Teil I: Grundlagen

- Kapitel 1: Einführung
- Kapitel 2: Basisalgorithmen
- Kapitel 3: Rekursive Algorithmen

1 Einführung

„Was genau ist ein Algorithmus?“ Auf diese Frage müssen wir zuerst eine Antwort finden, wenn wir uns weiter mit dem Thema beschäftigen wollen – sonst ist alles weitere Vorgehen sinnlos. Die Antwort ist aber eigentlich nicht so schwer zu finden, denn was ein Algorithmus ist, ist streng definiert. Die Definition lautet in etwa wie folgt:

Definition Algorithmus

Ein Algorithmus ist eine streng formale, ausführbare Rechenvorschrift, die in einer überschaubaren Zeit für eine bestimmte Ausgangsbedingung (z. B. in Form einer Zahl) ein reproduzierbares Ergebnis liefert. Hierbei ist es zulässig, dass einzelne Rechenschritte mehrmals wiederholt werden.

Ähnliche Definitionen gibt es zuhauf in zahlreichen Lehrbüchern, manchmal sind sie länger, manchmal kürzer. Ich habe eine möglichst kurze Fassung gewählt, weil es mir auf *die Essenz* ankommt. Außerdem möchte ich von Anfang an Missverständnissen vorbeugen, denn oft werden Algorithmen mit Dingen verglichen, die sie gewiss nicht sind. Einige Lehrbücher vergleichen z. B. Algorithmen mit Kochrezepten, bei deren strikter Einhaltung ein reproduzierbares Gericht herauskommt. Leider ist bei einem Kochrezept schon die Reproduzierbarkeit nicht gegeben, denn jeder Koch kocht anders. Bei dem einem Italiener ist z. B. die Tomatensauce sämiger, bei dem anderen Italiener ist sie schärfer. Außerdem enthalten Kochrezepte Anweisungen, wie „eine Prise Salz“ hinzuzugeben oder das Gericht „auf kleiner Stufe“ zu garen. Keine dieser Anweisungen ist exakt ausführbar. Aber selbst eine Anweisung wie „1,534 Gramm Salz hinzufügen“ oder „bei 102,885 Grad 582,35 Sekunden lang garen“ beachtet noch nicht, dass jeder Herd anders ist und dass Wasser bei hohem Luftdruck später kocht als bei niedrigem Luftdruck. Ein klassisches Kochrezept ist also weit davon entfernt, ein Algorithmus zu sein, weshalb Algorithmen auch eher im mathematischen Bereich angesiedelt sind. Dort und nur dort sind streng formale, ausführbare Vorschriften mit einer genau festgelegten Ausgangsbedingung und reproduzierbarem Ergebnis denkbar. Ein gutes erstes Beispiel für einen Algorithmus ist die schriftliche Division zweier Dezimalzahlen a und b. Wenn Sie für a = 12 und für b = 5 einsetzen, dann erhalten Sie als Ergebnis stets 2,4 – egal wie oft Sie (natürlich in korrekter Weise) nachrechnen.

Kommen wir nun zum letzten Punkt, nämlich zu der Aussage, dass ein Algorithmus „in einer überschaubaren Zeit“ ein Ergebnis liefern muss. Für die Division von 12 durch 5 tut

die schriftliche Division genau dies: Sie kommt in nur wenigen Schritten zu einem korrekten Ergebnis. Leider ändert sich die Sache schon dramatisch, wenn Sie z. B. 10 durch 3 teilen: Ab dem dritten Schritt wiederholen sich die Ziffern endlos und Sie bekommen 3,3333 … heraus, ohne jemals den Rest 0 zu erhalten. Ihr Algorithmus endet also nicht in einer überschaubaren Zeit, wenn Sie keine zusätzliche Abbruchbedingung einfügen. Die Abbruchbedingung, die Sie wahrscheinlich schon aus der Schule kennen, ist die Perioden-Schreibweise. Sobald sich ein Rechenschritt wiederholt, wird der Algorithmus abgebrochen und man gibt eine Periode an.

1.1 Berechenbarkeit von Algorithmen

Ein großes (und wahrscheinlich das größte) Problem von Algorithmen ist also zu entscheiden, ob diese berechenbar sind. „Berechenbar“ im mathematischen Sinne heißt, dass ein Algorithmus in einer endlichen Anzahl von Schritten anhält und dann auch ein korrektes Ergebnis liefert. Leider ist die Aussage, dass ein bestimmtes mathematisches Problem berechenbar ist, erst bewiesen, wenn es einen Algorithmus für dieses Problem gibt, der für jede mögliche Eingabe (z. B. in Form einer Zahl) stets in einer endlichen Anzahl von Schritten ein korrektes Ergebnis liefert. Genau dieser Beweis ist sehr schwierig und kann oft nur mit einem mathematischen Beweisverfahren wie der vollständigen Induktion erbracht werden. Man kann also innerhalb der Mathematik bestimmte Aussagen beweisen, aber eben nicht alle. So gibt es z. B. keinen perfekten Algorithmus, der mir innerhalb einiger Sekunden für eine beliebig lange Zahl sagt, ob diese eine Primzahl ist, und dieser Algorithmus kann wahrscheinlich auch überhaupt nicht gefunden werden. Weil die Sache mit der Berechenbarkeit in der Mathematik nicht so einfach ist, hat man sie in späteren Definitionen von Algorithmen abgeschwächt – sie ist nun für ein Berechnungsverfahren, das ein Algorithmus „sein möchte“, nicht mehr unbedingt nötig, sondern nur noch wünschenswert.

Trotzdem ist die Frage nach der Berechenbarkeit mathematischer Probleme an dieser Stelle immer noch nicht ganz beantwortet. Gibt es zumindest ernstzunehmende Versuche, um herauszufinden, wie wir feststellen können, ob eine bestimmte Fragestellung in einer endlichen bzw. vernünftigen Zeit beantwortbar ist? Leider ist es so, dass wir diese Frage bis jetzt nicht beantworten können. In der Vergangenheit, vor allem in den 20er-Jahren des letzten Jahrhunderts, gab es viele Ansätze und Vorschläge aus der Gruppentheorie, die aber spätestens aufgegeben wurden, als der Mathematiker Kurt Gödel seinen Unvollständigkeitssatz aufstellte. Der Unvollständigkeitssatz besagt Folgendes: In jedem mathematischen Axiomensystem (dies ist quasi eine Sammlung von Regeln, die für eine bestimmte Zahlengruppe wie z. B. die natürlichen Zahlen gilt) gibt es immer Annahmen, die weder beweisbar noch widerlegbar sind. D. h. natürlich auch, dass es immer Algorithmen gibt, von denen man nicht sagen kann, ob sie in einer endlichen Anzahl von Schritten ein Ergebnis liefern. Dennoch gibt es einige populäre Ansätze, um herauszufinden, ob ein bestimmter Algorithmus zumindest nachvollziehbare Ergebnisse liefert. Einer der populäreren Ansätze, den man auch immer im Studium kennenlernt, ist das verwenden einer Turing-Maschine. Auf den folgenden Seiten werden Sie nun an Beispielen erfahren, wie eine Turing-Maschine arbeitet.

1.2 Wie eine Turing-Maschine arbeitet

In seinem berühmten Werk „On computable numbers“ (über berechenbare Zahlen) entwickelte Alan Turing ein Modell, mit dem man feststellen kann, ob ein mathematisches Problem berechenbar ist. Dieses Modell ist die später nach ihm benannte Turing-Maschine. Die Turing-Maschine ist so konstruiert, dass sie immer dann anhält, wenn ein mathematisches Problem berechenbar ist.

Eine Turing-Maschine wird oft in Form einer Eingabeeinheit dargestellt, in die ein unendlich langes Band hineinführt. Dieses Band, das Eingabeband, führt im einfachsten Fall auch direkt wieder aus der Maschine heraus und ist so auch gleichzeitig das Ausgabeband. Auf dem Band können nun – auch schon am Anfang – Symbole stehen. Diese Symbole sind beliebig wählbar, in den meisten Einführungsbeispielen in Bezug auf Turing-Maschinen sind dies aber fast immer Punkte und Striche, Nullen und Einsen oder Zahlen und Buchstaben. Wird die Turing-Maschine nun gestartet (dies kann z. B. durch einen Startknopf oder Hebel geschehen), liest sie zuerst das Symbol ein, über dem sich der Lesekopf zurzeit befindet. Je nachdem, welches Symbol sich nun gerade unter dem Lesekopf befindet, und je nachdem, in welchem Zustand sich die Maschine aktuell befindet, wird eine entsprechende Aktion ausgeführt. Diese Aktion ist meistens eine Ersetzung des Symbols, das sich unter dem Lesekopf befindet, durch ein anderes Symbol und ein anschließendes Verschieben des Bands nach rechts oder links. Natürlich kann der Ersetzungsschritt auch entfallen. Nach Ausführen einer Aktion wechselt die Maschine dann in einen anderen Zustand. Welcher Zustand dies ist, entnimmt die Maschine einer internen Tabelle. Gleichzeitig zu den Zustandswechseln können in der internen Tabelle auch noch zusätzliche Kommandos stehen, die z. B. das Band anhalten, wenn ein bestimmter Endzustand erreicht wird.

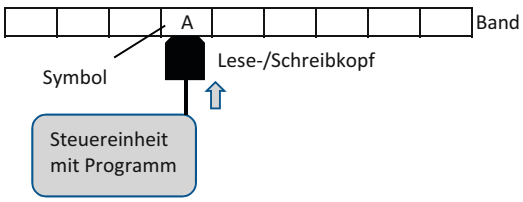

Bild 1.1 Arbeitsweise einer Turing-Maschine

Wie Sie in Bild 1.1 sehen können, hat das Eingabeband kein Ende und keinen Anfang. Ferner sind die Programme, die die Steuereinheit ausführt, fest verdrahtet. Weil die Turing-Maschine so ähnlich arbeitet wie ein sehr einfacher Computer, wird sie auch im Studium immer irgendwann besprochen. Meistens geschieht dies schon in den Vorlesungen zu den Grundlagen der Informatik, also in den ersten zwei Semestern. Deswegen habe auch ich die Turing-Maschine und die grundlegenden Überlegungen zur Berechenbarkeit mathematischer Probleme an den Anfang dieses Buchs gestellt. Die Betonung liegt hier auf „grundlegend“, ich werde also einige einfache Beispiele zu Turing-Maschinen anführen. Die ganzen komplexen mathematischen Formeln aus der Gruppentheorie, die Turing selbst verwendete, um sein Modell zu definieren, werde ich Ihnen natürlich ersparen.