7. Aufgabenblatt zur Vorlesung

Algorithmen, Datenstrukturen und Datenabstraktion WS 17/18

Wolfgang Mulzer, Katharina Klost

Abgabe am 08. Dezember 2017 bis 10 Uhr in die jeweiligen Tutorenfächer

Aufgabe 1 Hashing: Implementierung *10 Punkte*

(a) Lesen Sie die Dokumentation von `hashCode()` in `java.lang.Object`. Welche zwei Bedingungen muss diese Funktion erfüllen? Warum sollte man sie in eigenen Klassen überschreiben?

(b) Implementieren Sie Hashing mit Verkettung wie in der Vorlesung beschrieben. Benutzen Sie dabei die Funktion `hashCode()`. Für die verketteten Listen können Sie die Java-Bibliothek verwenden. Die Hashtabelle soll dynamisch wachsen, sobald ein bestimmter Belegungsfaktor erreicht ist.

Experimentieren Sie mit ihrer Implementierung: Wie hängen die Zugriffszeiten vom Belegungsfaktor ab? Vergleichen Sie zwei verschiedene Hashfunktionen (diese können für eine feste Größe der Hashtabelle hart kodiert sein).

Aufgabe 2 Hashing: Worst-case-Analyse *10 Punkte*

Sei A eine Hashtabelle der Größe N, und $n \in \mathbb{N}$ beliebig. Zeigen Sie: Für jede Schlüsselmenge K mit $|K| \geq (n-1)N + 1$ und jede Hashfunktion $h : K \to \{0, \ldots, N-1\}$ existiert eine Menge $S \subseteq K$ mit $|S| = n$, so dass alle Elemente von S auf denselben Eintrag in A abgebildet werden. Was bedeutet das für die worst-case Laufzeit von Hashing mit Verkettung? Wie verträgt sich das mit der Analyse aus der Vorlesung?

Aufgabe 3 Offene Adressierung *10 Punkte*

Wir fügen eine Menge S von n Einträgen in eine Hashtabelle T der Länge $m = 6n$ ein. Die Kollisionsbehandlung erfolgt durch offene Adressierung mit linearem Sondieren. Wir nehmen an, dass die Hashfunktion h sich zufällig verhält. Die Indizes in T werden modulo m gerechnet, d.h., $T[-1] = T[m-1]$ und $T[m] = T[0]$.

(a) Sei $k \in \mathbb{N}, k < m$. Ein *Klumpen der Länge* k in T ist eine Folge $i, i+1, \ldots, i+k-1$ von aufeinanderfolgenden Indizes, so dass $T[i], T[i+1], \ldots, T[i+k-1]$ alle belegt sind, aber $T[i-1]$ und $T[i+k]$ frei sind.

Zeigen Sie: Die Wahrscheinlichkeit, dass T nach dem Einfügen von S einen Klumpen der Länge k enthält, der bei i beginnt, ist höchstens

$$p_{i,k} := \binom{n}{k} \left(\frac{k}{m}\right)^k \left(\frac{m-k}{m}\right)^{n-k}.$$

Hinweis: Wenn ein Klumpen der Länge k bei i beginnt, so gibt es eine Teilmenge $S' \subseteq S$ von k Einträgen, die von h nach $i, i+1, \ldots, i+k-1$ abgebildet werden. Die anderen Hashwerte liegen außerhalb dieser Menge.

(b) Zeigen Sie mit Hilfe von $m = 6n$ und der Abschätzung $\binom{n}{k} \leq (\frac{ne}{k})^k$, dass

$$p_{i,k} \leq 2^{-k}$$

ist. Hierbei ist $e \approx 2.71\ldots$ die Eulersche Zahl.

(c) Geben Sie eine möglichst gute Abschätzung für die Wahrscheinlichkeit, dass T einen Klumpen der Länge $2 \log n$ enthält.

(d) (*freiwillig, 5 Zusatzpunkte*) Sei $x \notin S$ ein fester Schlüssel. Argumentieren Sie, dass die erwartete Laufzeit zum Nachschlagen von x in T höchstens proportional ist zu

$$1 + \sum_{k=0}^{n} k \Pr[h(x) \text{ liegt in einem Klumpen der Länge } k]. \qquad (*)$$

Zeigen Sie, dass sich (*) schreiben lässt als

$$1 + \sum_{k=0}^{n} k^2 \Pr[\text{ein Klumpen der Länge } k \text{ beginnt bei } h(x)] \qquad (*)$$

und folgern Sie, dass die erwartete Nachschlagezeit für x konstant ist.

Aufgabe 4 Kryptographische Hashfunktionen *freiwillig, 10 Zusatzpunkte*

(a) Was ist eine kryptographische Hashfunktion? Nennen Sie drei Beispiele.

(b) Welche kryptographischen Hashfunktionen sind in Java implementiert? Wie kann man sie verwenden?

(c) Passen Sie Ihre Hashtabelle aus Aufgabe 1(b) so an, dass sie eine kryptographische Hashfunktion verwendet und führen Sie geeignete Tests durch.

Nennen Sie einen Vorteil und einen Nachteil dieser Implementierung.

Hinweis: Bitte formatieren Sie Ihre Abgaben mit LaTeX oder einem vergleichbaren elektronischen Textverarbeitungssystem. Ansonsten droht der Abzug von bis zu 10% der erreichbaren Punkte.