3. Übung

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

$$(*): \qquad a_{ij} + a_{kl} \leq a_{il} + a_{kj}$$

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 wird genau dann wenn die Bedingung $(**)$

$$a_{ij} + a_{i+1j+1} \leq a_{ij+1} + a_{i+1j}$$

   für alle $1 \leq i \leq m-1$, $1 \leq j \leq n-1$ erfüllt ist.

   (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

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

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

2. (Fortsetzung Beispiel (1)):

   (a) Hier ein Divide–and–conquer Algorithmus zur Berechnung des am weitesten links gelegenen minimalen Elements jeder Reihe einer $(m \times n)$–Matrix A, welche die Bedingung $(*)$ erfüllt:
   Konstruiere eine Teilmatrix A', welche aus den geradzahligen 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 ungeradzahligen Zeile von A.

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

3. Ein n–elementiges Array $A[1], \ldots, A[n]$ kann folgendermaßen als binärer Baum dargestellt werden: $A[1]$ ist der Knoten des Baumes, $A[2]$ sind der linke bzw. $A[3]$ der rechte Nachfolger von $A[1]$, $A[4]$ sind der linke bzw. $A[5]$ der rechte Nachfolger von $A[2]$, $A[6]$ sind der linke bzw. $A[7]$ der rechte Nachfolger von $A[3]$... So erhält man (offenbar) einen Baum der Höhe $\lceil \log(n) \rceil$, dessen Knoten folgende Bedingungen erfüllen: $A[\lfloor i/2 \rfloor]$ ist der Vorgänger, $A[2i]$ ist der linke Nachfolger und $A[2i+1]$ ist der rechte Nachfolger von $A[i]$ (jeweils so existent).
   Ein n–elementiges Array $A[1], \ldots, A[n]$ erfüllt die max–Bedingung, falls jedes $A[i]$ größer oder gleich ist als $A[2i]$ und $A[2i+1]$, falls also in dem zugeordneten Baum die Nachfolger eines Knotens $A[i]$ stets kleiner oder gleich $A[i]$ sind.

   (a) Geben Sie den Baum zum Array

$$A[1, \ldots, 9] = (5, 13, 2, 25, 7, 17, 20, 8, 23)$$

   an.

   (b) Gegeben sei der folgende Algorithmus $SUBMAX(A, i)$:
   (1) $l = 2i$
   (2) $r = 2i + 1$
   (3) If $l \leq n$ and $A[l] > A[i]$
   (4) $\qquad largest = l$

(5) Else $largest = i$
(6) If $r \leq n$ and $A[r] > A[largest]$
(7) $\quad largest = r$
(8) If $largest \neq i$
(9) $\quad$ vertausche $A[i]$ und $A[largest]$
(10) $\quad SUBMAX(A, largest)$

Illustrieren Sie die Funktionsweise des Algorithmus, indem Sie $SUBMAX(A, 2)$ auf das Array von (a) anwenden.

(c) Überlegen Sie sich, dass der Algorithmus der Rekursion $T(n) \leq T(2n/3) + O(1)$ genügt. Verwenden Sie das Master Theorem, um das asymptotische Wachstum von $T(n)$ abzuschätzen.

4. (Fortsetzung Beispiel 3)

Betrachtet wird nun der Algorithmus $MAX(A)$:

(1) For $i = \lfloor length(A)/2 \rfloor$ downto 1
(2) $\quad SUBMAX(A, i)$

(a) Wenden Sie den Algorithmus auf das Array $A = (5, 3, 17, 10, 71, 19, 6, 22, 9)$ an, d.h., schreiben Sie die Gestalt des zu A gehörigen Baumes nach den einzelnen Schritten des Algorithmus auf.

(b) Formulieren Sie die Funktionsweise des Algorithmus allgemein.

(c) Bestimmen Sie eine obere Schranke (d.h. eine O Abschätzung) der asymptotischen Laufzeit des Algorithmus $MAX(A)$ bei einem Array der Länge n?

5. (Fortsetzung Beispiel 3) Betrachtet wird nun der Algorithmus $SORT(A)$ zu einem Array A der Länge n:

(1) $MAX(A)$
(2) For $i = length(A)$ downto 2
(3) $\quad$ tausche $A[1]$ und $A[i]$
(4) $\quad n = n - 1$
(5) $\quad SUBMAX(A[1, 2, \ldots, n], 1)$

(a) Überlegen Sie sich, wie $SORT$ auf den Array A aus Beispiel 3(a) wirkt, d.h. schreiben Sie die Gestalt des zu A gehörigen Baumes nach den einzelnen Schritten des Algorithmus auf.

(b) Überlegen Sie sich allgemein, dass $SORT$ einen Array der Länge n in $O(n \log n)$ Schritten sortiert.