Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Dr. Riko Jacob
Michael Schnupp

Wintersemester 2007/2008
Übungsblatt 12

Algorithmen für die Speicherhierarchie

Abgabetermin: 21.01.2008 **vor** *der Übung*

Connected Components

Aufgabe 1 (10 Punkte)

In der Vorlesung wurde gezeigt, dass man mit Hilfe des bekannten BFS-Algorithmuses die Komponenten eines Graphen bestimmen kann. Die sich daraus ergebende Laufzeit ist $O(|V| + sort(E))$ I/Os.

Um den $|V|$-Term in der Laufzeit vernachlässigen zu können, muss $|V| \leq |E|/B$ sein. Um dies zu erreichen reduziert *Algorithmus 1* die Anzahl der Knoten solange bis obige Bedingung erreicht ist.

Damit diese Reduktion nicht deutlich teurer wird als der eigentliche Algorithmus muss hierbei jeder Reduktionsschritt die Anzahl der Knoten stets um einen konstanten Reduktionsfaktor verkleinern.

Der Reduktionsschritt funktioniert, indem für jeden Knoten die Kante zum Nachbar mit der kleinsten Nummer bestimmt wird und die daraus entstehenden Pseudobäume jeweils zu einem einzigen Knoten kontrahiert werden.

Der Reduktionsfaktor hängt nun direkt mit der Anzahl der entstehenden Pseudobäume zusammen.

Wieviele Pseudobäume können minimal/maximal aus einem zusammenhängenden Graphen mit N Knoten entstehen?

Wenn wir uns die Nummerierung aussuchen könnten, könnten wir dadurch unabhängig von der Struktur des Graphen das Minimum oder Maximum erzwingen?

Falls eins davon nicht möglich ist, bei welcher Struktur des Graphen kann man den Fall erzwingen?

Aufgabe 2 (10 Punkte)

Wenn wir den Algorithmus so ändern, dass er nicht den Nachbarn mit der kleinsten Nummer sucht, sondern einfach eine zufällige Kante nimmt, was würde sich dadurch an der Struktur des Pseudobaums und damit am Listranking-Algorithmus zur Kontraktion der Pseudobäume ändern?

Aufgabe 3 (10 Punkte)

Durch Anwenden von *Algorithmus 1* ergab sich eine Verbesserung der Laufzeit auf $O(sort(|E|)\log(|B\frac{|V|}{|E|}))$. Eine weitere Verbesserung konnte *Algorithmus 2* erreichen, indem dieser für jeden Knoten mehr als eine Kante auswählt und Kanten des entstehenden Graphen kontrahierte.

Die Bestimmung der Spannbäume funktionierte sehr ähnlich, indem jeder Knoten einfach die Kante mit dem kleinsten Gewicht auswählt und kontrahiert. (Ein Boruvka-Schritt.)

Analog zu *Algorithmus 2* wurde in der Vorlesung (und Literatur) vorgeschlagen anstatt der einen leichtesten Kante die d leichtesten Kanten zu wählen und auf dem so entstehenden Graph mehrere Boruvka-Schritte zu machen. (Kontrahierte Kanten werden definitiv in den Spannbaum aufgenommen.)

Geben Sie einen Graphen an, auf dem dieser "verbesserte" Algorithmus, mit $d = 2$ und zwei Boruvka-Schritten eine falsche Kante kontrahiert.

Versuchen Sie eine Möglichkeit zu finden den Algorithmus so zu modifizieren, dass er eine korrekte Lösung findet. Welche Informationen könnte sich der Algorithmus dazu merken?

Bonusfrage: Bleibt die Laufzeit durch diese Modifikation erhalten?