Lernzettel

Komplexität von Algorithmen: Zeit- und Platzkomplexität

Universität: Technische Universität Berlin
Kurs/Modul: Theoretische Grundlagen der Informatik
Erstellungsdatum: September 19, 2025

Lernzettel: Komplexität von Algorithmen: Zeit- und Platzkomplexität

(1) Grundbegriffe.
Die Zeitkomplexität $T(n)$ beschreibt die Anzahl der Grundoperationen, die ein Algorithmus bei einer Eingabe der Größe n ausführt. Die Platzkomplexität $S(n)$ beschreibt den zusätzlich benötigten Speicher außerhalb der Eingangsdaten. Formal:

$$T(n) = \text{Anzahl der primitiven Operationen}$$

$$S(n) = \text{benötigter zusätzlicher Speicher (Bytes, Zellen, etc.)}$$

(2) Asymptotische Notationen.
Seien $f(n)$ und $g(n)$ nicht-negative Funktionen.:

$$f(n) \in O(g(n)) \quad \text{bedeutet:} \quad \exists\, c > 0,\ n_0 \text{ mit } \forall n \geq n_0 :\ f(n) \leq c\, g(n).$$

$$f(n) \in \Theta(g(n)) \quad \text{bedeutet:} \quad \exists\, c_1, c_2 > 0,\ n_0 \text{ mit } \forall n \geq n_0 :\ c_1 g(n) \leq f(n) \leq c_2 g(n).$$

$$f(n) \in \Omega(g(n)) \quad \text{bedeutet:} \quad \exists\, c > 0,\ n_0 \text{ mit } \forall n \geq n_0 :\ f(n) \geq c\, g(n).$$

(3) Worst-, Durchschnitts- und Amortisierte Komplexität.

- Worst-Case: maximaler Aufwand über alle Eingaben der Größe n.
- Durchschnittsfall: erwarteter Aufwand über eine Wahrscheinlichkeitsverteilung der Eingaben.
- Amortisiert: durchschnittlicher Aufwand pro Operation über eine Folge von Operationen.

(4) Zeitkomplexität – Beispiele.
Beispiel 1: lineare Schleife.

$$T(n) = c_1\, n + c_2$$
$$T(n) \in O(n)$$

Beispiel 2: zwei verschachtelte Schleifen.

$$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} c$$
$$T(n) = c\, n^2$$
$$T(n) \in O(n^2)$$

Beispiel 3: Halvierung (Divison by 2).

$$T(n) = T(\lfloor n/2 \rfloor) + c$$
$$T(n) \in O(\log n)$$

(5) Master-Theorem (Rekursionsformen).
Für Rekurrenzen der Form

$$T(n) = a\,T\Big(\frac{n}{b}\Big) + f(n), \quad a \geq 1,\ b > 1$$

gilt grob:

- Falls $f(n) = O(n^{\log_b a - \varepsilon})$ für ein $\varepsilon > 0$, dann $T(n) = \Theta(n^{\log_b a})$.
- Falls $f(n) = \Theta\big(n^{\log_b a} \log^k n\big)$, dann $T(n) = \Theta\big(n^{\log_b a} \log^{k+1} n\big)$.
- Falls $f(n) = \Omega\big(n^{\log_b a + \varepsilon}\big)$ mit einer Regularitätsbedingung, dann $T(n) = \Theta\big(f(n)\big)$.

Beispiele:

$$T(n) = 2\,T(n/2) + n \quad \Rightarrow \quad T(n) = \Theta(n \log n)$$
$$T(n) = 2\,T(n/2) + 1 \quad \Rightarrow \quad T(n) = \Theta(n)$$

(6) Platzkomplexität – Beispiele.
Beispiel 1: konstanter zusätzlicher Speicher.

$$S(n) = c \quad \Rightarrow \quad S(n) \in O(1)$$

Beispiel 2: rekursive Verfahren mit Stack-Tiefe.

$$S(n) = S(n/2) + d$$
$$S(n) \in O(\log n)$$

(7) Praktische Hinweise.
- Wähle Worst-/Best-/Average-Case-Analysen je nach Problem.
- Bevorzuge In-Place-Algorithmen, um Platz zu sparen.
- Prüfe, ob rekursive Lösungen sinnvoll sind; andernfalls Iteration verwenden.