Einzelprüfung „Theoretische Informatik / Algorithmen (vertieft)“

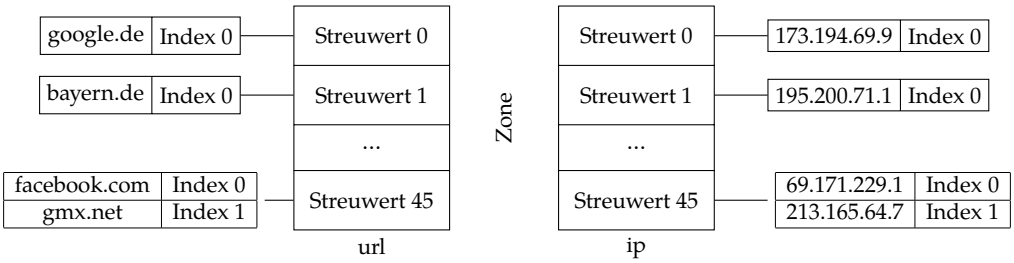
Einzelprüfungsnummer 66115 / 2013 / Frühjahr

Thema 1 / Aufgabe 6

(IP und ULR mit Hashes)

Stichwörter: Streutabellen (Hashing)

Um die URL (zum Beispiel google.de) und die zugehörige IP des Servers (hier 173.194.69.9) zu verwalten, werden Streutabellen verwendet, die eine bestimmte Zone von Adressen abbilden. Die Streutabellen werden als zwei dynamische Arrays (in Java: ArrayLists) realisiert. Kollisionen innerhalb einer Zone werden ebenfalls in dynamischen Arrays verwaltet.



Um zu einer URL die IP zu finden, berechnet man zunächst mittels der Funktion `hash()` den entsprechenden Streuwert, entnimmt dann den Index der Tabelle URL und sucht schließlich an entsprechender Stelle in der Tabelle IP die IP-Adresse.

(a) Erläutern Sie am vorgestellten Beispiel, wie ein Hash-Verfahren zum Speichern großer Datenmengen prinzipiell funktioniert und welche Voraussetzungen und Bedingungen daran geknüpft sind.

(b) Nun implementieren Sie Teile dieser IP- und URL-Verwaltung in einer objektorientierten Sprache Ihrer Wahl. Verwenden Sie dabei die folgende Klasse (die Vorgaben sind in der Sprache Java gehalten):

```
class Zone {
  private ArrayList<ArrayList<String>> urlList =
    new ArrayList<ArrayList<String>>();
  private ArrayList<ArrayList<String>> ipList =
    new ArrayList<ArrayList<String>>();
  public int hash(String url) {/* calculates hash-value h, >=0 */}
}
```

(i) Prüfen Sie in einer Methode `boolean exists(int h)` der Klasse `Zone`, ob bereits mindestens ein Eintrag für einen gegebenen Streuwert vorhanden ist. Falls `h` größer ist als die derzeitige Größe der Streutabelle, existiert der Eintrag nicht.

Lösungsvorschlag

```
boolean exists(int h) {
  if (urlList.size() - 1 < h || ipList.size() - 1 < h)
    return false;

  ArrayList<String> urlCollisionList = urlList.get(h);
  ArrayList<String> ipCollisionList = ipList.get(h);
```

```
    if (urlCollisionList.size() == 0 || ipCollisionList.size() == 0)
      return false;

    return true;
  }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java

(ii) Die Methode `int getIndex (string url, ArrayList<String> urlList)` soll den Index einer URL in der Kollisionsliste berechnen. Ist die URL in der Kollisionsliste nicht vorhanden, soll `-1` zurückgeliefert werden.

Lösungsvorschlag

```
int getIndex(String url, ArrayList<String> urlList) {
  for (int i = 0; i < urlList.size(); i++) {
    if (urlList.get(i).equals(url))
      return i;
  }
  return -1;
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java

(iii) Ergänzen Sie die Klasse Zone um eine Methode `String lookup (String url)`, die in der Streutabelle die IP-Adresse zur `url` zurückgibt. Wird eine nicht vorhandene Adresse abgerufen, wird eine Fehlermeldung zurückgegeben.

Lösungsvorschlag

```
String lookup(String url) {
  int h = hash(url);
  int collisionIndex = getIndex(url, urlList.get(h));
  if (collisionIndex == -1)
    return "Die URL kannte nicht in der Tabelle gefunden werden";
  return ipList.get(h).get(collisionIndex);
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java

Additum

Komplette Java-Datei

```
import java.util.ArrayList;

class Zone {

  private ArrayList<ArrayList<String>> urlList = new ArrayList<ArrayList<String>>();

  private ArrayList<ArrayList<String>> ipList = new ArrayList<ArrayList<String>>();

  /**
   * Der Konstruktor initialisert die Streutabellen mit 46 Buckets, damit wir den
```

```
     * Code testen können.
     */
    public Zone() {
      for (int i = 0; i <= 45; i++) {
        urlList.add(new ArrayList<String>());
        ipList.add(new ArrayList<String>());
      }
    }

    /**
     * Diese Methode wird zum Testen der Methode {@link getIndex} gebraucht.
     *
     * @param hash Der Hashwert, für den eine Kollisionsliste gefunden werden soll.
     *
     * @return Eine Kollisionsliste bestehend aus Strings.
     */
    public ArrayList<String> getUrlCollisionList(int hash) {
      return urlList.get(hash);
    }

    /**
     * Nicht wirklich eine Hashfunktion. Gibt die Werte wie im Schaubild zurück.
     * Diese Methode muss nicht implementiert werden. Sie ist nur dazu da, um die
     * Code testen zu können.
     *
     * @param url Eine URL.
     *
     * @return Ein Hashwert, der größer gleich 0 ist.
     */
    public int hash(String url) {
      /* calculates hash-value h, >=0 */
      switch (url) {
        case "google.de":
          return 0;

        case "bayern.de":
          return 1;

        case "facebook.com":
        case "gmx.net":
          return 45;

        default:
          return 42;
      }
    }

    /**
     * Prüfe, ob bereits mindestens ein Eintrag für einen gegebenen Streuwert
     * vorhanden ist. Falls h größer ist als die derzeitige Größe der Streutabelle,
     * existiert der Eintrag nicht.
     *
     * @param h Der Index-Wert, der durch die Hashfunktion erzeugt wird.
     *
```

```
   * @return Wahr, wenn in beiden Streutabellen an einer bestimmen Index-Position
   *         mindestes ein Wert hinterlegt ist, sonst falsch.
   */
  boolean exists(int h) {
    if (urlList.size() - 1 < h || ipList.size() - 1 < h)
      return false;

    ArrayList<String> urlCollisionList = urlList.get(h);
    ArrayList<String> ipCollisionList = ipList.get(h);
    if (urlCollisionList.size() == 0 || ipCollisionList.size() == 0)
      return false;

    return true;
  }

  /**
   * Berechne den Index einer URL in der Kollisionsliste. Ist die URL in der
   * Kollisionsliste nicht vorhanden, soll -1 zurückgeliefert werden.
   *
   * @param url     Die URL, für die der Index gesucht werden soll.
   * @param urlList Die URL-Kollisionsliste, in der gesucht werden soll.
   *
   * @return Die Index-Nummer der gefundenen URL oder -1, wenn die URL nicht
   *         gefunden wurde.
   */
  int getIndex(String url, ArrayList<String> urlList) {
    for (int i = 0; i < urlList.size(); i++) {
      if (urlList.get(i).equals(url))
        return i;
    }
    return -1;
  }

  /**
   * Gib in der Streutabelle die IP-Adresse zurück. Wird eine nicht vorhandene
   * Adresse abgerufen, wird eine Fehlermeldung zurückgegeben.
   *
   * @param url Die gesuchte URL.
   *
   * @return Die entsprechende IP-Adresse.
   */
  String lookup(String url) {
    int h = hash(url);
    int collisionIndex = getIndex(url, urlList.get(h));
    if (collisionIndex == -1)
      return "Die URL kannte nicht in der Tabelle gefunden werden";
    return ipList.get(h).get(collisionIndex);
  }

  /**
   * Nicht verlangt. Zum Einfügen von Testwerten gedacht.
   *
   * @param url Eine URL.
   * @param ip  Eine IP-Adresse.
```

```
   */
  public void addUrl(String url, String ip) {
    int h = hash(url);
    urlList.get(h).add(url);
    ipList.get(h).add(ip);
  }

  public static void main(String[] args) {
    Zone zone = new Zone();
    zone.addUrl("google.de", "173.194.69.9");
    zone.addUrl("bayern.de", "195.200.71.1");
    zone.addUrl("facebook.com", "69.171.229.1");
    zone.addUrl("gmx.net", "213.165.64.7");
  }
}
```

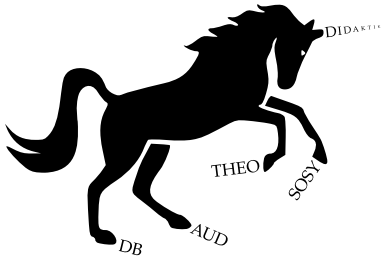
Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java`

Test-Datei

```
import static org.junit.Assert.assertEquals;

import java.util.ArrayList;

import org.junit.Test;
import org.junit.Before;

public class ZoneTest {

  Zone zone;

  @Before
  public void legeZoneAn() {
    zone = new Zone();
    zone.addUrl("google.de", "173.194.69.9");
    zone.addUrl("bayern.de", "195.200.71.1");
    zone.addUrl("facebook.com", "69.171.229.1");
    zone.addUrl("gmx.net", "213.165.64.7");
  }

  @Test
  public void methodExists() {
    assertEquals(true, zone.exists(0));
    assertEquals(true, zone.exists(1));
    assertEquals(false, zone.exists(2));
    assertEquals(true, zone.exists(45));
    assertEquals(false, zone.exists(46));
  }

  @Test
  public void methodGetIndex() {
    ArrayList<String> urlCollisionList = zone.getUrlCollisionList(45);
    assertEquals(0, zone.getIndex("facebook.com", urlCollisionList));
    assertEquals(1, zone.getIndex("gmx.net", urlCollisionList));
```

```java
    assertEquals(-1, zone.getIndex("bschlangaul.org", urlCollisionList));
  }

  @Test
  public void methodLookup() {
    assertEquals("173.194.69.9", zone.lookup("google.de"));
    assertEquals("195.200.71.1", zone.lookup("bayern.de"));
    assertEquals("69.171.229.1", zone.lookup("facebook.com"));
    assertEquals("213.165.64.7", zone.lookup("gmx.net"));
    assertEquals("Die URL kannte nicht in der Tabelle gefunden werden",
↪   zone.lookup("bschlangaul.org"));
  }
}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/ZoneTest.java`



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden: `https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2013/03/Thema-1/Aufgabe-6.tex`