LEHRERINFO ZUM VIDEO

# Algorithmen und Komplexität

| FACH | STUFE | DAUER | VIDEO |
|---|---|---|---|
| Informatik | Gymnasium · Sek II | 15 min | youtube.com/@Erklaerbaer |

## 01 Lernziele

Nach dem Video können die Schülerinnen und Schüler …

- die Laufzeit eines Algorithmus als Zahl der Grundoperationen in Abhängigkeit von $n$ beschreiben und besten, mittleren und schlechtesten Fall unterscheiden,
- lineare und binäre Suche durchführen, vergleichen und die binäre Suche in Python lesen,
- Selection Sort, Insertion Sort, Bubble Sort und Mergesort erklären und ihre Laufzeiten vergleichen,
- rekursive Algorithmen mit Basisfall und Rekursionsschritt verstehen und den Aufrufstapel nachverfolgen,
- Laufzeiten in O-Notation angeben und vereinfachen: $O(1)$, $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, $O(2^n)$,
- praktisch unlösbare Probleme erkennen (Rundreise, P gegen NP) und den Nutzen von Näherungsverfahren begründen.

## 02 Voraussetzungen

Algorithmus, Ablaufplan, Pseudocode und Halbieren beim Zahlenraten (Folge 6 · Grundlagen); Potenzen und Zweierpotenzen (Folge 2). Python wird nur gelesen; die Grundlagen dazu folgen in Folge 7 bis 9.

## 03 Ablauf des Videos

| ZEIT | KAPITEL | INHALT UND BEISPIEL |
|---|---|---|
| 0:00 | Einstieg | 1 Million Namen: über 8 min gegen 0,02 s · Ident · Fahrplan · Ausblick Rundreise |
| 0:40 | Was du brauchst | Algorithmus, Bausteine, Halbieren · Pseudocode und Python · Hinweis auf 06G |
| 1:10 | Schritte zählen statt Sekunden messen | Stoppuhr hängt vom Gerät ab · Eingabegrösse $n$ · bester, mittlerer, schlechtester Fall |

| | | |
|---|---|---|
| 1:58 | Suchen: linear und binär | O($n$) · 23 in [2 … 91]: Mitte 16 → 56 → 23 · Python mit while · 1 Million → 20 PAUSE-FRAGE 1 Milliarde → 30 · Merksatz |
| 3:40 | Einfache Sortierverfahren | Selection, Insertion, Bubble Sort an 8 Balken · $n(n-1)/2$ · doppelt → vierfach · 1 Million → über 8 min |
| 4:46 | Rekursion | Matrjoschka · 5! = 5 · 4! · Basisfall · Aufrufstapel 4! · ohne Basisfall: Fehler · Türme von Hanoi (7 Züge, 64 Scheiben ≈ 585 Mrd. Jahre) · Merksatz |
| 6:45 | Mergesort: Teile und herrsche | 7 3 9 2 5 8 1 6 als Teilungs- und Zusammenführungsbaum · $\log_2 n$ Ebenen × $n$ · $2 \cdot 10^7$ statt $5 \cdot 10^{11}$ · Quicksort |
| 7:57 | Die O-Notation | Konstanten fallen weg · Wachstumskurven · Tabelle $n$ = 10, 1000, $10^6 \cdot 2^{100}$ hat 31 Stellen PAUSE-FRAGE O($n^2$) · Merksatz |
| 9:24 | Zu schwer für jeden Computer | Rundreise mit 20 Orten: $19!/2 \approx 6 \cdot 10^{16}$ · P gegen NP · Näherung (nächster Nachbar) · Schach ≈ $10^{120}$ Partien |
| 10:38 | Projekt-Baustein | Mediathek: ohne Index 100 000 Zeilen lesen, mit Index ≈ 17 Vergleiche |
| 11:04 | Prüfungsfallen | fünf Fallen (siehe 04) |
| 11:58 | Zusammenfassung | Kernaussagen · zurück zu 8 min gegen 0,02 s |
| 12:38 | Quiz | 4 Fragen · Endkarte Folge 7 Advanced |

## 04 Prüfungsfallen

- Die binäre Suche funktioniert in jeder Liste – nur in sortierten Daten.
- O(2$n$) ist langsamer als O($n$) – konstante Faktoren fallen weg, beides ist linear.
- Rekursion ohne Basisfall ist elegant – sie endet nie (RecursionError, Stack Overflow).
- O($n$ log $n$) ist fast so schlimm wie O($n^2$) – bei $n = 10^6$ rund $2 \cdot 10^7$ statt $10^{12}$ Schritte, also 50 000-mal weniger.
- Ein schnellerer Computer löst exponentielle Probleme – doppelt so schnell heisst bei O($2^n$) nur ein Element mehr.

## 05 Einsatz im Unterricht

**Einstieg:** Zwei Gruppen sortieren je 16 Karten – eine mit Bubble Sort, eine mit Mergesort – und zählen die Vergleiche.

**Werkstatt:** Die binäre Suche aus dem Video in Python ausführen und mit `time` oder einem Vergleichszähler gegen die lineare Suche messen.

**Visualisieren:** Sortierverfahren mit VisuAlgo oder einer Tabellenkalkulation Schritt für Schritt nachspielen (kostenlos im Browser).

**Prüfungsvorbereitung:** Aufgaben 2, 4 und 5 sind im Stil schriftlicher Prüfungen gestellt.

**Hinweis:** Die Zeiten im Video nehmen $10^9$ Grundschritte pro Sekunde an und zählen beim Bubble Sort nur die Vergleiche ($n^2/2 \approx 5 \cdot 10^{11}$). Python sortiert mit Timsort bzw. seit Version 3.11 mit Powersort, beides Mischformen aus Mergesort und Insertion Sort. Die Wachstumskurven im Video sind schematisch skaliert. Die Shannon-Zahl $10^{120}$ ist eine grobe Schätzung der möglichen Schachpartien.

## 06 Quellen


- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: „Introduction to Algorithms“, 4. Aufl., MIT Press (2022)
- Donald E. Knuth: „The Art of Computer Programming“, Band 3: Sorting and Searching, 2. Aufl., Addison-Wesley (1998)
- Stephen A. Cook: „The Complexity of Theorem-Proving Procedures“, STOC (1971) · Clay Mathematics Institute: Millennium Prize Problems – P vs NP
- Claude E. Shannon: „Programming a Computer for Playing Chess“, Philosophical Magazine 41 (1950)
- Python-Dokumentation: Sorting Techniques / list.sort (Timsort, Powersort)
- Gesellschaft für Informatik: Bildungsstandards Informatik für die Sekundarstufe II (2016)

NAME _____ KLASSE _____ DATUM _____

# Algorithmen und Komplexität

## 1 O-Notation vereinfachen ●○○

**Gib** die O-Klasse **an**: a) $4n + 7$ b) $n^2 + 100n$ c) $3 \log n + 2$ d) $2^n + n^3$ e) 500

## 2 Binäre Suche ●●○

Gegeben ist die sortierte Liste [3, 8, 11, 15, 19, 24, 31, 42, 57, 66, 80]. **Führe** die binäre Suche nach 57 und nach 20 durch (Mitte = (links + rechts) // 2) und **notiere** jeweils die verglichenen Zahlen. Wie viele Vergleiche bräuchte die lineare Suche?

## 3 Bubble Sort von Hand ●●○

**Sortiere** [6, 2, 9, 4, 1] mit Bubble Sort. **Schreibe** die Liste nach jedem Durchgang auf und **zähle** die Vergleiche. **Vergleiche** mit der Formel $n(n-1)/2$.

## 4 Rekursion ●●●

Gegeben ist `def summe(n): return 0 if n == 0 else n + summe(n - 1)`. a) **Benenne** Basisfall und Rekursionsschritt. b) **Zeichne** den Aufrufstapel für `summe(4)` und **gib** die Rückgabewerte **an**. c) Was passiert bei `summe(-1)`?

## 5 Laufzeiten vergleichen ●●●

Ein Computer schafft $10^9$ Schritte pro Sekunde. **Berechne** für $n = 10^7$ die Zeit von $n^2$ und von $n \log_2 n$ Schritten ($\log_2 10^7 \approx 23$). **Beurteile**, welches Verfahren für ein Online-Shop-Ranking taugt.

## 6 Türme von Hanoi ●●○

**Begründe**, warum für $n$ Scheiben $2^n - 1$ Züge nötig sind (Tipp: $Z(n) = 2 \cdot Z(n-1) + 1$). Wie viele Züge braucht man für 5 Scheiben?

# Lösungen

## 1 O-Notation vereinfachen

a) $O(n)$ · b) $O(n^2)$ · c) $O(\log n)$ · d) $O(2^n)$ · e) $O(1)$

## 2 Binäre Suche

57: Mitte Index 5 → 24 (zu klein) → Bereich 6–10, Mitte 8 → 57 gefunden: 2 Vergleiche (linear: 9). 20: 24 (zu gross) → Bereich 0–4, Mitte 2 → 11 (zu klein) → Bereich 3–4, Mitte 3 → 15 (zu klein) → Bereich 4–4 → 19 (zu klein) → Bereich leer: nicht vorhanden nach 4 Vergleichen (linear: alle 11).

## 3 Bubble Sort von Hand

Durchgang 1: [2, 6, 4, 1, 9] (4 Vergleiche) · 2: [2, 4, 1, 6, 9] (3) · 3: [2, 1, 4, 6, 9] (2) · 4: [1, 2, 4, 6, 9] (1). Zusammen 10 = 5 · 4 / 2 Vergleiche.

## 4 Rekursion

a) Basisfall: $n = 0 \rightarrow 0$; Rekursionsschritt: $n$ + summe($n - 1$). b) summe(4) → summe(3) → summe(2) → summe(1) → summe(0) = 0; zurück: 1, 3, 6, 10. c) Der Basisfall wird nie erreicht (−1, −2, …): Python bricht mit RecursionError ab.

## 5 Laufzeiten vergleichen

$n^2 = 10^{14}$ Schritte → $10^5$ s ≈ 28 Stunden. $n \log_2 n \approx 10^7 \cdot 23 = 2{,}3 \cdot 10^8$ → 0,23 s. Für ein Shop-Ranking, das in Sekundenbruchteilen antworten muss, kommt nur das $O(n \log n)$-Verfahren in Frage.

## 6 Türme von Hanoi

Um die unterste Scheibe zu bewegen, müssen erst $n - 1$ Scheiben weg ($Z(n - 1)$), dann 1 Zug, dann wieder $Z(n - 1)$ obendrauf: $Z(n) = 2 \cdot Z(n - 1) + 1$ mit $Z(1) = 1 \rightarrow 1, 3, 7, 15, 31 = 2^n - 1$. Für 5 Scheiben: 31 Züge.

## Q Quiz im Video

1 B (logarithmisch) · 2 C (Mergesort) · 3 A (einen Basisfall) · 4 B ($O(n^2)$)