Prof. Dr. A. Weber
Hassan Errami

Universität Bonn
Institut für Informatik II
06.01.2014

Wintersemester 2013

Übungen zur Vorlesung
Algorithmisches Denken und imperative Programmierung (BA-INF-014)
Aufgabenblatt 7

Zu bearbeiten bis: 10.01.2014

Aufgabe 1 *(Dynamische Arrays - 6 Punkte)*

Ein dynamisches Array hat die Eigenschaft, dass man effizient am Ende des Arrays neue Elemente anfügen kann, indem man in einem statischen Array neben der Länge des statischen Arrays auch die aktuelle Größe des dynamischen Arrays speichert. Solange die aktuelle Größe kleiner als die maximale Länge ist, kann der Speicher des statischen Arrays verwendet werden

Wenn wir an ein Array ein Element anhängen wollen, wenn die aktuelle und maximale Größe , müssen wir neuen Speicher allokieren, der die gewünschte Länge hat (oder eine größere). Die Werte aus dem alten Array müssen dann in den neuen Speicher umkopiert werden. Danach kann das neue Element hinten angefügt werden, weil wir im neuen Array bereits Speicher für dieses Element reserviert haben.

Bei der naiven Implementation des dynamischen Arrays wird jeweils Speicher der benötigten Größe allokiert; eine andere Strategie ist es, die Größe des statischen Arrays jeweils zu verdoppeln, wenn der Speicherplatz nicht ausreichend war.

a) Schreiben Sie einen Datentyp *DynArray* zur Speicherung von dynamischen *int*-Arrays. Dieser soll die aktuelle und maximale Größe des Arrays speichern und die oben skizzierte Verdopplungsstrategie realisieren. Schreiben Sie einen Datentyp *DynArrayMin*, die die Strategie realisiert, dass jeweils nur der benötigte Speicherplatz allokiert wird.

Beachten Sie, dass Sie nicht nur Speicher für eine neues Array allokieren, sondern den Speicher für das alte auch wieder freigeben, sobald dieses nicht mehr benötigt wird.

b) Implementieren Sie einen Algorithmus (*InsertSort*) zum Sortieren Einfügen in einer Folge (F). Verwenden Sie dabei in zwei Versionen die Datentypen *DynArray* und *DynArrayMin*. Vergleichen Sie die Laufzeiten Ihrer Implementierungen bei sortierten Einfügen von 1000000 Zufallszahlen.

Aufgabe 2 *(Komplexität - 4 Punkte)*

Bestimmen Sie die Zeit-Komplexität der Funktionen für die Listenoperationen *insertFirst*, *insertLast*, *insertSorted* und *reverseList*.

Was ist die Zeit-Komplexität dieser Funktionen, wenn bei den Listen nicht nur ein Pointer auf den Anfang der Liste, sondern auch auf das Ende der Listen gespeichert wird?