

Lernzettel

Aufwandsabschätzung, Komplexitätstheorie und
formale Korrektheitsnachweise von Algorithmen

Universität: Technische Universität Berlin
Kurs/Modul: Algorithmen und Datenstrukturen
Erstellungsdatum: September 6, 2025



Zielorientierte Lerninhalte, kostenlos!
Entdecke zugeschnittene Materialien für deine Kurse:

<https://study.AllWeCanLearn.com>

Algorithmen und Datenstrukturen

Lernzettel: Aufwandsabschätzung, Komplexitätstheorie und formale Korrektheitsnachweise von Algorithmen

(1) Grundbegriffe und Ziele. Aufwandsabschätzung bezeichnet die Analyse von Laufzeit und Speicherbedarf eines Algorithmus.

- *Laufzeit* $T(n)$ – wie viele Grundoperationen hängen von der Eingröße n ab.
- *Speicherbedarf* $S(n)$ – benötigter zusätzlicher RAM.

Wichtige Arten der Analyse:

- Worst-case, Best-case, Average-case.

(2) Asymptotische Notationen. Für Funktionen $f, g : \mathbb{N} \rightarrow \mathbb{N}$ gelten die gängigen Einordnungen:

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

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

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

Beispiele:

$$T(n) = O(n^2), \quad S(n) = \Theta(n \log n).$$

(3) Aufwandsabschätzung – Vorgehen.

- Formuliere eine Rekursions- oder Iterationsgleichung $T(n)$ bzw. $S(n)$.
- Löse sie analytisch (z. B. Rekursionsgleichungen, Master-Theorem) oder schätze grob ab.
- Prüfe Worst-/Average-/Best-Case separat.

(4) Master-Theorem (kurzfassung). Für rekursive Gleichung der Form

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

gibt es typische Lösungen:

$$T(n) = \Theta(n^{\log_b a}) \quad \text{falls } f(n) = O(n^{\log_b a - \epsilon}),$$

$$T(n) = \Theta(f(n)) \quad \text{falls } f(n) = \Theta(n^{\log_b a}),$$

$$T(n) = \Theta(n^{\log_b a} \log^k n) \quad \text{falls } f(n) = \Omega(n^{\log_b a + \epsilon}) \text{ und Regularity-Bedingung.}$$

(5) Komplexitätstheorie – Grundkonzepte.

- P vs. NP : Sprachen, deren Mitgliedschaft in polynomieller Zeit verifiziert werden kann.
- $P \subseteq NP$ – offen, ob Gleichheit gilt.
- NP -vollständige Probleme: So schwer, wie jedes NP -Problem in NP .

(6) Formale Korrektheitsnachweise von Algorithmen. Korrektheit wird durch Spezifikationen und Beweise abgesichert.

- *Spezifikation* vor und nach der Ausführung:

$$\{P\} S \{Q\},$$

mit Vorbedingung P und Nachbedingung Q .

- *Schleifeninvariante* I : Vor jeder Iteration gilt I , und aus I folgt nach Abbruch Q .
- *Beweis durch Induktion*: Beweise über die Iterationen oder Rekursion.

(7) Beispiel: Binäre Suche – Korrektheit und Laufzeit. Gegeben sortiertes Array $A[1..n]$ und Suchschlüssel x .

- *Spezifikation*: Liefere den Index von x oder -1 wenn nicht vorhanden.
- *Schleifeninvariante*:

I : x ist im Bereich $A[l..r]$ oder x ist außerhalb des Bereichs und A ist sortiert.

- *Fortführung*:

$$m = \left\lfloor \frac{l+r}{2} \right\rfloor, \quad \text{Falls } A[m] = x \text{ dann Index; Wenn } A[m] < x \text{ setze } l = m+1; \text{ sonst } r = m-1.$$

$$T(n) = O(\log n)$$

Korrektur folgt aus Invariante und Abbruchbedingung ($l > r$).

(8) Praktische Hinweise.

- Trenne Zyklen von Rekursion klar; wähle passende Worst-/Average-Case-Szenarien.
- Nutze invariantenbasierte Beweise, statt rein numerischer Tests.
- Modellierung von Algorithmen mit klaren Vor-/Nachbedingungen erleichtert Korrektheit.

(9) Beispielskizze: Rekursion vs. Iteration.

$$T_{\text{rek}}(n) = a T_{\text{rek}}\left(\frac{n}{b}\right) + f(n) \quad \Rightarrow \quad T_{\text{iter}}(n) = \Theta(T_{\text{rek}}(n)) \text{ bei ähnlicher Wachstumsordnung}$$

Beide können durch Master-Theorem oder durch Akkumulation von Teilaufgaben analysiert werden.

(10) Weiterführende Konzepte (Ausblick).

- Kosten-Nutzen-Relation von Datenstrukturen (z. B. Join-/Split-Operationen in Mengenstrukturen).
- Modellierung komplexerer Graphenprobleme (Max-Flow, Min-Cut) als Formalnachweise.

- Scheduling-Problemstellungen und Heuristiken (z. B. Branch Bound, Backtracking) im Kontext der Optimierung.

Hinweis zur Notation. Alle formalen Ausdrücke sind sauber gesetzt:

$$O(\cdot), \quad \Theta(\cdot), \quad \Omega(\cdot), \quad T(n), S(n)$$

und logisch-verbal formulierte Bedingungen in Beweisen folgen dem Standard der formalen Korrektheit.