Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Dr. Hanjo Täubig
Tobias Lieber

Sommersemester 2011
Übungsblatt 9
6. Juli 2011

Grundlagen: Algorithmen und Datenstrukturen

Abgabetermin: In der jeweiligen Tutorübung

Hausaufgabe 1

Implementieren Sie in der Klasse `Graph` eine Repräsentation eines Graphen durch eine Adjazenzliste. Stellen Sie die Funktionen wie im Interface angegeben bereit.
Verwenden Sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die bereitgestellten Klassen aber nicht deren Interfaces.
Achten Sie bei der Abgabe Ihrer Aufgabe darauf, dass Ihre Klassen den Rechnern der Rayhalle (`rayhalle.informatik.tu-muenchen.de`) mit der bereitgestellten Datei `main_gr` kompiliert werden kann. 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.

Hausaufgabe 2

Implementieren Sie in der Klasse `DFS` eine Funktion, die auf der eben implementierten `Graph`-Klasse zu einem angegebenen Knoten s eine Tiefensuche durchführt und die Knoten ausgibt nachdem alle Kinder abgearbeitet sind. Achten Sie darauf, dass Ihre Implementierung im Gegensatz zu dem in der Vorlesung vorgestellten Ansatz nicht rekursiv ist.
Verwenden Sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die bereitgestellten Klassen aber nicht deren Interfaces.
Achten Sie bei der Abgabe Ihrer Aufgabe darauf, dass Ihre Klassen den Rechnern der Rayhalle (`rayhalle.informatik.tu-muenchen.de`) mit der bereitgestellten Datei `main_dfs` kompiliert werden kann. 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.

Aufgabe 1

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

Aufgabe 2

In dieser Aufgabe modifizieren wir die gestellten Bedingungen für einen (a, b)-Baum so dass wir einen B^*-Baum erhalten.
Wir ändern die Grad-Invariante wie folgt: Ein Knoten darf höchstens Grad b haben. Jeder Knoten außer der Wurzel hat mindestens $\frac{2b-1}{3}$ Kinder. Die Wurzel hat mindestens 2 Kinder und höchstens $2\lfloor\frac{2b-2}{3}\rfloor + 1$ Kinder.

- Wie müssen die `insert` und `delete`-Operationen des (a, b)-Baumes modifiziert werden, so dass die Grad-Invariante immer gegeben ist?
- Welche Vorteile und Nachteile ergeben sich aus praktischer und theoretischer Sicht für die Speicherausnutzung und die Laufzeit der `insert`-Operationen?

Aufgabe 3

Geben Sie Algorithmen `postNext(v)` und `preNext(v)` an, die zu einem Knoten v in einem Binärbaum den in der PreOrder bzw. PostOrder folgenden Knoten w berechnet. Analysieren Sie die asymptotische Worst-Case Laufzeit Ihres Pseudocodes.
Berechnen Sie außerdem die asymptotische Laufzeit wenn mittels der Operationen `postNext(v)` und `preNext(v)` die vollständige PreOrder bzw. PostOrder berechnet wird (also n-maliges Anwenden der Funktion).