Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Dr. Hanjo Täubig
Tobias Lieber

Sommersemester 2011
Übungsblatt 6
4. Juli 2011

Grundlagen: Algorithmen und Datenstrukturen

Abgabetermin: In der jeweiligen Tutorübung

Hausaufgabe 1

Implementieren Sie den QuickSelect-Algorithmus.
Implementieren Sie in der Klasse `UIsqsArray` in der Funktion `quickSelect` den QuickSelect-Algorithmus der das i-te Elemente in dem unsortierten Feld `A` liefert.
Verwenden Sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die Klasse `UIsqsArray`.
Achten Sie bei der Abgabe Ihrer Aufgabe darauf, dass Ihre Klasse `UIsqsArray` heißt und auf den Rechnern der Rayhalle (`rayhalle.informatik.tu-muenchen.de`) mit der bereitgestellten Datei `main_qs` kompiliert werden kann. Anderenfalls kann eine Korrektur nicht garantiert werden. Achten Sie darauf, dass Ihr Quelltext ausreichend kommentiert ist.
Schicken Sie die Lösung per Email mit dem Betreff `[GAD] Gruppe <Gruppennummer>` an Ihren Tutor.

Lösungsvorschlag

Siehe Übungswebseite.

Aufgabe 1

Beweisen Sie folgende in der Vorlesung benötigte Aussage: Ein fast vollständiger Binärbaum der Höhe h hat $2^{h-1} \leq n \leq 2^h - 1$ Knoten (Die Wurzel hat Höhe 1).
Definition; Ein fast vollständiger Binärbaum der Höhe h ist ein vollständiger Binärbaum der Höhe $h-1$ dessen Knoten so angeordnet werden können, dass es auf dem Level h einen Knoten e mit dem Vaterknoten v gibt so dass gilt:

- Alle Knoten auf dem Level $h-1$ die „links“ von v stehen zwei Kinder haben
- und alle Knoten auf dem Level $h-1$ die „rechts“ von v stehen keine Kinder haben
- oder e ist die Wurzel.

Lösungsvorschlag

Wir betrachten zwei Fälle:

- Ein vollständiger Binärbaum der Höhe h gibt immer eine obere Schranke für die Anzahl der Knoten in einem fast vollständigen Binärbaum an.

- Ein vollständiger Binärbaum der Höhe $h-1$ mit einem zusätzlichem Knoten (ganz links) gibt eine untere Schranke für die Anzahl der Knoten in einem fast vollständigen Binärbaum an.

Somit ergibt sich: $1 + \sum_{j=1}^{h-1} 2^{j-1} \leq n \leq \sum_{k=1}^{h} 2^{k-1}$. Mit Indextransformationen erhalten wir: $1 + \sum_{j=0}^{h-2} 2^j \leq n \leq \sum_{k=0}^{h-1} 2^k$. Mit $\sum_{k=0}^{n} 2^k = 2^{n+1} - 1$ (kann auch mit Induktion bewiesen werden) erhalten wir das angestrebte Resultat: $2^{h-1} \leq n \leq 2^h - 1$.

Aufgabe 2

Seien A und B nicht leere Mengen und $\leq_A$ und $\leq_B$ totale Ordnungen auf A bzw. B.
Definieren Sie eine totale Ordnung auf $A \times B$.
Wozu brauchen wir den Begriff der totalen Ordnung im Zusammenhang der aktuellen Vorlesungsthemen Sortieren und Selektieren?
Hinweis: Eine lineare Ordnung $\leq$ auf einer Menge X ist

- transitiv: $\forall x, y, z \in X: \quad x \leq y \ \wedge \ y \leq z \ \Rightarrow \ x \leq z$
- antisymmetrisch: $\forall x, y \in X: \quad x \leq y \ \wedge \ y \leq x \ \Rightarrow \ x = y$
- linear (total): $\forall x, y \in X: \quad x \leq y \ \vee \ y \leq x$
- reflexiv: $\forall x \in X: \quad x \leq x$ (folgt aus der Linearität)

Lösungsvorschlag

Seien $(a_1, b_1), (a_2, b_2) \in A \times B$. Dann ist die wie folgt definierte Ordnung $\leq_{A \times B}$

$$(a_1, b_1) \leq_{A \times B} (a_2, b_2) \quad \Leftrightarrow \quad (a_1 \leq_A a_2 \wedge \neg(a_2 \leq_A a_1)) \ \vee \ (a_1 \leq_A a_2 \wedge a_2 \leq_A a_1 \wedge b_1 \leq_B b_2)$$

eine lineare Ordnung auf der Menge $A \times B$.
Sinngemäß übersetzt bedeutet das soviel wie:

$$(a_1, b_1) \leq_{A \times B} (a_2, b_2) \quad \Leftrightarrow \quad (a_1 <_A a_2) \ \vee \ (a_1 =_A a_2 \ \wedge \ b_1 \leq_B b_2).$$

Der Begriff der totalen Ordnung ist essentiell für jeden vergleichsbasierten Sortieralgorithmus (oder auch Selektionsalgorithmus). Ist es nicht Möglich eine Menge total zu ordnen, so ist nicht für alle Paare x, y von Elemente klar, welches der beiden Elemente größer oder kleiner ist. Somit ist nicht klar wie diese Elemente anzuordnen sind.
Dies ist eine grundsätzliche Voraussetzung für die Eingabemenge für vergleichsbasierte Sortieralgorithmen. Diese Defintion stellt eine Verallgemeinerung der $\leq$-Relation für einelementige Schlüßel aus der Vorlesung dar.

Aufgabe 3

Wir betrachten die Anzahl der Blocktransfers die beim externen Sortieren auftreten. Diese ist in der Vorlesung mit $\mathcal{O}(\frac{n}{B} \log_{M/B} \frac{n}{M})$ hergeleitet worden.

- Zeigen Sie: Der Algorithmus für das externe Sortieren muss unter normalen Umständen (realistische Werte für M und B) für noch nicht einmal jedes Element eine Lese- oder Schreiboperation ausführen.

Gehen Sie dabei wie folgt vor: Gehen Sie von der asymptotischen Anzahl der Blocktransfers aus und geben Sie eine Bedingung für n an, so dass mindestens n Lese- oder Schreiboperation auf dem Hauptspeicher ausgeführt werden müssen. Argumentieren Sie dann warum diese Bedingung meist nicht zu erfüllen ist, oder wie M und B zu wählen ist, damit diese Bedingung realistisch erfüllbar ist.

- Man kann für externes Sortieren eine untere Schranke von $\Omega(\frac{n}{B}\log_{M/B}\frac{n}{B})$ benötigten Blocktransfers zeigen. Ist damit der in der in der Vorlesung gezeigte Algorithmus asymptotisch optimal?

Lösungsvorschlag

- Wir berechnen ein Kriterium, so dass $\frac{n}{B}\log_{M/B}\frac{n}{M} \geq n$:

$$\begin{aligned} & \frac{n}{B}\log_{M/B}\frac{n}{M} \geq n \\ \Leftrightarrow \quad & \log_{M/B}\frac{n}{M} \geq B \\ \Leftrightarrow \quad & \frac{n}{M} \geq \left(\frac{M}{B}\right)^B \\ \Leftrightarrow \quad & n \geq M\left(\frac{M}{B}\right)^B. \end{aligned}$$

Gehen wir nun davon aus, dass die Blockgröße 100 Zahlen beeinhalten kann und $\frac{M}{B} = 2$ sei, was sehr klein wäre. So müsste man schon 2^{100} Elemente sortieren (geschätze Anzahl der Atome im Universum).

- Die obere Schranke ist asymptotisch optimal, da für ein $0 < c < 1$

$$\begin{aligned} \frac{n}{B}\log_{M/B}\frac{n}{M} = \frac{n}{B}(\log_{M/B}\frac{n}{M} + 0) = \frac{n}{B}(\log_{M/B}\frac{n}{M} + \log_{M/B}\frac{B}{B}) = \\ \frac{n}{B}(\log_{M/B}\frac{nB}{BM}) = \frac{n}{B}(\log_{M/B}\frac{n}{B} + \log_{M/B}\frac{B}{M}) = \frac{n}{B}(\log_{M/B}\frac{n}{B} - 1) \leq \\ \frac{n}{B}(c\log_{M/B}\frac{n}{B}) \end{aligned}$$

gilt. Somit ist also $\frac{n}{B}\log_{M/B}\frac{n}{M} \in \mathcal{O}(\frac{n}{B}\log_{M/B}\frac{n}{B})$. Dies bedeutet, dass die obere Schranke höchstens so schnell wächst wie die untere Schranke. Langsamer wachsen kann sie aber auch nicht, da $\frac{n}{B}\log_{M/B}\frac{n}{B}$ sonst keine unteree Schranke wäre. Deshalb müßen Sie bis auf einen konstanten Faktor gleich sein.