Höhere Algorithmik, WS 2014/15 — 4. Übungsblatt

20 Gummipunkte. Schriftliche Einzelabgabe bis Dienstag, 11. November 2014

21. Rekursionsgleichungen, 0 Punkte

Geben Sie, wenn möglich, möglichst genaue untere und obere asymptotische Schranken für die Lösung $T(n)$ der folgenden Rekursionen an. Die Rekursionen gelten für alle $n \geq n_0$, für ein festes n_0.

(a) $T(n) = 2 \cdot T(\lfloor n/4 \rfloor) + \Theta(\sqrt{n})$

(b) $T(n) = 2 \cdot T(n-1) + 1$

(c) $T(n) = 2 \cdot T(\lfloor n/3 \rfloor) + O(n)$

(d) $T(n) = 3 \cdot T(\lceil n/3 \rceil) + O(n)$

22. Bestimmen eines dünn besetzten Teilintervalls, 0 Punkte

Gegeben sind n Zahlen $a_1, \ldots, a_n$ im Intervall $[0, 1)$. Das Intervall wird durch eine gegebene Folge von Grenzen $0 = b_0 < b_1 < b_2 < \cdots < b_{k-1} < b_k = 1$ in k Teilintervalle $[b_{i-1}, b_i)$ zerlegt, $i = 1, \ldots, k$. Nach dem Schubfachprinzip muss es dann ein Teilintervall mit höchstens $\lfloor n/k \rfloor$ Werten geben.

Entwerfen Sie einen teile-und-herrsche-Algorithmus zur Bestimmung eines solchen Intervalls, und analysieren Sie die Laufzeit (obere und untere Schranke).

Wie kann man analog ein *dicht* besetztes Intervall mit *mindestens* $\lceil n/k \rceil$ Werten finden?

23. Bestimmen der Mehrheit, 10 Punkte

Gegeben ist eine Folge $a_1, \ldots, a_n$ von Werten („Stimmen“). Ein *Mehrheitselement* ist ein Element, das mindestens $\lceil (n+1)/2 \rceil$-mal vorkommt.

(a) (0 Punkte) Zeigen Sie, dass man das häufigste Element einfach in $O(n \log n)$ Zeit mit Hilfe von einem Sortierlauf finden kann.

(b) Entwerfen Sie einen teile-und-herrsche-Algorithmus zur Bestimmung eines Mehrheitselements, falls ein solches existiert, und analysieren Sie die Laufzeit (obere und untere Schranke). Natürlich dürfen Sie hierbei nicht sortieren.

(c) Erweitern Sie den Algorithmus, sodass er alle Elemente bestimmt, die mindestens einen Anteil von 30 % haben.

24. Zusatzaufgabe zum Knobeln[1], 0 Punkte

Ein Mehrheitselement kann mit konstantem Zusatzspeicher in einem einzigen Durchlauf bestimmt werden, in dem die Eingabe nur sequentiell gelesen wird. Ein zweiter Durchlauf ist dann eventuell noch notwendig, um zu überprüfen, ob das gefundene Element tatsächlich die Mehrheit hat oder ob es gar kein Mehrkeitselement gibt.

25. Lokales Maximum in Bäumen, 10 Punkte

Gegeben ist ein binärer Baum mit beliebigen Werten in den Knoten. (Es muss kein Suchbaum sein.) Ein *lokales Maximum* ist ein Knotenwert, der mindestens so groß ist wie die Werte in seinem Elternknoten und seinen Kindern (sofern vorhanden).

(a) Wie kann man eine lokales Maximum in linearer Zeit bestimmen?

(b) Wie kann man eine lokales Maximum in einem Baum der Höhe h in $O(h)$ Zeit bestimmen?

(c) (0 Punkte) Wie kann man eine lokales Maximum in einem Baum mit n Knoten bestimmen, indem man höchstens $O(\log n)$ *Werte anschaut*? Der Algorithmus darf die Struktur des Baumes untersuchen und insgesamt lineare Zeit brauchen.

[1] Peter Winkler, *Mathematical Puzzles—a Connoisseur's Collection*, A. K. Peters, 2004, S. 102