

50 Algorithms Every Programmer Should Know

Second Edition

An unbeatable arsenal of algorithmic solutions
for real-world problems

Imran Ahmad, Ph.D



BIRMINGHAM—MUMBAI

50 Algorithms Every Programmer Should Know

Second Edition

Copyright © 2023 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Senior Publishing Product Manager: Denim Pinto

Acquisition Editor – Peer Reviews: Tejas Mhasvekar

Project Editor: Rianna Rodrigues

Content Development Editors: Rebecca Robinson and Matthew Davies

Copy Editor: Safis Editing

Technical Editor: Karan Sonawane

Proofreader: Safis Editing

Indexer: Pratik Shirodkar

Presentation Designer: Rajesh Shirasath

Developer Relations Marketing Executive: Vipanshu Parashar

First published: June 2020

Second edition: September 2023

Production reference: 1260923

Published by Packt Publishing Ltd.

Grosvenor House

11 St Paul's Square

Birmingham

B3 1RB, UK.

ISBN 978-1-80324-776-2

www.packt.com

Table of Contents

Preface	xxix
<hr/>	
Section I: Fundamentals and Core Algorithms	1
<hr/>	
Chapter 1: Overview of Algorithms	3
<hr/>	
What is an algorithm?	4
The phases of an algorithm • 5	
Development environment • 7	
Python packages	7
The SciPy ecosystem • 8	
<i>Using Jupyter Notebook</i> • 9	
Algorithm design techniques	9
The data dimension • 10	
The compute dimension • 12	
Performance analysis	12
Space complexity analysis • 13	
Time complexity analysis • 14	
Estimating the performance • 15	
<i>The best case</i> • 15	
<i>The worst case</i> • 16	
<i>The average case</i> • 16	

Big O notation • 16	
Constant time ($O(1)$) complexity • 18	
Linear time ($O(n)$) complexity • 19	
Quadratic time ($O(n^2)$) complexity • 19	
Logarithmic time ($O(\log n)$) complexity • 20	
Selecting an algorithm	21
Validating an algorithm	22
Exact, approximate, and randomized algorithms • 22	
Explainability • 23	
Summary	24

Chapter 2: Data Structures Used in Algorithms	25
--	-----------

Exploring Python built-in data types	26
Lists • 26	
<i>Using lists</i> • 27	
<i>Modifying lists: append and pop operations</i> • 29	
<i>The range() function</i> • 30	
<i>The time complexity of lists</i> • 31	
Tuples • 31	
<i>The time complexity of tuples</i> • 32	
Dictionaries and sets • 33	
<i>Dictionaries</i> • 33	
<i>Sets</i> • 35	
<i>Time complexity analysis for sets</i> • 37	
<i>When to use a dictionary and when to use a set</i> • 37	
Using Series and DataFrames • 38	
<i>Series</i> • 38	
<i>DataFrame</i> • 39	
<i>Creating a subset of a DataFrame</i> • 39	
Matrices • 42	
<i>Matrix operations</i> • 42	

<i>Big O notation and matrices</i> • 43	
Exploring abstract data types	43
Vector • 43	
<i>Time complexity of vectors</i> • 44	
Stacks • 45	
<i>Time complexity of stack operations</i> • 47	
<i>Practical example</i> • 47	
Queues • 47	
<i>Time complexity analysis for queues</i> • 49	
<i>The basic idea behind the use of stacks and queues</i> • 49	
Tree • 50	
<i>Terminology</i> • 50	
<i>Types of trees</i> • 51	
<i>Practical examples</i> • 52	
Summary	53
<hr/>	
Chapter 3: Sorting and Searching Algorithms	55
Introducing sorting algorithms	56
Swapping variables in Python • 56	
Bubble sort • 57	
<i>Understanding the logic behind bubble sort</i> • 57	
<i>Optimizing bubble sort</i> • 59	
<i>Performance analysis of the bubble sort algorithm</i> • 60	
Insertion sort • 61	
<i>Performance analysis of the insertion sort algorithm</i> • 62	
Merge sort • 63	
Shell sort • 66	
<i>Performance analysis of the Shell sort algorithm</i> • 68	
Selection sort • 68	
<i>Performance analysis of the selection sort algorithm</i> • 69	
Choosing a sorting algorithm • 70	

Introduction to searching algorithms	70
Linear search • 71	
<i>Performance analysis of the linear search algorithm</i> • 72	
Binary search • 72	
<i>Performance analysis of the binary search algorithm</i> • 73	
Interpolation search • 73	
<i>Performance analysis of the interpolation search algorithm</i> • 74	
Practical applications	74
Summary	77
<hr/> Chapter 4: Designing Algorithms	79
Introducing the basic concepts of designing an algorithm	80
Concern 1: correctness: will the designed algorithm produce the result we expect? • 81	
Concern 2: performance: is this the optimal way to get these results? • 82	
<i>Characterizing the complexity of the problem</i> • 83	
<i>Exploring the relationship between P and NP</i> • 83	
<i>Introducing NP-complete and NP-hard</i> • 84	
Concern 3 – scalability: how is the algorithm going to perform on larger datasets? • 86	
<i>The elasticity of the cloud and algorithmic scalability</i> • 87	
Understanding algorithmic strategies	87
Understanding the divide-and-conquer strategy • 87	
<i>A practical example – divide-and-conquer applied to Apache Spark</i> • 88	
Understanding the dynamic programming strategy • 90	
<i>Components of dynamic programming</i> • 91	
<i>Conditions for using dynamic programming</i> • 91	
Understanding greedy algorithms • 92	
<i>Conditions for using greedy programming</i> • 92	
A practical application – solving the TSP	93
Using a brute-force strategy • 94	
Using a greedy algorithm • 98	
Comparison of Three Strategies • 99	

Presenting the PageRank algorithm	100
Problem definition • 100	
Implementing the PageRank algorithm • 100	
Understanding linear programming	103
Formulating a linear programming problem • 104	
<i>Defining the objective function</i> • 104	
<i>Specifying constraints</i> • 104	
A practical application – capacity planning with linear programming • 104	
Summary	107
Chapter 5: Graph Algorithms	109
Understanding graphs: a brief introduction	110
Graphs: the backbone of modern data networks • 110	
<i>Real-world applications</i> • 110	
The basics of a graph: vertices (or nodes) • 111	
Graph theory and network analysis	112
Representations of graphs	112
Graph mechanics and types	113
Ego-centered networks • 114	
<i>Basics of egonets</i> • 114	
<i>One-hop, two-hop, and beyond</i> • 114	
<i>Applications of egonets</i> • 115	
Introducing network analysis theory	115
Understanding the shortest path • 116	
<i>Creating a neighborhood</i> • 116	
<i>Triangles</i> • 117	
<i>Density</i> • 118	
Understanding centrality measures • 119	
<i>Degree</i> • 119	
<i>Betweenness</i> • 120	
<i>Fairness and closeness</i> • 121	

<i>Eigenvector centrality</i> • 121	
Calculating centrality metrics using Python • 122	
1. <i>Setting the foundation: libraries and data</i> • 122	
2. <i>Crafting the graph</i> • 122	
3. <i>Painting a picture: visualizing the graph</i> • 123	
Social network analysis • 125	
Understanding graph traversals	125
BFS • 126	
<i>Constructing the adjacency list</i> • 126	
<i>BFS algorithm implementation</i> • 127	
<i>Using BFS for specific searches</i> • 129	
DFS • 130	
Case study: fraud detection using SNA	133
Introduction • 133	
What is fraud in this context? • 133	
Conducting simple fraud analytics • 135	
Presenting the watchtower fraud analytics methodology • 136	
<i>Scoring negative outcomes</i> • 136	
<i>Degree of suspicion</i> • 137	
Summary	139

Section II: Machine Learning Algorithms **141**

Chapter 6: Unsupervised Machine Learning Algorithms 143

Introducing unsupervised learning	143
Unsupervised learning in the data-mining lifecycle • 144	
<i>Phase 1: Business understanding</i> • 145	
<i>Phase 2: Data understanding</i> • 146	
<i>Phase 3: Data preparation</i> • 146	
<i>Phase 4: Modeling</i> • 146	

<i>Phase 5: Evaluation</i> • 147	
<i>Phase 6: Deployment</i> • 147	
Current research trends in unsupervised learning • 148	
Practical examples • 149	
<i>Marketing segmentation using unsupervised learning</i> • 149	
Understanding clustering algorithms	149
<i>Quantifying similarities</i> • 150	
<i>Euclidean distance</i> • 151	
<i>Manhattan distance</i> • 152	
<i>Cosine distance</i> • 153	
k-means clustering algorithm • 155	
<i>The logic of k-means clustering</i> • 155	
<i>Initialization</i> • 155	
<i>The steps of the k-means algorithm</i> • 156	
<i>Stop condition</i> • 157	
<i>Coding the k-means algorithm</i> • 158	
<i>Limitation of k-means clustering</i> • 160	
<i>Hierarchical clustering</i> • 161	
<i>Steps of hierarchical clustering</i>	161
<i>Coding a hierarchical clustering algorithm</i>	162
<i>Understanding DBSCAN</i>	163
<i>Creating clusters using DBSCAN in Python</i>	164
<i>Evaluating the clusters</i>	166
<i>Application of clustering</i> • 166	
Dimensionality reduction	167
<i>Principal component analysis</i> • 167	
<i>Limitations of PCA</i> • 172	
<i>Association rules mining</i> • 172	
<i>Examples of use</i> • 173	
<i>Market basket analysis</i> • 173	

Association rules mining	175
<i>Types of rules • 175</i>	
<i>Trivial rules • 175</i>	
<i>Inexplicable rules • 176</i>	
<i>Actionable rules • 176</i>	
<i>Ranking rules • 176</i>	
<i>Support • 177</i>	
<i>Confidence • 177</i>	
<i>Lift • 178</i>	
<i>Algorithms for association analysis • 178</i>	
<i>Apriori algorithm • 178</i>	
<i>Limitations of the apriori algorithm • 179</i>	
<i>FP-growth algorithm • 179</i>	
<i>Populating the FP-tree • 179</i>	
<i>Mining frequent patterns • 182</i>	
<i>Code for using FP-growth • 183</i>	
Summary	185

Chapter 7: Traditional Supervised Learning Algorithms **187**

Understanding supervised machine learning	188
Formulating supervised machine learning problems	189
Understanding enabling conditions • 192	
Differentiating between classifiers and regressors • 193	
Understanding classification algorithms	193
Presenting the classifiers challenge • 194	
<i>The problem statement • 194</i>	
<i>Feature engineering using a data processing pipeline • 195</i>	
<i>Scaling the features • 199</i>	
<i>Evaluating the classifiers • 200</i>	
Confusion matrices • 201	
<i>Understanding recall and precision • 202</i>	

Understanding the recall and precision trade-off • 204	
<i>Understanding overfitting</i> • 209	
<i>Specifying the phases of classifiers</i> • 211	
Decision tree classification algorithm	212
Understanding the decision tree classification algorithm • 213	
The strengths and weaknesses of decision tree classifiers • 216	
Use cases • 216	
Understanding the ensemble methods	217
Implementing gradient boosting with the XGBoost algorithm • 218	
Differentiating the Random Forest algorithm from ensemble boosting • 221	
Using the Random Forest algorithm for the classifiers challenge • 222	
Logistic regression	223
Assumptions • 224	
Establishing the relationship • 224	
The loss and cost functions • 225	
When to use logistic regression • 226	
Using the logistic regression algorithm for the classifiers challenge • 226	
The SVM algorithm	227
Using the SVM algorithm for the classifiers challenge • 229	
Understanding the Naive Bayes algorithm • 230	
Bayes' theorem	230
Calculating probabilities • 231	
Multiplication rules for AND events • 231	
The general multiplication rule • 232	
Addition rules for OR events • 232	
Using the Naive Bayes algorithm for the classifiers challenge • 233	
For classification algorithms, the winner is...	233
Understanding regression algorithms • 234	
Presenting the regressors challenge • 235	
The problem statement of the regressors challenge • 235	
Exploring the historical dataset • 235	

Feature engineering using a data processing pipeline • 236	
Linear regression	237
Simple linear regression • 238	
Evaluating the regressors • 239	
Multiple regression • 240	
Using the linear regression algorithm for the regressors challenge • 241	
When is linear regression used? • 242	
The weaknesses of linear regression • 242	
The regression tree algorithm • 242	
Using the regression tree algorithm for the regressors challenge • 243	
The gradient boost regression algorithm • 243	
Using the gradient boost regression algorithm for the regressors challenge • 244	
For regression algorithms, the winner is...	245
Practical example – how to predict the weather	245
Summary	248
Chapter 8: Neural Network Algorithms	249
The evolution of neural networks	250
Historical background • 250	
AI winter and the dawn of AI spring • 252	
Understanding neural networks	252
Understanding perceptrons • 252	
Understanding the intuition behind neural networks • 254	
Understanding layered deep learning architectures • 255	
<i>Developing an intuition for hidden layers • 256</i>	
<i>How many hidden layers should be used? • 257</i>	
<i>Mathematical basis of neural network • 258</i>	
Training a neural network	259
Understanding the anatomy of a neural network	259
Defining gradient descent	260
Activation functions	262

Step function • 264	
Sigmoid function • 264	
ReLU • 266	
<i>Leaky ReLU</i> • 267	
Hyperbolic tangent (tanh) • 268	
Softmax • 269	
Tools and frameworks	270
Keras • 270	
<i>Backend engines of Keras</i> • 270	
<i>Low-level layers of the deep learning stack</i> • 271	
<i>Defining hyperparameters</i> • 272	
<i>Defining a Keras model</i> • 272	
Choosing a sequential or functional model	276
Understanding TensorFlow • 276	
Presenting TensorFlow's basic concepts • 277	
Understanding Tensor mathematics • 278	
Understanding the types of neural networks	279
Convolutional neural networks • 279	
<i>Convolution</i> • 280	
<i>Pooling</i> • 280	
Generative Adversarial Networks • 281	
Using transfer learning	281
Case study – using deep learning for fraud detection	283
Methodology • 283	
Summary	287
Chapter 9: Algorithms for Natural Language Processing	289
Introducing NLP	290
Understanding NLP terminology	290
Text preprocessing in NLP • 292	
<i>Tokenization</i> • 292	

<i>Cleaning data • 294</i>	
Cleaning data using Python	297
Understanding the Term Document Matrix	299
Using TF-IDF • 300	
Summary and discussion of results • 302	
Introduction to word embedding	302
Implementing word embedding with Word2Vec	303
Interpreting similarity scores • 305	
Advantages and disadvantages of Word2Vec • 305	
Case study: Restaurant review sentiment analysis	306
Importing required libraries and loading the dataset • 306	
Building a clean corpus: Preprocessing text data • 307	
Converting text data into numerical features • 307	
Analyzing the results • 308	
Applications of NLP	309
Summary	309
Chapter 10: Understanding Sequential Models	311
Understanding sequential data	312
Types of sequence models • 313	
<i>One-to-many • 313</i>	
<i>Many-to-one • 315</i>	
<i>Many-to-many • 316</i>	
Data representation for sequential models	317
Introducing RNNs	318
Understanding the architecture of RNNs • 318	
<i>Understanding the memory cell and hidden state • 318</i>	
<i>Understanding the characteristics of the input variable • 319</i>	
Training the RNN at the first timestep • 320	
<i>The activation function in action • 320</i>	
<i>Training the RNN for a whole sequence • 322</i>	

<i>Calculating the output for each timestep</i> • 324	
Backpropagation through time • 325	
<i>Predicting with RNNs</i> • 326	
Limitations of basic RNNs • 327	
<i>Vanishing gradient problem</i> • 327	
<i>Inability to look ahead in the sequence</i> • 328	
GRU	329
Introducing the update gate • 330	
Implementing the update gate • 331	
Updating the hidden cell • 331	
<i>Running GRUs for multiple timesteps</i> • 332	
Introducing LSTM	332
Introducing the forget gate • 333	
The candidate cell state • 334	
The update gate • 334	
Calculating memory state • 335	
The output gate • 335	
Putting everything together • 337	
Coding sequential models • 337	
<i>Loading the dataset</i> • 338	
<i>Preparing the data</i> • 339	
<i>Creating the model</i> • 339	
<i>Training the model</i> • 342	
<i>Viewing some incorrect predictions</i> • 343	
Summary	343
Chapter 11: Advanced Sequential Modeling Algorithms	345
The evolution of advanced sequential modeling techniques	346
Exploring autoencoders	347
Coding an autoencoder • 348	
Setting up the environment • 349	

<i>Data preparation</i> • 349	
<i>Model architecture</i> • 349	
<i>Compilation</i> • 349	
<i>Training</i> • 350	
<i>Prediction</i> • 350	
<i>Visualization</i> • 350	
Understanding the Seq2Seq model	351
Encoder • 352	
Thought vector • 352	
Decoder or writer • 352	
Special tokens in Seq2Seq • 352	
The information bottleneck dilemma • 353	
Understanding the attention mechanism	353
What is attention in neural networks? • 353	
<i>Basic idea</i> • 353	
<i>Example</i> • 354	
Three key aspects of attention mechanisms • 355	
A deeper dive into attention mechanisms • 356	
The challenges of attention mechanisms • 357	
Delving into self-attention	357
Attention weights • 358	
Encoder: bidirectional RNNs • 359	
Thought vector • 360	
Decoder: regular RNNs • 360	
Training versus inference • 360	
Transformers: the evolution in neural networks after self-attention	361
Why transformers shine • 362	
A Python code breakdown • 362	
Understanding the output • 363	
LLMs	364
Understanding attention in LLMs • 365	

Exploring the powerhouses of NLP: GPT and BERT • 365	
<i>2018's LLM pioneers: GPT and BERT</i> • 366	
Using deep and wide models to create powerful LLMs • 367	
Bottom of Form	367
Summary	369

Section III: Advanced Topics	371
-------------------------------------	------------

Chapter 12: Recommendation Engines	373
Introducing recommendation systems	374
Types of recommendation engines	374
Content-based recommendation engines • 375	
<i>Determining similarities in unstructured documents</i> • 375	
Collaborative filtering recommendation engines • 376	
<i>Issues related to collaborative filtering</i> • 377	
Hybrid recommendation engines • 378	
<i>Generating a similarity matrix of the items</i> • 379	
<i>Generating reference vectors of the users</i> • 379	
<i>Generating recommendations</i> • 380	
<i>Evolving the recommendation system</i> • 381	
Understanding the limitations of recommendation systems	381
The cold start problem • 381	
Metadata requirements • 382	
The data sparsity problem • 382	
The double-edged sword of social influence in recommendation systems • 382	
Areas of practical applications	383
Netflix's mastery of data-driven recommendations • 383	
The evolution of Amazon's recommendation system • 384	
Practical example – creating a recommendation engine	385
1. Setting up the framework • 385	

2. Data loading: ingesting reviews and titles • 385	
3. Merging data: crafting a comprehensive view • 386	
4. Descriptive analysis: gleaning insights from ratings • 387	
5. Structuring for recommendations: crafting the matrix • 387	
6. Putting the engine to test: recommending movies • 388	
<i>Finding movies correlating with Avatar (2009) • 389</i>	
10,000 BC (2008) • 389	-0.075431
<i>Understanding correlation • 389</i>	
<i>Evaluating the model • 390</i>	
<i>Retraining over time: incorporating user feedback • 390</i>	
Summary	390
<hr/>	
Chapter 13: Algorithmic Strategies for Data Handling	393
<hr/>	
Introduction to data algorithms	394
Significance of CAP theorem in context of data algorithms • 394	
Storage in distributed environments • 394	
Connecting CAP theorem and data compression • 395	
Presenting the CAP theorem	395
CA systems • 397	
AP systems • 397	
CP systems • 398	
Decoding data compression algorithms	398
Lossless compression techniques • 398	
<i>Huffman coding: Implementing variable-length coding • 399</i>	
<i>Understanding dictionary-based compression LZ77 • 403</i>	
<i>Advanced lossless compression formats • 404</i>	
Practical example: Data management in AWS: A focus on CAP theorem and compression algorithms	405
1. Applying the CAP theorem • 405	
2. Using compression algorithms • 406	

3. Quantifying the benefits • 406	
Summary	407
Chapter 14: Cryptography	409
Introduction to cryptography	410
Understanding the importance of the weakest link • 410	
The basic terminology • 411	
Understanding the security requirements • 411	
<i>Step 1: Identifying the entities</i> • 412	
<i>Step 2: Establishing the security goals</i> • 412	
<i>Step 3: Understanding the sensitivity of the data</i> • 413	
Understanding the basic design of ciphers • 414	
<i>Presenting substitution ciphers</i> • 414	
<i>Cryptanalysis of substitution ciphers</i> • 416	
<i>Understanding transposition ciphers</i> • 417	
Understanding the types of cryptographic techniques	417
Using the cryptographic hash function • 418	
<i>Implementing cryptographic hash functions</i> • 419	
<i>An application of the cryptographic hash function</i> • 422	
<i>Choosing between MD5 and SHA</i> • 422	
Using symmetric encryption • 423	
<i>Coding symmetric encryption</i> • 423	
<i>The advantages of symmetric encryption</i> • 424	
<i>The problems with symmetric encryption</i> • 424	
Asymmetric encryption • 424	
<i>The SSL/TLS handshaking algorithm</i> • 425	
<i>Public key infrastructure</i> • 427	
<i>Blockchain and cryptography</i> • 428	
Example: security concerns when deploying a machine learning model	430
MITM attacks • 430	
<i>How to prevent MITM attacks</i> • 431	

Avoiding masquerading • 432	
Data and model encryption • 432	
Summary	435
Chapter 15: Large-Scale Algorithms 437	
Introduction to large-scale algorithms	438
Characterizing performant infrastructure for large-scale algorithms	439
Elasticity • 439	
Characterizing a well-designed, large-scale algorithm • 439	
<i>Load balancing</i> • 440	
<i>ELB: Combining elasticity and load balancing</i> • 441	
Strategizing multi-resource processing	442
Understanding theoretical limitations of parallel computing	444
Amdahl's law • 444	
Deriving Amdahl's law • 444	
CUDA: Unleashing the potential of GPU architectures in parallel computing • 447	
<i>Bottom of form</i> • 448	
<i>Parallel processing in LLMs: A case study in Amdahl's law and diminishing returns</i> • 449	
Rethinking data locality • 451	
Benefiting from cluster computing using Apache Spark • 452	
How Apache Spark empowers large-scale algorithm processing	454
Distributed computing • 454	
In-memory processing • 454	
Using large-scale algorithms in cloud computing	455
Example • 455	
Summary	456
Chapter 16: Practical Considerations 457	
Challenges facing algorithmic solutions	458
Expecting the unexpected • 458	

Failure of Tay, the Twitter AI bot	459
The explainability of an algorithm	460
Machine learning algorithms and explainability • 460	
<i>Presenting strategies for explainability</i> • 461	
Understanding ethics and algorithms	467
Problems with learning algorithms • 468	
Understanding ethical considerations • 468	
Factors affecting algorithmic solutions • 469	
<i>Considering inconclusive evidence</i> • 470	
<i>Traceability</i> • 470	
<i>Misguided evidence</i> • 470	
<i>Unfair outcomes</i> • 470	
Reducing bias in models	471
When to use algorithms	472
Understanding black swan events and their implications on algorithms • 472	
Summary	474
Other Books You May Enjoy	475
<hr/> Index	481