AuD – Algorithmen und Datenstrukturen

Aufgabenblatt 3: Divide & Conquer Verfahren und Laufzeitanalyse

Abgabe bis 9. November, 1600 Uhr – Besprechung am 11.-13. November 2015

von
3

Übungsaufgabe 3.1: Gegeben seien die folgenden Code-Fragmente. Geben Sie eine möglichst dichte asymptotische obere Schranke für die Laufzeit der einzelnen Code-Fragmente jeweils in Abhängigkeit von n an. Begründen Sie Ihre Behauptung. (Es geht hier nicht um die Bedeutung des Codes, nur um die Laufzeit. An der Stelle von $sum = sum + j$ könnte sinnvoller(er) Code stehen. Die Einrückung gibt den Skopus der Schleifenkonstrukte an.)

```
ALGO1()
1  for i = 0 to n
2      for j = n downto 1
3          sum = sum + j
4      for j = 1 to n
5          sum = sum + j
```

```
ALGO2()
1  i = 1
2  while i < 2 · n
3      for j = 1 to i
4          sum = sum + j
5      i = i + 2
```

```
ALGO3()
1  i = 1
2  while i · i < n
3      i = i + 1
4      j = n
5      while j > 1
6          sum = sum + j
7          j = j/2
```

von
2

Übungsaufgabe 3.2: Gegeben sei folgender Algorithmus, der ein Array A als Eingabe erwartet (dessen genaue Funktionsweise nachfolgend aber nicht wichtig ist):

```
FUNC(A)
 1  if A.länge < 4
 2      return 5
 3  else
 4      sum = 0
 5      for i = 1 to A.länge
 6          sum = sum + A[i]
 7      x = A.länge/4
 8      y = FUNC(A[x + 1 .. 2 · x])
 9      z = FUNC(A[3 · x + 1 .. A.länge])
10      r = y
11      for i = 1 to A.länge
12          r = r + z · A[i]
13      return sum + r
```

Leiten Sie eine Rekurrenzgleichung für die Laufzeit der Methode FUNC in Abhängigkeit von der Arraygröße n von A ab. Begründen Sie Ihre Gleichung.

Übungsaufgabe 3.3: Gegeben seien die folgenden Rekurrenzgleichungen T_1, T_2 und T_3, wobei die c_i und d_i Konstanten seien.

von
3

$$T_1(n) := \begin{cases} c_1, & \text{für } n = 1 \\ 8 \cdot T_1(\frac{n}{2}) + d_1 \cdot n^3, & \text{sonst} \end{cases}$$

$$T_2(n) := \begin{cases} c_2, & \text{für } n = 1 \\ 5 \cdot T_2(\frac{n}{4}) + d_2 \cdot n^2, & \text{sonst} \end{cases}$$

$$T_3(n) := \begin{cases} c_3, & \text{für } n = 1 \\ 6 \cdot T_3(\frac{n}{3}) + d_3 \cdot n \cdot \log n, & \text{sonst} \end{cases}$$

Bestimmen Sie die Größenordnung der Funktionen $T_i : \mathbb{N} \to \mathbb{N}$ mittels des Mastertheorems. Ist dies nicht möglich, geben Sie an warum. Geben Sie Ihre Ergebnisse nachvollziehbar an.

Übungsaufgabe 3.4:

von
8

1. Geben Sie für die Funktion *merge* in *mergesort* (siehe Folie 9, Kapitel 3) sinnvollen Pseudocode an. Orientieren Sie sich dabei an der natürlichsprachlichen Formulierung auf Folie 10, Kapitel 3. Ihr Pseudocode sollte diese Formulierung möglichst nah umsetzen. Weisen Sie dann Ihren Pseudocode von *merge* als korrekt nach, indem Sie eine sinnvolle Schleifeninvariante formulieren und nutzen. Nutzen Sie dies dann, um auch *mergesort* selbst als korrekt nachzuweisen. (Siehe auch Folie 11, Kapitel 3.)

2. Gegeben sei eine Sequenz $A = \langle a_1, a_2, \ldots, a_n \rangle$ von Zahlen (als Array). Ein Paar (i, j) mit $i < j$, aber $a_i > a_j$ nennen wir einen *Konflikt* (in A). Gesucht ist nun, gegeben das Array A, die Anzahl der Konflikte in A. Ihre Aufgabe ist es, einen Divide & Conquer Algorithmus zu entwerfen, der das Problem in $O(n \cdot \log n)$ löst. Beschreiben Sie zunächst Ihre Idee. Geben Sie dann den Algorithmus konkreter an und begründen Sie dann überzeugend, warum Ihr Algorithmus das Problem korrekt löst (sie müssen dazu nicht umbedingt auf Schleifeninvarianten zurückgreifen). Zuletzt begründen Sie auch, warum die Laufzeit Ihres Algorithmus tatsächlich in $O(n \cdot \log n)$ liegt.

Informationen und Unterlagen zur Veranstaltung unter:

`http://www.informatik.uni-hamburg.de/TGI/lehre/vl/WS1516/AuD`