Übung zur Vorlesung

Theoretische Informatik II

Prof. Christoph Kreitz, Dipl. Math. Eva Richter

Universität Potsdam, Theoretische Informatik — Wintersemester 2003/04

Blatt 6 — Abgabetermin: 23.01.04

Aufgabe 6.1

Neben der oberen Abschätzung des asymptotischen Verhaltens einer Funktion ($\mathcal{O}$) werden gelegentlich auch andere Arten der Abschätzung benutzt:
Untere Schranken: Definiere für eine Funktion $g : \mathbb{N} \to \mathbb{N}$ die Klasse von Funktionen $\Omega(g)$ durch

$$\Omega(g) = \{f : \mathbb{N} \to \mathbb{N} \mid (\exists c > 0)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)[f(n) \geq c \cdot g(n)]\}$$

Exakte Schranken: Definiere für eine gegebene Funktion $g : \mathbb{N} \to \mathbb{N}$ die Klasse von Funktionen $\Theta(g)$ durch

$$\Theta(g) = \mathcal{O}(g) \cap \Omega(g)$$

oder etwas ausführlicher:

$$\Theta(g) = \{f : \mathbb{N} \to \mathbb{N} \mid (\exists c_1, c_2 > 0)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)[c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)]\}.$$

Stellen Sie für die folgenden Paare von Funktionen jeweils fest, ob $f = \mathcal{O}(g)$, $f = \Omega(g)$ oder $f = \Theta(g)$ gilt.

(i)	$f(n) = \sqrt{n}$	$g(n) = 100n$
(ii)	$f(n) = \log_{10} n$	$g(n) = \log_2 n$
(iii)	$f(n) = \sqrt[4]{n}$	$g(n) = \sqrt{n}$
(iv)	$f(n) = n^3$	$g(n) = n \log_2 n$
(v)	$f(n) = 155n^2 + 24n + 13$	$g(n) = n^2$
(vi)	$f(n) = n \log_2 n + \sqrt{n}$	$g(n) = n$

Aufgabe 6.2

Beweisen Sie die folgenden Rechenregeln.$(\forall g_1, g_2 : \mathbb{N} \to \mathbb{N})(\forall f_1 \in \mathcal{O}(g_1))(\forall f_2 \in \mathcal{O}(g_2))$:

(i) $f_1 + f_2 \in \mathcal{O}(g_1 + g_2)$,

(ii) $f_1 \cdot f_2 \in \mathcal{O}(g_1 \cdot g_2)$.

Aufgabe 6.3

Diese Aufgabe befaßt sich mit der Berechnung der Komplexität eines „echten“Algorithmus– des sogenannten KRUSKAL-Algorithmus. Das Problem, das mit diesem Algorithmus gelöst werden soll, ist die Suche nach einem einen Graphen aufspannenden Baum mit minimalem Gewicht (Minimum Weight Spanning Tree MWST). Informell stellen wir uns einen Graphen vor als eine Menge von Knoten, die durchnumeriert werden und einer Menge von Kanten, die einige der Knoten miteinander verbinden. Zusätzlich wollen wir annehmen, daß jede Kante mit einem Gewicht ausgestattet wird, das durch eine Zahl gekennzeichnet werde. Ein aufspannender Baum ist eine Teilmenge von Kanten, sodaß alle Knoten durch Kanten miteinander verbunden sind; es gibt jedoch keine Zyklen. Ein aufspannender Baum mit minimalem Gewicht hat das kleinstmögliche Gesamtkantengewicht aller aufspannenden Bäume.

Der KRUSKAL-Algorithmus zur Lösung dieses Problem kann informell wie folgt beschrieben werden.

1. Speichere für jeden Knoten die *Zusammenhangskomponente*, in der der Knoten erscheint, und verwende dazu alle bisher ausgewählten Kanten des Baumes. Anfänglich sind keine Kanten ausgewählt, und somit steht jeder Knoten in einer Zusammenhangskomponente für sich allein.

2. Wähle aus der Menge der Kanten, die bisher noch nicht betrachtet wurden eine mit geringstem Gewicht aus. Verbindet diese Kante zwei Knoten, die sich zum aktuellen Zeitpunkt in verschiedenen Zusammenhangskomponenten befinden, dann

   a) wähle die Kante für den aufspannenden Baum und

   b) vereinige die beiden betroffenen Zusammenhangskomponenten durch die Änderung der Komponentennummer aller Knoten in beiden Zusammenhangskomponenten, sd. ihre Nummern übereinstimmen.

   Verbindet die gewählte Kante jedoch zwei Knoten der gleichen Zusammenhangskomponente, dann gehört sie nicht in den aufspannenden Baum; diese Kante würde zu einem Zyklus führen.

3. Betrachte weiterhin Kanten, bis entweder alle Kanten berücksichtigt wurden, oder die Anzahl der für den aufspannenden Baum ausgewählten Kanten um 1 niedriger ist als die Anzahl der Knoten ist. Beachten Sie, daß im zweiten Fall alle Knoten in der gleichen Zusammenhangskomponente liegen und wir daher keine weiteren Kanten berücksichtigen müssen.

a) Gegeben sei ein Graph mit 4 Knoten 1,2,3,4 und vier Kanten mit folgenden Gewichten: Kante(1,2):15, Kante(2,4):20, Kante(3,4):18, Kante(1,3):10. Überlegen Sie sich wie der MWST für diesen Graphen aussieht.

b) Angenommen wir ändern das Gewicht der Kante (1,3) von 10 in 25, welcher MWST würde daraus resultieren?

c) Überlegen Sie sich, wie man diesen Algorithmus implementieren kann. Benutzen Sie zur Darstellung einer solchen Implementation eine Pseudo-Programmiersprache, ähnlich wie auf den Folien 2,3 und 4 aus Abschnitt 8.2. Geben Sie eine Funktion g in e, der Anzahl der Kanten, und m, der Anzahl der Knoten eines vorgegebenen Graphen an, sd. Ihre Implementation in $\mathcal{O}(g)$ liegt.(Die Abschätzung muß nicht allzu genau sein.)