Prof. Dr. Leif Kobbelt
Fabian Emmes, Sven Middelberg,
Michael Kremer

Informatik 8
RWTH Aachen

Datenstrukturen und Algorithmen (SS 2013)

Übungsblatt 1

Abgabe: Montag, **22.04.2013**, 14:00 Uhr

- Die Übungen sollen in Gruppen von zwei bis drei Personen bearbeitet werden.
- Schreiben Sie die Namen jedes Gruppenmitglieds sowie alle Matrikelnummern auf die abgegebenen Lösungen.
- Schreiben Sie die Namen jedes Gruppenmitglieds sowie alle Matrikelnummern auch in die Quellcode-Dateien.
- Geben Sie Ihre Lösungen am **Anfang** der Globalübung, montags, 14:00 Uhr, ab.
- Schicken Sie den jeweiligen Quellcode bitte per **E-Mail** direkt an Ihre/n Tutor/in.
- Geben Sie außerdem den ausgedruckten Quellcode zusammen mit den schriftlichen Lösungen ab.
- Zu spät abgegebene Lösungen werden nicht bewertet.
- Sofern nicht anders gefordert, müssen alle Lösungen und Zwischenschritte kommentiert werden.

Prof. Dr. Leif Kobbelt
Fabian Emmes, Sven Middelberg,
Michael Kremer

Informatik 8
RWTH Aachen

Aufgabe 1 (Fibonacci-Folge [**20 Punkte**])

Im Laufe des Übungsbetriebs, der begleitend zur Vorlesung stattfinden wird, werden Sie verschiedene Algorithmen und Datenstrukturen implementieren. Dafür verwenden wir die Programmiersprache *Java*. In der Regel werden wir Ihnen alle notwendigen Quelldateien zu Beginn der Globalübung zur Verfügung stellen. Dies beinhaltet die Datei `Main.java` sowie Dateien, in der die zum Problem gehörigen Klassen implementiert sind. Diese Klassen werden aus einem Grundgerüst inklusive Methodendeklarationen bestehen, welches dann an den dafür vorgesehenen Stellen durch Ihren Quellcode ergänzt werden soll. Bitte editieren Sie **niemals** die Datei `Main.java`, sondern immer nur die von uns bereitgestellten Problemklassen. Ändern Sie des Weiteren bitte **niemals** von uns bereitgestellte Methodensignaturen (also den Namen oder die Parameter und Rückgabewerte einer Methode). Sie dürfen selbstverständlich immer eigene Variablen und Methoden hinzufügen und benutzen, sofern Sie das von uns vorgegebene Grundgerüst nicht verändern.
Für den Einstieg in die Programmiersprache Java werden Sie in dieser Übung die *Fibonacci-Folge* implementieren. Die Fibonacci-Folge f_i lässt sich durch folgende Gleichung rekursiv definieren:

$$f_i = \begin{cases} 0 & \text{falls } i = 0 \\ 1 & \text{falls } i = 1 \\ f_{i-2} + f_{i-1} & \text{falls } i \geq 2 \end{cases}$$

Wir stellen Ihnen für diese Aufgabe die Klassen `Main.java` und `Fibonacci.java` zur Verfügung. Sofern Sie die Teilaufgaben implementiert haben gibt Ihnen das Programm die Fibonacci-Zahlen f_0 bis f_{40} aus, inklusive der Zeit in Sekunden, welche für die Berechnung der jeweiligen Fibonacci-Zahl benötigt wurde. Für sämtliche Teilaufgaben können Sie davon ausgehen, dass nur valide Werte, also Werte für die die Fibonacci-Folge definiert ist eingegeben werden.

(a) In der Klasse `Fibonacci.java` finden Sie die Methode `fibonacciRecursive`, welche eine Ganzzahl `i` als Parameter erwartet. Ergänzen Sie die Methode so, dass Sie *rekursiv*, also durch Aufrufe von `fibonacciRecursive(i-2)` und `fibonacciRecursive(i-1)`, die `i`-te Fibonacci-Zahl berechnet und zurückgibt. Notieren Sie die Zeiten, welche für die Berechnung der einzelnen Fibonacci-Zahlen benötigt wurde. [**7 Punkte**]

(b) Neben der rekursiven Berechnung, kann die `i`-te Fibonacci-Zahl auch *iterativ* ermittelt werden, indem bei $j = 2$ angefangen die Werte der Fibonacci-Zahlen f_{j-2} und f_{j-1} in Variablen zwischengespeichert und in jeder Iteration aktualisiert werden, solange $j < i$ ist.

Implementieren Sie die Berechnung der `i`-ten Fibonacci-Zahl iterativ, also ohne rekursive Methodenaufrufe, in der Methode `fibonacciIterative`. Notieren Sie erneut die Berechnungszeiten. [**7 Punkte**]

(c) Vergleichen Sie die gemessenen Zeiten aus Aufgabe (a) mit denen aus Aufgabe (b). Interpretieren Sie die Resultate. [**6 Punkte**]

Bitte geben Sie nur die von Ihnen bearbeitete `Fibonacci.java` ab.

Lösungsvorschlag

(a) Siehe `Fibonacci.java`

(b) Siehe `Fibonacci.java`

(c) Tabelle 1 zeigt die Zeiten der rekursiven und iterativen Methode gemittelt über 100 Berechnungen. Offensichtlich steigen die Zeiten für die rekursive Methode schnell an, während sie für die iterative Methode nahezu konstant bleiben.

Dies lässt sich durch die Anzahl der Fibonacci-Zahlen, welche insgesamt für die Berechnung der Fibonacci-Zahl f_i ermittelt werden, erklären. Bei der iterativen Methode wird jede Fibonacci-Zahl f_j, $0 \leq j \leq i$, genau einmal berechnet. Also ist die Anzahl der Fibonacci-Zahlen, welche für die Berechnung der Fibonacci-Zahl f_i ermittelt werden, $N_{it}(i) = i + 1$.

Bei der rekursiven Methode ist dies etwas komplizierter. Die Zahl N_{rek} lässt sich rekursiv definieren:

$$N_{rek}(i) = \begin{cases} 1 & \text{falls } i = 0 \\ 1 & \text{falls } i = 1 \\ N_{rek}(i-2) + N_{rek}(i-1) + 1 & \text{falls } i \geq 2 \end{cases} \tag{1}$$

Beispielsweise nimmt N_{rek} für die ersten 10 Fibonacci-Zahlen die folgenden Werte an:

i	f_i	$N_{rek}(i)$
0	0	1
1	1	1
2	1	3
3	2	5
4	3	9
5	5	15
6	8	25
7	13	41
8	21	67
9	34	109

Genaues hinsehen lässt den folgenden Zusammenhang vermuten:

$$N_{rek}(i) = 2 \cdot f_{i+1} - 1 \ . \tag{2}$$

Diese Vermutung lässt sich leicht durch vollständige Induktion beweisen:

Induktionsanfang:
$$N_{rek}(0) = 1 = 2 \cdot f_1 - 1$$

Induktionsschluss:
$$\begin{aligned} N_{rek}(i+1) &\overset{(1)}{=} N_{rek}(i-1) + N_{rek}(i) + 1 \\ &\overset{\text{IV}}{=} 2 \cdot f_i - 1 + 2 \cdot f_{i+1} - 1 + 1 \\ &= 2 \cdot (f_i + f_{i+1}) - 1 \\ &= 2 \cdot f_{i+2} - 1 \end{aligned}$$

□

Dieser Zusammenhang spiegelt sich auch grafisch wider, wenn man die Plots von f_i und die entsprechenden Berechnungszeiten der rekursiven Methode für $0 \leq i \leq 40$ miteinander vergleicht:

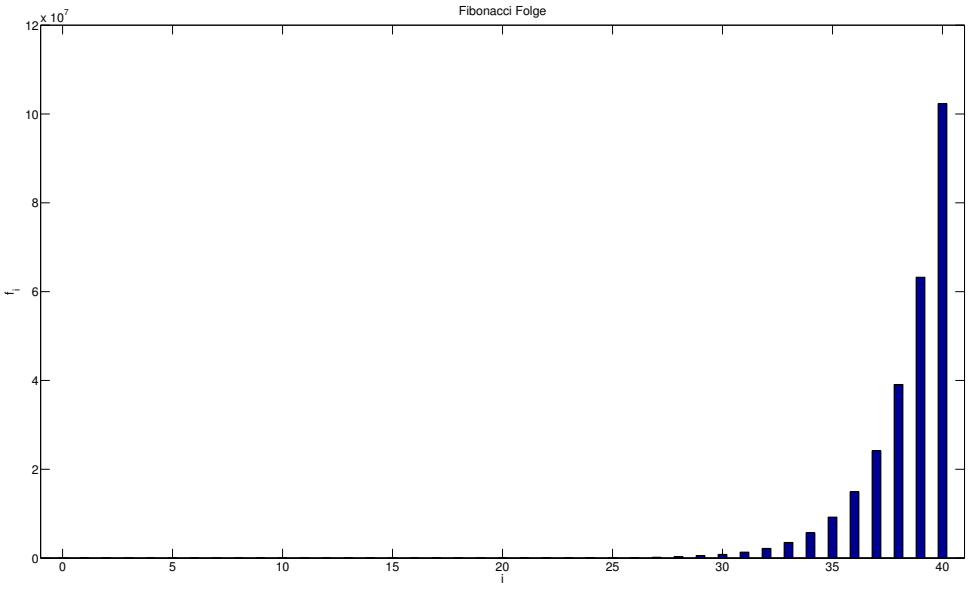

Figure 1: f_i for $0 \leq i \leq 40$.

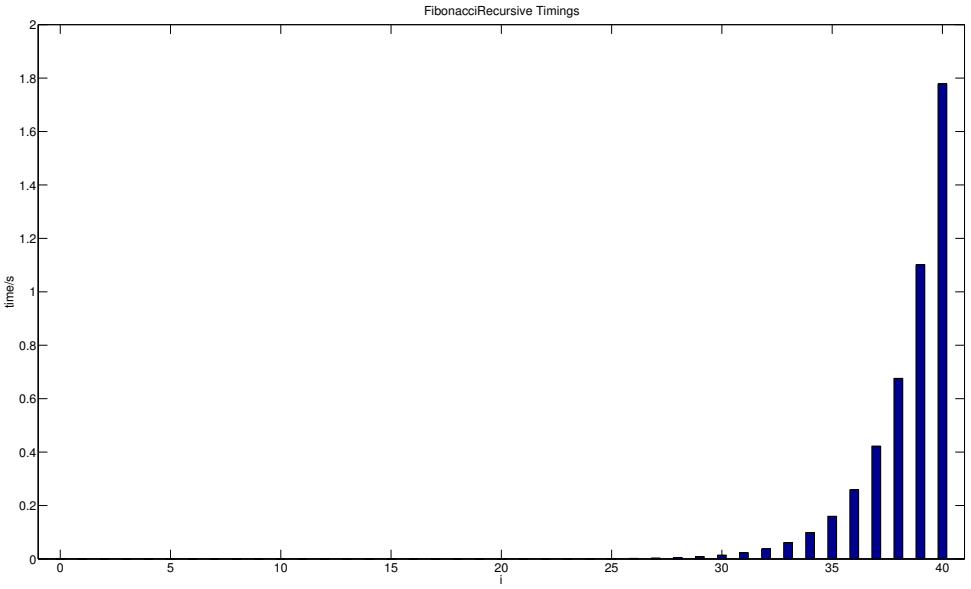

Figure 2: Durchschnittliche Berechnungszeit für f_i, $0 \leq i \leq 40$.

`i`	time(`fibonacciRecursive(i)`)	time(`fibonacciRecursive(i)`)
0	0.00000038	0.00000052
1	0.00000034	0.00000087
2	0.00000153	0.00000037
3	0.00000110	0.00000037
4	0.00000174	0.00000041
5	0.00000288	0.00000046
6	0.00000452	0.00000047
7	0.00000772	0.00000048
8	0.00001370	0.00000171
9	0.00002069	0.00000073
10	0.00000215	0.00000115
11	0.00000172	0.00000076
12	0.00000250	0.00000081
13	0.00000449	0.00000081
14	0.00000716	0.00000083
15	0.00001087	0.00000108
16	0.00001712	0.00000095
17	0.00002849	0.00000102
18	0.00004708	0.00000102
19	0.00007403	0.00000156
20	0.00011960	0.00000157
21	0.00020700	0.00000167
22	0.00036167	0.00000175
23	0.00052614	0.00000169
24	0.00088942	0.00000019
25	0.00133558	0.00000027
26	0.00214821	0.00000023
27	0.00351774	0.00000023
28	0.00555246	0.00000024
29	0.00899743	0.00000023
30	0.01437606	0.00000025
31	0.02344586	0.00000022
32	0.03783847	0.00000023
33	0.06116425	0.00000024
34	0.09915604	0.00000024
35	0.15973625	0.00000023
36	0.25901642	0.00000023
37	0.42260498	0.00000025
38	0.67593269	0.00000020
39	1.10114408	0.00000024
40	1.77799733	0.00000025

Table 1: Berechnungszeiten (in Sekunden) der ersten 41 Fibonacci-Zahlen für die rekursive und iterative Methode