CHRISTIAN-ALBRECHTS-UNIVERSITÄT ZU KIEL
Institut für Informatik, Arbeitsgruppe Theorie der Parallelität
Prof. Dr. K. Jansen, K.-M. Klein, M.Maack

14. Mai 2015

Präsenzaufgaben zur Vorlesung »Algorithmen und Datenstrukturen«

Blatt 6

Präsenzaufgabe 6.1 (Iteratives Mergesort)
Sortieren Sie das Array $[4,3,2,4,1,42,3,1,23,2,5]$ mittels iterativen Mergesorts. Geben Sie das Array am Ende jedes Durchlaufs der äußeren `while`-Schleife an.

Präsenzaufgabe 6.2 (Stack)
In der folgenden Sequenz steht ein Buchstabe für eine *push*-Operation, bei der der entspechende Buchstabe auf den Stack gelegt wird, und ein Sternchen für eine *pop*-Operation eines Stacks:

E A S * Y * Q U E * * * S T * * * I O * N * * *

(a) Geben Sie die Sequenz der Werte an, die die *pop*-Operationen zurückgeben, wenn diese Sequenz von Operationen auf einem anfänglich leeren Stack ausgeführt wird.

(b) Modifizieren Sie die Sequenz durch Verschieben der Sternchen derart, dass folgende Ausdrücke zurückgeliefert werden, oder belegen Sie durch Argumente, dass es für die einzelnen Fälle keine derartige Sequenz gibt.

(1) E A S Y Q U E S T I O N

(2) N O I T S E U Q Y S A E

(3) U E Q S Y T S I A O E N

Präsenzaufgabe 6.3 (Backtracking)
Gegeben ist eine Landkarte mit n Ländern. Die Länder sollen mit k unterschiedlichen Farben eingefärbt werden, sodass benachbarte Länder jeweils unterschiedliche Farben haben. Entwerfen Sie einen auf Backtracking basierenden Algorithmus, der eine zulässige Färbung für die Länder findet, oder korrekt feststellt, dass keine existiert.

Hinweis: Sie können davon ausgehen, dass eine Funktion $\textsc{Nachbar}(i,j)$ gegeben ist die angibt, ob Land i und Land j benachbart sind.

Präsenzaufgabe 6.4 (Linked-List)
Implementieren Sie eine einfach verkettete Liste in Java. Vervollständigen Sie dafür die folgende Klasse:

```
public class SinglyLinkedList<T> {

  private T value;
  private SinglyLinkedList<T> next;

  public SinglyLinkedList() {
  }

  /* returns the first element of this list */
  public T getFirst() {
  }

  /* returns the rest list */
  public SinglyLinkedList<T> getNext() {
  }

  /* returns true, if the list is empty */
  public boolean isEmpty(){
  }

  /* returns the size of the list */
  public int size() {
  }

  /* returns the element at position index in the list */
  public T get(int index) throws IndexOutOfBoundsException {
  }

  /* Adds the given value at the given index to the list */
  public SinglyLinkedList<T> add(int index, T value)
    throws IndexOutOfBoundsException {
  }

  /* removes the node at the given index */
  public SinglyLinkedList<T> remove(int index)
    throws IndexOutOfBoundsException {
  }

  /* append the given list to this list */
  public SinglyLinkedList<T> append(SinglyLinkedList<T> list) {
  }

  /* converts the list to a String */
  public String toString() {
  }
}
```