Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
Y. Maus, O. Saukh

13 Mai, 2016

Algorithmen und Datenstrukturen
Sommersemester 2016
Übungsblatt 4

Abgabe bis 12:00, Freitag, 27 Mai, 2016

Achtung: Für dieses Aufgabenblatt haben Sie zwei Wochen Zeit. Bitte fügen Sie Ihrer Lösung eine (kurze) Datei *erfahrungen.txt* mit Ihren Erfahrungen mit dem jeweiligen Aufgabenblatt hinzu. Das Forum können (und sollen) Sie nicht nur für Fragen zur Übung, sondern auch für Fragen zur Vorlesung benutzen.

Aufgabe 1: Mystische Multiplikationen (10 Punkte)

Gegeben sei ein Integer Array A der Länge n und folgender Code:

```
boolean myst(int[] A) {
   int n = A.length;
   for (int i = 0; i < n-1; i++) {
      for (int j = i+1; j < n; j++) {
         for (int k = 0; k < n; k++) {
            if (A[i] * A[j] == A[k]) {
               return true;
            }
         }
      }
   }
   return false;
}
```

(a) Was berechnet die Funktion *myst* (bzw., in welchen Fällen gibt *myst* true zurück)?

(b) Welche (asymptotische) Laufzeit hat *myst(int[] A)* im Worst Case und im Best Case (in Abhängigkeit von n, in Landau-Notation und mit kurzer Begründung)?

(c) Geben Sie, z.B. unter Zuhilfenahme einer geeigneten Datenstruktur, einen Algorithmus mit gleicher Ausgabe wie *myst* an, dessen *asymptotische* Laufzeit im Worst Case jedoch strikt besser als diejenige von *myst* ist. Was ist die Laufzeit Ihres Algorithmus?

Aufgabe 2: Hashtabelle (10 Punkte)

Implementieren Sie eine Datenstruktur *HashTable* und benutzen Sie zur Kollisionsbehandlung die *DoublyLinkedList* aus Aufgabenblatt 3 (Hashtabelle mit Chaining). Dazu soll die Klasse *ListElement* um einen Eintrag *data* erweitert werden. Beim Erstellen einer neuen Hasttabelle soll die fixe Größe der Tabelle als Parameter übergeben werden. Die Wahl einer geeigneten Hashfunktion (mit Hilfe der Vorlesung) ist Ihnen überlassen.

Im public-*Ordner finden Sie vorgefertigte Strukturdateien, die spezifizieren, welche Operationen Ihre Hashtabelle anbieten soll. Sie können die Strukturdateien herunterladen und anpassen, bzw. mit Programmcode füllen.*
Hinweis: Falls Sie die DoublyLinkedList aus Aufgabenblatt 3 nicht implementiert haben, so koennen Sie die Beispiellösung aus dem public-*Ordner im SVN verwenden.*

Aufgabe 3: Zusatzaufgabe (6 Punkte*)

Im *public* Ordner des SVNs finden Sie eine Datei *input.txt*, die 80.000 Integer Zahlen enthält. Erstellen Sie mit Ihrer Lösung zu Aufgabe 2 eine Hashtabelle der Größe 100.000 und fügen Sie die Zahlen aus der Datei *input.txt* mit der jeweiligen Zahl als *key* hinzu. Analysieren Sie danach die Länge der Listen, welche in Ihrer Hashtabelle vorkommen, d.h. sie sollten ein Histogramm (hier ist eine Textdatei ausreichend) erstellen, aus welchem ersichtlich ist, welche Listenlänge wie häufig in Ihrer Hasttabelle vorkommt.
**: Die Punkte in dieser Aufgabe sind Bonuspunkte. Die Aufgabe verschiebt nicht die Grenze, die erreicht werden muss um zur Klausur zugelassen zu werden; jedoch werden Ihnen alle hier erreichten Punkte zum Erreichen dieser Grenze mitgezählt.*