4. Aufgabenblatt zur Vorlesung

Höhere Algorithmik

WiSe 2011/12

Wolfgang Mulzer, Claudia Dieckmann, Romain Grunert, Tillmann Miltzow, Lena Schlipf

Abgabe am 18. November 2011 vor der Vorlesung in die Mitarbeiterschachteln

Aufgabe 1 Bestimmen des engsten Paares *10 Punkte*

Sei P eine Menge von n Punkten in der Ebene. In der Vorlesung haben Sie einen Algorithmus kennengelernt, der ein engstes Paar von P in Zeit $O(n \log n)$ bestimmt. Dabei haben wir angenommen, dass alle x-Koordinaten in P verschieden sind. Zeigen Sie, wie man den Algorithmus anpassen kann, damit wir diese Annahme nicht mehr benötigen. Die Laufzeit soll immer noch $O(n \log n)$ betragen.

Aufgabe 2 Finden eines lokalen Minimums *10 Punkte*

Sei $G = (V, E)$ ein Gittergraph mit n Zeilen und n Spalten. Formal heißt das: $V = \{(i, j) \mid 1 \leq i, j \leq n\}$ und $E = \{\{(i, j), (i', j')\} \mid |i - i'| + |j - j'| = 1\}$. Des Weiteren sei jeder Knoten v von G mit einer Zahl $x_v \in \mathbb{R}$ beschriftet, so dass die x_v paarweise verschieden sind. Ein *lokales Minimum* von G ist ein Knoten, dessen Beschriftung kleiner ist als die Beschriftungen seiner Nachbarn.

(a) Zeigen Sie, dass immer ein lokales Minimum existiert.

(b) Der folgende Algorithmus heißt *lokale Suche*: Beginne bei einem beliebigen Knoten v von G. Falls v ein lokales Minimum ist, sind wir fertig. Ansonsten hat v einen Nachbarn w mit $x_w < x_v$ (wenn es mehr als einen solchen Nachbarn gibt, wählen wir denjenigen mit dem kleinsten x_w). Gehe zu w und wiederhole, bis ein lokales Minimum erreicht ist.

Zeigen Sie, dass lokale Suche im schlimmsten Fall $\Omega(n^2)$ viele Schritte benötigt.

(c) Zeigen Sie, wie man ein lokales Minimum in $O(n)$ Zeit finden kann. Begründen Sie Korrektheit und Laufzeit Ihres Algorithmus.

Hinweis: Benutzen Sie das Abschneiden-und-Suchen Prinzip. Überlegen Sie sich, wie man für eine Spalte (Zeile) entscheiden kann, ob sich ein lokales Minimum darauf bzw. links oder rechts davon (darunter oder darüber) befindet.

Aufgabe 3 Matrizenkettenmultiplikation *10 Punkte*

Für eine gegebene Folge M_1, M_2, ..., M_n von n Matrizen ist das *Matrizenkettenprodukt* $M_1 \cdot M_2 \cdot \cdots \cdot M_n$ zu berechnen. Die Matrizen haben dabei verschiedene Dimensionen. M_1 ist eine $p_1 \times p_2$ Matrix, M_2 ist eine $p_2 \times p_3$ Matrix, usw.

Um eine $a \times b$ Matrix mit einer $b \times c$ Matrix (naiv) zu multiplizieren, benötigen wir bekanntlich acb Multiplikationen und $ac(b-1)$ Additionen, also insgesamt $ac(2b-1)$ elementare Operationen. Daraus folgt, dass es einen Unterschied macht, wie

man das Matrizenprodukt klammert, also in welcher Reihenfolge man die Matrizen multipliziert.

Sei beispielsweise M_1 eine 100×200 Matrix, M_2 eine 200×10 Matrix und M_3 eine 10×1 Matrix. Dann haben wir zwei Möglichkeiten, die Multiplikation durchzuführen: $(M_1 M_2)M_3$ oder $M_1(M_2 M_3)$. Im ersten Fall benötigen wir zunächst $100 \cdot 10 \cdot (400 - 1) = 399.000$ Operationen, um die 100×10 Matrix $M_1 M_2$ zu berechnen, und dann $100 \cdot 1 \cdot (20 - 1) = 1.900$ Operationen, um das Ergebnis zu erhalten. Insgesamt ergibt das 400.900 Operationen. Im zweiten Fall brauchen wir erst $200 \cdot 1 \cdot (20 - 1) = 3.800$ Operationen für die 200×1 Matrix $M_2 M_3$ und dann $100 \times 1 \times (400 - 1) = 39.900$ Operationen für das Ergebnis. Insgesamt sind es hier 43.700 Operationen. Die zweite Klammerung ist also fast zehnmal so schnell wie die erste. Ziel dieser Aufgabe ist es, eine optimale Klammerung für eine Matrizenkettenmultiplikation zu bestimmen.

(a) Bezeichne mit $\mathrm{P}[i, j]$ die Kosten für eine optimale Klammerung des Matrizenkettenprodukts $M_i \cdot \cdots \cdot M_j$ $(1 \leq i \leq j \leq n)$. Unser Ziel ist, $\mathrm{P}[1, n]$ zu berechnen. Finden Sie eine geeignete Rekursionsgleichung für $\mathrm{P}[i, j]$.

(b) Benutzen Sie Ihre Rekursion, um Pseudocode für einen Algorithmus anzugeben, welcher die optimalen Kosten $\mathrm{P}[1, n]$ bestimmt. Analysieren Sie Laufzeit und Platzbedarf.

(c) Erweitern Sie Ihren Algorithmus so, dass er auch eine optimale Klammerung ausgibt.

(d) (*freiwillig, 5 Zusatzpunkte*) Implementieren Sie ein Programm, das eine optimale Klammerung für eine gegebene Matrizenkettenmultiplikation bestimmt und die Multiplikation durchführt. Vergleichen Sie Ihr Programm mit einer naiven Implementierung, welche die Matrizen von links nach rechts multipliziert.