



Gesamtübersicht Algorithmen & Datenstrukturen – Entscheidungs- & Lernhilfe

Diese Übersicht ist **prüfungsorientiert** und stellt sicher, dass **alle im Modul behandelten Themen** klar **kategorisiert**, **vergleichbar** und **entscheidungsrelevant** dargestellt sind.

Ziel:

- Wann verwende ich **welchen Algorithmus / welche Struktur**?
- Wie ist die **Performance**?
- Wie grenzt sich das Thema von ähnlichen ab?

1 Sortialgorithmen

Algorithmus	Kategorie	Wann einsetzen?	Laufzeit (Best / Avg / Worst)	Speicher	Stabil
Selection-Sort	Vergleichssort	sehr kleine Daten, Lehrzwecke	$O(n^2)$ / $O(n^2)$ / $O(n^2)$	$O(1)$	✗
Bubble-Sort	Vergleichssort	fast sortierte Daten	$O(n)$ / $O(n^2)$ / $O(n^2)$	$O(1)$	✓
Insertion-Sort	Vergleichssort	kleine / fast sortierte Daten	$O(n)$ / $O(n^2)$ / $O(n^2)$	$O(1)$	✓
Merge-Sort	Divide & Conquer	stabile Sortierung, große Daten	$O(n \log n)$ / $O(n \log n)$ / $O(n \log n)$	$O(n)$	✓
Quick-Sort	Divide & Conquer	sehr schnell in der Praxis	$O(n \log n)$ / $O(n \log n)$ / $O(n^2)$	$O(\log n)$	✗
Heap-Sort	Heap-basiert	garantierte Laufzeit	$O(n \log n)$ / $O(n \log n)$ / $O(n \log n)$	$O(1)$	✗
Binary-Sort	Einfügesort	Sortieren via binärer Suche	$O(n^2)$ / $O(n^2)$ / $O(n^2)$	$O(1)$	✓

✦ **Merksatz:** Stabilität wichtig → Merge / Insertion Garantierte Laufzeit → Heap / Merge Praxis-Speed → Quick-Sort

2 Such- & Ordnungsstrukturen (Bäume & Listen)

Struktur	Kategorie	Wann einsetzen?	Operationen
Binary-Search-Tree	Suchbaum	einfache Ordnung	$O(n)$ Worst
AVL-Tree	balancierter Suchbaum	schnelle Suche	$O(\log n)$
Rot-Schwarz-Tree	balancierter Suchbaum	viele Updates	$O(\log n)$
Skip-List	probabilistisch	Alternative zu Bäumen	$O(\log n)$ avg

✚ **Merksatz:** Beste Suchzeit → AVL Viele Inserts/Deletes → Rot-Schwarz Einfach & probabilistisch → Skip-List

3 String- & Textstrukturen

Algorithmus / Struktur	Kategorie	Einsatz	Laufzeit
Tries	Prefixbaum	Autocomplete, Prefix-Suche	$O(L)$
Knuth-Morris-Pratt	String-Suche	viele Pattern-Wiederholungen	$O(n + m)$
Boyer-Moore	String-Suche	große Alphabete	sehr schnell (Praxis)

✚ **Merksatz:** Prefix-Suche → Trie Garantiert linear → KMP Praxis-Performance → Boyer-Moore

4 Hashing & probabilistische Verfahren

Struktur	Kategorie	Wann einsetzen?	Laufzeit
Hashing / Hash-Table	Nachschlagen	Key-Value-Zugriff	$O(1)$ avg

✚ **Merksatz:** Schnellstes Nachschlagen → Hashing (keine Ordnung, kein Sortieren!)

5 Graphen – Grundlagen

Thema	Zweck
Graphen	Modellierung von Beziehungen
BFS	Traversierung, kürzester Weg (ungewichtet)
DFS	Traversierung, Struktur, Zyklen

✚ **Merksatz:** Graphen = Knoten + Kanten BFS → breit, DFS → tief

6 Graphen – Kürzeste Wege

Algorithmus	Graph-Typ	Wann einsetzen?	Laufzeit
Dijkstra	gewichtet, keine neg. Kanten	Single-Source	$O((V+E) \log V)$
Warshall	alle Paare	kleine Graphen	$O(V^3)$

🚩 **Merksatz:** Ein Startknoten → Dijkstra Alle Paare → Warshall

7 Graphen – Optimierung

Algorithmus	Kategorie	Ziel
Minimaler Spannbaum (Prim/Kruskal)	Optimierung	minimale Gesamtkosten

🚩 **Merksatz:** MST \neq kürzester Weg Ziel = minimale Summe aller Kanten

8 Formale Modelle

Modell	Zweck
Endliche Automaten (DFA / NFA)	Erkennung regulärer Sprachen

🚩 **Merksatz:** DFA = deterministisch NFA = nichtdeterministisch Beide gleich mächtig

9 Globale Entscheidungs-Merksätze (Prüfungs-Gold)

- *Sortiert?* → Binäre Suche
- *Kürzester Weg, ungewichtet* → BFS
- *Kürzester Weg, gewichtet* → Dijkstra
- *Alle kürzesten Wege* → Warshall
- *Minimale Gesamtkosten* → MST
- *Schnelles Nachschlagen* → Hashing
- *Prefix-Suche* → Trie
- *Pattern-Suche* → KMP / Boyer-Moore