sie, wie man für $k \geq 2$ in Zeit $O(\log(k) \cdot f(n))$ die k-te Potenz A^k einer Adjazenzmatrix A berechnen kann. Sie können davon ausgehen, dass das Boolesche Matrixprodukt assoziativ ist. Das bedeutet insbesondere, dass $A^{k_1} * A^{k_2} = A^{k_1+k_2}$ für beliebige natürliche Zahlen k_1 und k_2 gilt.

Hinweis: Überlegen Sie sich zunächst, wie man mit möglichst wenigen Multiplikationen spezielle große Potenzen von A berechnet. Nutzen Sie diese Idee, um A^k für beliebige Exponenten k schnell zu bestimmen.

Aufgabe 8.2

4+2 Punkte

(a) Bestimmen Sie mit Hilfe des Algorithmus von Kruskal einen minimalen Spannbaum des unten abgebildeten Graphen.



(b) Einige Probleme lassen sich mit Hilfe einer Divide-and-Conquer-Strategie lösen. Dabei zerlegt man ein kompliziertes Problem in einfachere Teilprobleme, löst diese (ggf. rekursiv) und bestimmt aus den Lösungen der Teilprobleme die Lösung des Gesamtproblems. Wir wollen diesen Ansatz für die Berechnung von minimalen Spannbäumen untersuchen.

Gegeben seien ein zusammenhängender ungerichteter Graph $G = (V, E)$ mit einer Kantengewichtung $w \colon E \to \mathbb{R}_{>0}$ und eine 2-Partition (V_1, V_2) der Knotenmenge V, d.h. $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$ und $V_1, V_2 \neq \emptyset$. Ein Divide-and-Conquer-Ansatz könnte wie folgt aussehen.

(1) Bestimme minimale Spannbäume von $G[V_1]$ und $G[V_2]$. (Für eine Menge $U \subseteq V$ bezeichnet $G[U]$ den auf U eingeschränkten Untergraphen von G, bestehend aus allen Knoten $u \in U$ und den Kanten von G, die innerhalb von U verlaufen.)

(2) Bestimme eine kostenminimale Kante zwischen Knotenmenge V_1 und Knotenmenge V_2.

(3) Vereinige die beiden Spannbäume und die Kante zu einem Spannbaum von G.

Geben Sie ein Beispiel für G, V_1 und V_2 an, bei dem dieser Ansatz keinen minimalen Spannbaum liefert.

Aufgabe 8.3 6 Punkte

Seien $G = (V, E)$ ein zusammenhängender ungerichteter Graph mit einer Kantengewichtung $w\colon E \to \mathbb{R}_{>0}$, C ein Kreis in G und e eine teuerste Kante von C. Zeigen Sie, dass es einen minimalen Spannbaum von G gibt, der die Kante e nicht enthält.

Hinweis: Nutzen Sie Ideen aus dem Korrektheitsbeweis des Algorithmus von Kruskal.

Aufgabe 8.4 3+3 Punkte

(a) Sei $G = (V, E)$ ein zusammenhängender ungerichteter Graph mit einer Kantengewichtung $w\colon E \to \mathbb{R}_{>0}$. Geben Sie einen Algorithmus an, der in Zeit $O(m \log m)$ einen **maximalen** Spannbaum bestimmt, d.h. einen Spannbaum T mit maximalem Gewicht $w(T)$. Dabei ist $m = |E|$ die Anzahl der Kanten von G.

(b) Beweisen Sie die Korrektheit Ihres Algorithmus.

Hinweis: Alle Spannbäume von G haben genau $|V| - 1$ Kanten.

Aufgabe 8.5 5+1 Zusatzpunkte

Gegeben sei ein zweidimensionales Feld A mit $s := m \cdot n$ Einträgen. Wir wollen zwei Funktionen `Init` und `Sum` bereitstellen. Die Funktion `Init` wird einmal am Anfang aufgerufen und dient als Vorverarbeitungsschritt. Das Ausführen dieser Funktion soll in Zeit $O(s)$ möglich sein. Die `Sum`-Funktion erhält als Parameter Indizes i_1, i_2 sowie j_1, j_2 mit $1 \le i_1 \le i_2 \le m$ und $1 \le j_1 \le j_2 \le n$ und soll in konstanter Zeit die Summe

$$\sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} A[i, j]$$

berechnen.

(a) Implementieren Sie die `Init`- und die `Sum`-Funktion.

(b) Die Idee lässt sich auf beliebige Dimensionen d übertragen, doch die Konstante c, die in der O-Notation der `Sum`-Funktion steckt, hängt von d ab. Geben Sie die Größe von c in Θ-Notation bzgl. d an.