Grundzüge der Informatik 1

Vorlesung 4 - flipped classroom

Wiederholung

Grundideen (asymptotische Analyse)

- Ignoriere konstante Faktoren
- Betrachte das Verhältnis von Laufzeiten für $n \to \infty$
- Klassifizieren Laufzeiten durch Angabe von „einfachen Vergleichsfunktionen“

O-Notation

- Lässt Konstanten und Terme niederer Ordnung weg
- Beschreibt Menge von Funktionen, die höchstens genau so schnell wachsen wie die Vergleichsfunktion
- Kann genutzt werden, um obere Schranken für die Laufzeit anzugeben

Landau Notation

Definition 4.1 (O-Notation)

- $O(g(n)) = \{f(n)$: Es gibt positive Konstanten c und n_0, so dass für alle $n \geq n_0$ gilt: $0 \leq f(n) \leq c \cdot g(n)\}$

Landau Notation

Aufgabe 1

- Die Laufzeit von InsertionSort war $T(n)= (3n^2+7n-8)/2$. Was ist eine möglichst gute Abschätzung in der O-Notation?
- Welche der folgenden Aussagen sind wahr?
- $n^3 \in O(n^2)$
- $\log^5 n \in O(\sqrt{n})$
- Finden Sie eine möglichst gute Abschätzung in der O-Notation für:
- $130n + 2 n^2 + 15 n \log n + 12 \log n$
- $150n^{10} + 20 \log n + 21 n^{10} \log n$

Wiederholung

Ω-Notation

- Lässt Konstanten und Terme niederer Ordnung weg
- Beschreibt Menge von Funktionen, die mindestens genau so schnell wachsen wie die Vergleichsfunktion
- Kann genutzt werden, um untere Schranken für die Laufzeit anzugeben

Landau Notation

Definition 4.2 (Ω-Notation)

- $\Omega(g(n)) = \{f(n)$: Es gibt positive Konstanten c und n_0, so dass für alle $n \geq n_0$ gilt: $0 \leq c \cdot g(n) \leq f(n)\}$

Landau Notation

Definition 4.3 (Θ-Notation)

- $f(n) \in \Theta(g(n)) \Leftrightarrow f(n) \in O(g(n))$ und $f(n) \in \Omega(g(n))$

Interpretation

- $f(n) \in \Theta(g(n))$ bedeutet, dass f(n) genauso stark wächst wie g(n) für $n \to \infty$
- Dabei ignorieren wir beim Wachstum konstante Faktoren
- Die Θ-Notation liefert eine obere und untere Schranke
- Die Funktionen f(n) und g(n) müssen asymptotisch nicht-negativ sein (für n groß genug, sind die Funktionen nicht-negativ)

Landau Notation

Definition 4.4 (o-Notation)

- $o(g(n)) = \{f(n)$: Für jede Konstante $c>0$ gibt es eine Konstante $n_0>0$, so dass für alle $n \geq n_0$ gilt: $0 \leq f(n) \leq c \cdot g(n)\}$

Interpretation

- $f(n) \in o(g(n))$ bedeutet, dass $f(n)$ weniger stark wächst als $g(n)$ für $n \to \infty$
- Dabei ignorieren wir beim Wachstum konstante Faktoren
- Die Funktionen $f(n)$ und $g(n)$ müssen asymptotisch nicht-negativ sein (für n groß genug, sind die Funktionen nicht-negativ)

Landau Notation

Definition 4.5 (ω-Notation)

- $f(n) \in \omega(g(n)) \Leftrightarrow g(n) \in o(f(n))$

Interpretation

- $f(n) \in \omega(g(n))$ bedeutet, dass f(n) echt stärker wächst als g(n) für $n \to \infty$
- Dabei ignorieren wir beim Wachstum konstante Faktoren
- Die Funktionen f(n) und g(n) müssen asymptotisch nicht-negativ sein (für n groß genug, sind die Funktionen nicht-negativ)

Landau Notation

Schreibweise

- Es ist üblich $f(n)=O(n)$ zu schreiben anstelle von $f(n)\in O(n)$
- Dies gilt auch für Ω-, Θ-, o- und ω-Notation.

Landau Notation

Aufgabe 2: Welche Aussagen sind korrekt?

- $n^2 = O(n^3)$
- $n = \Omega(n)$
- $n = \omega(n)$
- $n^2 = \Omega(n)$
- $100n^3+n^2 = \Theta(n^2)$
- $100n^3+n^2 = \Theta(n^3)$

Korrektheitsbeweise

Korrektheitsbeweis

- Formale Argumentation, dass ein Algorithmus korrekt arbeitet

Problembeschreibung

- Definiert für eine Menge von zulässigen Eingaben die zugehörigen gewünschten Ausgaben

Korrektheit

- Wir bezeichnen einen Algorithmus für eine vorgegebene Problembeschreibung als korrekt, wenn er für jede zulässige Eingabe die in der Problembeschreibung spezifizierte Ausgabe berechnet
- Streng genommen kann man also nur von Korrektheit sprechen, wenn vorher festgelegt wurde, was der Algorithmus eigentlich tun soll

Korrektheitsbeweise

EinfacherAlgorithmus(n)

1. X=10
2. Y=X
3. X=X*Y
4. **return** X

Aufgabe 3

- Welche Funktion berechnet der Algorithmus?
- Zeigen Sie die Korrektheit Ihrer Aussage.

Korrektheitsbeweise
- Schleifeninvarianten

Schleifeninvariante

- A(n) ist eine Aussage über den Zustand des Algorithmus vor dem n-ten Durchlauf einer Schleife
- Eine Schleifeninvariante ist korrekt, wenn Sie zu Beginn jedes Schleifendurchlaufs erfüllt ist.
- A(1) wird auch als Initialisierung bezeichnet.

Korrektheitsbeweis für Invarianten

- Induktionsanfang: Die Aussage A(1) gilt
- Induktionsschluss: Gilt A(n) und ist die Eintrittsbedingung der Schleife erfüllt so gilt auch A(n+1)

Korrektheitsbeweise
- Schleifeninvarianten

Schleifeninvariante bei for-Schleifen - Annahmen

- Die Laufvariable wird am Ende der for-Schleife erhöht
- Zur Initialisierung wurde die Laufvariable bereits auf ihren Startwert gesetzt

Abhängigkeit von der Laufvariable

- Da bei for-Schleifen die Laufvariable i.a. eindeutig mit der Anzahl der Schleifendurchläufe zusammenhängt, können wir die Invariante auch in Abhängigkeit der Laufvariablen formulieren

InsertionSort

Einfach(n)
1. k = 1
2. **for** j=1 **to** n **do**
3.   k = k + 1

Aufgabe 4: Welche der beiden Aussagen sind Invarianten?

- Aussage 1
- Aussage 2
- Beide
- Keine

Aussage 1

- k wird in jedem Durchlauf der **for**-Schleife um 1 erhöht

Aussage 2

- k hat den Wert j