Algorithmen und Datenstrukturen
Übungszettel 7

Martin Avanzini <martin.avanzini@uibk.ac.at>
Thomas Bauereiß <thomas.bauereiss@uibk.ac.at>
Herbert Jordan <herbert@dps.uibk.ac.at>
René Thiemann <rene.thiemann@uibk.ac.at>

10. Mai, zur Besprechung am 17. Mai

Aufgabe 1) Hashfunktionen Betrachten Sie die folgenden beiden Klassen, die Tripel von Daten beliebigen Typs bzw. Listen von Integers repräsentieren:

```
1  public class Triple<X,Y,Z> {
2
3     X x;
4     Y y;
5     Z z;
6
7     public boolean equals(Object other) {
8        if (other instanceof Triple) {
9           Triple<X,Y,Z> t = (Triple<X,Y,Z>)other;
10          return this.x.equals(t.x) && this.y.equals(t.y) && this.z.equals(t
                .z);
11       } else {
12          return false;
13       }
14    }
15
16    public int hashCode() {
17       throw new RuntimeException("zu implementieren!");
18    }
19 }
```

```
1  public class List {
2     Element root;
3
4     public boolean equals(Object other) {
5        if (other instanceof List) {
6           Element one = this.root;
7           Element two = ((List)other).root;
8           while (one != null && two != null) {
9              if (one.x != two.x) {
10                return false;
11             }
12             one = one.next;
13             two = two.next;
```

```
14            }
15            return (one == null && two == null);
16        } else {
17            return false;
18        }
19    }
20
21    public int hashCode() {
22        throw new RuntimeException("zu implementieren!");
23    }
24 }
25
26
27 class Element {
28    int x;
29    Element next;
30
31    Element(int x, Element next) {
32        this.x = x;
33        this.next = next;
34    }
35 }
```

Die `equals`-Methoden definieren dabei Gleichheit so, dass zwei Tripel bzw. Listen genau dann gleich sind, wenn alle ihre Elemente gleich sind und an der gleichen Position stehen. Es werden Hashfunktionen für diese beiden Klassen gesucht.

1. Geben Sie an, ob universelles Hashing für Tripel und/oder Listen als Schlüssel anwendbar ist. Gehen Sie dabei davon aus, dass für die Datentypen der Elemente (d.h. `X`, `Y`, `Z` und `int`) universelles Hashing implementiert werden kann. Begründen Sie Ihre Antwort kurz.

2. Implementieren Sie jeweils die Methode `hashCode` für beide Klassen. Verwenden Sie dabei universelles Hashing, falls möglich, ansonsten eine andere sinnvolle Hashfunktion.

Aufgabe 2) Ketten-Multiplikation von Matrizen In der Vorlesung wurde ein Algorithmus zur Optimierung der Klammerung bei der Ketten-Multiplikation von Matrizen vorgestellt, der das Prinzip der dynamischen Programmierung nutzt (siehe Folien 251–258). Gegeben seien vier Matrizen $A_1 \in \mathbb{R}^{10\times500}$, $A_2 \in \mathbb{R}^{500\times30}$, $A_3 \in \mathbb{R}^{30\times1000}$, $A_4 \in \mathbb{R}^{1000\times50}$. Der Dimensionsvektor ist demnach $\vec{d} = (10, 500, 30, 1000, 50)$.

1. Geben Sie die Tabellen `m[4][4]` und `s[4][4]` an, die sich aus der Ausführung des Algorithmus auf Folie 258 für die oben gegebenen Dimensionen ergeben.

2. Verwenden Sie die Tabellen `m` und `s` aus der vorherigen Teilaufgabe, um eine optimale Klammerung für das Produkt $A_1 \times A_2 \times A_3 \times A_4$ zu konstruieren.

3. Entwerfen Sie einen Algorithmus, der nur auf Grundlage der Tabelle `m`, d.h. ohne die Zusatzinformationen aus `s`, die Klammerung des Produkts konstruiert. Was ist die Komplexität Ihres Verfahrens?