

Theoretische Informatik III

1. Serie

Abgabe bis 13:00 Uhr am 4. Mai

Aufgabe 1

[4 Punkte]

Seien $f, g: \mathbb{N} \rightarrow \mathbb{N}$ Funktionen. Beweisen oder widerlegen Sie jede der folgenden Vermutungen.

- (a) $2^{n+1001} = O(2^n)$.
- (b) $2^{2^n} = O(2^n)$.
- (c) $f(n) + g(n) = O(\max\{f(n), g(n)\})$.
- (d) $f(n) + g(n) = \Theta(\min\{f(n), g(n)\})$.
- (e) $f(n) + g(n) = \Theta(\max\{f(n), g(n)\})$.
- (f) $\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$.
- (g) Wenn $\log(f(n)) = \Theta(\log(g(n)))$, dann gilt $f(n) = \Theta(g(n))$.
- (h) Wenn $f(n) = O(g(n))$, dann gilt $2^{f(n)} = O(2^{g(n)})$.

Aufgabe 2

[4 Punkte]

Betrachten Sie das spezielle Suchproblem:

Eingabe: aufsteigend sortierte Liste natürlicher Zahlen a_1, \dots, a_n und ein Wert $v \in \mathbb{N}$

Ausgabe: Index $i \in [n]$ mit $v = a_i$, falls v in a_1, \dots, a_n vorkommt, sonst 0

Die *binäre Suche* ist ein Algorithmus, der das mittlere Element der Liste mit v vergleicht und damit die Hälfte der Sequenz für die weitere Betrachtung ausschließt und diese Prozedur wiederholt.

- (a) Schreiben Sie ein rekursives Programm in Pseudocode für die binäre Suche und beweisen Sie die Korrektheit des Programms.
- (b) Geben Sie eine Rekursionsgleichung für die Laufzeit der binären Suche im schlechtesten Fall an und schätzen Sie diese asymptotisch ab.

Aufgabe 3

[4 Punkte]

Geben Sie (möglichst scharfe) asymptotische obere Schranken für die folgenden Rekursionsgleichungen an. Gehen Sie davon aus, dass $T(n)$ für hinreichend kleine n konstant ist.

- (a) $T(n) = T(n-1) + n$.
- (b) $T(n) = 7T(n/3) + n^2$.
- (c) $T(n) = T(\alpha n) + n$ mit $0 < \alpha < 1$.
- (d) $T(n) = 4T(n/2) + n \log n$

Aufgabe 4

[4 Punkte]

Sortieren durch Vertauschen ist ein Sortieralgorithmus, der wiederholt benachbarte Elemente vertauscht, die sich nicht in der richtigen Reihenfolge befinden.

- (1) Für $i = 1$ bis n :
- (2) Für $j = 1$ bis $n - i$:
- (3) Falls $a_j > a_{j+1}$:
- (4) dann Vertausche a_j und a_{j+1} .
- (5) Gib a_1, \dots, a_n aus.

- (a) Geben Sie zuerst eine Schleifeninvariante für die for-Schleife in den Zeilen (2-4) an und beweisen Sie, dass diese Schleifeninvariante erhalten bleibt.
- (b) Geben Sie unter Verwendung der Abbruchbedingung für die in (a) bewiesene Schleifeninvariante eine Schleifeninvariante für die for-Schleife in den Zeilen (1-4) an, die es Ihnen erlaubt, die Korrektheit des Algorithmus' zu beweisen.
- (c) Schätzen Sie die Laufzeit des Algorithmus' ab und geben Sie allgemein (für jedes $n \in \mathbb{N}$) eine Eingabesequenz a_1, \dots, a_n an, für die (größenordnungsmäßig) die angegebene Laufzeit benötigt wird.