Image Analysis Lab
Jun.-Prof. Dr.
Olaf Ronneberger
Claudius Korzen

Algorithmen und Datenstrukturen (ESE + IEMS) WS 2013 / 2014

http://lmb.informatik.uni-freiburg.de/lectures/

UNI FREIBURG

Übungsblatt 7

Abgabe für ESE: bis Dienstag, den 17. Dezember um 10:00 Uhr

Abgabe für IEMS: bis Donnerstag, den 26. Dezember um 16:00 Uhr

Erweitern Sie die Klasse *DynamicIntArray* aus der Vorlesung. Im SVN Ordner zur Vorlesung 7 finden Sie sowohl die Java Dateien als auch eine vergleichbare Version in C++.

Aufgabe 1 (5 Punkte)

Implementieren Sie analog zu der Methode *append* aus der Vorlesung die Methode *remove*, die das letzte Element aus der Liste entfernt. Schrumpfen Sie das unterliegende fixed-size Array immer dann, wenn es nur noch zu weniger als einem Drittel gefüllt ist. Eine Folge von n append/remove Operationen sollte amortisiert lineare Laufzeit haben.

Aufgabe 2 (5 Punkte)

Erweitern Sie die Klasse *DynamicArrayMain*, so dass verschiedene Tests ausgewählt werden können und dabei eine Ausgabe produziert wird, die Sie gut in eine Kurve für die akkumulierte Laufzeit über eine Folge von n Operationen verwandeln können, z.B. wie in der Vorlesung mit *gnuplot*.

Test 1 Eine Folge aus 10 Millionen *append* Operationen auf ein anfangs leeres Feld.

Test 2 Eine Folge aus 10 Millionen *remove* Operationen auf ein Feld mit anfangs 10 Millionen Elementen.

Test 3 Eine Folge aus 10 Millionen Operationen auf ein Feld mit anfangs 1 Millionen Elementen, wobei die Operationen mit *append* anfangen und nach jeder Reallokation zwischen *append* und *remove* alternieren.

Test 4 Wie Test 3, aber beginnend mit *remove*.

Erzeugen Sie jeweils eine Kurve für die Laufzeit und committen Sie diese in einem Ordner *non-code* als Teil ihrer Lösung in das SVN.

Aufgabe 3 (7 Punkte)

Für eine Echtzeit-Anwendung benötigen Sie ein dynamisches Array mit konstanter Ausführungszeit (auch im worst-case) für alle Opertionen. D.h. das lange Umkopieren bei bestimmten append() Operationen ist nicht tolerierbar. Schreiben Sie eine Klasse WorstCaseConstantAccessTimeArray mit den Methoden append(int item), get(int index), und put(int index, int item), die diese Eigenschaften erfüllt, und erstellen Sie eine Laufzeit-Kurve dazu im Order *non-code*. Tipp: Speichern Sie die Elemente in bis zu zwei fixed-size Arrays und beginnen Sie, die Werte umzukopieren, bevor die Kapazität des kleineren Arrays erschöpft ist.

Aufgabe 4 (3 Punkte)

Finden Sie heraus, welche Vergößerungs- und Verkleinerungsstrategie die ArrayList-Klasse (Java), bzw. der std::vector (C++) nutzen.

Aufgabe 5 (Zusatzaufgabe: 7 Punkte)

Der Programmierer aus dem Nachbarbüro hätte gerne ein Formel für seine Formelsammlung, mit der er ausrechnen kann, um welchen Vergößerungsfaktor r er ein Array vergößern muss, damit im Mittel (amortisiert) eine append()-Operation die Kosten $k \cdot A$ nicht übersteigt. In der Vorlesung haben wir bewiesen, dass für den Vergößerungsfaktor $r = 3/2$ diese Kosten $4A$ sind, also $k = 4$.

Committen Sie, wie gehabt, Ihren Code in das SVN, in einen neuen Unterordner *uebungsblatt_07*, sowie, ebendort, Ihr Feedback in einer Textdatei *erfahrungen.txt*. Insbesondere: Wie lange haben Sie ungefähr gebraucht? An welchen Stellen gab es Probleme und wieviel Zeit hat Sie das gekostet?