technische universität dortmund

Fakultät für Informatik
Lehrstuhl 11 / Algorithm Engineering

Petra Mutzel
Nicola Beume, Christian Bockermann, Christian Horoba,
Ingo Schulz, Dirk Sudholt, Christine Zarges

Sommersemester 2009

DAP2 Übung – Blatt 1

Ausgabe: 16. April, **Abgabe:** 23. April, 14:00 Uhr, **Block:** A

Aufgabe 1.1 (4 Punkte)

Beschreibe das asymptotische Wachstum folgender Funktion in der Θ-Notation und beweise deine Aussage:

$$f(n) := \begin{cases} 3n^5 & \text{falls } n < 1000 \\ 2n^3 & \text{falls } n \geq 1000. \end{cases}$$

Aufgabe 1.2 (4 Punkte)

Zähle für jede der unten aufgeführten Beziehungen *alle* Zeichen aus o, O, Θ, Ω und ω auf, durch die man $\star$ ersetzen kann um eine wahre Aussage zu erhalten. Hier braucht ihr die Aussagen nicht zu beweisen, allerdings sollen für jeden Fall *alle* zulässigen Ersetzungen für $\star$ angegeben werden, also *nicht* nur eine zulässige oder die genaueste Charakterisierung der linken Seite.

1. $3n^2 = \star(5n + (\log n)^3)$
2. $17n = \star\left(\frac{11n^2-8n+15}{5n-3}\right)$
3. $n \log n = \star\left(\frac{n(n-1)}{4}\right)$
4. $n^{\log n} = \star(2^n)$

Aufgabe 1.3 (4 Punkte)

In einer Eingabefolge in einem Feld A (d.h. $A[1], A[2], \ldots, A[n]$) können Elemente mehrfach vorkommen. Wir wollen die Anzahl der *verschiedenen* Elemente ermitteln. Für die Eingabefolge $\langle 7, 5, 2, 6, 1, 2, 7, 42, 3, 5, 6\rangle$ sollte zum Beispiel 7 ausgegeben werden. Gib ein Pseudocode-Programm an, das dieses Problem löst, und erkläre dein Programm. Da du aus der Vorlesung bereits den Algorithmus *Insertion-Sort* kennst, darfst du diesen Algorithmus natürlich als Unterprogramm aufrufen, dein „Befehlssatz“ ist also um den Aufruf einer Sortierfunktion erweitert.

Präsenzaufgabe 1.4

Eingaben des folgenden Algorithmus sind $n \times n$-Matrizen A und B, auf deren Einträge in der i-ten Zeile und j-ten Spalte mittels $A[i, j]$ bzw. $B[i, j]$ zugegriffen werden kann. Ausgabe ist eine $n \times n$-Matrix C. Alle Matrixeinträge sind Zahlen.

```
1: function WasBinIch(A, B)
2:     for i:=1,...,n do
3:         for j := 1, ..., n do
4:             C[i, j] := 0
5:             for k := 1, ..., n do
6:                 C[i, j] := C[i, j] + A[i, k] · B[k, j]
7:             end for
8:         end for
9:     end for
10:    return C
11: end function
```

Was berechnet dieser Algorithmus? Was ist die Laufzeitfunktion (Anzahl der primitiven Operationen als Funktion von n) dieses Algorithmus?

Was ist die asymptotische Worst-Case-Laufzeit des Algorithmus in Θ-Notation? Begründe deine Antwort.