Übung: Algorithmen und Datenstrukturen
SS 2007

Prof. Lengauer
Sven Apel, Michael Claßen, Christoph Zengler, Christof König

Blatt 3

– Votierung in der Woche vom 14.05.07–18.05.07 –

Aufgabe 6 Kopieren von Arrays

In Java sollen in einem Array Objekte gespeichert werden, die die Koordinaten eines Punktes in der Ebene enthalten:

```
public class Point {
  public double x;
  public double y;
  public Point(double x, double y) {
    this.x = x;
    this.y = y;
  }
}
```

Überlegen Sie sich, wie Sie eine komplette Kopie (deep copy) des Arrays erzeugen können und welche Ansätze nicht funktionieren würden, z. B. für ein Array `a` soll `b` die Kopie sein:

```
Point[] a = new Point[3];
a[0] = new Point(0,0);
a[1] = new Point(1,1);
a[2] = new Point(2,2);
Point[] b;
/* ... */
```

Aufgabe 7 Arrays und Listen

Aufgabe ist es, eine Datenstruktur `Buffer` mit folgenden Operationen zu implementieren:

- `void insert(int i, Element e)` fügt an der i-ten Position ein Element in den Puffer ein, ohne dass dabei andere Elemente überschrieben werden. Es ist darauf zu achten, nicht über das bisherige Ende des Puffers hinauszuschreiben. (Tip: Verwenden Sie im Falle der Array-basierten Implementierung eine Variable `size`, die für die aktuelle Größe des Puffers steht.)
- `void push_back(Element e)` fügt ein Element an das Ende des Puffers an, ohne dass dabei andere Elemente überschrieben werden.
- `void push_front(Element e)` fügt ein Element vor dem bisher ersten Element des Puffers ein, ohne dass dabei andere Elemente überschrieben werden.

Verwenden Sie als interne Speicherstruktur (a) ein Array und (b) eine einfach verkettete Liste. Geben sie jeweils an, welche Zeitkomplexität die einzelnen Operationen haben. (Tipp: Wieviele elementare Rechenschritte benötigt die Ausführung einer Operation ungefähr?)

Aufgabe 8 Listen durchlaufen

Stellen Sie sich vor, Sie haben eine Java-Implementierung einer Notizliste gegeben:

```
LinkedList<String> notes = new LinkedList<String>();
notes.add("note 1");
notes.add("note 2");
notes.add("note 3");
notes.add("note 4");
/* ... */
```

Überlegen Sie sich, wieviele verschiedene sinnvolle Möglichkeiten existieren, die Liste in Java zu durchlaufen. Zeigen Sie, wie eine solche Liste unter Verwendung der unten stehenden Mechanismen durchlaufen werden kann:

(a) `for`-Schleife (for-each-Schleife)

(b) `while`-Schleife und Zugriff über Index

(c) `while`-Schleife und Zugriff über Iterator