Universität Karlsruhe (TH)
Lehrstuhl Informatik für Ingenieure und Naturwissenschaftler
Dr. Thomas Worsch
Matthias Schulz

5. Mai 2008

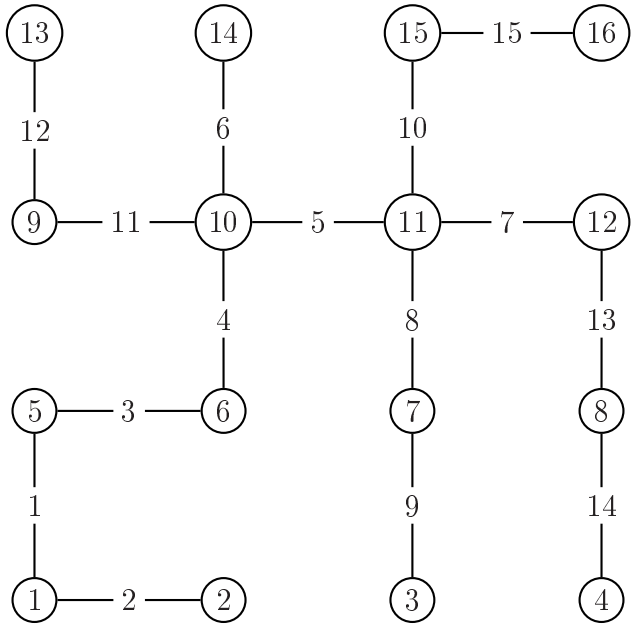
Einführung in die Informatik

Lösungen zu Übungsblatt 3

– O-Kalkül, Graphen –

Aufgabe 1:

a) Führt man den Algorithmus von Prim beginnend bei Knoten 1 durch (immer die Kante kleinsten Gewichtes zwischen den verbundenen Knoten und den nicht verbundenen Knoten hinzufügen), so ergibt sich folgender Graph:



Die Reihenfolge der Kanten ist durch die Kantenbeschriftungen gegeben.

b) Nach jeder während des Algorithmus zum Baum hinzugefügten Kante muss die minimale Kante zwischen den bereits im Baum enthaltenen und den nicht im Baum enthaltenen Knoten bestimmt werden. Geht man davon aus, dass Hinzufügen und Entfernen von Knoten aus Mengen in konstanter Zeit geht (man markiert die Knoten entsprechend), so bleibt n-mal die Suche nach einer minimalen Kante, die höchstens $O(m)$ Schritte braucht. Der Algorithmus läuft also in der Zeit $O(n \cdot m)$

   Eine geschicktere Lösung: Man sorgt dafür, dass die Kanten, die von den Knoten des Baumes ausgehen und nicht im Baum selbst enthalten sind, in einer aufsteigend geordneten Liste stehen, und überprüft, ob die erste (also kleinste) Kante in dieser Liste zwischen einem Knoten des Baumes und einem Knoten, der noch nicht im Baum enthalten ist, verläuft.

   Für das Einsortieren wird man jede Kante höchstens zweimal betrachten, und jede Kante wird höchstens einmal aus der Liste ausgelesen werden. Das Einsortieren in und Auslesen aus einer geordneten Liste von höchstens m Elementen geht (bei geschickter Datenstruktur $\rightarrow$ *AVL-Bäume*) in $O(\log m)$ und man erhält somit einen Aufwand von $O(n + m \log m)$, da auch jeder Knoten einmal besucht werden muss.

   An dieser Stelle wäre noch zu erwähnen, dass es einen aufspannenden Baum nur bei zusammenhängenden Graphen gibt, für die $n \leq m + 1$ gilt und man somit einen Aufwand in $O(m \log m)$ erhält.

Aufgabe 2: Für den Durchmesser D von U gilt $D = 3$. Dazu wird einfach die Länge aller kürzesten Wege bestimmt: Von einem Knoten aus werden zuerst alle Knoten bestimmt, die über eine Kante erreicht werden können; danach werden alle Knoten bestimmt, die von diesen Knoten über eine Kante erreicht werden können (und somit von dem Ausgangsknoten über zwei Kanten erreichbar sind). Dies wird wiederholt, bis alle Knoten erreicht wurden oder nur noch Knoten über neue Kanten erreicht werden können, die schon über weniger Kanten erreicht werden konnten.
Bei dem gegebenen Graphen verläuft dies wie folgt:
Von 1 aus können die Knoten 2, 4 und 6 erreicht werden; von den Knoten 2, 4 und 6 können die Knoten 1,2, 3, 4 und 5 erreicht werden. Somit können von Knoten 1 aus alle Knoten in V über zwei Kanten erreicht werden. (Kurzschreibweise: $1 \rightarrow 2, 4, 6 \rightarrow 3, 5$ (bereits besuchte Knoten werden nicht noch einmal aufgeführt.))
Für die anderen Knoten verläuft das Verfahren folgendermaßen:
$2 \rightarrow 1, 3, 4, 5 \rightarrow 6$
$3 \rightarrow 2, 4 \rightarrow 1, 5 \rightarrow 6$
$4 \rightarrow 1, 2, 3, 5 \rightarrow 6$
$5 \rightarrow 2, 4, 6 \rightarrow 1, 3$
$6 \rightarrow 1, 5 \rightarrow 2, 4, \rightarrow 3$
Der längste Pfad hat die Länge 3, und damit ist der Durchmesser 3.

Um die Bisektionsweite zu bestimmen, muß nach Definition eine minimale Kantenmenge M gefunden werden, durch deren Entfernen der Graph in zwei Teilgraphen zerfällt, deren Knotenzahl sich um höchstens eins unterscheidet. Bei 9 Kanten gibt es $2^9 = 512$ mögliche Kantenmengen, unter denen man die richtige finden muss. Andererseits gibt es nur $\binom{6}{3} = 20$ Möglichkeiten, V in zwei gleich große Teilmengen V_1 und V_2 mit $V_1 \cup V_2 = V, V_1 \cap V_2 = \emptyset$ aufzuteilen, wobei nur 10 Möglichkeiten wirklich verschieden ist, da man jeweils V_1 und V_2 vertauschen kann und eine andere "legale" Unterteilung erhält.
Sei V_1 jeweils die Teilmenge, die den Knoten 1 enthält; dann gibt es die folgenden möglichen Unterteilungen.
$V_1 = \{1,2,3\}, V_1 = \{1,3,4\}, V_1 = \{1,4,5\}, V_1 = \{1,5,6\}$
$V_1 = \{1,2,4\}, V_1 = \{1,3,5\}, V_1 = \{1,4,6\}$
$V_1 = \{1,2,5\}, V_1 = \{1,3,6\}$
$V_1 = \{1,2,6\}$
Die Kanten, die man jeweils entfernen muss, damit es zwischen Knoten in V_1 und Knoten in V_2 keine Verbindung gibt, ergibt sich jeweils dadurch, dass man die Anzahl der Kanten in E, die zwischen Knoten in V_1 verlaufen, sowie die Anzahl der Kanten in E, die zwischen Knoten in V_2 verlaufen, von der Anzahl der Knoten in E abzieht, da alle anderen Kanten zwischen Knoten in V_1 und Knoten in V_2 verlaufen müssen.
Die entsprechenden Werte sind jeweils
5, 5, 6, 4
5, 8, 5
6, 5
5.
Die Bisektionsweite B beträgt somit 4, $V_1 = \{1,5,6\}, V_2 = \{2,3,4\}$ und die Kanten, die man entfernen muss, sind $\{1,2\}, \{1,4\}, \{2,5\}$ und $\{4,5\}$.

Aufgabe 3:

a) $n \in O(n^2) \Leftrightarrow$ Es gibt eine positive Konstante C, so dass es eine natürliche n_0 gibt mit der Eigenschaft, dass $n < C \cdot n^2$ für alle $n \geq n_0$ gilt. Für jede Zahl $n \geq 2$ gilt $n < 2 \cdot n \leq n^2$, und somit gilt $n \in O(n^2)$.

b) Sei f eine Funktion in $O(n^2)$. Dann gibt es eine positive Konstante C und eine natürliche Zahl n_0, so dass $f(n) \leq C \cdot n^2$ für alle $n \geq n_0$ gilt. Da für $n \geq 2$ die Ungleichung $n^2 < 2 \cdot n^2 \leq n^3$ gilt, folgt für $n \geq \max\{n_0, 2\} : f(n) \leq C \cdot n^2 \leq C \cdot n^3$, und somit gilt $f \in O(n^3)$. Nach Teilmengendefinition gilt $O(n^2) \subseteq O(n^3)$.

c) Die Aussage ist wahr.

Es gilt auf jeden Fall $f \in O(f) = O(g)$ und $g \in O(g) = O(f)$. Nach Bemerkung 83 gilt damit auch $f \in \Omega(g)$ und $g \in \Omega(f)$.

Sei nun h eine Funktion in $\Theta(f)$. Das heißt, es gibt zwei positive reelle Zahlen c_1 und c_2 und eine natürliche Zahl n_0, so dass für alle $n \geq n_0$ gilt: $c_1 f(n) \leq h(n) \leq c_2 f(n)$.

Da $f \in O(g)$ gilt, gibt es eine positive reelle Zahl c_3 und eine natürliche Zahl n_1, so dass für alle $n \geq n_1$ gilt: $f(n) \leq c_3 g(n)$.

Da $f \in \Omega(g)$ gilt, gibt es eine positive reelle Zahl c_4 und eine natürliche Zahl n_2, so dass für alle $n \geq n_2$ gilt: $f(n) \geq c_4 g(n)$ oder $c_4 g(n) \leq f(n)$.

Sei nun $n_3 = \max\{n_0, n_1, n_2\}$. Dann gilt für alle $n \geq n_3$: $c_4 c_1 g(n) \leq c_1 f(n) \leq h(n) \leq c_2 f(n) \leq c_2 c_3 g(n)$
$\Rightarrow \forall n \geq n_3 : c_4 c_1 g(n) \leq h(n) \leq c_2 c_3 g(n)$.

Da $c_1 c_4$ und $c_2 c_3$ positive reelle Zahlen sind, liegt h in $\Theta(g)$ und es gilt somit $\Theta(f) \subseteq \Theta(g)$.

Umgekehrt lässt sich zeigen, dass $\Theta(g) \subseteq \Theta(f)$ gilt, und somit folgt die Behauptung.

d) Die Aussage ist falsch.

$2^n \in O(2^n)$. Angenommen, $2^n \in O((n+1)^{-1} \cdot 2^n)$.

Dann gibt es eine positive reelle Zahl c und eine natürliche Zahl n_0, so dass für alle $n \geq n_0$ gilt:

$2^n \leq c \cdot (n+1)^{-1} \cdot 2^n \iff 1 \leq c \cdot (n+1)^{-1} \iff n+1 \leq c$.

Da für $n = \max\{\lfloor c \rfloor, n_0\}$ ist $n+1$ aber größer als c und größer als n_0.

Damit kann 2^n nicht in $O((n+1)^{-1} \cdot 2^n)$ liegen.

e) Sei $g \in O(f(n)) \cap \Omega(f(n))$. Dann gilt:

$\exists C_1 \in \mathbb{R}_+ \exists n_1 \in \mathbb{N}_+ : \forall n > n_1 : g(n) \leq C_1 f(n)$
$\exists C_2 \in \mathbb{R}_+ \exists n_2 \in \mathbb{N}_+ : \forall n > n_2 : g(n) \geq C_2 f(n)$.

Sei nun $n_0 = \max\{n_1, n_2\}$. Dann gilt:

$\forall n > n_0 : C_2 f(n) \leq g(n) \leq C_1 f(n)$
$\Rightarrow O(f(n)) \cap \Omega(f(n)) \subseteq \Theta(f(n))$.

Sei $g \in \Theta(f(n))$. Dann gilt:

$\exists C_1 \in \mathbb{R}_+ \exists C_2 \in \mathbb{R}_+ \exists n_0 \in \mathbb{N}_+ \forall n > n_0 : C_2 f(n) \leq g(n) \leq C_1 f(n)$
$\Rightarrow \exists C_1 \in \mathbb{R}_+ \exists n_0 \in \mathbb{N}_+ \forall n > n_0 : g(n) \leq C_1 f(n) \wedge$
$\exists C_2 \in \mathbb{R}_+ \exists n_0 \in \mathbb{N}_+ \forall n > n_0 : g(n) \geq C_2 f(n)$
$\Rightarrow g \in O(f(n)) \cap \Omega(f(n)) \Rightarrow \Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$.

f) Angenommen, $O(n^2) \subseteq \Theta(n^3)$ stimmt; dann müsste auch $n^2 \in \Omega(n^3)$ gelten (wegen Aufgabe e)), und somit müsste es eine Konstante $C > 0$ und eine natürliche Zahl n_0 geben, so dass $n^2 \geq C \cdot n^3$ für alle $n > n_0$ gilt.

$\Rightarrow \forall n > n_0 : 1 \geq C \cdot n$.

Da $C \cdot n$ jedoch unbeschränkt wächst, ergibt sich ein Widerspruch, und somit gilt $O(n^2) \not\subseteq \Theta(n^3)$.

g) Die Aussage ist wahr.

Sei $h \in O(f)$ eine Funktion. Dann gibt es eine positive reelle Zahl c_1 und eine natürliche Zahl n_0, so dass für alle $n \geq n_0$ gilt: $h(n) \leq c_1 f(n)$.

Da $f \in O(g)$ liegt, gibt es eine positive reelle Zahl c_2 und eine natürliche Zahl n_1, so dass für alle $n \geq n_1$ gilt: $f(n) \leq c_2 g(n)$.

Sei $n_2 = \max\{n_0, n_1\}$. Dann gilt für alle $n \geq n_2$: $h(n) \leq c_1 f(n) \leq c_1 c_2 g(n)$, und es folgt $h \in O(g)$.

Damit gilt $O(f) \subseteq O(g)$.

Umgekehrt lässt sich zeigen, dass $O(g) \subseteq O(f)$ gilt, woraus die Behauptung folgt.

Aufgabe 4: Sei $k = L_1(1)$. Per Induktion kann man zeigen, dass für alle Zweierpotenzen 2^n die Ungleichung $L_1(2^n) \geq k(2^n)^2$ gilt:
$L_1(1) = k$
$L_1(2^{n+1}) = 4L_1(2^n) + 2 \geq 4(k(2^n)^2) + 2 = k(2^{n+1})^2 + 2 \geq k(2^{n+1})^2$.
Somit liegt die Laufzeit von A_1 in $\Omega(n^2)$, falls n eine Zweierpotenz ist.
Sei $l = L_2(1)$. Es lässt sich durch Induktion zeigen, dass für alle Zweierpotenzen 2^n $L_2(2^n) = 2^n(l + n)$ gilt:
$L_2(1) = 2^0(l + 0)$
$L_2(2^{n+1}) = 2L_2(2^n) + 2^{n+1} = 2(2^n(n + l)) + 2^{n+1} = 2^{n+1}(n + l + 1) = 2^{n+1}((n + 1) + l)$.
Somit liegt die Laufzeit von A_2 in $O(n(\log_2 n + l)) = O(n \log_2 n)$, wenn n eine Zweierpotenz ist.
Geht man davon aus, dass auch für n zwischen zwei Zweierpotenzen die angegebenen Formeln in etwa stimmen (zumindest von der Größenordnung her), folgt daraus:
Da $\frac{n^2}{n \log n} = \frac{n}{\log n}$ für $n \to \infty$ gegen unendlich geht, ist Algorithmus A_2 vorzuziehen.