Ludwig-Maximilians-Universität München
Department Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme
Prof. Dr. Hans-Peter Kriegel
Tobias Emrich, Johannes Niedermayer

München, 06.06.2012

Algorithmen und Datenstrukturen

SS 2012

Übungsblatt 8: Hashing + Sortieralgorithmen

Besprechung: 18.06.2012 - 22.06.2012

Abgabe dieses Übungsblattes bis spätestens Montag, 18.06.12, 10:00 Uhr.

Aufgabe 8-1 *Lineares Hashing mit partiellen Erweiterungen*

Betrachten Sie lineares Hashing mit $n_0 = 2$ partiellen Erweiterungen. Gegeben seien

- $2N$: anfängliche Größe der Hashtabelle (konstant)
- $n \in \{1, \ldots, n_0\}$: aktuelle partielle Expansion, anfangs $n = 1$
- $p \in \{0, \ldots, N-1\}$: Expansionszeiger, d.h. nächste zu expandierende Seite, anfangs $p = 0$
- L: Level, d. h. Anzahl der Verdoppelungen der Tabelle seit Beginn, anfangs $L = 0$
- $h_L(n, k)$: Folge von Hashfunktionen
  $h_0(1, k) = k \mod 2N, h_0(2, k) = k \mod 3N$,
  $h_1(1, k) = k \mod 4N, h_1(2, k) = k \mod 6N$,
  $h_2(1, k) = k \mod 8N, \ldots$

  Beachte: Hier ist $n \in \{1, \ldots, n_0 + 1\}$ und $h_L(n_0 + 1, k) = h_{L+1}(1, k)$, z.B. $h_0(3, k) = h_1(1, k)$.
- Expansionsregel / Kontrollfunktion

(a) Geben Sie die nächsten 5 Hashfunktionen obiger Folge an. Wie ist das allgemeine Bildungsgesetz dieser Hashfunktionenfolge (d.h. geben Sie die Formel für $h_L(n, k)$ an)?

(b) Fügen Sie in die Hashtabelle ($N = 1$, Primärseiten Größe 2, Überlaufsseiten Größe 1, Expansionsregel Speicherplatzausnutzung $\alpha = 0.7$) die Schlüssel 27, 23, 22, 21, 19, 16, 15, 12, 11, 7, 3 und 2 ein. Stellen Sie den Zustand der Hashtabelle (Primärseiten, Überlaufseiten) nach jeder Expansion und nach der letzten Einfügung graphisch dar.
(Hinweis: Verwenden Sie die Speicherplatzausnutzung, nicht den Belegungsfaktor – expandieren Sie, falls die Kontrollfunktion verletzt werden würde.)

Zur Wiederholung:

Eine Sortierung von Objekten basiert auf der Definition einer vollständigen Ordnung "$<$" auf dem Wertebereich der Objekte. Eine vollständige Ordnung ist u.a. dadurch charakterisiert, dass für zwei beliebige Objekte a und b genau eine der folgenden Beziehungen gilt:

(a) $a < b$

(b) $b < a$

(c) $a = b$

Diese Eigenschaft ist auch bekannt als "*Gesetz der Trichotomie*" (obwohl "*Tritomie*" eigentlich der korrektere Name wäre).

Üblicherweise nimmt man auch an, dass für drei beliebige Objekte a, b und c gilt: Wenn $a < b$ und $b < c$, dann $a < c$. Diese Eigenschaft ist bekannt als "*Transitivität*". Als Ergebnis eines Sortierverfahrens auf n Schlüsseln K_i, $i = 1, \ldots, n$ erwartet man eine Permutation $p(1), \ldots, p(n)$ der Indizes, so dass die Schlüssel in nicht-absteigender Ordnung angeordnet sind:

$$K_{p(1)} \leq K_{p(2)} \leq \ldots \leq K_{p(n)}$$

Ein Sortierverfahren kann zusätzlich *stabil* sein. Dann wird von der Permutation p verlangt, dass gilt:

$$(K_{p(i)} = K_{p(j)} \wedge i < j) \Rightarrow p(i) < p(j)$$

Aufgabe 8-2 *Stabilität von Sortierverfahren*

Entscheiden Sie, ob folgende Aussagen zutreffen und begründen Sie Ihre Entscheidung:

(a) "Sortieren durch Abzählen ist stabil."

(b) "Sortieren durch direktes Einfügen (Insertion-Sort) ist stabil."

Aufgabe 8-3 *Eindeutigkeit von Sortierverfahren*

(a) Gegeben sei ein *stabiles* Sortierverfahren, basierend auf einer vollständigen Ordnung, die den Gesetzen der *Trichotomie* und *Transitivität* genügt. Beweisen Sie, dass die Permutation $p(1), p(2), \ldots, p(n)$, die man als Ergebnis des Sortierverfahrens erhält, *eindeutig bestimmt* ist.

(b) Nun sei eine Ordnung "$<$" auf $K_1, \ldots, K_n$ gegeben, die dem Gesetz der *Trichotomie* genügt, aber *nicht transitiv* ist. Begründen Sie, warum es auch möglich ist, die Schlüssel *stabil* zu sortieren, wenn die vorausgesetzte Ordnung *nicht transitiv* ist. Nennen Sie zwei Beispiele für Sortierverfahren aus der Vorlesung, die im nicht-transitiven Fall eine stabile Sortierung ermöglichen.

Aufgabe 8-4 *Merge-Sort*

Sortieren Sie das folgende Array mit dem MergeSort-Verfahren *exakt wie es in der Vorlesung besprochen wurde*.

Geben Sie die Zwischenergebnisse des MergeSort-Algorithmus in dem untenstehenden Graphen an, der die in der Vorlesung verwendete Struktur besitzt. Die Obere HÏfte des Graphen stellt die verschiedenen Rekursionsstufen des Divide-Schritts dar, der obere Knoten des Graphen enthÏlt das unsortierte Array. In die untere HÏfte des Graphen soll der Merge-Schritt eingetragen werden.

Zu sortieren ist die folgende Liste von Zahlen:
25, 76, 3, 12, 35, 2, 70, 20