4. Übungsblatt zu Grundzüge von Algorithmen und Datenstrukturen, WS 12/13

Prof. Dr. Markus Bläser, Radu Curticapean, Christian Engels
http://www-cc.cs.uni-sb.de/course/38/

Abgabe: Donnerstag, 15. November 2012, 12:00 Uhr

Aufgabe 4.1 In dieser Aufgabe werden Sie Radix-Sort kennenlernen. Dieser Algorithmus funktioniert nach folgendem Prinzip.

- Betrachten Sie die erste Ziffer jeder Zahl und sortieren[1] Sie die Zahlen nach ihrer ersten Ziffer. Sie erhalten Blöcke von Zahlen, die in ihrer ersten Ziffer übereinstimmen.
- Führen Sie diese Prozedur rekursiv in jedem Block aus. Betrachten Sie hierbei allerdings die nächste Ziffer.

Sie können annehmen, dass es eine Funktion `ziffer`(x, k) gibt, die von der Zahl x die k-te Ziffer in Zeit $O(1)$ ausgibt. Zudem dürfen Sie annehmen, dass die Anzahl der Ziffern durch eine Konstante $c \in O(1)$ nach oben beschränkt ist.

a) (2 Punkt) Implementieren Sie Radix-Sort via Pseudo-Code.

b) (2 Punkte) Zeigen Sie, dass die Laufzeit durch $O(n)$ beschränkt ist.

c) ~~(1 Punkt) Erläutern sie, warum dies kein Widerspruch zu der unteren Schranke für Sortieralgorithmen ist, die Sie in der Vorlesung kennengelernt haben.~~

Aufgabe 4.2 Sie erhalten n Bitstrings $B_1, \ldots, B_n$. Jeder Bitstring hat die Länge k. Zusätzlich erhalten Sie eine Zahl $t \leq n$.

Zeigen Sie, dass es einen Algorithmus gibt, der entscheidet, ob es t paarweise verschiedene Bitstrings $A_1, \ldots, A_t$ der Länge k gibt, so dass

$$\{A_1, \ldots, A_t\} \cap \{B_1, \ldots, B_n\} = \emptyset$$

gilt. Ihr Algorithmus soll eine Laufzeit von $O(kn)$ haben.

Aufgabe 4.3 Ein m-adischer Heap ist ähnlich zu einem binärem Heap. Bei einem m-adischem Heap hat allerdings jeder Knoten, der kein Blatt ist, m Kinder, wobei $m \geq 2$ ist. Einzige Ausnahme ist ein Knoten auf der vorletzten Ebene, der weniger Kinder haben darf.

a) (1 Punkt) Wie würden Sie einen m-adischen Heap in einem Array repräsentieren?

b) (1 Punkt) Was ist die Höhe eines m-adischen Heaps mit n Elementen in Abhängigkeit von m und n? Beweisen Sie ihre Aussage.

[1] eher: "gruppieren"

c) (1 Punkt) Geben Sie eine effiziente Implementierung von $\texttt{Extract} - \texttt{Max}$ in einem m-adischen Max-Heap an.[2] Analysieren sie ihre Laufzeit in Abhägigkeit von m und n und zeigen Sie die Korrektheit.

d) (1 Punkt) Geben Sie eine effiziente Implementierung von $\texttt{Increase} - \texttt{Key}(x, k)$ in einem m-adischen Max-Heap an.[3] Analysieren sie ihre Laufzeit in Abhängigkeit von m und n und zeigen Sie die Korrektheit.

Aufgabe 4.4 Die Fibonacci-Zahlen $\{F_n\}$, $n \geq 0$, werden definiert als $F_0 = 0$, $F_1 = 1$ und $F_{n+1} = F_n + F_{n-1}$ für $n \geq 1$. Überlegen Sie sich, wie man mittels $O(\log n)$ arithmetischen Operationen die Fibonacci-Zahl F_n ausrechnen kann. Erlaubt sind in dieser Aufgabe nur Operationen mit ganzen Zahlen. Die Variablen Ihres Algorithmus dürfen Werte bis zu $2^{O(n)}$ beinhalten. (Insbesondere passt also die n-te Fibonacci-Zahl in eine einzelne Variable.)

Aufgabe 4.5 (2 Zusatzpunkte) In dieser Aufgabe betrachten wir erneut Datenstrom-Algorithmen. Ein derartiger Algorithmus erhält nach wie vor ein Array $A = [a_1, \ldots, a_n]$ als Eingabe, darf jedes Element von A jedoch nur einmal lesen und muss diese Elemente in der Reihenfolge $a_1, \ldots, a_n$ lesen.

Wir betrachten Eingaben der Art $[a_1, \ldots, a_{n-2}]$ mit paarweise verschiedenen a_i und $\{a_1, \ldots, a_{n-2}\} \subseteq \{1, \ldots, n\}$. In anderen Worten: Wir erhalten alle Elemente der Menge $\{1, \ldots, n\}$, mit Ausnahme von zwei Elementen $i, j \in \mathbb{N}$, in beliebiger Reihenfolge.

Geben Sie einen Datenstrom-Algorithmus an, der auf solchen Eingaben i und j bestimmt. Ihr Algorithmus darf nur $O(1)$ Speicherzellen nutzen.

[2] Die Operation $\texttt{Extract} - \texttt{Max}$ gibt das Maximum aus einem Heap aus und löscht dieses Element anschließend aus dem Heap.

[3] Die Operation $\texttt{Increase} - \texttt{Key}(x, k)$ erhöht den Wert des Schlüssels x auf k. x ist hierbei ein Zeiger, der auf eine Stelle im Heap zeigt.