Universität Karlsruhe
Fakultät für Informatik
ITI Wagner

2. Übungsblatt

Ausgabe: 15. November 2005
Abgabe: 21. November, 14 Uhr im ITI Wagner (Informatik-Hauptgebäude, 3. Stock)

Die Bearbeitung in Zweiergruppen ist ausdrücklich erwünscht.

Problem 1: Das Postamtplatzierungsproblem **

Zu n Punkten $p_1, \ldots, p_n$ in der Ebene mit assoziierten Gewichten $w_1, \ldots, w_n$ soll ein Punkt p gefunden werden, der die Summe $\sum_{i=1}^{n} w_i d(p, p_i)$ der gewichteten Abstände zu den anderen Punkten minimiert. In der Manhatten-Metrik haben zwei Punkte $p = (x_p, y_p)$ und $q = (x_q, y_q)$ den Abstand $d(p, q) = |x_p - x_q| + |y_p - y_q|$. Entwerfen Sie einen Algorithmus, der dieses Problem bzgl. der Manhatten-Metrik in linearer Zeit löst. (Hinweis: Überlegen Sie sich erst den eindimensionalen Fall.)

Problem 2: Union-Find ***

Eine Folge von m Operationen Makeset, Find (mit Pfadkompression) und Union (mit *balancing*) werde so ausgeführt, dass alle Union-Operationen vor der ersten Find-Operation ausgeführt werden.

(a) Zeigen Sie, dass der Gesamtaufwand für die m Operationen in $\mathcal{O}(m)$ liegt.

(b) Was passiert, wenn zwar die Find-Operationen weiterhin mit Pfadkompression, die Union-Operationen aber ohne *balancing* ausgeführt werden?

(c) Wie sieht es aus, wenn die Union-Operationen wieder mit *balancing*, die Find-Operationen jetzt aber ohne Pfadkompression ausgeführt werden?

Problem 3: **

In dem *Tiefen-Bestimmungsproblem* wird eine Menge $\mathcal{F}$ von Bäumen verwaltet, auf denen die folgenden drei Operationen definiert sind.

Make-Tree(v) erzeugt einen neuen Baum, dessen einziger Knoten v ist.

Find-Depth(v) bestimmt die Tiefe des Knotens v innerhalb des Baumes T, zu dem v gehört. Die Tiefe von v ist die Anzahl der Kanten auf dem Weg von v zur Wurzel des Baumes T.

Attach(r, v) macht den Knoten v zum Vorgänger des Knotens r. Dabei wird vorausgesetzt, dass r die Wurzel eines Baumes ist und dass v zu einem anderen Baum gehört als r (v ist nicht unbedingt die Wurzel eines Baumes).

Geben Sie Prozeduren MAKE-TREE(v), FIND-DEPTH(v) und ATTACH(r, v) in Pseudocode an, so dass die Laufzeit von m Operationen MAKE-TREE, FIND-DEPTH und ATTACH in $\mathcal{O}(m \cdot G(m))$ ist.

Hinweis: Orientieren Sie sich an den Algorithmen FIND und UNION aus der Vorlesung. Benutzen Sie für jeden Baum $T \in \mathcal{F}$ einen neuen Wurzelbaum S_T, der die gleichen Knoten wie T enthält. Weder Struktur von S_T und T noch ihre Wurzeln müssen sich entsprechen. Um die Tiefe von T zu bestimmen, wird eine Pseudodistanz $d(v)$ eingeführt, die folgende Eigenschaft haben soll. Ist v ein Knoten von T und ist $v = v_0, v_1, \ldots, v_k$ der Weg in S_T von v zur Wurzel v_k von S_T, dann ist $\sum_{j=0}^{k} d(v_i)$ die Tiefe von v in T.

Problem 4: Graustufenbilder **

Graustufenbilder seien gegeben als $(m \times n)$-Matrizen mit den Werten $\{1, 2, \ldots, k\}$ (den Graustufen). Zwei Bildpunkte (i_1, j_1) und (i_2, j_2) heißen *benachbart*, wenn sie den gleichen Grauwert haben und $|i_1 - i_2| + |j_1 - j_2| \leq 1$ gilt. Der transitive Abschluss dieser symmetrischen Relation ist eine Äquivalenzrelation.

(a) Formulieren Sie einen Algorithmus zur Bestimmung der Äquivalenzklassen dieser Äquivalenzrelation auf der Basis von UNION-FIND, der mit $\mathcal{O}(m \cdot n)$ UNION- und FIND-Operationen auskommt.

(b) Formulieren Sie eine aussagekräftige Invariante für Ihren Algorithmus, aus der die Korrektheit leicht ersichtlich ist.

Problem 5: d-Heaps **

Ein d-Heap ist eine Verallgemeinerung des aus der Vorlesung bekannten Heaps. Der einzige Unterschied zwischen einem Heap und einem d-Heap besteht darin, dass in einem d-Heap jeder innere Knoten (bis zu) d unmittelbare Nachfolger hat. Ein „normaler“ Heap ist also ein 2-Heap.

Die Speicherung der Werte eines d-Heap in einem Array erfolgt analog zum 2-Heap: Die Indizes des Arrays entsprechen einer Indizierung der Knoten im Baum von oben (der Wurzel) nach unten, in jedem Level des Baumes von links nach rechts.

(a) Geben Sie ein Beispiel für einen 5-Heap mit 13 Werten an. Geben Sie dabei die Baumstruktur und das entsprechende Array an.

(b) Welche Höhe hat der einem d-Heap mit n Knoten entsprechende Baum (für $n \in \mathbb{N}$)?

(c) Für einen Knoten mit Index i in einem d-Heap nummerieren wir die d unmittelbaren Nachfolger von links nach rechts durch. SUCC(i, j) bezeichne die Position des j-ten unmittelbaren Nachfolgers des Knotens mit Index i. PRED(i) bezeichne die Position des Vorgängers von i. Geben Sie Formeln für SUCC(i, j) und für PRED(i) an.

(d) Sei d nun fest gewählt. Schreiben Sie die für 2-Heaps bekannte Prozedur HEAPIFY(A, i) neu (in Pseudo-Code), so dass sie für d-Heaps funktioniert.

(e) Beweisen Sie, dass für die Worst-Case-Laufzeit $T(n)$ dieser neuen Prozedur HEAPIFY(A, i) ebenfalls gilt: $T(n) \in \mathcal{O}(\log_2 n)$