Übungsblatt: Algorithmen

Übungsaufgaben zur Digitalisierung und Programmierung

Prof. Dr. Nicolas Meseth

Zusammenfassung

Algorithmen sind ein fundamentales Konzept der Informatik und der digitalen Problemlösung. In diesem Kapitel befassen wir uns mit den grundlegenden Aspekten von Algorithmen und beantworten dabei zentrale Fragen:

- Was ist ein Algorithmus und wie grenzt er sich von einem Computerprogramm ab?
- Welche verschiedenen Arten von Algorithmen existieren und durch welche Beispiele lassen sie sich veranschaulichen?
- Wie können wir Algorithmen systematisch und präzise formulieren?
- Nach welchen Kriterien bewerten wir die Effizienz und Eignung verschiedener Algorithmen für spezifische Problemstellungen?

Was ist ein Algorithmus?

Herkunft des Begriffs

Der Begriff "**Algorithmus**" stammt vom Namen des persischen Mathematikers Muhammad al-Khwarizmi, der um das Jahr 780 n. Chr. geboren wurde. Al-Khwarizmi war ein bedeutender Gelehrter am Hofe des Kalifen al-Mamun und verfasste dort Schriften, die den Gebrauch der indischen Zahlzeichen erklärten. Diese Schriften wurden im 12. Jahrhundert ins Lateinische übersetzt, wobei der Titel "Algoritmi de numero Indorum" verwendet wurde. Im Laufe der Zeit wurde der Name al-Khwarizmi zur Bezeichnung für die von ihm beschriebenen Rechenverfahren und entwickelte sich schließlich zum modernen Begriff "Algorithmus".

Heute bezeichnet ein **Algorithmus** eine präzise Abfolge von Anweisungen, die ein bestimmtes Problem lösen oder eine Aufgabe erfüllen sollen. Im Alltag begegnen uns Algorithmen ständig, oft, ohne dass wir es merken: beim Kochen, bei der Wegbeschreibung oder beim Aufbau eines IKEA-Regals.

Algorithmen und Programme

Ein wichtiger Aspekt von Algorithmen ist ihre Universalität: Sie sind nicht an Computer gebunden. Ein Algorithmus ist im Kern eine strukturierte Anleitung zur Problemlösung, unabhängig davon, wer oder was diese Anleitung ausführt. Diese Flexibilität zeigt sich besonders deutlich in unserem Alltag, wo wir ständig algorithmische Anleitungen befolgen - sei es beim Aufbau eines Möbelstücks oder beim Kochen nach einem Rezept. Bei diesen Tätigkeiten führen wir Menschen die algorithmischen Schritte aus, ganz ohne Beteiligung eines Computers:

- Kochen: Ein Rezept ist ein Algorithmus für die Zubereitung eines Gerichts.
- Wegbeschreibung: Eine Schritt-für-Schritt-Anleitung, um von Punkt A nach Punkt B zu gelangen.
- Bastelanleitung: Die Anweisungen, um ein Modellflugzeug zusammenzubauen.

Viele Algorithmen können von Computern ausgeführt werden. Dafür ist jedoch eine Übersetzung in eine maschinenverständliche Form notwendig. Diese Übersetzung erfolgt durch das Programmieren, wobei wir den Algorithmus in einer Programmiersprache formulieren. Um die Beziehung zwischen Algorithmen und Computerprogrammen besser zu verstehen, ist es hilfreich, drei zentrale Begriffe zu unterscheiden:

- **Algorithmus**: Die abstrakte Beschreibung einer Lösungsmethode in Form einer präzisen, endlichen Sequenz von individuellen Anweisungen. Ein Algorithmus ist unabhängig von der konkreten Umsetzung und kann sowohl von Menschen als auch von Maschinen ausgeführt werden.
- **Programm**: Die konkrete Implementation eines oder mehrerer Algorithmen in einer Programmiersprache. Das Programm übersetzt die abstrakten Anweisungen des Algorithmus in eine Form, die ein Computer verstehen und ausführen kann.
- **Prozess**: Die tatsächliche Ausführung eines Programms durch einen Computer. Dabei werden die programmierten Anweisungen Schritt für Schritt abgearbeitet, um das gewünschte Ergebnis zu erzielen.

Im Folgenden konzentrieren wir uns zunächst auf das Konzept des Algorithmus an sich. Die praktische Implementierung in Form von Programmen werden wir später am Beispiel der Programmiersprache Python kennenlernen. Um Algorithmen jedoch bereits jetzt systematisch beschreiben und analysieren zu können, benötigen wir geeignete Darstellungsformen und Notationen.

Wie stellen wir Algorithmen dar?

Natürliche Sprache

Die natürliche Sprache bietet eine intuitive Möglichkeit, Algorithmen zu beschreiben. Ein klassisches Beispiel hierfür sind Kochrezepte, die als informelle algorithmische Beschreibungen

verstanden werden können. Diese Art der Darstellung folgt keinen festgelegten Regeln - jeder Autor kann die Anweisungen nach eigenem Ermessen formulieren. Entscheidend ist dabei nur, dass andere Menschen die Beschreibung *lesen* und *verstehen* können.

Für die professionelle Informatik ist diese informelle Darstellungsform jedoch nur bedingt geeignet. Während handschriftliche oder natürlichsprachliche Notizen für erste Entwürfe und Skizzen durchaus nützlich sein können, erfordern die präzise Dokumentation und der fachliche Austausch über Algorithmen eine formellere Notation.

Die natürliche Sprache weist für die präzise Beschreibung von Algorithmen zwei wesentliche Nachteile auf. Erstens ist sie mehrdeutig: Wörter und Sätze können je nach Kontext und Formulierung unterschiedlich interpretiert werden. Zweitens ist sie oft unnötig ausführlich, was die klare und effiziente Kommunikation von Algorithmen erschwert. Um diese Herausforderungen zu bewältigen, haben sich in der Informatik zwei formellere und präzisere Darstellungsformen durchgesetzt: Pseudocode und Flussdiagramme.

Pseudocode

Pseudocode ist eine strukturierte, programmiersprachenähnliche Notation zur Beschreibung von Algorithmen. Er kombiniert Elemente der natürlichen Sprache mit grundlegenden Programmierkonzepten wie Schleifen, Bedingungen und Funktionen. Der Vorteil des Pseudocodes liegt in seiner Präzision und Klarheit, ohne dabei an die strengen syntaktischen Regeln einer echten Programmiersprache gebunden zu sein.

Ein wichtiges Merkmal des Pseudocodes ist seine Flexibilität: Er kann je nach Bedarf formeller oder informeller gestaltet werden, solange die grundlegende Logik und Struktur des Algorithmus klar erkennbar bleiben. Dabei werden häufig standardisierte Schlüsselwörter wie “IF”, “THEN”, “WHILE” oder “REPEAT” verwendet, die die algorithmische Struktur verdeutlichen.

Flussdiagramme

Flussdiagramme bieten eine visuelle Darstellung von Algorithmen durch standardisierte grafische Symbole und Verbindungslinien. Diese Notation ist besonders hilfreich, um den Ablauf eines Algorithmus und die logischen Verzweigungen auf einen Blick zu erfassen. Die wichtigsten Elemente eines Flussdiagramms sind Start- und Endpunkte, Prozessschritte (dargestellt durch Rechtecke), Entscheidungen (dargestellt durch Rauten) und Verbindungspfeile, die den Kontrollfluss anzeigen.

Welche Arten von Algorithmen gibt es?

Algorithmen können nach ihrer Funktion und ihrem Anwendungsbereich in verschiedene Kategorien eingeteilt werden. Jede Kategorie repräsentiert einen spezifischen Problemlösungsansatz:

- **Mathematische Algorithmen**: Berechnen oder approximieren Werte
- **Suchalgorithmen**: Finden bestimmte Elemente in einer Datenmenge
- **Sortieralgorithmen**: Ordnen Daten nach bestimmten Kriterien
- **Optimierungsalgorithmen**: Finden die bestmögliche Lösung für ein Problem
- **Graphenalgorithmen**: Arbeiten mit vernetzten Strukturen
- **Stochastische Algorithmen**: Verwenden Zufallselemente, um ein Problem zu lösen
- **Maschinelle Lernalgorithmen**: Erkennen Muster und treffen Vorhersagen

Diese Kategorien sind weder vollständig noch strikt voneinander getrennt. Viele Algorithmen lassen sich mehreren Kategorien zuordnen. Ein anschauliches Beispiel hierfür ist der **Dijkstra-Algorithmus**, der die kürzeste Route zwischen zwei Punkten findet. Er ist sowohl ein Graphenalgorithmus, da er auf vernetzten Strukturen arbeitet, als auch ein Optimierungsalgorithmus, da er die optimale (kürzeste) Route ermittelt.

Im folgenden beleuchten wir ein oder mehr Beispiele für jeder der genannten Klassen.

Mathematische Algorithmen

Größter gemeinsamer Teiler (GGT)

Der Algorithmus zur Berechnung des größten gemeinsamen Teilers (GGT) ist ein klassisches Beispiel für einen eleganten mathematischen Algorithmus. Er wurde vom griechischen Mathematiker Euklid um 300 v. Chr. in seinem Werk “Die Elemente” beschrieben und demonstriert eindrucksvoll die zeitlose Natur algorithmischen Denkens.

Das Verfahren basiert auf einem einfachen, aber genialen Prinzip: Der GGT zweier Zahlen ist identisch mit dem GGT der kleineren Zahl und der Differenz beider Zahlen (Abbildung 1). Zum Beispiel haben die Zahlen 48 und 18 den gleichen GGT wie 18 und 30 (48-18). Durch wiederholtes Anwenden dieser Regel wird der GGT systematisch ermittelt. Die Eleganz dieses Verfahrens liegt in seiner Einfachheit und mathematischen Präzision - Eigenschaften, die auch heute noch moderne Algorithmen auszeichnen.

Abbildung 2 zeigt die Schritte des Euklidschen Algorithmus für das obige Zahlenbeispiel.

```
Identify the larger number. If a < b, swap numbers so
that a > b

Subtract b from a and replace a with the result

Repeat until one of the numbers becomes 0

Return the number that is not zero
```

%%algorithms_euclidean_textual%%

Abbildung 1: Vorgehen des Algorithmus nach Euklid.

```
Loop 1:
a = 18, b = 48 → swap → a = 48, b = 18
a = 48 - 18 = 30

Loop 2:
a = 30, b = 18 → no swap
a = 30 - 18 = 12

Loop 3:
a = 12, b = 18 → swap → a = 18, b = 12
a = 18 - 12 = 6

Loop 4:
a = 6, b = 12 → swap → a = 12, b = 6
a = 12 - 6 = 6

Loop 5:
a = 6, b = 6 → no swap
a = 6 - 6 = 0

return b = 6
```

%%algorithms_euclidean_example%%

Abbildung 2: Beispiel für die Anwendung des Algorithmus nach Euklid.

Babylonisches Wurzelziehen

Das Babylonische Verfahren zur Berechnung der Quadratwurzel ist ein weiteres Beispiel für einen mathematischen Algorithmus. Während der Euklid'sche Algorithmus exakte Ergebnisse liefert, zeigt das Babylonische Verfahren eine andere wichtige Eigenschaft mathematischer Algorithmen: die schrittweise Annäherung an einen Zielwert. Der Algorithmus *approximiert* die Quadratwurzel durch wiederholte Verfeinerung der Schätzung und *konvergiert* dabei gegen den tatsächlichen Wert. Diese Methode demonstriert, wie auch ohne exakte Berechnung präzise Ergebnisse erzielt werden können.

Suchalgorithmen

Lineare Suche

Binäre Suche

Suche in Bäumen

Sortieralgorithmen

Optimierungsalgorithmen

Dijkstra-Algorithmus

Graphenalgorithmen

Stochastische Algorithmen

Annäherung der Kreiszahl π

Maschinelle Lernalgorithmen

Maschinelles Lernen hat sich in den letzten Jahren zu einem der wichtigsten Bereiche der Algorithmenentwicklung entwickelt. Diese Algorithmenklasse bildet die technische Grundlage für viele praktische Anwendungen der Künstlichen Intelligenz (KI). Maschinelle Lernalgorithmen zeichnen sich dadurch aus, dass sie aus vorhandenen Daten Muster erkennen und auf dieser Basis Vorhersagen oder Entscheidungen treffen können.

Welche wichtigen algorithmischen Denkmuster gibt es?

Sequenzen

Die Sequenz ist das grundlegendste Muster in der Algorithmik. Sie beschreibt eine geordnete Abfolge von Anweisungen, die nacheinander ausgeführt werden. Wie bei einer Wegbeschreibung folgt dabei ein Schritt dem anderen in einer festgelegten Reihenfolge. Die korrekte und vollständige Ausführung aller Schritte ist entscheidend - das Auslassen oder Vertauschen von Anweisungen führt in der Regel nicht zum gewünschten Ergebnis.

Sowohl Menschen als auch Computer verarbeiten Anweisungen standardmäßig sequentiell - also Schritt für Schritt von oben nach unten. Diese intuitive Vorgehensweise bildet die Grundlage für das Verständnis von Algorithmen. Allerdings können Algorithmen auch komplexere Strukturen enthalten, die von diesem linearen Ablaufmuster abweichen und alternative Ausführungspfade ermöglichen.

Verzweigungen

Verzweigungen stellen eine grundlegende Form der nicht-linearen Ausführung eines Algorithmus dar. Sie ermöglichen es, dass der Algorithmus basierend auf bestimmten Bedingungen unterschiedliche Ausführungspfade einschlägt. Eine Verzweigung führt also je nach erfüllter oder nicht erfüllter Bedingung zu verschiedenen Anweisungsfolgen. Ein alltägliches Beispiel findet sich in Kochrezepten: “Wenn der Teig zu flüssig ist, füge mehr Mehl hinzu”. Die Anweisung, mehr Mehl hinzuzufügen, wird nur dann ausgeführt, wenn die Bedingung “Teig ist zu flüssig” zutrifft. Ist der Teig bereits von idealer Konsistenz, wird diese Anweisung übersprungen.

Iterationen

Kapselung

Rekursion

Übungsaufgaben

1. Woher stammt der Begriff “Algorithmus”?
2. Definiere, was ein Algorithmus ist, und gib drei Beispiele für Algorithmen aus dem Alltag, die keinen direkten Bezug zu Computern haben.
3. Welche grundlegenden Ansätze zur Klassifizierung von Algorithmen gibt es?
4. Erläutere, was mit der Komplexität eines Algorithmus gemeint ist. Warum ist die Komplexität eines Algorithmus wichtig? Wie wird sie angegeben?
5. Welche Komplexitätsklassen kennst du? Bringe sie in eine Reihenfolge von der geringsten zur höchsten Komplexität.
6. Berechne den größten gemeinsamen Teiler der Zaheln 56 und 98 mithilfe des euklidischen Algorithmus! Dokumentiere jeden Schritt!
7. Wir haben exemplarisch für einen Algorithmus die babylonische Methode zur Approximation einer Quadratwurzel kennengelernt. Beantworte die nachfolgenden Fragen in diesem Kontext:
    a. Berechne die Quadratwurzel von 25 mit der babylonischen Methode und dokumentiere jeden Schritt! Wähle einen sinnvollen Startwert!
    b. Vergleiche die Ergebnisse der babylonischen Methode nach 3, 5 und 7 Iterationen mit dem exakten Wert der Quadratwurzel.
    c. Erkläre die Funktionsweise des babylonischen Algorithmus zur Berechnung der Quadratwurzel. Verwende dazu visuelle Hilfsmittel. Warum konvergiert der Algorithmus gegen den exakten Wert der Quadratwurzel?
8. Erläutere die Monte-Carlo-Methode zur Schätzung von π und erkläre, wie man mithilfe von Zufallszahlen eine Annäherung an π erreichen kann.
9. Finde weitere Probleme, die sich durch Monte-Carlo-Simulationen lösen lassen. Weshalb sind manche dieser Probleme mit anderen Methoden nicht lösbar?
10. Betrachte den Pseudocode in der Abbildung unten und beantworte die folgenden Fragen!