Prof. Dr. Gerd Stumme, Folke Eisterlehner, Dominik Benz
Fachgebiet Wissensverarbeitung

27.4.2009

2. Hausübung „Algorithmen und Datenstrukturen“

Sommersemester 2009

Abgabetermin: Montag, 04.05.2009, 10:00 Uhr

1 Theorie

1.1 Berechenbarkeit.

Entscheiden Sie, ob sich die im folgenden beschriebenen Aufgabenstellungen algorithmisch lösen lassen.

Funktionale Sortierung: Für die Programmierumgebung Eclipse soll ein Plugin geschrieben werden, welches alle lokal verfügbaren Java-Quelldateien durchsucht und nach Funktionalität sortiert in einer Baumstruktur auflistet. So sollen z.B. unter dem Knoten „*Sortieren*“ alle Quelldateien aufgelistet werden, in denen ein Sortierverfahren implementiert ist.

Programmverifikation: Ein weiteres Plugin für Eclipse soll in großen Projekten dabei helfen, fehlerhafte Programmteile zu identifizieren. Hierzu sollen alle Programmschleifen (`for`,`while`, etc.) untersucht werden, ob sie bei beliebigen Eingabeparametern gesichert terminieren.

Nicht terminierende Schleifen sollen farbig (rot) hervorgehoben werden.

Programmverifikation mit Restriktionen: Für einige sicherheitskritische Anwendungen mit fest definiertem Einsatzgebiet lassen sich in der Phase der Programmspezifikation alle mögliche Eingabeparameter genau festlegen und auf einen endlichen Bereich einschränken. Für solche Anwendungen sollen alle innerhalb eines Projekts programmierten Funktionen überprüft werden, ob sie auf den spezifizierten Parametern nach einer vorgegebenen Anzahl von Schritten die gewünschte Ausgabe liefern.

(6 Punkte)

1.2 Komplexität.

Fingerübungen zur O-Notation: Gegeben sind folgende Funktionen $f_i \colon \mathbb{N} \to \mathbb{N}$, $i = 1, \ldots, 5$:

$$f_1(n) \coloneqq \tfrac{2}{3}n \qquad f_2(n) \coloneqq \tfrac{1}{3}n^3 + 10n^2 - n \qquad f_3(n) \coloneqq n^3 \qquad f_4(n) \coloneqq 10^{42} \cdot n^2 \qquad f_5(n) \coloneqq a^n$$

Untersuchen Sie, ob $f_i \in \mathrm{O}(f_{i+1})$ für $i = 1, \ldots, 4$ gilt. Begründen Sie Ihre Behauptung. Dabei können Sie folgende Eigenschaften verwenden:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty \implies f \in \mathrm{O}(g)$$
$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \implies f \notin \mathrm{O}(g)$$

(8 Punkte)

Ternäre Suche: Überlegen Sie sich, wie der Algorithmus **BinarySearch** aus der Vorlesung abgewandelt werden muss, so dass in jeder Runde das Eingabefeld in drei gleichgroße Abschnitte aufgeteilt und die Suche im passenden Drittel fortgesetzt wird (falls der Schlüssel noch nicht gefunden ist und der Abschnitt mehr als ein Element enthält).

Diskutieren Sie die Laufzeit analog zur Vorlesung und entscheiden Sie, ob und gegebenenfalls wann dieser Algorithmus der ursprünglichen Variante **BinarySearch** vorzuziehen ist und wann nicht. (6 Punkte)

Matrix-Rechnung: Betrachten Sie folgenden Ausschnitt eines Java-Programms und beantworten Sie die anschließenden Fragen:

algorithm Matrix (U, e) $\to M$
Eingabe: $n \times n$-Matrix U über $\mathbb{Z}$, natürliche Zahl e

```
long V[][] = new long[n][n];
long W[][] = new long[n][n];
copyMatrix(V, U);

for( int h=0; h<e; h++ ) {
   for( int i=0; i<n; i++ ) {
      for( int j=0; j<n; j++) {
         W[i][j] = 0;
         for( int k=0; k<n; k++ )
            W[i][j] += V[i][k]*U[k][j];
      }
   };
   copyMatrix(V,W);
}
return V;
```

Dabei erzeugt **copyMatrix(Target, Source)** eine Kopie des Arrays **Source** in das Array **Target**.

1. Was berechnet der angegebene Algorithmus bei Eingabe einer Matrix U und natürlichen Zahl e?

2. Wieviele arithmetische Operationen (Additionen und Multiplikationen) braucht der Algorithmus zur Berechnung bei Eingabe einer $n \times n$-Matrix U und natürlichen Zahl e? Geben Sie Ihr Ergebnis als Funktion von n und e an.

(6 Punkte)

2 Programmieren

Schreiben Sie ein Java-Programm zur Primfaktorzerlegung einer natürlichen Zahl. Das heißt, zu gegebener Zahl n sollen alle Primzahlen p bestimmt werden, welche n ohne Rest teilen.
Die Ausgabe Ihres Programms bei Eingabe der Zahl 2600 soll wie folgt aussehen (es gilt $2600 = 2^3 * 5^2 * 13$):

```
Bitte geben Sie eine natürliche Zahl n ein: 2600
Primteiler von 2600 sind:
   2
   5
   13
```

Begründen Sie kurz die Korrektheit ihres Programms und geben Sie *obere Schranken* für folgende Größen in Abhängigkeit von n an:

a) die Anzahl von Divisionen und Modulo-Operationen

b) die Anzahl von Multiplikationen

Zum Beispiel ist $t(n) := 2n^2$ eine obere Schranke für $\frac{2}{3}n^2+\frac{4}{2}n$, da für alle natürlichen Zahlen n gilt:

$$\begin{aligned} 2n^2 &= n^2 + n^2 \\ &\geq \frac{2}{3}n^2 + \frac{4}{2}n^2 \\ &\geq \frac{2}{3}n^2 + \frac{4}{2}n \end{aligned}$$

(10 Punkte)

Viel Spaß und viel Erfolg!