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

Sommersemester 2011
Übungsblatt 3
16. September 2011

Grundlagen: Algorithmen und Datenstrukturen

Abgabetermin: 16. September 2011

Hausaufgabe 1

Implementieren Sie die Methoden `insert`, `remove` und `find` für Hashing mit Linear Probing. Verwenden Sie die Hashfunktion

$$h(x) = ax \bmod m.$$

Stellen Sie sicher, dass nach dem Löschen eines Elements die Invariante gilt, dass für jedes Element e in der Hashtabelle mit idealer Position $i = h(\texttt{key}(e))$ und aktueller Position j gilt: $T[i], T[i+1], ..., T[j]$ sind besetzt.
Verwenden sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die Klasse `LinHash`.

Hausaufgabe 2

Implementieren Sie die Methoden `insert`, `remove` und `find` für Cuckoo-Hashing.
Als Hashfunktionen werden Funktionen vom Typ

$$\left(\left(\sum_{j=0}^{k-1} a_j x^j\right) \bmod p\right) \bmod n$$

mit einem Vektor $a = (a_0, ..., a_{k-1})$ und ganzen Zahlen k, p und n verwendet, die bei der Initialisierung übergeben werden. Beachten Sie, dass x hier nur eine ganze Zahl und *kein* Vektor ist. Die Zahl n gibt hier die Größe der Hashtabelle an. Außerdem soll bei der Initialisierung der Hashtabelle ein Wert `max` übergeben werden, der einen Höchstwert für die Anzahl der Verschiebungen angibt (in der Vorlesung sind dies $2 \log n$). Wenn dieser Wert der Verschiebungen erreicht wird, so dürfen Sie das Programm sofort beenden.
Verwenden Sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die Klasse `DynHash`.

Hausaufgabe 3

Implementieren Sie in der Klasse `UIsmsArray` den MergeSort-Algorithmus, in der Funktion `sort`, der die Elemente in dem Feld `A` sortiert.
Verwenden Sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die Klasse `UIsmsArray`.

Aufgabe 1

Konstruieren Sie eine statische, perfekte Hashtabelle für die Elemente:

$$(16,10,11) \quad (8,2,15) \quad (7,12,8) \quad (1,10,3) \quad (13,11,14) \quad (6,11,14)$$
$$(7,3,16) \quad (2,2,8) \quad (10,5,15) \quad (7,3,14) \quad (2,10,1) \quad (14,11,6)$$

Jedes Element x besteht aus den Stellen (x_0, x_1, x_2). Verwenden Sie jeweils passend eine der Hashfunktionen:

$$(\textstyle\sum_{i=0}^{2} 2^i x_i) \bmod 17$$
$$(\textstyle\sum_{i=0}^{2} a_i x_i) \bmod 7 \text{ mit } \mathbf{a} = (0,0,1) \text{ oder } \mathbf{a} = (6,6,2)$$
$$(\textstyle\sum_{i=0}^{2} a_i x_i) \bmod 3 \text{ mit } \mathbf{a} = (1,0,0) \text{ oder } \mathbf{a} = (0,2,2).$$

Aufgabe 2

Wir betrachten ein Negativbeispiel für eine Hashfunktion, die auf einem String s der Länge n aus Charactern $(s_1, \ldots, s_n)$ arbeitet:

$$h(s) := \sum_{i=1}^{n} \text{Anzahl der Einsen in der Binärdarstellung von } s_i.$$

Nehmen Sie an, dass jedes der 256 Zeichen in dem Text gleichwahrscheinlich vorkommt. Begründen Sie, warum Sie die Hashfunktion nicht für geeignet halten.

Aufgabe 3

Ein Hotelmanager hat n Buchungen für die nächste Saison. Sein Hotel hat k identische Räume. Die Buchungen enthalten ein Ankunfts- und ein Abreisedatum. Er will herausfinden, ob er zu allen Zeiten genügend Räume für die Buchungen zur Verfügung hat. Entwickeln Sie einen Algorithmus, der dieses Problem in Zeit $\mathcal{O}(n + \text{sort}(n))$ löst.

Aufgabe 4

Geben Sie ein Verfahren an, um 5 Elemente mit 7 Vergleichen zu sortieren.

Aufgabe 5

Wir haben uns in der Vorlesung beim Beweis, dass die Laufzeit von MergeSort in $\mathcal{O}(n \log n)$ liegt, auf das Mastertheorem berufen.
Zeigen Sie ohne Benutzung des Mastertheorems, dass die geschlossende Darstellung der Rekursionsformel

$$\begin{aligned} T(n) &= T(\lfloor \tfrac{n}{2} \rfloor) + T(\lceil \tfrac{n}{2} \rceil) + \Theta(n) \\ T(1) &= \Theta(1) \end{aligned}$$

in $\mathcal{O}(n \log n)$ liegt. Sie dürfen vereinfachend annehmen, dass n eine Zweierpotenz ist.

Aufgabe 6

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?