

# Algorithmen kapiern

Visuell lernen und verstehen

» Hier geht's  
direkt  
zum Buch

# DAS VORWORT

# Vorwort



Immer mehr Menschen müssen das Programmieren lernen. Einige verdienen ihr täglich Brot damit, darunter Softwareentwickler oder Webentwickler. Für viele weitere Berufe, die traditionell nichts mit dem Programmieren am Hut hatten, spielt diese Fertigkeit jetzt oder in der Zukunft eine Rolle. Wer weiß, wie man programmiert, versteht zudem die Technologie um uns herum besser.

Leider ist das Wissen um die Vorteile der Programmierung ungleich verteilt. So ist der Anteil der Frauen und bestimmter Ethnien in den Informatikstudiengängen in Nordamerika sehr gering. Wir müssen unbedingt dafür sorgen, dass viel mehr Menschen aus allen Schichten und Ethnien viel mehr Wissen über Programmierung und Informatik erhalten. Dazu müssen wir in verschiedenen Bereichen Fortschritte machen und beispielsweise Vorurteile überwinden, mehr Lehrkräfte ausbilden und diversifiziertere Lernpfade bieten. Wir müssen es den Menschen leicht machen, in das Thema einzusteigen.

Bhargavas Buch begeistert mich, weil es einen neuen Ansatz dafür bietet, mehr über Algorithmen zu erfahren, denn Algorithmen sind ein zentraler Baustein für effektives Programmieren. Einige behaupten, man könne nur etwas über Algorithmen lernen, indem man sich einen dicken und staubtrockenen mathematischen Wälzer zu diesem Thema besorgt, ihn von der ersten bis zur letzten Seite durchackert und jede Information darauf verinnerlicht. Doch solche Bücher nutzen nur Menschen, die auf genau diese Weise lernen können, die über die notwendige Zeit verfügen, auf diese Weise zu lernen, und die überhaupt erst gelernt haben, auf diese Weise zu lernen. Diese Leute gehen auch davon aus, dass wir wissen, *wieso* jemand sich mit Algorithmen beschäftigen will. Ganz ehrlich: Das ist eine ziemlich gewagte Annahme.

Versteh mich nicht falsch: Ich habe einige Lieblingsbücher zum Thema Informatik, die das Wissen über Algorithmen streng mathematisch vermitteln. Ich komme gut damit zurecht. Die meisten Informatikprofessoren kommen gut damit zurecht. Aber genau darin liegt das Problem: Wir halten es viel zu schnell für einen Fakt, dass andere auf dieselbe Art und Weise wie wir selbst lernen. Dabei benötigen wir möglichst viele unterschiedliche Ressourcen, die Wissen über die verschiedensten Informatikthemen vermitteln und sich jeweils an eine bestimmte Zielgruppe richten.

Bhargavas Buch richtet sich ausdrücklich an Menschen, die eine Einführung in Algorithmen ohne mathematische Vorkenntnisse suchen. Am meisten beeindruckt mich, dass Bhargava bestimmte Dinge weggelassen hat. Ein Buch wie dieses kann nicht jedes Detail berücksichtigen. Das würde die Leserschaft überfordern und am Ziel vorbeischießen.

Dank seiner Lehrkompetenz schafft Bhargava es, auf wenigen Seiten viel Stoff verständlich zu vermitteln. Im Kapitel »Dynamische Programmierung« zeigt sich das auf großartige Weise: Bhargava nimmt Fragen vorweg, die in vielen Köpfen auftreten, denen aber andere Bücher über Algorithmen keine Zeile widmen würden.

Ich hoffe sehr, dass dieses Buch auch dir hilft, Ihre Kenntnisse zu erweitern. Dabei spielt es gar keine Rolle, ob du dich erstmals mit Algorithmen befasst oder ob du bisher einfach nicht die für dich richtigen Erklärungen gefunden hast. Viel Vergnügen beim Lesen!

– Daniel Zingaro, Universität Toronto

# Geleitwort



Anfangs war Programmieren für mich einfach nur ein Hobby. Die Grundlagen erlernte ich mit dem Buch *Visual Basic 6 für Dummies* und ich las weitere Bücher, um mehr zu erfahren. Das Thema Algorithmen war für mich allerdings undurchschaubar. Ich kann mich noch gut daran erinnern, dass ich mir das Inhaltsverzeichnis meines ersten Lehrbuchs zu diesem Thema zu Gemüte führte und dachte: »Endlich werde ich das Ganze mal verstehen!« Der Inhalt war jedoch so kompakt und informationsreich, dass ich die Lektüre nach einigen wenigen Wochen aufgab. Erst als ich auf einen wirklich guten Professor für Algorithmen traf, wurde mir klar, wie einfach und elegant die zugrunde liegenden Ideen tatsächlich sind.

Vor einigen Jahren verfasste ich meinen ersten bebilderten Blogbeitrag. Ich kann am besten lernen, wenn mir der Stoff visuell präsentiert wird, daher gefielen mir die illustrierten Beiträge besonders gut. Seitdem habe ich selbst einige bebilderte Beiträge über funktionale Programmierung, Git, Machine Learning und Nebenläufigkeit geschrieben. Ich war anfangs übrigens nur ein mittelmäßiger Autor. Technische Begriffe zu erklären ist schwierig. Es erfordert viel Zeit, gute Beispiele zu finden und komplizierte Konzepte zu erklären, daher ist es am einfachsten, die schwierigen Dinge unter den Teppich zu kehren. Ich dachte eigentlich, ich machte die Sache ganz gut, nachdem sich einer meiner Artikel ziemlich weit verbreitete. Bis ein Kollege zu mir kam und sagte: »Ich habe deinen Artikel gelesen, aber das Thema noch immer nicht verstanden.« Ich musste noch viel über das Schreiben lernen.

Während ich eine Reihe dieser Blogbeiträge verfasste, kam der Manning-Verlag auf mich zu und fragte, ob ich Lust hätte, ein illustriertes Buch zu verfassen. Es stellte sich heraus, dass die Redakteure des Verlags sich hervorragend mit dem

Erklären technischer Konzepte auskannten und sie zeigten mir, wie man Wissen vermittelt. Ich habe dieses Buch verfasst, weil mir eine Sache besonders am Herzen lag: Ich wollte ein Buch schreiben, das schwierige technische Themen gut erklärt, ein leicht verständliches Buch über Algorithmen.

Die erste Auflage dieses Buchs wurde 2016 veröffentlicht. Seitdem wurde es von über 100.000 Menschen gelesen. Ich freue mich sehr, dass der visuelle Stil so gut angekommen ist.

Auch für die zweite Auflage bleibt mein Ziel unverändert. Ich nutze Abbildungen und einprägsame Beispiele, damit die Konzepte in den Köpfen hängen bleiben. Das Buch richtet sich an Personen, die wissen, wie man programmiert, und die ohne mathematisches Kauderwelsch mehr über Algorithmen erfahren möchten.

Diese zweite Auflage füllt ein paar Lücken der Erstaufgabe. Ich habe häufig den Wunsch erhalten, das Konzept der Bäume ausführlich zu erläutern. Mit zwei neuen Kapiteln zu diesem Thema komme ich dem Wunsch gern nach. Auch der Abschnitt zum Stichwort NP-vollständig fällt in dieser Auflage länger aus. NP-Vollständigkeit ist ein sehr abstraktes Konzept. Ich habe versucht, es mit meiner Erklärung zu konkretisieren, und hoffe sehr, dass ich mein Ziel erreicht habe.

Meine Fähigkeit zu schreiben, hat sich seit meinem ersten Blogbeitrag weiterentwickelt und ich hoffe, dass dieses Buch eine leicht verständliche und informative Lektüre sein wird.

# Einleitung



Dieses Buch soll leicht verständlich sein, deshalb vermeide ich große Gedankensprünge. Bei der Vorstellung eines neuen Konzepts erkläre ich es entweder sofort oder weise darauf hin, wann es erläutert wird. Kernkonzepte werden durch Übungen und mehrfache Erklärungen vertieft, sodass du deine Vermutungen überprüfen kannst und dich auf diese Weise vergewisserst, dass du dem Inhalt folgst.

Ich verwende stets Beispiele. Ich möchte keinen Zeichensalat präsentieren, sondern habe zum Ziel, dass es dir leicht fällt, die Konzepte zu visualisieren. Ich bin davon überzeugt, dass man am besten lernt, wenn man sich an bereits Bekanntes erinnern kann – und Beispiele vereinfachen es, sich zu erinnern. Wenn du dir beispielsweise den Unterschied zwischen Arrays und verketteten Listen (die in Kapitel 2 erläutert werden) merken möchtest, brauchst du nur daran zu denken, in einem Kinosaal Platz zu nehmen. Auch wenn ich Gefahr laufe, das Offensichtliche zu verkünden: Ich bin ein visueller Lerner. Dieses Buch enthält also haufenweise Abbildungen.

Der Inhalt des Buchs wurde sorgfältig zusammengestellt. Für ein Buch, das sämtliche Sortieralgorithmen abhandelt, gibt es keinen Bedarf – zu diesem Zweck gibt es die Wikipedia und die Khan Academy. Alle vorgestellten Algorithmen sind praktisch anwendbar. Bei meiner Tätigkeit als Softwareentwickler haben sie sich als nützlich erwiesen und bilden eine gute Grundlage für komplexere Themen. Viel Vergnügen beim Lesen!

## Verwendung dieses Buchs

Die Reihenfolge des Inhalts dieses Buchs und der Inhalt selbst wurden sorgfältig zusammengestellt. Wenn du an einem Thema besonders interessiert bist, steht es dir natürlich frei, einen Teil des Buchs zu überspringen. Andernfalls solltest du die Kapitel jedoch der Reihe nach lesen, da sie aufeinander aufbauen.

Ich empfehle nachdrücklich, dass du den Code der Beispiele auch tatsächlich ausführst. Ich kann es gar nicht oft genug wiederholen. Gib die Codebeispiele genau wie angegeben ein (oder lade sie unter ([https://github.com/egonschiele/grokking\\_algorithms](https://github.com/egonschiele/grokking_algorithms) herunter) und führe sie aus. Wenn du das machst, wirst du sehr viel mehr davon haben.

Darüber hinaus rate ich auch dazu, die Übungen zu bearbeiten. Sie sind nicht zeitaufwändig – in der Regel dauert es nur ein oder zwei Minuten, vielleicht auch mal fünf oder zehn, sie zu bearbeiten. Die Übungen helfen dir dabei, dein Verständnis zu überprüfen, sodass du rechtzeitig bemerkst, wenn du dem Inhalt nicht mehr folgen kannst.

## Wer sollte dieses Buch lesen?

Das Buch wendet sich an Leser, die über grundlegende Kenntnisse der Programmierung verfügen und Algorithmen besser verstehen möchten. Vielleicht stehst du schon vor einer ganz konkreten Aufgabe, für die du eine Lösung in Form eines Algorithmus suchst. Oder du möchtest einfach nur wissen, wofür Algorithmen gut sind. Nachfolgend eine kurze, unvollständige Liste von Lesern, für die das Buch von Nutzen sein kann:

- Hobbyprogrammierer
- Schüler, die einen Programmierkurs besuchen
- Informatiker, die ihre Kenntnisse auffrischen möchten
- Physiker, Mathematiker oder andere Akademiker, die an der Programmierung interessiert sind

## Aufbau dieses Buchs: Überblick

Die ersten drei Kapitel des Buchs behandeln die Grundlagen:

- **Kapitel 1** – Du lernst den ersten praxisnahen Algorithmus kennen: die binäre Suche. Außerdem erfährst du, wie man die Geschwindigkeit eines Algorithmus mithilfe von Landau-Symbolen (engl. *Big-O-Notation*) analysiert. Diese Notation wird im gesamten Buch verwendet, um zu beschreiben, wie langsam oder wie schnell ein Algorithmus arbeitet.

- **Kapitel 2** – Dieses Kapitel hat zwei fundamentale Datenstrukturen zum Thema: Arrays und verkettete Listen. Diese beiden Datenstrukturen werden im gesamten Buch verwendet und dienen dazu, komplexere Datenstrukturen wie Hashtabellen (siehe Kapitel 5) zu erstellen.
- **Kapitel 3** – In diesem Kapitel geht es um die Rekursion. Hierbei handelt es sich um ein praktisches Verfahren, das von vielen Algorithmen verwendet wird (wie z. B. Quicksort, das in Kapitel 4 zur Sprache kommt).

Meiner Erfahrung nach sind die Landau-Notation und Rekursion für Einsteiger ziemlich anspruchsvolle Themen. Deshalb lasse ich es langsam angehen und widme diesen beiden Abschnitten zusätzlichen Raum.

Die verbleibenden Kapitel stellen Algorithmen mit einem breiten Spektrum von Anwendungsmöglichkeiten vor:

- *Problemlösungsverfahren*: Die Kapitel 4, Kapitel 10 und Kapitel 11 behandeln Problemlösungsverfahren. Wenn du einer Aufgabe gegenüberstehst und dir nicht sicher bist, wie sie effizient gelöst werden kann, solltest du das Teile-und-herrsche-Verfahren (siehe Kapitel 4) oder dynamische Programmierung (siehe Kapitel 11) ausprobieren. Möglicherweise stellst du jedoch fest, dass es keine effiziente Lösung gibt. In diesem Fall kannst du einen Greedy-Algorithmus (siehe Kapitel 10) verwenden, um eine Näherungslösung zu berechnen.
- *Hashtabellen*: Kapitel 5 hat Hashtabellen zum Thema. Eine Hashtabelle ist eine äußerst nützliche Datenstruktur, die Schlüssel-und-Wert-Paare enthält, wie beispielsweise den Namen einer Person und deren E-Mail-Adresse oder einen Benutzernamen und das dazugehörige Passwort. Die Bedeutung von Hashtabellen kann kaum überbewertet werden. Wenn ich eine Aufgabe in Angriff nehme, stelle ich zunächst die folgenden beiden Fragen: »Kann ich eine Hash-tabelle verwenden?« und »Kann ich das als Graph darstellen?«.
- *Graphen- und Baumalgorithmen*: Die Kapitel 6, Kapitel 7, Kapitel 8 und Kapitel 9 behandeln diese beiden Themen. Graphen bieten die Möglichkeit, ein Netzwerk zu modellieren: ein soziales Netzwerk, ein Netzwerk aus Straßen oder Neuronen oder irgendeine andere aus Verbindungen bestehende Menge. Die Breitensuche (engl. *Breadth-First Search*, kurz BFS, siehe Kapitel 6) und der Dijkstra-Algorithmus (siehe Kapitel 9) ermöglichen es, die kürzeste Verbindung zwischen zwei Punkten eines Netzwerks zu finden. Du kannst diesen Ansatz beispielsweise dazu verwenden, um die Verschiedenartigkeit zweier Personen oder die kürzeste Verbindung zu einem Ziel zu berechnen. Bäume sind eine Art von Graphen. Sie werden in Datenbanken (häufig in Form von B-Bäumen), in deinem Browser (als DOM-Baum) und im Dateisystem deines Computers verwendet.
- *k nächste Nachbarn (KNN)*: In Kapitel 12 geht es um k nächste Nachbarn, einen einfachen Machine-Learning-Algorithmus. Mit KNN kann beispielsweise ein

Empfehlungssystem, eine optische Zeichenerkennung (engl. *Optical Character Recognition*, OCR) oder ein System zur Vorhersage von Aktienkursen erstellt werden – also Systeme, die irgendeine Vorhersage treffen (»Der Besucher bewertet diesen Film mit 4 Sternen«) oder Objekte klassifizieren (»Dieser Buchstabe ist ein Q«).

- *Die nächsten Schritte:* In Kapitel 13 werden weitere Algorithmen vorgestellt, die gut geeignet sind, um das Thema Algorithmen weiter zu vertiefen.

## Konventionen und Downloads

Die Codebeispiele in diesem Buch sind in Python 3 programmiert. Für den im Buch abgedruckten Code wird eine nicht-proportionale Schrift verwendet, um ihn vom Fließtext zu unterscheiden. Einige der Listings enthalten Anmerkungen, die wichtige Konzepte hervorheben.

Die Codebeispiele kannst du unter folgender Adresse herunterladen:  
[https://github.com/egonschiele/grokking\\_algorithms](https://github.com/egonschiele/grokking_algorithms)

Ich bin davon überzeugt, dass du am besten lernst, wenn es dir Spaß macht – also gönne dir das Vergnügen und führe die Codebeispiele aus!

## Über den Autor



**Aditya Bhargava** ist als Softwareentwickler tätig. Er hat einen Master in Computer Science von der Universität Chicago. Außerdem betreibt er unter <http://adit.io> ein beliebtes, reich bebildertes Tech-Blog.

## Über den Fachkorrektor

David Eisenstat ist als Softwareentwickler in der Forschung tätig. Er hat an der Brown University in Computer Science promoviert.

## Danksagungen

Ich danke dem amerikanischen Originalverlag Manning, der mir die Möglichkeit gab, dieses Buch zu schreiben und mir eine Menge kreativer Freiheit ließ. Ich danke dem Herausgeber Marjan Bace, Mike Stephens, der mich engagierte, Bert Bates, der mich lehrte, wie man schreibt, und der unglaublich aufgeschlossenen und zuvorkommenden Redakteurin Jennifer Stout. Dank gebührt auch dem Produktionsteam: Kevin Sullivan, Mary Piergies, Tiffany Taylor, Leslie Haimes und vielen anderen, die hinter den Kulissen tätig waren. Darüber hinaus möchte ich den vielen Menschen danken, die das Manuskript gelesen und Verbesserungsvorschläge geliefert haben: Karen Bensdon, Rob Green, Michael Hamrah, Ozren Harlovic, Colin Hastie, Christopher Haupt, Chuck Henderson, Pawel Kozlowski, Amit Lamba, Jean-François Morin, Robert Morrison, Sankar Ramanathan, Sander Rosel, Doug Sparling und Damien White.

Dank gebührt auch den Menschen, die mir dabei geholfen haben, dieses Buch zu verwirklichen: den Mitarbeitern des Flashkit Game Boards, die mich das Programmieren lehrten, den vielen Freunden, die verschiedene Kapitel überprüft haben, Ratschläge gaben und es mir ermöglichten, unterschiedliche Erklärungen auszuprobieren. Hierzu gehören Ben Vinegar, Karl Puzon, Alex Manning, Esther Chan, Anish Bhatt, Michael Glass, Nikrad Mahdi, Charles Lee, Jared Friedman, Hema Manickavasagam, Hari Raja, Murali Gudipati, Srinivas Varadan und andere. Ich danke Gerry Brady, der mir Algorithmen erklärt hat. Großer Dank gebührt auch den Algorithmen-Koryphäen CLRS (Cormen, Leiserson, Rivest, Stein), Knuth und Strang. Ich stehe wahrhaft auf den Schultern von Giganten.

Ich danke meinem Vater, meiner Mutter, Priyanka und der übrigen Familie für die beständige Unterstützung. Und ein besonderes Dankeschön an meine Frau Maggie. Uns stehen noch viele Abenteuer bevor – und damit meine ich nicht, am Freitagabend zuhause zu bleiben und Abschnitte umzuschreiben.

An all meine Testleser und Lektoren: Abhishek Koserwal, Alex Lucas, Andres Sacco, Arun Saha, Becky Huett, Cesar Augusto Orozco Manotas, Christian Sutton, Diógenes Goldoni, Dirk Gómez, Ed Bacher, Eder Andres Avila Niño, Frans Oilinki, Ganesh Swaminathan, Giampiero Granatella, Glen Yu, Greg Kreiter, Javid Asgarov, João Ferreira, Jobinesh Purushothaman, Joe Cuevas, Josh McAdams, Krishna Anipindi, Krzysztof Kamyczek, Kyrylo Kalinichenko, Lakshminarayanan AS, Laud Benteil, Matteo Battista, Mikael Byström, Nick Rakochy, Ninoslav Cerkez, Oliver Korten, Ooi Kuan San, Pablo Varela, Patrick Regan, Patrick Wanjau, Philipp Konrad, Piotr Pindel, Rajesh Mohanan, Ranjit Sahai, Rohini Uppuluri, Roman Levchenko, Sambaran Hazra, Seth MacPherson, Shankar Swamy, Srihari Sridharan, Tobias Kopf, Vivek Veerappan, William Jamir Silva, and Xiangbo Mao – eure Vorschläge haben dazu beigetragen, das Buch zu verbessern.

Und schließlich ein großes Dankeschön an alle Leser, die sich auf dieses Buch eingelassen haben und an die Leser, die im Forum zu diesem Buch Feedback gegeben haben. Ihr habt wirklich dazu beigetragen, das Buch zu verbessern.