5. 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, 22. November 2012, 12:00 Uhr

Aufgabe 5.1 Eine Min-Prioritätswarteschlange ist eine Datenstruktur, die eine Menge S von Elementen verwaltet, von denen jedes eine bestimmte Priorität hat. Folgende Operationen muss eine Prioritätswarteschlange unterstützen:

- `Insert`(S, x) - fügt Element x in die Menge S ein.
- `Minimum`(S) - gibt das Element mit der niedrigsten Priorität zurück.
- `Extract-Key`(S) - gibt das Element mit der niedrigsten Priorität zurück und entfernt es aus S.
- `Decrease-Key`(S, x, k) - verringert die Priorität des Elementes x auf den neuen Wert k, Vorraussetzung ist, dass k höchstens so groß wie die bisherige Priorität von x ist.

a) *(2 Punkte)* Zeigen Sie, wie man mit einem Heap eine Min-Prioritätswarteschlange implementieren kann, so das die obigen Operationen Laufzeit $O(\log n)$ haben, wenn $n = |S|$ ist.

b) *(2 Punkte)* Geben Sie einen Algorithmus mit Laufzeit in $O(n \log k)$, der gegeben k sortierte Listen eine sortierte Liste produziert. n ist hierbei die gesamte Anzahl der Elemente über alle Listen.

Aufgabe 5.2 In dieser Aufgabe betrachten wir Anwendungen des Median-Algorithmus, den Sie in der Vorlesung kennen gelernt haben.

a) *(2 Punkte)* Gegeben ist ein Array A von n Zahlen und k mit $1 \leq k \leq n$. Es sei b_k die Zahl, die an der Stelle k steht, wenn man A sortiert. Zeigen Sie, wie sich b_k in Zeit $\mathrm{O}(n)$ bestimmen lässt.

b) *(2 Punkte)* Es schreiben n Studenten eine Klausur. Jeder Student i erreicht hierbei eine gewisse Punktzahl $a_i \in \mathbb{N}$. Der Dozent der Vorlesung wird misstrauisch, wenn ein unerwartet großer Anteil der Studenten exakt dieselbe Punktzahl erreicht.

Es sei $\alpha \in \mathbb{Q}$ mit $0 < \alpha < 1$ eine beliebige Konstante. Zeigen Sie, dass sich in Zeit $\mathrm{O}(n)$ entscheiden lässt, ob es eine Punktzahl gibt, die von echt mehr als αn Studenten erreicht wird.

Hinweis: Diese Teilaufgabe lässt sich beispielsweise durch $\leq 1/\alpha$ geschickte Aufrufe des Algorithmus aus der vorigen Teilaufgabe lösen.

Aufgabe 5.3 Wenn der Quicksort-Algorithmus aus der Vorlesung ein Array A sortiert, wählt er das Element an der ersten Stelle von A als Pivot-Element aus. Dies kann zu sehr unbalancierten Partitionen von A führen.

a) *(3 Punkte)* Modifizieren Sie Quicksort so, dass in jeder Iteration der Median des aktuellen Arrays als Pivot-Element gewählt wird. Geben Sie die Rekurrenzgleichung für die Laufzeit Ihres Algorithmus an.

b) *(1 Punkt)* Warum wird die modifizierte Variante von Quicksort in der Praxis trotz ihrer asymptotisch optimalen Laufzeit nicht verwendet?

c) *(1 Zusatzpunkt)* Zeigen Sie, dass die modifizierte Variante von Quicksort eine Laufzeit von $\mathrm{O}(n \log n)$ hat.

Aufgabe 5.4 Gegeben eine Menge von n Elementen, n sei gerade.

a) *(2 Punkte)* Geben Sie einen vergleichsbasierten Algorithmus an, der das Maximum *und* das Minimum mittels $\frac{3}{2}n - 2$ Vergleiche findet.

b) *(1 Punkt)* Wir definieren uns eine Potentialfunktion anhand der bisherigen partiellen Ordnung eines Algorithmus. Wir bezeichnen als Maxima, die Maxima in dieser partiellen Ordnung und als Minima die Minima. Wir definieren Singletons als, alle Elemente die sowohl Minima und Maxima sind. Unsere Potentialfunktion ist nun gegeben durch $\Phi = 2^{(\#\text{Maxima}+\#\text{Minima}-\frac{1}{2}\#\text{Singletons})}$. Beweisen Sie, dass dies eine Potentialfunktion ist.

c) *(1 Punkt)* Zeigen Sie: Jeder vergleichsbasierten Algorithmus, der Maximum und Minimum von n Elementen bestimmt, benötigt im schlechtesten Fall mindestens $\frac{3}{2}n - 2$ Vergleiche.