TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Computer Vision Group
Einführung in die Informatik I für Ingenieure (MSE)
Prof. Dr. Daniel Cremers, Mohamed Souiai

WS 2012
Übungsblatt 10
11.01.2013

Aufgabe 10.1 (Ü) Queues

Eine Schlange ist eine Datenstruktur zur Speicherung von Elementen nach dem FIFO-Prinzip (First-In-First-Out). Die zweite der in der Vorlesung vorgestellten Implementierungen von Schlangen (Klasse: `Queue`) verwendet zur Speicherung der Elemente ein Array. Zwei Indizes, `first` und `last`, geben dabei die Position des ersten bzw. letzten Wertes an. Bei Bedarf wird das Array beim Einfügen von Elementen durch `enqueue()` durch ein größeres Array ersetzt. Eine Verkleinerung erfolgt nicht.

Erweitern Sie die in der Vorlesung vorgestellte Methode `dequeue()` (zum Entnehmen des nächsten Elements), so dass das Array bei Bedarf auch verkleinert wird.

Hinweis: Die zu ergänzende Klasse `Queue` finden Sie auf der Internetseite zur Übung.

Lösungsvorschlag 10.1

Analog zur Stack-Implementierung aus der Vorlesung wird das Feld halbiert, wenn es nur noch zu einem Viertel gefüllt ist. Somit wird vermieden, dass ein gerade verkleinertes Feld bei einem unmittelbar folgenden Aufruf von `enqueue()` sofort wieder vergrößert werden muss (siehe Vorlesung).

Bei der Verkleinerung des Feldes ist zu berücksichtigen, dass die Elemente in der Queue (im Gegensatz zum Stack) nicht unbedingt am Anfang des Feldes liegen, sondern

- in `a[first]`, ..., `a[last]`, falls `first < last`.
- in `a[first]`, ..., `a[a.lengh-1]`, `a[0]`, ..., `a[last]`, sonst.

Die erweiterte Methode `dequeue()`:

```
public int dequeue() {
    if(isEmpty())
        return 0;
    int result = a[first];
    if (first == last) {
        // letzter Wert wurde gelesen; das Feld ist nun leer
        first = last = -1;
    } else {
        // Anfangszeiger ein Element weiter setzen
        int length_of_array = a.length;
        first = (first + 1) % length_of_array;

        // Anzahl der enthaltenen Elemente == Differenz+1
        int number_of_elements;
        if (first < last)
            number_of_elements = last - first + 1;
```

```
            else
                number_of_elements = length_of_array - first + last + 1;

            // verkleinern falls nur ein Viertel des Feldes gefuellt ist
            if (number_of_elements <= length_of_array / 4) {
                int[] b = new int[length_of_array / 2]; // Feld halbieren
                int index = 0;

                // Ueberpruefe wo im Feld die Element liegen
                if (first <= last) {
                    // kopiern der Elemente von first bis last
                    for (int i = first; i <= last; i++) {
                        b[index++] = a[i];
                    }
                } else {
                    // kopieren der Elemente von first bis Ende
                    for (int i = first; i < length_of_array; i++) {
                        b[index++] = a[i];
                    }
                    // kopieren der Element vom Anfang bis last
                    for (int i = 0; i <= last; i++) {
                        b[index++] = a[i];
                    }
                }
                // altes Feld durch neues ersetzen; Zeiger neu setzen
                a = b;
                first = 0;
                last = index - 1;
            }
        }
        return result;
    }
```

Aufgabe 10.2 (H) Queues II

Benutzen Sie nun Ihre Klasse `LinkedList` aus Blatt 9 (siehe Lösungsblatt und File auf der Übungshomepage) , um String-Warteschlangen mit doppelt verketteten Listen zu realisieren. Die Einträge der Warteschlange sollen diesmal vom Typ `String` sein.

Ein Objekt der Klasse `QueueDLL` enthält nur noch einen Verweis, `header`, auf ein Objekt der Klasse `LinkedList`. Die Klasse `LinkedList` enthält bereits die Funktionalitäten aus der Hausaufgabe 9.2. Implementieren Sie die Klasse `QueueDLL`:

a) Erstellen Sie einen Konstruktor, der eine leere `QueueDLL` erzeugt.
b) Implementieren die Methoden:
   - `boolean isEmpty()` testet auf Leerheit
   - `String dequeue()` liefert das erste Element und entfernt dieses
   - `void enqueue(String x)` reiht x in die Warteschlange ein (am Ende)
   - `String toString()` liefert eine String-Darstellung der ganzen Schlange

   Benutzen Sie dabei gegebenenfalls die Methoden der Klasse `LinkedList`.

Lösungsvorschlag 10.2

```
public class QueueDLL {
    // Warteschlange mit doppelt verketteter Liste
```

```
    private LinkedList header;

    // Konstruktor
    public QueueDLL() {
        header = new LinkedList();
    }

    public boolean isEmpty() {
        return header.size() == 0;
    }

    public String toString() {
        return header.toString();
    }

    public void enqueue(String x) {
        header.addFirst(x);
    }

    public String dequeue() {
        String result = header.last();
        header.removeLast();
        return result;
    }

    public static void main(String args[]) {
        QueueDLL queue = new QueueDLL();
        for (int i = 0; i < 100; i++) {
            if (i % 4 == 2) {
                System.out.println("entfernt: " + queue.dequeue());
                System.out.println("queue: " + queue);
            }

            queue.enqueue("" + i);
            System.out.println("queue: " + queue);
        }
        while (!queue.isEmpty()) {
            System.out.println("entfernt: " + queue.dequeue());
            System.out.println("queue: " + queue);
        }
    }
}
```