CHRISTIAN-ALBRECHTS-UNIVERSITÄT ZU KIEL
Institut für Informatik, Arbeitsgruppe Algorithmen und Komplexität
Prof. Dr. K. Jansen, K.-M. Klein, F. Land M. Rau

26. Mai 2016

Hausaufgaben zur Vorlesung »Algorithmen und Datenstrukturen«

Blatt 7

Hausaufgabe 7.1 (Hashing (2 Punkte))
Wenden Sie die Instanz aus Präsenzaufgabe 7.1

```
47 17 24 70 22 01 40 45 36 59
```

auf eine Hashtabelle der Größe $m = 23$ mit offener Adressierung an. Verwenden Sie als Hashfunktion die Divisionsmethode kombiniert mit

1. linearer Sondierung und
2. quadratischer Sondierung.

Hausaufgabe 7.2 (Suchbäume (2 Punkte))
Entwerfen Sie einen Algorithmus der für einen gegebenen Suchbaum die Zahlen in sortierter Reihenfolge ausgibt.

Welche Laufzeit hat Ihr Algorithmus?

Hausaufgabe 7.3 (Untere Schranken (3 Punkte))
Zeigen Sie, dass jedes vergleichsbasierte Sortierverfahren im *Durchschnitt* mindestens $\Omega(n \log n)$ Vergleiche benötigt.

Hinweis: Modifizieren Sie den Beweis der unteren Schranken im worst case von Sortierverfahren.

Hausaufgabe 7.4 (Hashing (3 Punkte))
Entwerfen Sie eine Implementierung einer Hash-Tabelle für Hashing mit offener Adressierung und linearer Sondierung in Java auf Papier. Verwenden Sie als Hash-Funktion die erweiterte Divisionsmethode. Um ein Element wieder zu entfernen, verwenden Sie eine spezielle Markierung *DELETED* um zukünftiges Suchen nicht zu behindern.

Entwickeln Sie die auf der Rückseite beschriebenen Operationen *hash*, *locate*, *lookup*, *maybelocate*, *insert* und *delete*.

```
//HashTable for keys that are strictly positive integers.
public class HashTable {

    static final int DEFAULT_M=101;
    private static final int EMPTY=0;
    private static final int DELETED=-1;

    private int[] t;
    private int m;
    //Definiert die Parameter der Hashfunktion
    private final int A=13, B=3;

    //Konstruktoren
    public HashTable(){
        this(DEFAULT_M);
    }

    public HashTable(int size){
        m=size;
        t = new int[m];
        // Vorbelegung der Elemente ist 0 = EMPTY.
    }

   // Gibt den Wert der Hash-Funktion zurueck
    private int hash(int x){}

   //Gibt die Position eines abgespeicherten Elementes
      zurueck
    private int locate(int x){}

   // Ueberpruefe, ob x in der Hashtabelle vorhanden ist.
    public boolean lookup(int x){}

   // Sucht einen freien Platz fuer ein Element x mittels
      linearer Sondierung
    private int maybelocate(int x){}

    // Fuegt ein Element x in die Tabelle ein. Wirft eine
       Exception, wenn x bereits vorhanden ist oder kein
       freier Platz vorhanden ist.
    public void insert(int x) throws HashTableException {}

    //Loescht das Element x aus der Hashtabelle. Wirft eine
       Exception, wenn das Element nicht vorhanden ist.
    public void delete(int x) throws HashTableException {}
}
```

Abgabe der theoretischen und praktischen Aufgaben Donnerstag, den 02. Juni, bis spätestens 14 Uhr. Die Abgabe der theoretischen Aufgaben erfolgt im Schrein. Die Abgabe der praktischen Aufgaben erfolgt im iLearn.