Algotithmen und Komplexität [1]

Noah Wollenhaupt

16. Juni 2020

Inhaltsverzeichnis

1 Groß-$\mathcal{O}$-Notation

1.1 Motivation

Wie berechnen Rechner die Zeiten eines Algorithmus?

1. Implementierung in Programmiersprache
2. Zählen von arithmetischen Operationen und Speicherzugriffen
3. Nachschlagen der Zeit der Operationen im Prozessorhandbuch
4. Berechnung der Rechenzeit

Die Groß-O-Notation ist eine abstrakte Möglichkeit, die das Wachstum der Rechenzeit in Abhängigkeit von der Größe der Eingabe beschreiben. Die O-Notation soll von konstanten Faktoren und unwesentlichen Thermen abstrahieren. Man definiert einen X-Wert (k), ab dem die Funktion f(x) immer unter g(x) liegt. Das **c** muss definiert werden und gibt einen Faktor der Funktion g(x) an, für welche dann das Wachstum von f(x) ab **k** immer unterhalb von g(x) verläuft.

1.2 Defintion der $\mathcal{O}$-Notation

$$\mathcal{O}(g) := \{f \in \mathbb{R}_+^{\mathbb{N}} | \exists k \in \mathbb{N} : \exists c \in \mathbb{R}_+ \forall n \in \mathbb{N} : (n \geq k \rightarrow f(n) \leq c \cdot g(n))\}$$

1.3 Eigenschaften der $\mathcal{O}$-Notation

Die $\mathcal{O}$ - Notation verfügt über mehrere Eigenschaften, welche im Folgenden einzeln bewiesen werden.

1.3.1 Reflexivität

Behauptung: Für alle $f : \mathbb{N} \rightarrow \mathbb{R}_+$ gilt, dass $f \in \mathcal{O}(f)$.
Beweis: Wir definieren $k := 0$ und $c := 1$. Dann folgt unsere Behauptung direkt aus der Ungleichung.

$$\forall n \in \mathbb{N} : f(n) \leq f(n)$$

1.3.2 Multiplikation mit konstanten Faktoren

Behauptung: Angenommen sei, dass $f, g : \mathbb{N} \rightarrow \mathbb{R}_+$ gegeben sei und $d \in \mathbb{R}_+$ gegeben sei. Dann gilt:

$$g \in \mathcal{O}(f) \rightarrow d \cdot g \in \mathcal{O}(f)$$

Beweis: Die Voraussetzung $g \in \mathcal{O}(f)$ impliziert, dass es die Konstanten $c' \in \mathbb{R}_+$ und $k' \in \mathbb{N}$ gibt, so dass

$$\forall n \in \mathbb{N} : (n \geq k' \rightarrow g(n) \leq c' \cdot f(n))$$

gilt. Wenn man nun Ungleichung, welche $g(n)$ enthält, mit d multipliziert, bekommt man folgende Gleichung:

$$\forall n \in \mathbb{N} : (n \geq k' \rightarrow d \cdot g(n) \leq d \cdot c' \cdot f(n))$$

Wenn man nun die Konstanten als $k := k'$ und $c := d \cdot c'$ definiert, dann erhält man wiederum folgende Gleichung:

$$\forall n \in \mathbb{N} : (n \geq k \rightarrow g(n) \leq c \cdot f(n))$$

Nach der Definition der $\mathcal{O} - Notaion$, dass $d \cdot g \in \mathcal{O}(f)$ gilt.

1.3.3 Addition

Behauptung: Angenommen sei, dass $f, g, h : \mathbb{N} \to \mathbb{R}_+$ gegeben sei. Dann gilt:

$$f \in \mathcal{O}(h) \wedge g \in \mathcal{O}(h) \to f + g \in \mathcal{O}(h)$$

Beweis: Die Voraussetzungen $f \in \mathcal{O}(h)$ und $g \in \mathcal{O}(h)$ impliziern, dass es die Konstanten $c_1, c_2 \in \mathbb{R}_+$ und $k_1, k_2 \in \mathbb{N}$ gibt, so dass die folgenden Aussagen zutreffend sind:

$$\forall n \in \mathbb{N} : (n \geq k_1 \to f(n) \leq c_1 \cdot h(n))$$

$$\forall n \in \mathbb{N} : (n \geq k_2 \to g(n) \leq c_2 \cdot h(n))$$

Wenn mann nun $k := max(k_1, k_2)$ und $c := c_1 + c_2$ definiert, dann gelten für $n \geq k$:

$$f(n) \leq c_1 \cdot h(n) \quad \text{und} \quad g(n) \leq c_2 \cdot h(n)$$

Diese beiden Gleichungen werden nun addiert und $h(n)$ wird ausgeklammert:

$$f(n) + g(n) \leq (c_1 + c_2) \cdot h(n)$$

Da die Defintion $c := c_1 + c_2$ gegeben ist, kann die Formel vereinfacht werden:

$$f(n) + g(n) \leq c \cdot h(n)$$

und dies bestätigt die Behauptung, dass $f + g \in \mathcal{O}(h)$ gilt.

1.3.4 Transitivität der $\mathcal{O}$-Notation

Behauptung: Angenommen sei, dass $f, g, h : \mathbb{N} \to \mathbb{R}_+$ gegeben sei. Dann gilt:

$$f \in \mathcal{O}(g) \wedge g \in \mathcal{O}(h) \to f \in \mathcal{O}(h)$$

Beweis: Die Voraussetzungen $f \in \mathcal{O}(g)$ und $g \in \mathcal{O}(h)$ implizieren, dass es die Konstanten $c_1, c_2 \in \mathbb{R}_+$ und $k_1, k_2 \in \mathbb{N}$ gibt, so dass die folgenden Aussagen zutreffend sind:

$$\forall n \in \mathbb{N} : (n \geq k_1 \to f(n) \leq c_1 \cdot g(n))$$

$$\forall n \in \mathbb{N} : (n \geq k_2 \to g(n) \leq c_2 \cdot h(n))$$

Wenn man nun $k := max(k_1, k_2)$ und $c := c_1 \cdot c_2$ definiert, dann gelten für $n \geq k$:

$$f(n) \leq c_1 \cdot g(n) \quad \text{und} \quad g(n) \leq c_2 \cdot h(n)$$

Nun wird die rechte Ungleichung mit c_1 multipliziert:

$$f(n) \leq c_1 \cdot g(n) \quad \text{und} \quad c_1 \cdot g(n) \leq c_1 \cdot c_2 \cdot h(n)$$

Durch die Definition von c kann die Gleichung vereinfacht werden:

$$f(n) \leq c_1 \cdot g(n) \quad \text{und} \quad c_1 \cdot g(n) \leq c \cdot h(n)$$

Die Transitivität der Relation $\leq$ impliziert sofot, dass $f(n) \leq c \cdot h(n)$ für $n \geq k$.

1.3.5 Grenzwert der $\mathcal{O}$-Notation

Behauptung: Angenommen sei, dass $f, g : \mathbb{N} \to \mathbb{R}_+$ gegeben sei. Des Weiteren sei anzunehmen, dass der Grenzwert von

$$\lim_{n \to \infty} \frac{f(n)}{g(n)}$$

existiert. Dann gilt $f \in \mathcal{O}(g)$

Beweis: Wir definieren

$$\lambda := \lim_{n \to \infty} \frac{f(n)}{g(n)}$$

Da wir durch unsere Annahme wissen, dass der Grenzwert existiert, wissen wir, dass folgendes gilt:

$$\forall \varepsilon \in \mathbb{R}_+ : \exists k \in \mathbb{R} : \forall n \in \mathbb{N} : (n \geq k \rightarrow |\frac{f(n)}{g(n)} - \lambda| < \varepsilon)$$

Da sies eine valide für alle Werte von ε ist, definieren wir $\varepsilon := 1$. Dann existiert eine Zahl $k \in \mathbb{N}$, so dass für alle $n \in \mathbb{N}$ und $n \geq k$ die folgende Ungleichung gilt:

$$|\frac{f(n)}{g(n)} - \lambda| \leq 1$$

Wir multiplizieren die Ungleichung mit $g(n)$. Da $g(n)$ positiv ist, erhält man:

$$|f(n) - \lambda \cdot g(n)| \leq g(n)$$

Durch die Dreiecksungleichung erhält man:

$$f(n) \leq (1 + \lambda) \cdot g(n)$$

Nun definieren wir $c := 1 + \lambda$ und zeigen, dass $f(n) \leq c \cdot g(n)$ für alle $n \geq k$ gilt.

1.4 Lösen einer Rekurrenzgleichung

Das Lösen einer Rekurrenzgleichung erfolgt in drei wesentlichen Schritten, allerdings muss man immer beachten, dass es zwei Sonderfälle gibt, die eine Abweichung dieses Verfahrens führen. Die Schritte und die Abweichungen bei einer Ausnahme werden nun kurz und knapp beschrieben.

1.4.1 Lösen einer Rekurrenzgleichung

1. Homogenen Teil lösen
   (a) Man betrachtet nur den homogenen Teil der inhomogenen Rekurrzenzgleichung und verwendet den Ansatz $x_n = \lambda^n$
2. Inhomogenen Teil lösen
   (a) Nun wird der inhomogene Teil der Gleichung auch betrachtet und man verwendet den Ansatz $x_n = \gamma^n$
3. α und β mit Hilfe der Anfangsbedingungen bestimmen
   (a) Man setzt alle nun bestimmten Variablen ein und stellt ein Gleichungssystem mit den gegebenen Anfangsbedingungen auf, um α und β zu bestimmen.
   (b) Nun werden alle Variablen in die allgemeine Gleichung eingesetzt und man hat das fertige Ergebnis der inhomogenen Rekurrenzgleichung vorliegen.

1.4.2 Ausnahmen beim Lösen von Rekurrenzgleichungen

1. $\lambda_1 = \lambda_2$: $x_n = \alpha \cdot \lambda^n + \beta \cdot \lambda^n \cdot n + \gamma$
2. a + b = 1: Ansatz: $n \cdot \gamma$ im ihnhomogenen Rekurrenzteil

Literatur

[1] K. Stroetmann, *Algorithms*. Karl Stroetmann, 2020.