Aufgabensammlung zum Buch „Algorithmen und Datenstrukturen“ (Kapitel 4)

1 Aufgaben aus dem Buch

Zu folgenden Aufgaben, die direkt aus dem Buch entnommen sind, gibt es an der Universität Freiburg am Lehrstuhl Ottmann Musterlösungen. In der Version mit Lösungen sind diese angegeben. Hinter der fortlaufenden Aufgabennummer steht in Klammern die Nummer der Aufgabe im Buch.

Aufgabe 1 (Aufgabe 4.5):

Gegeben sei eine anfangs leere Hashtabelle mit 13 Elementen, in die der Reihe nach die Schlüssel 14, 21, 27, 28, 8, 18, 15, 36, 5 und 2 mit Double Hashing eingefügt werden sollen. Die zu verwendenden Hashfunktionen seien $h(k) = k \bmod 13$ und $h'(k) = 1 + k \bmod 11$. Geben Sie die Belegung der Hashtabelle an, wenn die Schlüssel

1. in der gegebenen Reihenfolge;
2. in sortierter Reihenfolge;
3. in der gegebenen Reihenfolge mit Brents Algorithmus;
4. in sortierter Reihenfolge mit Brents Algorithmus;
5. in der gegebenen Reihenfolge mit Binärbaum-Sondieren;
6. in sortierter Reihenfolge mit Binärbaum-Sondieren;
7. in der gegebenen Reihenfolge mit Ordered Hashing;
8. in sortierter Reihenfolge mit Ordered Hashing

eingefügt werden.

Wieviele Hashtabellenplätze müssen beim Einfügen eines der Schlüssel, bei der erfolgreichen und bei der erfolglosen Suche jeweils höchstens inspiziert werden?

2 Ähnlich Aufgaben

Bei den folgenden Aufgaben handelt es sich um Aufgaben die an der ETH Zürich, am Institut für Theoretische Informatik und an der Universität Freiburg im Institut für Informatik in diversen Vorlesungen gestellt wurden. Inhaltlich sind diese Aufgaben mit dem behandelten Stoff im Buch verwandt. Zu allen Aufgaben gibt es Musterlösungen, die allerdings nur in der Version mit Lösungen enthalten sind.

Aufgabe 2:

Programmieren Sie Brent’s Algorithmus zum effizienten Hashing mit offener Adressierung als Klasse `BrentOpenHashTable extends OpenHashTable`. Implementieren Sie insbesondere die Funktion `public void insert (Object key,Object value)`. Testen Sie Ihr Programm in der Klasse `BrentOpenHash` mit dem *Standart-Array* 19, 53, 72, 5, 12. Nutzen Sie dabei den Rahmen für Hashtabellen aus der Vorlesung (im Netz zu finden).

Aufgabe 3:

In der Vorlesung wurden unterschiedliche Hashverfahren besprochen. Programmieren Sie jeweils eine Klasse für die angegebene Methode und testen Sie ihre Programme mit verschiedenen Eingabefolgen. Nutzen Sie auch hier den Rahmen aus der Vorlesung.

1. Die *multiplikative Methode* wählt eine irrationale Zahl Θ zwischen Null und Eins aus, multipliziert den Schlüssel mit dieser Zahl und entnimmt die Nachkommastellen. Diese werden wiederum mit der Tabellengröße multipliziert und letztendlich der Nachkommabereich weggestrichen. Wählen Sie für Θ den goldenen Schnitt $(\sqrt{5}-1)/2$.

2. *Universelles Hashing* umgeht den worst-case im Hashing, der sich durch eine ungünstige Eingabe immer ergeben kann, mit Randomisierung. In der Vorlesung wurde gezeigt, daß die Klasse $\mathcal{H} = \{((ax+b) \bmod p) \bmod m \text{ mit } 1 \leq a < p \text{ und } 0 \leq b < p\}$ für eine Primzahl p universell ist. Dabei sei m die Größe der Hashtabelle und das Universum aller Schlüssel bestehe o.B.d.A. aus p Elementen.

Aufgabe 4:

Fügen Sie die Schlüsselfolge 16,44,21,5,19,22,8,33,27,30 gemäß der Hashfunktion $h(k) = k \; mod \; m$ für $m = 11$ in eine Hashtabelle mittels offener Adressierung ein. Als Sondierungsfolge nutzen Sie die Funktion $h(k) - s(j,k) \; mod \; m$. Berechnen Sie dabei den Quotient aus der Anzahl von Vergleichen für die Suche der einzelnen Elemente in der endgültigen Tabelle mit der Anzahl der insgesamt gespeicherten Elemente.

1. Verwenden Sie lineares Sondieren $s(j,k) = c \cdot j$ mit $c = 1$.

2. Verwenden Sie die quadratische Sondierungsfolge $s(j,k) = (-1)^j \lceil j/2 \rceil^2$.

3. Verwenden Sie Doppel-Hashing $s(j,k) = h'(k) \; j$. Bestimmen Sie ein geeignetes h' (Die Eignung von h' muß nicht exakt bewiesen werden).

4. Geben sie für den Aufgabenteil (a) und $m = 14$ möglichst allgemein alle c an, die sich als unbrauchbar erweisen. Begründen Sie Ihre Lösung.

Aufgabe 5:

Gegeben sei eine anfangs leere Hashtabelle mit 13 Elementen, in die der Reihe nach die Schlüssel 14, 21, 27, 28, 8, 18, 15, 36, 5, 2 mit Double Hashing eingefügt werden sollen. Die zu verwendenden Hashfunktionen seien $h(k) = k \bmod 13$ und $h'(k) = 1 + k \bmod 11$.

Verwirklichen Sie aufbauend auf die folgende Hashtabellenimplementation

```
public abstract class Hashtable {
  public static final int len=13;        // Laenge der Tabelle
  public int table[] = new int[len];     // Array zum Speichern der
                                         // Schluessel
  public int marke[] = new int[len];     // frei(0) belegt(1) oder
                                         // geloescht(-1)
  public Hashtable()  { for(int i=0;i<len;i++) marke[i]=0; }
  public int h(int k) { return k%13;}    // die erste Hashfunktion
  public int h_(int k){ return 1+k%11; } // die zweite Hashfunktion
  public String toString(){              // ueberschreibt toString()
    String s="";
```

```
    for(int i=0;i<len;i++)
      s+= (marke[i]==1) ? ("["+table[i]+"]  ") : "[]  ";
    return s;
  }
  public abstract void insert(int k);    // abzuleitende Methode
  public abstract boolean search(int k); // abzuleitende Methode
}
```

und die folgende Testklasse

```
public class Hashtest {
  static final int[] ds={14,21,27,28,8,18,15,36,5,2};
  static final int[] sort_ds={2,5,8,14,15,18,21,27,28,36};

  public static void main(String[] argv) {

    DoubleHashtable dht=new DoubleHashtable();
    for(int i=0;i<ds.length;i++) dht.insert(ds[i]);
    System.out.println("Double Hashing in gegebener Reihenfolge:");
    System.out.println(dht);
    System.out.println("27:"+dht.search(27));

    ... (Analog fuer sortierte Reihenfolge)
  }
}
```

die abgeleitete Klasse `DoubleHashtable extends Hashtable`, die die beiden Funktionen `insert` und `search` anbietet. Geben Sie als Ausgabe eine Belegung der Hashtabelle an, wenn die Schlüssel in der

1. gegebenen Reihenfolge
2. in sortierter Reihenfolge

eingefügt werden. Wieviele Hashtabellenplätze müssen beim Einfügen eines der Schlüssel, bei der erfolgreichen und bei der erfolglosen Suche inspiziert werden?

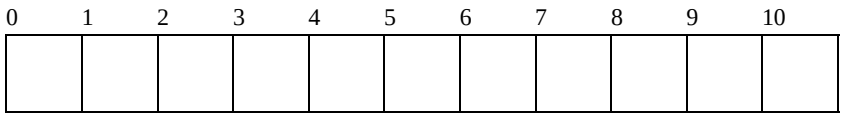
Aufgabe 6:

Gegeben sei eine anfangs leere Hashtabelle mit 11 Feldern, in die der Reihe nach die folgenden Schlüssel eingefügt werden sollen:

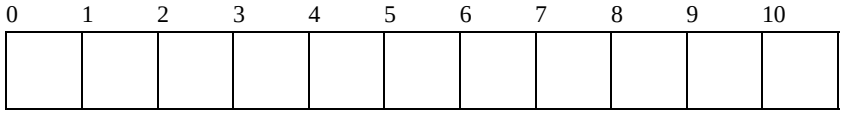
16, 21, 15, 10, 5, 19, 8

Fügen Sie die angegebene Schlüsselmenge mittels der unten angegebenen Hashverfahren ein und geben Sie die Belegung der Tabelle nach der letzten Einfügeoperation an. Verwenden Sie dazu die Hashfunktion: $h(x) = x \ mod \ 11$ und zum Sondieren eine zweite Hashfunktion: $h'(x) = 1 + (x \ mod \ 9)$.

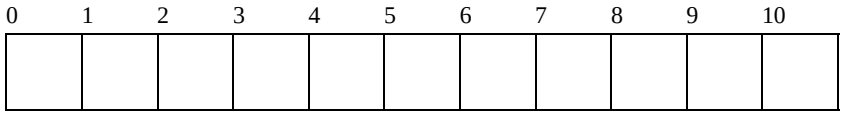
1. Double Hashing



2. Double Hashing in Verbindung mit Brent's Algorithmus



3. Ordered Hashing



Welches der unter b) und c) genannten Methode verbessert gegenüber a) die erfolglose bzw. die erfolgreiche Suche?