Kapitel 2: Algorithmen und Algorithmuseigenschaften

Algorithmen sind die Grundlage jeder modernen Technologie. Sie stecken in den Navigationssystemen unserer Autos, bestimmen die Ergebnisse von Suchmaschinen und sind das Herzstück von künstlicher Intelligenz. Doch was genau ist ein Algorithmus? Warum sind sie so entscheidend für die Informatik und das tägliche Leben?

Ein Algorithmus ist eine **präzise Beschreibung**, wie eine bestimmte Aufgabe **Schritt für Schritt** gelöst werden kann. Dabei kann es sich um alltägliche Abläufe wie das Zubereiten eines Rezepts oder das Zusammenbauen eines Möbelstücks handeln, aber auch um komplexe Berechnungen in der Informatik.

Ein **Beispiel aus dem Alltag** ist das Schreiben einer WhatsApp-Nachricht: Zuerst entsperrt man das Handy, öffnet die App, wählt einen Chat aus, tippt die Nachricht ein und sendet sie schließlich ab. Dieser Prozess läuft immer nach einer bestimmten Abfolge ab und kann als Algorithmus betrachtet werden.

In diesem Kapitel beschäftigen wir uns mit dem Begriff des „Algorithmus'", seiner Herkunft und den wesentlichen Eigenschaften, die ein Algorithmus besitzen muss. Dabei werden wir sehen, dass Algorithmen nicht nur in der Programmierung eine Rolle spielen, sondern auch in vielen anderen Bereichen unseres Lebens.

2.1. Algorithmusbegriff

Es existiert eine **Vielzahl von Versuchen** einer **Definition** für den **Algorithmusbegriff**. Zum Beispiel:

- Als „Algorithmus [...] bezeichnet [man] heute in der Arithmetik einen Rechenvorgang (bzw. die ihn beschreibenden Regeln), der nach einem bestimmten sich wiederholenden Schema abläuft.“ (Meyers Enzyklopädie, 1971)
- „Ein Algorithmus [...] ist eine vollständige, präzise und in einer Notation oder Sprache mit exakter Definition abgefasste, endliche Beschreibung eines schrittweisen Problemlösungsverfahrens zur Ermittlung gesuchter Datenobjekte (ihrer Werte) aus gegebenen Werten von Datenobjekten, in dem jeder Schritt aus einer Anzahl ausführbarer, eindeutiger Aktionen und einer Angabe über den nächsten Schritt besteht.“ (Pomberger, Gustav; Dobler, Heinz: Algorithmen und Datenstrukturen - Eine systematische Einführung in die Programmierung. München: Pearson Education Deutschland GmbH, 2008)
- „Ein Algorithmus ist eine detaillierte und explizite Vorschrift zur schrittweisen Lösung eines Problems.“ (Gumm, Heinz P.; Sommer, Manfred: Einführung in die Informatik. München: Oldenbourg Wissenschaftsverlag, 2006)

Anschauliche Erklärung

Ein Algorithmus ist eine endliche Folge von eindeutigen und ausführbaren Anweisungen zur Lösung einer Gruppe von Problemen.

Herkunft des Algorithmusbegriffs

Die Bezeichnung "Algorithmus" leitet sich aus dem Namen Al-Khwarizmi ab. Abu Abd Allah Mohammed Ibn Musa Al-Khwarizmi - so der vollständige Name - war ein Universalgelehrter und Mathematiker, der etwa von 780 bis 850 n. Chr. lebte. Er stammte aus Choresm (arab. Khwarizmi) – einer Region, die heute zu Usbekistan und Turkmenistan gehört – und arbeitete in Bagdad.

Al-Khwarizmi beschäftigte sich unter anderem mit Verfahren zur Lösung von Gleichungen. Diese Verfahren können aus heutiger Sicht als Algorithmen betrachtet werden. Dies ist wohl der Grund, weshalb Al-Khwarizmi als Namensgeber für systematische, maschinell verarbeitbare Verfahren gilt.

2.2. Algorithmuseigenschaften

Ein Algorithmus besitzt bestimmte **fundamentale Eigenschaften**, die ihn kennzeichnen.

Allgemeinheit

Ein Algorithmus ist allgemeingültig, d. h. er löst eine Vielzahl von Problemen derselben Art. Die Auswahl eines einzelnen konkreten Problems erfolgt über Eingabedaten oder Parameter.

Beispiel: Ein Navigationssystem kann für unterschiedliche Start- und Zielpunkte die beste Route berechnen.

Eindeutigkeit / Determinismus

An jeder Stelle des Algorithmus muss eindeutig festgelegt sein, was zu tun ist und welcher Schritt als nächstes folgt. Jede Anweisung muss unmissverständlich formuliert sein.

Beispiel: Eine Kochanleitung, die besagt „Salz hinzufügen“, ist nicht eindeutig. Eine präzisere Formulierung wäre: „Füge einen Teelöffel Salz hinzu.“

Ausführbarkeit

Jede einzelne Anweisung eines Algorithmus muss für den Computer oder Menschen ausführbar sein.

Beispiel: Eine Bauanleitung für Möbel enthält nur Werkzeuge und Materialien, die tatsächlich vorhanden und nutzbar sind.

Endlichkeit / Finitheit

Die Beschreibung eines Algorithmus besitzt eine endliche Länge, d. h. er besteht aus einer begrenzten Anzahl von Anweisungen mit begrenzter Länge. Zudem darf ein Algorithmus zu jedem Zeitpunkt nur endlich viel Speicherplatz belegen.

Beispiel: Eine Matheformel zur Berechnung des Flächeninhalts eines Kreises ist eine endliche Vorschrift und kann nicht unendlich viele Schritte enthalten.

Determiniertheit

Wird ein Algorithmus mit denselben Eingabewerten und Startbedingungen wiederholt, so liefert er stets dasselbe Ergebnis.

Beispiel: Der Algorithmus zur Berechnung des Durchschnitts von drei Zahlen liefert bei gleichen Eingaben immer denselben Wert.

Terminiertheit

Ein Algorithmus muss nach einer endlichen Anzahl von Schritten beendet sein.

Beispiel: Ein Kuchenrezept endet mit dem fertigen Kuchen – es läuft nicht unendlich weiter.

2.3. Aufgaben

a) Finde Beispiele aus dem Alltag und aus der Informatik, in denen Algorithmen eine Rolle spielen.

Beispiel: Das Aufbauen eines IKEA-Regals mit Hilfe der Aufbauanleitung ist ein Algorithmus.

Alltag:

- Kochrezept: Zutaten abmessen, Reihenfolge einhalten, Garzeit prüfen → schrittweise, reproduzierbar.
- Navigationssystem: Kürzeste Route berechnen, bei Stau umplanen → algorithmische Pfadsuche.
- Wäsche sortieren: Nach Farbe/Material trennen, Waschprogramm wählen → Entscheidungsregeln.
- Treppenhaus putzen: Von oben nach unten, Raum für Raum → determinierte Abfolge.
- Suche im Bücherregal: Alphabetisch vorgehen → lineare/binäre Suche je nach Ordnung.

Informatik:

- Sortieralgorithmen
- Suchalgorithmen
- Verschlüsselung
- Bildkompression
- Routenplanung

b) Finde zu jeder Algorithmuseigenschaft ein Beispiel aus dem Alltag.

Beispiel: Eine Formel, in Mathe oder Physik, liefert mit denselben Eingangswerten immer dasselbe Endergebnis (= Determiniertheit).

Allgemeinheit:
Backrezept für „Rührkuchen" gilt für beliebige Mengen/Backformen; konkrete Eingaben (Mengen) wählen den Spezialfall.

Eindeutigkeit / Determinismus:
„Füge 1 TL Salz hinzu" ist eindeutig; „Salz dazugeben" ist mehrdeutig und führt zu unterschiedlichen Ergebnissen.

Ausführbarkeit:
IKEA-Anleitung verwendet nur vorhandene Werkzeuge (Inbus, Schraubendreher). „Benutze Laserschneider" wäre im Haushalt nicht ausführbar.

Endlichkeit / Finitheit:
Anleitung „Zimmer streichen" enthält eine endliche Liste von Schritten; kein Schritt verlangt unendliche Wiederholung oder unendliche Textmenge.

Determiniertheit (gleiche Eingaben → gleiches Ergebnis):
Dreisatz zur Preisberechnung: Bei identischen Zahlen entsteht stets derselbe Endpreis.

Terminiertheit (endet nach endlich vielen Schritten):
Kaffeemaschine: Start → Mahlen → Brühen → Stopp. Der Ablauf endet und liefert Kaffee.

Hinweis:
„Determinismus" (eindeutige nächste Aktion) und „Determiniertheit" (gleiches Ergebnis bei gleichen Eingaben) werden oft verwechselt. Ersteres betrifft den Ablauf, letzteres das Resultat.

c) Beschreibe einen Algorithmus, den du täglich anwendest, in einfachen Worten.

Beispiel: Die Reihenfolge der Morgenroutine (Aufstehen, Zähneputzen, Anziehen, Frühstücken, Haus verlassen).

Schulweg mit Bus:

- Wecker aus, aufstehen.
- Uhrzeit prüfen.
- Wenn vor 6:50, dann frühstücken; sonst Frühstück überspringen.
- Tasche kontrollieren: Laptop, Hefte, Stifte, Schlüssel.
- Wohnungstür abschließen.
- Zur Haltestelle gehen.
- Wenn Bus X in ≤ 3 Minuten, dann Bus X nehmen; sonst Bus Y.
- Während im Bus: Nächste Haltestellen prüfen.
- Wenn „Schule“ erreicht, dann aussteigen → Ziel erreicht (Algorithmus endet).

d) Überlege, warum ein Algorithmus unbedingt terminiert sein muss. Was würde passieren, wenn er unendlich weiterläuft?

Leitfrage: Gibt es Beispiele für Programme oder Prozesse, die nicht terminieren? Wie kann man dieses Problem lösen?

Warum Terminiertheit wichtig ist …

- Ergebnisgarantie: Ohne Ende gibt es kein auslieferbares Ergebnis (z. B. nie fertig werdende Steuerberechnung).
- Ressourcen: Endlosprozesse blockieren CPU, Speicher, Energie → System wird langsam/instabil.
- Vorhersagbarkeit: In Technik/Unterricht/Produktion müssen Abläufe planbar enden.

Was passiert bei Nicht-Terminiertheit?

- Endlosschleife: Programm „hängt“ .
- Starvation/Deadlock-Risiken: Andere Aufgaben kommen nicht zum Zug.
- Alltagsanalog: „Putze, bis es perfekt ist“ ohne Kriterium ⇒ nie fertig.

Gibt es absichtlich nicht terminierende Software?

- Ja, Dienste/Server (z. B. Webserver) laufen „dauerhaft“. Aber: Sie lösen kein endliches Problem, sondern warten auf Anfragen. Für Algorithmus im engeren Sinn (konkrete Eingaben → Ergebnis) ist Terminiertheit erforderlich.

Wie stellt man Terminiertheit sicher? (Strategien)

- Abbruchbedingung klar definieren (z. B. „solange Temperatur > 65 °C“).
- Schleifen-Variante: Eine Größe, die sich streng verringert (z. B. Restmenge, Distanz, Zähler).
- Maximale Iterationszahl / Timeout als Sicherheitsnetz.
- Eingabebedingungen prüfen (z. B. keine leeren/ungültigen Werte, die Fortschritt verhindern).

→ Auf lange Zeitskalen gedacht ist **JEDER** Prozess terminiert, aufgrund endlicher Ressourcen oder der menschlichen Sterblichkeit.