Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Dr. Hanjo Täubig
Jeremias Weihmann

Sommersemester 2014
Übungsblatt 10
23. Juni 2014

Grundlagen: Algorithmen und Datenstrukturen

Abgabetermin: Jeweilige Tutorübung in der Woche vom 30. Juni bis 4. Juli

Tutoraufgabe 1

Führen Sie auf einem anfangs leeren $(2,4)$-Baum folgende Operationen aus und zeichnen Sie die Zwischenergebnisse: `insert(23)`, `insert(30)`, `insert(13)`, `insert(6)`, `insert(40)`, `insert(80)`, `insert(62)`, `insert(75)`, `insert(28)`, `insert(21)`, `insert(29)`, `remove(62)`, `remove(75)`, `remove(13)`.

Anmerkung: Zum leichteren Verständnis der Operationen beobachten wir, dass ein Split-Schlüssel stets dem größten (d.h. rechtesten) Blatt entspricht, das sich in dem zu dem Split-Schlüssel korrespondierenden Unterbaum befindet.

Tutoraufgabe 2

Diese Tutoraufgabe hat eine Eigenschaft von $(2,3)$-Bäumen zum Gegenstand, die für die Hausaufgabe ausgenutzt werden kann. In der Hausaufgabe wiederum wird eine Aussage aus der Vorlesung bewiesen.

Wir betrachten einen $(2,3)$-Baum, der nur das Dummy-Element ∞ enthält, und fügen in den Baum der Reihe nach die Zahlen $1, 2, \ldots, n$ ein. Zeigen Sie, dass der resultierende Baum ein vollständiger Binärbaum ist, wenn $n+1$ eine Zweierpotenz ist.

Zur Erinnerung: Ein Binärbaum ist echt, wenn jeder innere Knoten genau zwei Kinder hat. Ein Binärbaum ist vollständig, wenn er echt ist und alle Blätter dieselbe Tiefe besitzen.

Hinweis: Betrachten Sie den Pfad von der Wurzel bis zum rechtesten/größten Blatt des Baumes (das Dummy-Element ∞). Sehen Sie einen Zusammenhang zwischen den Knotengraden der inneren Knoten, die auf diesem Pfad liegen, und der Zahl der Blätter im Baum? Führen Sie drei/vier Einfügeoperationen durch, um ein Gefühl für den Zusammenhang zu entwickeln.

Hausaufgabe 1

Beweisen Sie folgende Aussage: Für einen $(2,3)$-Baum gibt es eine Folge von n insert bzw. remove-Operationen, sodass die Anzahl der nötigen Aufspaltungen und Vereinigungen in $\Omega(n \log n)$ ist.

Hinweis: Verwenden Sie in Abhängigkeit von n zwischen einem Viertel und der Hälfte der Operationen dafür, einen großen vollständigen binären $(2,3)$-Baum zu konstruieren, und den Rest der Operationen auf solche Weise, möglichst viele Aufspaltungen und Vereinigungen zu erhalten. Die Aussage der zweiten Tutoraufgabe kann hilfreich sein.

Hausaufgabe 2

Führen Sie auf einem anfangs leeren $(2, 3)$-Baum folgende Operationen aus und zeichnen Sie die Zwischenergebnisse: `insert(11)`, `insert(66)`, `insert(22)`, `insert(55)`, `insert(33)`, `insert(44)`, `insert(99)`, `insert(7)`, `insert(6)`, `insert(5)`, `insert(4)`, `remove(22)`, `remove(11)`, `remove(7)`, `remove(44)`, `remove(66)`, `remove(55)`, `remove(4)`,

Hausaufgabe 3

Implementieren Sie in der Klasse `UIbiHeap` einen Binomial-Heap. Hierzu muss auch ein BinomialBaum in der Klasse `binomialTree` implementiert werden.

Verwenden Sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die Klassen `UIbiHeap` und `binomialTree`.

Achten Sie bei der Abgabe Ihrer Aufgabe darauf, dass Ihre Klassen `UIbiHeap` und `binomialTree` heißen und auf den Rechnern der Linuxhalle (`lxhalle.informatik.tu-muenchen.de`) mit der bereitgestellten Datei `main_bi` kompiliert werden können. Anderenfalls kann eine Korrektur nicht garantiert werden. Achten Sie darauf, dass Ihr Quelltext ausreichend kommentiert ist.

Schicken Sie die Lösung per Email mit dem Betreff `[GAD] Gruppe <Gruppennummer>` an Ihren Tutor.