

## Übungsblatt 4 für “Diskrete und geometrische Algorithmen”

1.) Ein bißchen Kombinatorik.

- (a) Ein geordnetes  $k$ -Tupel  $(n_1, \dots, n_k)$  von natürlichen Zahlen  $n_i \in \mathbb{N}$  mit  $n = n_1 + \dots + n_k$  nennen wir eine Komposition von  $n$  in  $k$  Summanden. Geben Sie eine explizite Formel für die Anzahl  $C_{n,k}$  der Kompositionen von  $n$  in  $k$  Summanden ( $n, k \in \mathbb{N}$ ) an (beispielsweise indem Sie die Idee des Beweises der Formel für die Anzahl der “Auswahlen einer Teilmultimenge” aus der Vorlesung verwenden).
- (b) Üblicherweise betrachtet man aber (an Stelle der obigen Definition) geordnete  $k$ -Tupel  $(n_1, \dots, n_k)$  von positiven natürlichen Zahlen, also  $n_i \in \mathbb{N} \setminus \{0\}$ , mit  $n = n_1 + \dots + n_k$ , welche wir Kompositionen von  $n$  in  $k$  positive Summanden nennen wollen. Geben Sie nun auch eine explizite Formel für die Anzahl  $\tilde{C}_{n,k}$  der Kompositionen von  $n$  in  $k$  positive Summanden an (beispielsweise indem Sie dieses Problem auf das vorige zurückführen via “Abziehen von 1 bei jedem Summanden”).

2.) (a) Wie schnell können Sie eine  $(kn \times n)$ -Matrix  $A$  mit einer  $(n \times kn)$ -Matrix  $B$  multiplizieren, d.h.  $C = A \cdot B$  berechnen, wenn Sie Strassens Algorithmus als Unterprogramm verwenden?

- (b) Beantworten Sie die gleiche Frage, wenn die Reihenfolge der Eingabematrizen vertauscht ist, man also  $\tilde{C} = B \cdot A$  bestimmen möchte.

3.) Zeigen Sie, wie man komplexe Zahlen  $a + bi$  und  $c + di$  mit nur drei Multiplikationen reeller Zahlen multiplizieren kann. Der Algorithmus sollte  $a, b, c$  und  $d$  als Eingabe bekommen und den Realteil  $ac - bd$  sowie den Imaginärteil  $ad + bc$  getrennt ausgeben.

4.) Verbesserung des “Schul-Algorithmus” zum Multiplizieren  $n$ -stelliger Dezimalzahlen. Um zwei 2-stellige Dezimalzahlen  $M = (ab)_{10} := 10a + b$  und  $N = (cd)_{10} := 10c + d$ , mit Ziffern  $a, b, c, d \in \{0, \dots, 9\}$  miteinander zu multiplizieren, kann man (zurückgehend auf Karatsuba und Ofman) anstatt den “Schul-Algorithmus” zu verwenden folgendermaßen vorgehen (wie man leicht nachrechnen kann):

$$A := ac; \quad B := bd; \quad C := (a-b) \cdot (d-c); \quad MN = (ab)_{10} \cdot (cd)_{10} = 100A + 10A + 10B + B + 10C.$$

Im Gegensatz zum Schul-Algorithmus (welcher 4 einstellige Multiplikationen benötigt) benötigt dieser Algorithmus zum Multiplizieren zweier 2-stelliger Dezimalzahlen nur drei einstellige Multiplikationen, wohingegen alle anderen notwendigen Operationen Additionen bzw. “shifts” darstellen.

- (a) Obige Idee kann man verwenden um einen “Divide and Conquer”-Algorithmus zum Multiplizieren zweier  $n$ -stelliger Dezimalzahlen zu erhalten, indem man  $M$  und  $N$  jeweils in zwei (ungefähr)  $\frac{n}{2}$ -stellige Dezimalzahlen  $a, b, c, d$  aufteilt,  $A, B$  und  $C$  rekursiv berechnet und daraus  $MN$  erhält. Formulieren Sie einen solchen Algorithmus in Pseudocode.
- (b) Unter der Annahme, daß  $n$  eine Zweierpotenz ist, also  $n = 2^k$ , für  $k \in \mathbb{N}$  gilt, gebe man eine Rekursion für die Anzahl  $T(n)$  der einstelligen Multiplikationen an (alle anderen Operationen werden vernachlässigt), welche bei diesem Verfahren beim Multiplizieren zweier  $n$ -stelliger Dezimalzahlen durchgeführt werden. Mit Hilfe des Master-Theorems folgere man daraus das asymptotische Verhalten von  $T(n)$ .

5.) Betrachtet werden reelle  $(m \times n)$ -Matrizen  $A$ , welche für alle  $1 \leq i < k \leq m$  und  $1 \leq j < \ell \leq n$  die Bedingung

$$(*) : \quad a_{i,j} + a_{k,\ell} \leq a_{i,\ell} + a_{k,j}$$

erfüllen (d.h.: wählt man beliebig zwei Zeilen und zwei Spalten und betrachtet man die vier Schnittelemente, so soll die Summe der “links oben” + “rechts unten” Einträge kleiner oder gleich der Summe der “links unten” + “rechts oben” Einträge sein).

(a) Zeigen Sie, daß Bedingung  $(*)$  erfüllt genau dann erfüllt wird, wenn die Bedingung

$$(**) : \quad a_{i,j} + a_{i+1,j+1} \leq a_{i,j+1} + a_{i+1,j}$$

für alle  $1 \leq i \leq m - 1$ ,  $1 \leq j \leq n - 1$  erfüllt ist. (**Hinweis:** Induktion getrennt nach Zeilen und Spalten.)

(b) Sei  $\min(i)$  der Index der Spalte, in welcher das am weitesten links stehende minimale Element der Zeile  $i$  steht. Überlegen Sie sich, daß

$$\min(1) \leq \min(2) \leq \min(3) \leq \dots \leq \min(m)$$

für jede Matrix gilt, welche Bedingung  $(*)$  erfüllt.

6.) Fortsetzung des vorigen Beispiels.

(a) Hier ist die Beschreibung eines “Divide and Conquer”-Algorithmus zur Berechnung des am weitesten links gelegenen minimalen Elements jeder Zeile einer  $(m \times n)$ -Matrix  $A$ , welche die Bedingung  $(*)$  erfüllt:

Konstruiere eine Teilmatrix  $A'$  von  $A$ , welche aus den geradzahigen Zeilen von  $A$  besteht. Bestimme rekursiv das am weitesten links gelegene minimale Element jeder Zeile von  $A'$ . Dann bestimme das am weitesten links gelegene minimale Element jeder ungeradzahigen Zeile von  $A$ .

Überlegen Sie sich insbesondere, wie in  $O(m+n)$  Schritten das am weitesten links gelegene minimale Element jeder ungeradzahigen Zeile von  $A$  bestimmt werden kann, wenn man die Minima in den geradzahigen Zeilen bereits kennt.

- (b) Geben Sie eine Rekursion für die Laufzeit des in (a) beschriebenen Algorithmus an. Weisen Sie nach, daß die Lösung  $O(m + n \log m)$  ist.

- 7.) In der Vorlesung wurde der Fisher-Yates-Algorithmus zum “in-place” permutieren eines Feldes vorgestellt:

```
RANDOMIZE-IN-PLACE( $A$ )
 $n := A$ .länge
FOR  $i = 1$  TO  $n$  DO
    vertausche  $A[i]$  mit  $A[\text{RANDOM}(i, n)]$ 
END DO
```

Hierbei liefert  $\text{RANDOM}(i, n)$  eine durch einen Zufallszahlengenerator erzeugte Zufallszahl aus der Menge  $\{i, i + 1, \dots, n\}$ . Überlegen Sie sich, daß der Algorithmus  $\text{RANDOMIZE-IN-PLACE}$  tatsächlich eine zufällige Permutation des Feldes  $A$  erzeugt, daß also die Ausgabe  $\tilde{A}[1..n] = A[\pi(1), \pi(2), \dots, \pi(n)]$  ist, wobei  $A[1..n]$  die Eingabe bezeichnet und  $\pi$  eine zufällige Permutation von  $\{1, \dots, n\}$  darstellt.

- 8.) Nehmen Sie an, Sie wollen die Ausgaben 0 und 1 jeweils mit Wahrscheinlichkeit  $1/2$  erhalten. Ihnen steht die Prozedur  $\text{BIASED-RANDOM}$  zur Verfügung, die jeweils entweder 0 oder 1 ausgibt. Die Ausgabe 1 tritt mit Wahrscheinlichkeit  $p$  und die Ausgabe 0 mit Wahrscheinlichkeit  $1 - p$  auf, wobei  $0 < p < 1$  gilt. Sie wissen aber nicht, wie groß  $p$  ist. Geben Sie einen Algorithmus an, der  $\text{BIASED-RANDOM}$  als Unteroutine verwendet und die Werte 0 und 1 mit Wahrscheinlichkeit  $1/2$  zurückgibt. Wie groß ist die erwartete Laufzeit für Ihren Algorithmus als Funktion von  $p$ ?