8. Aufgabenblatt zur Vorlesung

Algorithmen, Datenstrukturen und Datenabstraktion WS 17/18

Wolfgang Mulzer, Katharina Klost

Abgabe am 15. Dezember 2017 bis 10 Uhr in die jeweiligen Tutorenfächer

Aufgabe 1 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 $n/2^{j-1}$ besteht L aus mindestens j horizontalen Listen.

Hinweis: Für Ereignisse $A_1, \ldots, A_\ell$ gilt: $\Pr[A_1 \cup \cdots \cup A_\ell] \leq \sum_{i=1}^{\ell} \Pr[A_i]$.

Aufgabe 2 Implementierung des geordneten Wörterbuchs *10 Punkte*

(a) Formulieren Sie den abstrakten Datentyp `OrderedDictionary` 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) Informieren Sie sich über die Spezifikation von `java.util.Iterator`. 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.

Achten Sie darauf, dass Sie die Spezifikation von und `java.util.Iterator` aus der API-Doku erfüllen. 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 *Observer Pattern*, und beschreiben Sie knapp, wie es unser Problem lösen kann.

Bei dieser Teilaufgabe müssen Sie nichts implementieren.

Aufgabe 3 Binäre Suchbäume *10 Punkte*

(a) Fügen Sie die Schlüssel A, L, P, D, R, E, I, X, Y, Z in dieser Reihenfolge in einen anfangs leeren binären Suchbaum ein. Löschen Sie sodann die Schlüssel Z, A, L. Zeichnen Sie den Baum nach jedem Einfüge- und Löschvorgang.

(b) Seien T_1 und T_2 zwei binäre Suchbäume, in denen jeweils die gleiche Menge S von Einträgen gespeichert ist, mit $|S| = n$. Zeigen Sie: Es gibt eine Folge von höchstens $2n$ einfachen Rotationen, die T_1 nach T_2 überführt.

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?

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