UE DGA WS2020-2021
Übungsblatt 3

Aufgabe 13:

Gegeben sei ein zusammenhängender ungerichteter Graph $G = (V, E)$ mit einer geraden Anzahl an Knoten. Zeigen Sie, dass es einen (nicht notwendigerweise zusammenhängenden) Untergraph mit Knotenmenge V gibt (also einen Graph $G' = (V, E')$ mit $E' \subseteq E$), in dem alle Knotengrade ungerade sind.

(Hinweis: beweisen Sie die Behauptung für Bäume und begründen Sie, warum diese Annahme reicht.)

Aufgabe 14:

Sei $A[1, \dots, n]$ ein Feld mit n verschiedenen Zahlen. Das Paar (i, j) wird Inversion genannt, wenn $i < j$ und $A[i] > A[j]$ gilt.

(a) Welches Feld mit Elementen der Menge $\{1, \dots, n\}$ besitzt die meisten Inversionen und wie viele Inversionen sind in diesem Feld enthalten?

(b) Welche Beziehung gibt es zwischen der Anzahl von Inversionen im Eingabefeld und der Laufzeit von Insertion-Sort (Einfügesortieren)?

(c) Geben Sie einen Algorithmus an, der die Anzahl von Inversionen einer Permutation von n Elementen bestimmt und dessen Laufzeit im schlechtesten Fall $\Theta(n \log n)$ ist.
(Hinweis: Modifizieren Sie Merge-Sort (Sortieren durch Verschmelzen) in passender Weise)

Aufgabe 15:

Natural Merge-Sort ist eine Variante von Merge-Sort, die bereits vorsortierte Teilfolgen (so genannte runs) ausnutzt. Ein run ist eine Teilfolge aufeinanderfolgender Glieder $x_i, x_{i+1}, \dots, x_{i+k}$ mit $x_i \leq x_{i+1} \leq \dots \leq x_{i+k}$. Die Basis für den Verschmelzen-Vorgang bilden hier nicht die rekursiv oder iterativ gewonnenen Zweiergruppen, sondern die runs. Im ersten Durchlauf des Algorithmus bestimmt man die runs, anschließend fügt man die runs mittels VERSCHMELZEN zusammen.

(a) Sortieren Sie folgende Liste mittels Natural Merge-Sort:

$$2, 4, 3, 1, 7, 6, 8, 9, 0, 5$$

(b) Schreiben Sie einen Pseudocode für Natural Merge-Sort.

(c) Machen Sie eine Best- sowie eine Worst-Case-Analyse für das Laufzeitverhalten von Natural Merge-Sort bei einem Eingabefeld der Größe n.

Aufgabe 16:

Die Fibonacci-Zahlen seien durch die Rekursion $F_n = F_{n-1} + F_{n-2}$ für $n \geq 2$ mit Anfangswerten $F_0 = 0$ und $F_1 = 1$ definiert. Die Fibonacci-Zahlen können effizient mittels folgender auf Matrizenmultiplikation beruhender Formel berechnet werden:

$$\begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \qquad \text{für } n \geq 1.$$

a Beweisen Sie diese Formel durch vollständige Induktion.

b Überlegen sie sich einen Algorithmus, der $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$ in nur logarithmisch vielen Schritten berechnet.

Aufgabe 17:

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

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

Aufgabe 18:

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.