

## Datenstrukturen und Algorithmen – Informatik I

### 8. Übung

Abgabe der Lösungen: 28. Juni 2004

---

**Hinweis:** Die Implementierungsaufgaben sind in Java oder C/C++ zu lösen. Der Quellcode sowie ein dokumentierter Beispiellauf sind beim Betreuer der Kleingruppe abzugeben.

#### Aufgabe 35: Implementierung: Hashing

**2+2 Punkte**

1. Implementieren Sie folgende Klasse:

```
class HashMap {
    public HashMap(int n);           //Leere Hashmap für maximal n Einträge zur
                                     //Verfügung stellen.
    public void Insert(int x);       //Schlüssel x in die Hashmap einfügen.
    public void Delete(int x);       //Schlüssel x aus der Hashmap entfernen.
    public boolean Search(int x);    //Feststellen, ob x in der Hashmap
                                     //enthalten ist.
    private int h(int x, int s);     //die Hashfunktion beim s-ten Einfügeversuch.
                                     //    d.h. h(x,0) gibt den Hash-Wert
                                     //          h(x,1) gibt die erste Ausweichposition
                                     //          h(x,2) gibt die zweite Ausweichposition
};
```

Verwenden Sie dabei eine Sondierungs- und Hashstrategie Ihrer Wahl. Testen Sie ihre Implementierung mit den Werten aus Aufgabe 34.

2. Lesen Sie die Datei `vielezahlen.txt`, die es auf der Website zur Vorlesung gibt, ein und fügen Sie die Zahlen in eine Hashtabelle ein. Zählen Sie dabei die Anzahl der Kollisionen mit Ihrer Hash- und Sondierungsstrategie. Welche Methode könnte zu besseren Ergebnissen führen?

#### Aufgabe 36: Algorithmenentwurf: Doppelte Bitvektoren finden

**4 Punkte**

Gegeben sei eine Datei mit 1 050 000 20-Bit Integerwerten. Beschreiben Sie einen möglichst effizienten Algorithmus, der einen Integerwert findet, der mindestens zweimal vorkommt.

#### Aufgabe 37: Algorithmen für Binäre Suchbäume

**1+1+1 Punkte**

Die Schlüssel eines binären Suchbaumes seien ganze Zahlen. Geben Sie einen Algorithmus an, der für einen solchen Suchbaum

1. ein Schüsselpaar mit minimaler Differenz bestimmt.
2. den Median der Schlüssel berechnet (hierzu seien die Schlüssel paarweise verschieden).

Welche Laufzeitkomplexitäten besitzen diese Algorithmen?

**Aufgabe 38: Hashing – Reihenfolge****2+2 Punkte**

Gegeben sei eine Hashtabelle der Größe 7 mit der Belegung

0	1	2	3	4	5	6
1	164	8	21	73	22	89

sowie die Hashfunktion  $h(k) = \text{Quersumme von } k \pmod{7}$ . Als Kollisionsstrategie wurde quadratisches, alternierendes Sondieren (s.u.) gewählt.

1. Geben Sie alle Reihenfolgen an, in denen die Schlüssel in die anfangs leere Hashtabelle eingefügt worden sein können.
2. Gibt es eine bessere Reihenfolge die Schlüssel einzufügen, die zu einer geringeren Anzahl zu inspizierender Hashtabellenplätze bei erfolgreicher Suche führt (dabei sei die Suche nach jedem Schlüssel gleichwahrscheinlich)?

**Zu Aufgabe 38:**

Quadratisches, alternierendes Sondieren entspricht dem in der Vorlesung vorgestellten quadratischen Sondieren mit Verfeinerung, d.h.

$$\begin{aligned} h(x, 0) &= h(x) \\ \text{und für } 1 \leq j \leq \frac{m-1}{2} : & \quad h(x, 2j-1) = (h(x) + j^2) \pmod{m} \\ & \quad h(x, 2j) = (h(x) - j^2) \pmod{m} \end{aligned}$$