Hochschule für Technik und Wirtschaft des Saarlandes
University of Applied Sciences

Programmiersprachen II

Fortsetzung Datenstrukturen

Einfache Bäume

Übung 14

Prof. Dr. Reiner Güttler
Fachbereich GIS
HTW

Kapitel 2: Folge Datenstrukturen

Hintergrund:

In vielen realen Anwendungsfällen besteht für eine Menge von Items irgendeines Typs (Bsp. Personen) der Bedarf

- sowohl für Operationen, die eine sortierte Datenhaltung nahe legen (z.B. find(), insert(), remove())
- als auch für Operationen, die dies nicht erfordern (z.B. alle Operationen, die sich ohnehin auf alle Elemente beziehen).

Deshalb kann es sinnvoll sein

- die Items in einem unsortierten Array zu halten
- und nur die Sortierungsreihenfolge in einem Baum mit Verweisen auf das jeweilige Arrayelement.

Kapitel 2: Folge Datenstrukturen

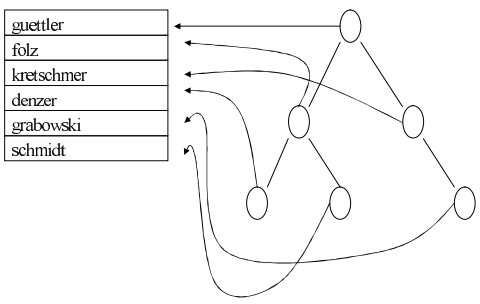
Ausserdem: unter welchen Rahmenbedingungen ist in diesem Fall die Verwendung der contigous sequential representation nicht sinnvoll?

Wie soll das aussehen:

Bsp. Für eine Namensliste, bei der zu jedem Namen irgendeine Menge von Daten gespeichert wird

Eine Eingabefolge der Namen „guettler“, „folz“, „kretschmer“, „denzer“, „grabowski“, „schmidt“ würde zu folgender Konstellation führen:

Kapitel 2: Folge Datenstrukturen



Kapitel 2: Folge Datenstrukturen

Aufgabe 1

- Implementieren sie diese Datenstruktur zunächst unter der Voraussetzung, dass keine Löschoperationen vorkommen. Folge: Alle eingefügten Knoten werden jeweils hinter den letzten im Array abgelegt. Im Baum wird die richtige Stelle gesucht und „normal“ eingefügt. Überlegen sie sich geeignete Definitionen für die Klassen.
- Bemerkung: Unter diesen Voraussetzungen wäre die „contigous sequential representation“ aus der Vorlesung eventuell anwendbar. Aber : siehe Vorbemerkung und zweite Aufgabe

Kapitel 2: Folge Datenstrukturen

Aufgabe 2

Erweitern sie ihre Lösung von Aufgabe 1 um die „delete“-Operation.

Bzgl. des Baums wendet man die „normale“ delete-Funktionalität an.

Ziel: man möchte mit einem möglichst kleinen Array auskommen

Folge: dies erfordert eine geeignete Freispeicherverwaltung im Array,

- d.h. bei einem „insert“ sollen jeweils durch „delete“ entstandene Lücken im Array wieder aufgefüllt werden.
- Dies ist möglich, da die Einträge im Array ja nicht sortiert zu sein brauchen.
- Es erfordert aber, dass man zu jedem Zeitpunkt einen Überblick über alle Lücken hat (von denen aber nur die erste gebraucht wird).