Algorithmen und Datenstrukturen
WS 2025
von Luxburg/Bordt

Präsenzübung 1

Besprechung: 20.10.– 25.10.25

Aufgabe 1: Landau-Notation (mündlich, keine Punkte)

Bestimmen Sie für alle folgenden Beispiele, welcher der drei folgenden Fälle vorliegt: $f \in o(g)$, oder $f \in \omega(g)$, oder $f \in \Theta(g)$.

	$f(n)$	$g(n)$		$f(n)$	$g(n)$
(a)	$n/2$	$4n + 250$	(e)	$n^{1.01}$	$n \log(n)^5$
(b)	$10n^2 + 8n + 100$	n^3	(f)	2^n	2^{n+1}
(c)	$10 \cdot \log(n)$	$\log(n^2)$	(g)	$n!$	2^n
(d)	n	$n \log(n)$	(h)	$(\log_2 n)^{\log_2 n}$	$2^{(\log_2 n)^2}$

Aufgabe 2: Induktion (mündlich, keine Punkte)

Folgender rekursiver Algorithmus wird bei Eingabe $n \geq 1$ insgesamt $C(n)$ Mal aufgerufen (d.h. Zeile 1 ausgeführt).

1: **function** StupidAlg(n)
2:     **if** $n = 1$ **then**
3:         **return** 1
4:     **else**
5:         **return** StupidAlg($n-1$) $\cdot$ StupidAlg($n-1$)
6:     **end if**
7: **end function**

(a) Was berechnet der Algorithmus?

(b) Ermitteln Sie $C(n)$. Beweisen Sie durch vollständige Induktion.

Aufgabe 3: Laufzeitanalyse (mündlich, keine Punkte)

Die folgende Funktion Fun erhält als Eingabe ein Array $X = [x_1, x_2, \ldots, x_n]$ der Länge n. Fun ruft sich rekursiv mit Teilarrays auf, wobei z.B. $X[5 \ldots 42]$ das Teilarray der Indizes $5 \ldots 42$ bezeichnet.

function Fun(X)
    $\ell \leftarrow \mathit{length}(X)$
    if $\ell \leq 1$ **then**
        Print("Hallo")
        return
    end if
    $p \leftarrow \lceil \ell/3 \rceil$
    Fun($X[1 \ldots p]$)
    Fun($X[(\ell - p + 1) \ldots \ell]$)
    $x \leftarrow 0$
    while $x < \ell$ **do**
        $x \leftarrow x + \sqrt{\ell}$
    end while
end function

a) Bestimmen Sie eine Rekurrenzgleichung für die Laufzeit von Fun in Abhängigkeit von n (in der Form, wie wir sie für das Master-Theorem benötigen).

b) Nutzen Sie das Master-Theorem, um die resultierende asymptotische Laufzeit zu ermitteln.

c) Nehmen Sie an, Fun wird mit einem Array der Länge n aufgerufen, wobei n eine 3er-Potenz ist. Wie oft wird "Hallo" ausgegeben?

Aufgabe 4: Suchen in einem Array (mündlich, keine Punkte)

Sei A ein sortiertes Array der Größe n und sei z eine gegebene Zahl. Das Ziel dieser Übung ist es, folgende Frage zu beantworten: Gibt es in A zwei verschiedene Elemente x und y, so dass $x+y = z$?

(a) Finden Sie einen (naiven) Algorithmus, der diese Frage in quadratischer Zeit $\mathcal{O}(n^2)$ beantwortet. Geben Sie den Pseudocode für einen solchen Algorithmus an und erklären Sie, warum Ihr Algorithmus die Laufzeit $\mathcal{O}(n^2)$ hat.

(b) Es ist klar, dass ein Algorithmus für die obige Frage *mindestens* die Laufzeit $\Omega(n)$ benötigt. Versuchen Sie, einen Algorithmus zu finden, der obige Frage in einer Laufzeit von tatsächlich $\mathcal{O}(n)$ beantwortet. Geben Sie den Pseudocode an und begründen Sie die Laufzeit und die Korrektheit Ihres Algorithmus. Tipp: Nennen Sie die Variablen x und y in *links* und *rechts* um - und verwenden Sie die Variablen entsprechend. Für den Korrektheitsbeweis machen Sie sich eine Invariante zunutze: In einem beliebigen Schritt im Algorithmus, was muss sicher gestellt sein, damit der Algorithmus korrekt ist.