Algorithmen und Komplexität

Semester Nr. 2 Algorithmen und Komplexität bei Karl Stroetmann

Inhaltsverzeichnis

Überblick

1. Komplexität von Algorithmen
   - Groß O-Notation
2. Rekurrenz Gleichungen
   - Master Theorem
3. Sortier-Algorithmen:
   - Sortieren durch einfügen (Insertion Sort)
   - Sortieren durch Auswahl (Selection Sort)
   - Sortieren durch Mischen (Merge Sort)
   - Quicksort
   - Radix Sort
   - Heapsort
4. Abstrakte Datentypen
5. Dictionaries (und Mengen)
   - Binäre Bäume
   - AVL – Bäume + 2-3-Bäume
   - Hash Tabellen
   - Tries (Spezialfall Strings)
6. Prioritäts Warteschlangen
7. Graphentheoretische Algorithmen

Man muss nun zwischen einem Algorithmus und einem Programm unterscheiden. Ein Algorithmus ist eine abstrakte Darstellung, wie ein gegebenes Problem gelöst werden kann. Ein Programm hingegen ist die konkrete Implementierung dessen.

Des Weiteren sollte ein Algorithmus drei Kriterien erfüllen:

1. Ein Algorithmus muss korrekt sein
2. Ein Algorithmus sollte effizient sein in Bezug auf die Rechenzeit und den Speicherverbrauch
3. Ein Algorithmus sollte einfach sein.

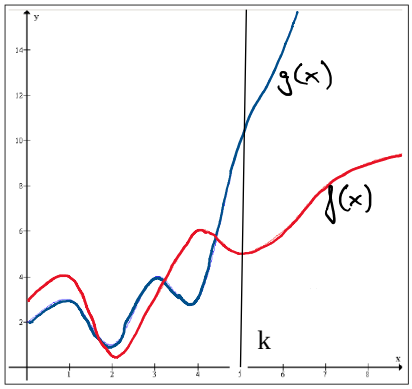
Groß-O-Notation

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 Termen abstrahieren.
Man definiert einen X-Wert (k), ab dem die Funktion $f(x)$ (rot) immer unter $c \cdot g(x)$ liegt. Das c muss ebenfalls definiert werden und gibt einen Faktor der Funktion g(x) (blau) an, für welche dann das Wachstum von $f(x)$ ab k immer unterhalb von $c \cdot g(x)$ verläuft.



Definition der Groß-O-Notation

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

Aussagen der Groß-O-Notation

Reflexivität der Groß-O-Notation

Proposition 2 (Reflexivity of Big $\mathcal{O}$ Notation) For all functions $f : \mathbb{N} \rightarrow \mathbb{R}_+$ we have that

$$f \in \mathcal{O}(f) \quad \text{holds.}$$

Proof: Let us define $k := 0$ and $c := 1$. Then our claim follows immediately from the inequality

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

□

Multiplikation von Konstanten

Assume that we have functions $f, g : \mathbb{N} \rightarrow \mathbb{R}_+$ and a number $d \in \mathbb{R}_+$. Then we have

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

Proof: The premiss $g \in \mathcal{O}(f)$ implies that there are constants $c' \in \mathbb{R}_+$ and $k' \in \mathbb{N}$ such that

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

holds. If we multiply the inequality involving $g(n)$ with d, we get

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

Let us therefore define $k := k'$ and $c := d \cdot c'$. Then we have

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

and by definition this implies $d \cdot g \in \mathcal{O}(f)$.

□

Remark: The previous proposition shows that the big $\mathcal{O}$ notation does indeed abstract from constant factors. ◇

Addition

Proposition 4 (Addition) Assume that $f, g, h : \mathbb{N} \to \mathbb{R}_+$. Then we have

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

Proof: The preconditions $f \in \mathcal{O}(h)$ and $g \in \mathcal{O}(h)$ imply that there are constants $k_1, k_2 \in \mathbb{N}$ and $c_1, c_2 \in \mathbb{R}$ such that both

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

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

holds. Let us define $k := \max(k_1, k_2)$ and $c := c_1 + c_2$. For all $n \in \mathbb{N}$ such that $n \geq k$ it then follows that both

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

holds. Adding these inequalities we conclude that

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

holds for all $n \geq k$. □

Exercise 2: Assume that $f_1, f_2, h_1, h_2 : \mathbb{N} \to \mathbb{R}_+$. Prove that

$$f_1 \in \mathcal{O}(h_1) \wedge f_2 \in \mathcal{O}(h_2) \to f_1 \cdot f_2 \in \mathcal{O}(h_1 \cdot h_2) \quad \text{holds.}$$ ◇

Exercise 3: Assume that $f_1, f_2, h_1, h_2 : \mathbb{N} \to \mathbb{R}_+$. Prove or refute the claim that

$$f_1 \in \mathcal{O}(h_1) \wedge f_2 \in \mathcal{O}(h_2) \to f_1 / f_2 \in \mathcal{O}(h_1 / h_2) \quad \text{holds.}$$ ◇

Transitivität der Groß-O-Notation

Proposition 5 (Transitivity of Big $\mathcal{O}$ Notation) Assume $f, g, h : \mathbb{N} \to \mathbb{R}_+$. Then we have

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

Proof: The precondition $f \in \mathcal{O}(g)$ implies that there exists a $k_1 \in \mathbb{N}$ and a number $c_1 \in \mathbb{R}$ such that

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

holds, while the precondition $g \in \mathcal{O}(h)$ implies the existence of $k_2 \in \mathbb{N}$ and $c_2 \in \mathbb{R}$ such that

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

holds. Let us define $k := \max(k_1, k_2)$ and $c := c_1 \cdot c_2$. Then for all $n \in \mathbb{N}$ such that $n \geq k$ we have the following:

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

Let us multiply the second of these inequalities with c_1. Keeping the first inequality this yields

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

The transitivity of the relation $\leq$ immediately implies $f(n) \leq c \cdot h(n)$ for $n \geq k$. □

Grenzwert der Groß-O-Notation

Proposition 6 (Limit Proposition) Assume that $f, g: \mathbb{N} \to \mathbb{R}_+$. Furthermore, assume that the limit

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

exists. Then we have $f \in \mathcal{O}(g)$.

Proof: Define

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

Since the limit exists by our assumption, we know that

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

Since this is valid for all positive values of ε, let us define $\varepsilon := 1$. Then there exists a number $k \in \mathbb{N}$ such that for all $n \in \mathbb{N}$ satisfying $n \geq k$ the inequality

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

holds. Let us multiply this inequality with $g(n)$. As $g(n)$ is positive, this yields

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

The triangle inequality $|a+b| \leq |a| + |b|$ for real numbers tells us that

$$f(n) = \left|f(n)\right| = \left|f(n) - \lambda \cdot g(n) + \lambda \cdot g(n)\right| \leq \left|f(n) - \lambda \cdot g(n)\right| + \lambda \cdot g(n)$$

holds. Combining the previous two inequalities yields

$$f(n) \leq \left|f(n) - \lambda \cdot g(n)\right| + \lambda \cdot g(n) \leq g(n) + \lambda \cdot g(n) = (1+\lambda) \cdot g(n).$$

Therefore, we define

$$c := 1 + \lambda$$

and have shown that $f(n) \leq c \cdot g(n)$ holds for all $n \geq k$. □

Lösen einer Rekurrenzgleichung

1. Homogenen Teil lösen ($x_n = \lambda^n$)
    - Die Rekurrenzgleichung ohne +1 am Ende
2. Inhomogenen Teil lösen ($x_n = \gamma^n$)
    - Die gesamte Rekurrenzgleichung
3. Einsetzen in die allgemeine Lösung
    - $x_n = \alpha \cdot \lambda_1^n + \beta \cdot \lambda_2^n + \gamma$
4. Gleichungssystem mit den Anfangsbedingungen Aufstellen
5. Lösen des Gleichungssystems und α & β bestimmen
6. Allgemeine Lösung aufstellen
    - $x_n = \alpha \cdot \lambda_1^n + \beta \cdot \lambda_2^n + \gamma$

Ausnahmefälle

a) Gilt $\lambda_1 = \lambda_2$, dann ist die allgemeine Lösung
    - $x_n = \alpha \cdot \lambda^n + \beta \cdot n \cdot \lambda^n + \gamma$

b) Wenn $\alpha + \beta = 1$, dann ist der Ansatz zur Lösung der inhomogenen Rekurrenzgleichung
    - $x_n = \gamma \cdot n$

Arten der Groß-O-Notation

1. Die O-Notation:
    - $\mathcal{O}(g) := \{f \in \mathbb{R}_+^{\mathbb{N}} \mid \exists k \in \mathbb{N} : \exists c \in \mathbb{R}_+ : \forall n \in \mathbb{N} : (n \geq k \to f(n) \leq c \cdot g(n))\}$
2. Die Ω-Notation:
    - $\Omega(g) := \{f \in \mathbb{R}_+^{\mathbb{N}} \mid \exists k \in \mathbb{N} : \exists c \in \mathbb{R}_+ : \forall n \in \mathbb{N} : (n \geq k \to f(n) \geq c \cdot g(n))\}$
3. Die Θ-Notation:
    - $\Theta(g) := \mathcal{O}(g) \cap \Omega(g)$

Master-Theorem

Mit dem Master-Theorem kann man einfacher die Komplexität einer rekursiven Funktion bestimmen.

Bedingungen für das Master-Theorem

a. $\alpha, \beta \in \mathbb{N}$, so dass $\beta \geq 2, \delta \in \mathbb{R}$, so dass $\delta \geq 0$ und
b. die Funktion $f : \mathbb{N} \to \mathbb{R}_+$ erfüllt die Rekurrenzrelation

$$f(n) = \alpha \cdot f(n\,//\,\delta) + \mathcal{O}(n^{\delta})$$

Anwendung des Master-Theorems

Man kann durch das Master-Theorem alle benötigten Variablen schnell ablesen und so die Komplexität des Algorithmus in einen der folgenden Fälle einordnen.

1. $\alpha < \beta^{\delta} \to f(n) \in \mathcal{O}(n^{\delta})$
2. $\alpha = \beta^{\delta} \to f(n) \in \mathcal{O}(log_{\beta}(n) \cdot n^{\delta})$
3. $\alpha > \beta^{\delta} \to f(n) \in \mathcal{O}(n^{log_{\beta}(\alpha)})$