7. Aufgabenblatt zur Vorlesung

Algorithmen und Programmierung III

Wolfgang Mulzer WiSe 2013/14

Abgabe am 5. Dezember 2013 vor der Vorlesung in die jeweiligen Tutorenfächer

Aufgabe 1 Offene Adressierung *10 Punkte*

Sei T eine leere Hashtabelle der Länge m. Wir fügen nacheinander $n \leq m/2$ Einträge in T ein. Die Kollisionsbehandlung erfolgt durch offene Adressierung mit linearem Sondieren. Wir nehmen an, dass die Hashfunktion sich zufällig verhält.

(a) Zeigen Sie: für jede Einfügeoperation ist die Wahrscheinlichkeit, dass mindestens k Positionen in T inspiziert werden, höchstens 2^{1-k}.
Achtung: Sie dürfen annehmen, dass jeder Eintrag der Tabelle unabhängig mit Wahrscheinlichkeit $1/2$ besetzt ist. Das stimmt zwar nicht, aber andernfalls ist die Analyse schwierig

(b) Folgern Sie: für jede Einfügeoperation ist die Wahrscheinlichkeit, dass mindestens $2 \log n$ Positionen in T inspiziert werden, höchstens $1/n^2$.

(c) Mit Wahrscheinlichkeit mindestens $1 - 1/n$ benötigen alle Einfügeoperationen jeweils höchstens $O(\log n)$ Schritte.
Hinweis: Für Ereignisse $A_1, \ldots, A_\ell$ gilt: $\Pr[A_1 \vee \cdots \vee A_\ell] \leq \sum_{i=1}^{\ell} \Pr[A_i]$.

Aufgabe 2 Analyse von Skiplisten *10 Punkte*

Sei L eine Skipliste mit n Einträgen. Zeigen Sie:

(a) Die erwartete Anzahl von Knoten in L ist $O(n)$.

(b) Mit Wahrscheinlichkeit höchstens $n/2^j$ besteht L aus mindestens j Listen.
Hinweis: Beachten Sie den Hinweis von 1c.

Aufgabe 3 Implementierung des geordneten Wörterbuchs *10 Punkte*

(a) Formulieren Sie den abstrakten Datentyp *geordnetes Wörterbuch* aus der Vorlesung als Java-Schnittstelle. Achten Sie dabei auf eine geeignete Spezifikation der Methoden in ihren Kommentaren. Implementieren Sie die Schnittstelle mit einer Skipliste.
Hinweis: Pseudocode für Skiplisten findet sich auf der Website.

(b) Fügen Sie zwei Methoden `reverseIterator()` und `iterator()` hinzu, welche jeweils ein `Iterator`-Objekt erzeugen. Der Iterator von `iterator()` soll die Einträge in sortierter Reihenfolge durchlaufen, `reverseIterator()` soll die Einträge in umgekehrter Reihenfolge liefern.
Informieren Sie sich über die Spezifikation von `java.util.Iterator`, und setzen Sie sie um. Sie müssen nicht die `remove()`-Operation unterstützen.

(c) (*freiwillig* 5 Zusatzpunkte)

Was passiert, wenn die Skipliste verändert wird, während ein `Iterator`-Objekt aktiv ist? Beschreiben Sie ein mögliches Problem.

Ein Lösung bieten *fail-fast* Iteratoren: sobald die Skipliste via `put` oder `remove` verändert wird, werden *alle* aktiven `Iterator`-Objekte ungültig. Spätere Methodenaufrufe liefern dann eine `ConcurrentModificationException`. Iteratoren, welche nach der Modifikation erzeugt wurden, sollen aber weiterhin funktionieren. Wie kann man dieses Verhalten umsetzen? Informieren Sie sich über das Entwurfsmuster *Observer*, und beschreiben Sie knapp, wie es unser Problem lösen kann.

Bei dieser Teilaufgabe müssen Sie nichts implementieren.

Aufgabe 4 Cuckoo Hashing (freiwillig) *10 Zusatzpunkte*

Sehen Sie sich die Bachelorarbeit zum Thema Cuckoo-Hashing an, und beantworten Sie folgende Fragen.

(a) Was ist der Kuckucksgraph?

(b) Wie ist der Zusammenhang zwischen dem Kuckucksgraphen und dem Erfolg der Einfügeprozedur? Warum (kurze Begründung)?

(c) Was ist eine Kodierung?

(d) Wie benutzt man Kodierungen, um zu zeigen, dass ein Ereignis geringe Wahrscheinlichkeit haben muss?

(e) Welche Variante von Cuckoo-Hashing ist in der Praxis am effizientesten?