Algorithms A Functional Programming Approach

This book constitutes the proceedings of the 13th International Conference on Parallel Computing Technologies, PaCT 2015, held in Petrozavodsk, Russia, during August / September 2015. The 37 full papers and 14 short papers presented were carefully reviewed and selected from 87 submissions. The papers are organized in topical sections on parallel models, algorithms and programming methods; unconventional computing; cellular automata; distributed computing; special processors programming techniques; applications.

This book presents a variety of widely used algorithms, expressing them in a pure functional programming language to make their structure and operation clearer to readers. In the opening chapter the author introduces the specific notations that constitute the variant of Scheme that he uses. The second chapter introduces many of the simpler and more general patterns available in functional programming. The chapters that follow introduce and explain data structures, sorting, combinatorial constructions, graphs, and sublist search. Throughout the book the author presents the algorithms in a purely functional version of the Scheme programming language, which he makes available on his website. The book is supported with exercises, and it is suitable for undergraduate and graduate courses on programming techniques.

This practically-focused textbook presents a concise tutorial on data structures and algorithms using the object-functional language Scala. The material builds upon the foundation established in the title Programming with Scala: Language Exploration by the same author, which can be treated as a companion text for those less familiar with Scala. Topics and features: discusses data structures and algorithms in the form of design patterns; covers key topics on arrays, lists, stacks, queues, hash tables, binary trees, sorting, searching, and graphs; describes examples of complete and running applications for each topic; presents a functional approach to implementations for data structures and algorithms (excepting arrays); provides numerous challenge exercises (with solutions), encouraging the reader to take existing solutions and improve upon them; offers insights from the author's extensive industrial experience; includes a glossary, and an appendix supplying an overview of discrete mathematics. Highlighting the techniques and skills necessary to quickly derive solutions to applied problems, this accessible text will prove invaluable to time-pressured students and professional software engineers.

This collection of 17 papers drawn from an August 1999 workshop held in Scotland presents advances in parallel functional programming, type systems, architectures and implementation, language applications, and theory. Topics include BSP-based cost analysis of skeletal programs, how to combine the benefits of strict and soft typing, interfacing Java with Haskell, a functional design framework for genetic algorithms, and list homomorphisms with accumulation and indexing. No index. Distributed by ISBS. c. Book News Inc.

Your guide to the functional programming paradigm Functional programming mainly sees use in math computations, including those used in Artificial Intelligence and gaming. This programming paradigm makes algorithms used for math calculations easier to understand and provides a concise method of coding algorithms by people who aren't developers. Current books on the market have a significant learning curve because they're written for developers, by developers—until now. Functional Programming for Dummies explores the differences between the pure (as represented by the Haskell language) and impure (as represented by the Python language) approaches to functional programming for readers just like you. The pure approach is best suited to researchers who have no desire to create production code but do need to test algorithms fully and demonstrate their usefulness to peers. The impure approach is best suited to production environments because it's possible to mix coding paradigms in a single application to produce a result more quickly. Functional Programming For Dummies uses this two-pronged approach to give you an all-in-one approach to a coding methodology that can otherwise be hard to grasp. Learn pure and impure when it comes to coding Dive into the processes that most functional programmers use to derive, analyze and prove the worth of algorithms Benefit from examples that are provided in both Python and Haskell Glean the expertise of an expert author who has written some of the market-leading programming books to date If you're ready to massage data to understand how things work in new ways, you've come to the right place!

Third International Conference, MPC '95, Kloster Irsee, Germany, July 17 - 21, 1995. Proceedings

WORK EFFECT LEG CODE _p1

Volume 27: Supplement 6

Type Theory and Functional Programming

13th International Conference, RAMiCS 2012, Cambridge, United Kingdom, September 17-21, 2012, Proceedings

Functional Programming

Your guide to the functional programming paradigm Functional programming mainly sees use in math computations, including those used in Artificial Intelligence and gaming. This programming paradigm makes algorithms used for math calculations easier to understand and provides a concise method of coding algorithms by people who aren't developers. Current books on the market have a significant learning curve because they're written for developers, by developers—until now. Functional Programming for Dummies explores the differences between the pure (as represented by the Haskell language) and impure (as represented by the Python language) approaches to functional programming for readers just like you. The pure approach is best suited to researchers who have no desire to create production code but do need to test algorithms fully and demonstrate their usefulness to peers. The impure approach is best suited to production environments because it's possible to mix coding paradigms in a single application to

produce a result more quickly. Functional Programming For Dummies uses this two-pronged approach to give you an all-in-one approach to a coding methodology that can otherwise be hard to grasp. Learn pure and impure when it comes to coding Dive into the processes that most functional programmers use to derive, analyze and prove the worth of algorithms Benefit from examples that are provided in both Python and Haskell Glean the expertise of an expert author who has written some of the market-leading programming books to date If you're ready to massage data to understand how things work in new ways, you've come to the right place!

Use Kotlin to build Android apps, web applications, and more—while you learn the nuances of this popular language. With this unique cookbook, developers will learn how to apply thisJava-based language to their own projects. Both experienced programmers and those new to Kotlin will benefit from the practical recipes in this book. Author Ken Kousen (Modern Java Recipes) shows you how to solve problems with Kotlin by concentrating on your own use cases rather than on basic syntax. You provide the contextand this book supplies the answers. Already big in Android development, Kotlin can be used anywhere Java is applied, as well as for iOS development, native applications, JavaScriptgeneration, and more. Jump in and build meaningful projects with Kotlin today. Apply functional programming concepts, including lambdas, sequences, and concurrency See how to use delegates, late initialization, and scope functions Explore Java interoperability and access Java libraries using Kotlin Add your own extension functions Use helpful libraries such as JUnit 5 Get practical advice for working with specific frameworks, like Android and Spring

Grokking Functional Programming is a practical book written especially for object-oriented programmers. Grokking Functional Programming teaches you first to break down problems in a new way so you can approach them from a FP mindset. Following carefully-selected examples with thorough, carefully-paced explanations, you'll immerse yourself in FP concept by concept. Along the way, exercises, checks for understanding, and even the occasional puzzler give you opportunities to think and practice what you're learning. Grokking Functional Programming is a practical book written especially for object-oriented programmers. It will help you map familiar ideas like objects and composition to FP concepts such as programming with immutable data and higher-order functions. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications.

A programming course should concentrate as much as possible on a program's logical structure and design rather than simply show how to write code. The Functional Approach to Programming achieves this aim because logical concepts are evident and programs are transparent so can be written quickly and cleanly. In this book the authors emphasise the notions of function and function application which relate programming to familiar concepts from mathematics and logic. They introduce functional programming via examples but also explain what programs compute and how to reason about them. They show how the ideas can be implemented in the Caml language, a dialect of the ML family, and give examples of how complex programs from a variety of areas (such as arithmetic, tree algorithms, graph algorithms, text parsing and geometry) can be developed in close agreement with their specifications. Many exercises and examples are included throughout the book; solutions are also available.

Achieving Synergy Between Computer Power and Human Resources to Temporal and Modal Logic Programming Languages.

Relational and Algebraic Methods in Computer Science

13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31-September 4, 2015, Proceedings

Real World OCaml

A Problem-Focused Approach

Functional Programming for Java Developers

Working Effectively with Legacy Code

Programming is hard. Building a large program is like constructing a steam locomotive through a hole the size of a postage stamp. An artefact that is the fruit of hundreds of person-years is only ever seen by anyone through a lOO-line window. In some ways it is astonishing that such large systems work at all. But parallel programming is much, much harder. There are so many more things to go wrong. Debugging is a nightmare. A bug that shows up on one run may never happen when you are looking for it - but unfailingly returns as soon as your attention moves elsewhere. A large fraction of the program's code can be made up of marshalling and coordination algorithms. The core application can easily be obscured by a maze of plumbing. Functional programming is a radical, elegant, high-level attack on the programming problem. Radical, because it dramatically eschews side-effects; elegant, because of its close connection with mathematics; high-level, be cause you can say a lot in one line. But functional programming is definitely not (yet) mainstream. That's the trouble with radical approaches: it's hard for them to break through and become mainstream. But that doesn't make functional programming any less fun, and it has turned out to be a won derful

laboratory for rich type systems, automatic garbage collection, object models, and other stuff that has made the jump into the mainstream.

Richard Bird takes a radical approach to algorithm design, namely, design by calculation. These 30 short chapters each deal with a particular programming problem drawn from sources as diverse as games and puzzles, intriguing combinatorial tasks, and more familiar areas such as data compression and string matching. Each pearl starts with the statement of the problem expressed using the functional programming language Haskell, a powerful yet succinct language for capturing algorithmic ideas clearly and simply. The novel aspect of the book is that each solution is calculated from an initial formulation of the problem in Haskell by appealing to the laws of functional programming. Pearls of Functional Algorithm Design will appeal to the aspiring functional programmer, students and teachers interested in the principles of algorithm design, and anyone seeking to master the techniques of reasoning about programs in an equational style.

Completely revised and updated, this best-selling introduction to programming in JavaScript focuses on writing real applications. JavaScript lies at the heart of almost every modern web application, from social apps like Twitter to browser-based game frameworks like Phaser and Babylon. Though simple for beginners to pick up and play with, JavaScript is a flexible, complex language that you can use to build full-scale applications. This much anticipated and thoroughly revised third edition of Eloquent JavaScript dives deep into the JavaScript language to show you how to write beautiful, effective code. It has been updated to reflect the current state of Java¬Script and web browsers and includes brand-new material on features like class notation, arrow functions, iterators, async functions, template strings, and block scope. A host of new exercises have also been added to test your skills and keep you on track. As with previous editions, Haverbeke continues to teach through extensive examples and immerses you in code from the start, while exercises and full-chapter projects give you hands-on experience with writing your own programs. You start by learning the basic structure of the JavaScript language as well as control structures, functions, and data structures to help you write basic programs. Then you'll learn about error handling and bug fixing, modularity, and asynchronous programming before moving on to web browsers and how JavaScript is used to program them. As you build projects such as an artificial life simulation, a simple programming language, and a paint program, you'll learn how to: - Understand the essential elements of programming, including syntax, control, and data - Organize and clarify your code with object-oriented and functional programming techniques - Script the browser and make basic web applications - Use the DOM effectively to interact with browsers - Harness Node.js to build servers and utilities Isn't it time you became fluent in the language of the Web? * All source code is available online in an inter¬active sandbox, where you can edit the code, run it, and see its output instantly.

Numerical algorithms, modern programming techniques, and parallel computing are often taught serially across different courses and different textbooks. The need to integrate concepts and tools usually comes only in employment or in research - after the courses are concluded - forcing the student to synthesise what is perceived to be three independent subfields into one. This book provides a seamless approach to stimulate the student simultaneously through the eyes of multiple disciplines, leading to enhanced understanding of scientific computing as a whole. The book includes both basic as well as advanced topics and places equal emphasis on the discretization of partial differential equations and on solvers. Some of the advanced topics include wavelets, high-order methods, non-symmetric systems, and parallelization of sparse systems. The material covered is suited to students from engineering, computer science, physics and mathematics.

Software development today is embracing functional programming (FP), whether it's for writing concurrent programs or for managing Big Data. Where does that leave Java developers? This concise book offers a pragmatic, approachable introduction to FP for Java developers or anyone who uses an object-oriented language. Dean Wampler, Java expert and author of Programming Scala (O'Reilly), shows you how to apply FP principles such as immutability, avoidance of side-effects, and higher-order functions to your Java code. Each chapter provides exercises to help you practice what you've learned. Once you grasp the benefits of functional programming, you'll discover that it improves all of the code you write. Learn basic FP principles and apply them to object-oriented programming Discover how FP is more concise and modular than OOP Get useful FP lessons for your Java type design—such as avoiding nulls Design data structures and algorithms using functional programming principles Write concurrent programs using the Actor model and software transactional memory Use functional libraries and frameworks for Java—and learn where to go next to deepen your functional programming skills

Patterns and Skeletons for Parallel and Distributed Computing

Parallel Scientific Computing in C++ and MPI

A Modern Introduction to Programming

Encyclopedia of Microcomputers

A Functional Programming Approach

A Seamless Approach to Parallel Algorithms and their Implementation

A Functional Start to Computing with Python enables students to quickly learn computing without having to use loops, variables, and object abstractions at the start. Requiring no prior programming experience, the book draws on Python's flexible data types and operations as well as its capacity for defining new functions. Along with the specifics of Python, the text covers important concepts of computing, including software engineering motivation, algorithms behind syntax rules, advanced functional programming ideas, and, briefly, finite state machines. Taking a student-friendly, interactive approach to teach computing, the book addresses more difficult concepts and abstractions later in the text. The author presents ample explanations of data types, operators, and expressions. He also describes comprehensions—the powerful specifications of lists and dictionaries—before introducing loops and variables. This approach helps students better understand assignment syntax and iteration by giving them a mental model of sophisticated data first. Web Resource The book's supplementary website at http://functionalfirstpython.com/ provides many ancillaries, including: Interactive flashcards on

Python language elements Links to extra support for each chapter Unit testing and programming exercises An interactive Python stepper tool Chapter-by-chapter points Material for lectures

Even experienced developers struggle with software systems that sprawl across distributed servers and APIs, are filled with redundant code, and are difficult to reliably test and modify. Grokking Simplicity is a friendly, practical guide that will change the way you approach software design and development. Even experienced developers struggle with software systems that sprawl across distributed servers and APIs, are filled with redundant code, and are difficult to reliably test and modify. Grokking Simplicity is a friendly, practical guide that will change the way you approach software design and development. Grokking Simplicity guides you to a crystal-clear understanding of why certain features of modern software are so prone to complexity and introduces you to the functional techniques you can use to simplify these systems so that they're easier to read, test, and debug. Through hands-on examples, exercises, and numerous self-assessments, you'll learn to organize your code for maximum reusability and internalize methods to keep unwanted complexity out of your codebase. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications.

Functional C teaches how to program in C, assuming that the student has already learnt how to formulate algorithms in a functional style. By using this as a starting point, the student will become a better C programmer, capable of writing programs that are easier to comprehend, maintain and that avoid common errors and pitfalls. All program code that appears in Functional C is available on our ftp server - see below. How to find a code fragment? To access a particular code fragment, use the book to locate the section or subsection in which the code fragment appears, then click on that section in the code index . This will open the appropriate page at the beginning of the section. The code fragment may then be selected using the copy/paste facilities of your browser. Each chapter is represented by a separate page, so as an alternative to the procedure above you can use the save-as menu of your browser to up-load all code fragments in a particular chapter at once. Also available on our ftp server is errata for Functional C.

Richard Bird takes a radically new approach to algorithm design, namely, design by calculation. These 30 short chapters each deal with a particular programming problem drawn from sources as diverse as games and puzzles, intriguing combinatorial tasks, and more familiar areas such as data compression and string matching. Each pearl starts with the statement of the problem expressed using the functional programming language Haskell, a powerful yet succinct language for capturing algorithmic ideas clearly and simply. The novel aspect of the book is that each solution is calculated from an initial formulation of the problem in Haskell by appealing to the laws of functional programming. Pearls of Functional Algorithm Design will appeal to the aspiring functional programmer, students and teachers interested in the principles of algorithm design, and anyone seeking to master the techniques of reasoning about programs in an equational style.

Get more out of your legacy systems: more performance, functionality, reliability, and manageability Is your code easy to change? Can you get nearly instantaneous feedback when you do change it? Do you understand it? If the answer to any of these questions is no, you have legacy code, and it is draining time and money away from your development efforts. In this book, Michael Feathers offers start-to-finish strategies for working more effectively with large, untested legacy code bases. This book draws on material Michael created for his renowned Object Mentor seminars: techniques Michael has used in mentoring to help hundreds of developers, technical managers, and testers bring their legacy systems under control. The topics covered include Understanding the mechanics of software change: adding features, fixing bugs, improving design, optimizing performance Getting legacy code into a test harness Writing tests that protect you against introducing new problems Techniques that can be used with any language or platform—with examples in Java, C++, C, and C# Accurately identifying where code changes need to be made Coping with legacy systems that aren't object-oriented Handling applications that don't seem to have any structure This book also includes a catalog of twenty-four dependency-breaking techniques that help you work with program elements in isolation and make safer changes.

An Introduction to Functional Programming Through Lambda Calculus

A Functional Start to Computing with Python

Grokking Simplicity

PHP 7 Data Structures and Algorithms

A Practitioner's Approach with Emphasis on Functional Programming

Data Structures and Algorithms with Scala

Fourteen papers from the 1999 Stirling Workshop highlight major research goals and engineering concerns in the field. These include: making profitable use of modern parallel architectures, designing and defining modern type systems, performance comparisons between different functional languages, and applying functional programming languages. Specific chapters discuss cloning in a fuzzy language, transformation and optimization, genetic algorithms, GpH and Eden, parallel heuristics search in Haskell, operational semantics, graph- reduction semantics, CAMLFLOW, quilting, and type inference for MLj. Author index only. Distributed by ISBS. c. Book News Inc.

Well-respected text for computer science students provides an accessible introduction to functional programming. Cogent examples illuminate the central ideas, and numerous exercises offer reinforcement. Includes solutions. 1989 edition.

This volume constitutes the proceedings of the Third International Conference on the Mathematics of Program Construction, held at Kloster Irsee, Germany in July 1995. Besides five invited lectures by distinguished researchers there are presented 19 full revised papers selected from a total of 58 submissions. The general theme is the use of crisp, clear mathematics in the discovery and design of algorithms and in the development of corresponding software and hardware; among the topics

addressed are program transformation, program analysis, program verification, as well as convincing case studies.

This book explores the role of Martin-Lof s constructive type theory in computer programming. The main focus of the book is how the theory can be successfully applied in practice. Introductory sections provide the necessary background in logic, lambda calculus and constructive mathematics, and exercises and chapter summaries are included to reinforce understanding.

Helping readers to understand and learn how to use F#, this book covers basic algorithms and data structures using an innovative functional programming approach. The authors first use analogies of high school mathematics to cover code in order to make the code easy to understand. They then connect functional programming topics to important computer science areas and employ the popular .NET environment to give readers real-world skills and a solid programming platform. The text also includes an appendix on .NET programming using F# that contains additional information.

Functional C

Learning Functional Data Structures and Algorithms

Computational Semantics with Functional Programming

Functional Programming For Dummies

Learning Functional Programming in Go

Algorithms for Functional Programming

This book is a revised edition of the monograph which appeared under the same title in the series Research Notes in Theoretical Computer Science, Pit man, in 1986. In addition to a general effort to improve typography, English, and presentation, the main novelty of this second edition is the integration of some new material. Part of it is mine (mostly jointly with coauthors). Here is brief guide to these additions. I have augmented the account of categorical combinatory logic with a description of the confluence properties of rewriting systems of categor ical combinators (Hardin, Yokouchi), and of the newly developed cal culi of explicit substitutions (Abadi, Cardelli, Curien, Hardin, Levy, and Rios), which are similar in spirit to the categorical combinatory logic, but are closer to the syntax of A-calculus (Section 1.2). The study of the full abstraction problem for PCF and extensions of it has been enriched with a new full abstraction result: the model of sequential algorithms is fully abstract with respect to an extension of PCF with a control operator (Cartwright, Felleisen, Curien). An order extensional model of error-sensitive sequential algorithms is also fully abstract for a corresponding extension of PCF with a control operator and errors (Sections 2.6 and 4.1). I suggest that sequential algorithms lend themselves to a decomposition of the function spaces that leads to models of linear logic (Lamarche, Curien), and that connects sequentiality with games (Joyal, Blass, Abramsky) (Sections 2.1 and 2.6).

Computational semantics is the art and science of computing meaning in natural language. The meaning of a sentence is derived from the meanings of the individual words in it, and this process can be made so precise that it can be implemented on a computer. Designed for students of linguistics, computer science, logic and philosophy, this comprehensive text shows how to compute meaning using the functional programming language Haskell. It deals with both denotational meaning (where meaning comes from knowing the conditions of truth in situations), and operational meaning (where meaning is an instruction for performing cognitive action). Including a discussion of recent developments in logic, it will be invaluable to linguistics students wanting to apply logic to their studies, logic students wishing to learn how their subject can be applied to linguistics, and functional programmers interested in natural language processing as a new application area.

Get up to speed on Scala, the JVM language that offers all the benefits of a modern object model, functional programming, and an advanced type system. Packed with code examples, this comprehensive book shows you how to be productive with the language and ecosystem right away, and explains why Scala is ideal for today's highly scalable, data-centric applications that support concurrency and distribution. This second edition covers recent language features, with new chapters on pattern matching, comprehensions, and advanced functional programming. You'll also learn about Scala's command-line tools, third-party tools, libraries, and language-aware plugins for editors and IDEs. This book is ideal for beginning and advanced Scala developers alike. Program faster with Scala's succinct and flexible syntax Dive into basic and advanced functional programming (FP) techniques Build killer big-data apps, using Scala's functional combinators Use traits for mixin composition and pattern matching for data extraction Learn the sophisticated type system that combines FP and object-oriented programming concepts Explore Scala-specific

concurrency tools, including Akka Understand how to develop rich domain-specific languages Learn good design techniques for building scalable and robust Scala applications

Richard Bird is famed for the clarity and rigour of his writing. His new textbook, which introduces functional programming to students, emphasises fundamental techniques for reasoning mathematically about functional programs. By studying the underlying equational laws, the book enables students to apply calculational reasoning to their programs, both to understand their properties and to make them more efficient. The book has been designed to fit a first- or second-year undergraduate course and is a thorough overhaul and replacement of his earlier textbooks. It features case studies in Sudoku and pretty-printing, and over 100 carefully selected exercises with solutions. This engaging text will be welcomed by students and teachers alike.

Patterns and Skeletons for Parallel and Distributed Computing is a unique survey of research work in high-level parallel and distributed computing over the past ten years. Comprising contributions from the leading researchers in Europe and the US, it looks at interaction patterns and their role in parallel and distributed processing, and demonstrates for the first time the link between skeletons and design patterns. It focuses on computation and communication structures that are beyond simple message-passing or remote procedure calling, and also on pragmatic approaches that lead to practical design and programming methodologies with their associated compilers and tools. The book is divided into two parts which cover: skeletons-related material such as expressing and composing skeletons, formal transformation, cost modelling and languages, compilers and run-time systems for skeleton-based programming.- design patterns and other related concepts, applied to other areas such as real-time, embedded and distributed systems. It will be an essential reference for researchers undertaking new projects in this area, and will also provide useful background reading for advanced undergraduate and postgraduate courses on parallel or distributed system design.

Functional Programming in F#

Literate Programming

Grokking Functional Programming

Scalability = Functional Programming + Objects

Graph Algorithms in a Lazy Functional Programming Language

Change the way you approach your applications using functional programming in Go

This book constitutes the thoroughly refereed post-conference proceedings of the 13th International Conference on Relational and Algebraic Methods in Computer Science, RAMiCS 13, held in Cambridge, UK, in September 2012. The 23 revised full papers presented were carefully selected from 39 submissions in the general area of relational and algebraic methods in computer science, adding special focus on formal methods for software engineering, logics of programs and links with neighboring disciplines. The papers are structured in specific fields on applications to software specification and correctness, mechanized reasoning in relational algebras, algebraic program derivation, theoretical foundations, relations and algorithms, and properties of specialized relations.

A student introduction to the design of algorithms for problem solving. Written from a functional programming perspective, the text should appeal to anyone studying algorithms. Included are end-of-chapter exercises and bibliographic references.

Ideal for learning or reference, this book explains the five main principles of algorithm design and their implementation in Haskell.

Literate programming is a programming methodology that combines a programming language with a documentation language, making programs more easily maintained than programs written only in a high-level language. A literate programmer is an essayist who writes programs for humans to understand. When programs are written in the recommended style they can be transformed into documents by a document compiler and into efficient code by an algebraic compiler. This anthology of essays includes Knuth's early papers on related topics such as structured programming as well as the Computer Journal article that launched literate programming. Many examples are given, including excerpts from the programs for TeX and METAFONT. The final essay is an example of CWEB, a system for literate programming in C and related languages. Index included.

This fast-moving tutorial introduces you to OCaml, an industrial-strength programming language designed for expressiveness, safety, and speed. Through the book’s many examples, you’ll quickly learn how OCaml stands out as a tool for writing fast, succinct, and readable systems code. Real World OCaml takes you through the concepts of the language at a brisk pace, and then helps you explore the tools and techniques that make OCaml an effective and practical tool. In the book’s third section, you’ll delve deep into the details of the compiler toolchain and OCaml’s simple and efficient runtime system. Learn the foundations of the language, such as higher-order functions, algebraic data types, and modules Explore advanced features such as functors, first-class modules, and objects Leverage Core, a comprehensive general-purpose standard

library for OCaml Design effective and reusable libraries, making the most of OCaml’s approach to abstraction and modularity Tackle practical programming problems from command-line parsing to asynchronous network programming Examine profiling and interactive debugging techniques with tools such as GNU gdb

Trends in Functional Programming

Categorical Combinators, Sequential Algorithms, and Functional Programming

Algorithm Design with Haskell

Mathematics of Program Construction

Parallel Computing Technologies

Thinking Functionally with Haskell

In this approach, laziness plays an essential role to build a cyclic data structure, a graph, and to implement iteration as streams. The resulting algorithm is not optimal on uniprocessors but, avoiding side effects, the algorithm suggests a promising, more general approach to multiprocessor solutions."

Learn functional data structures and algorithms for your applications and bring their benefits to your work now About This Book Moving from object-oriented programming to functional programming? This book will help you get started with functional programming. Easy-to-understand explanations of practical topics will help you get started with functional data structures. Illustrative diagrams to explain the algorithms in detail. Get hands-on practice of Scala to get the most out of functional programming. Who This Book Is For This book is for those who have some experience in functional programming languages. The data structures in this book are primarily written in Scala, however implementing the algorithms in other functional languages should be straight forward. What You Will Learn Learn to think in the functional paradigm Understand common data structures and the associated algorithms, as well as the context in which they are commonly used Take a look at the runtime and space complexities with the O notation See how ADTs are implemented in a functional setting Explore the basic theme of immutability and persistent data structures Find out how the internal algorithms are redesigned to exploit structural sharing, so that the persistent data structures perform well, avoiding needless copying. Get to know functional features like lazy evaluation and recursion used to implement efficient algorithms Gain Scala best practices and idioms In Detail Functional data structures have the power to improve the codebase of an application and improve efficiency. With the advent of functional programming and with powerful functional languages such as Scala, Clojure and Elixir becoming part of important enterprise applications, functional data structures have gained an important place in the developer toolkit. Immutability is a cornerstone of functional programming. Immutable and persistent data structures are thread safe by definition and hence very appealing for writing robust concurrent programs. How do we express traditional algorithms in functional setting? Won't we end up copying too much? Do we trade performance for versioned data structures? This book attempts to answer these questions by looking at functional implementations of traditional algorithms. It begins with a refresher and consolidation of what functional programming is all about. Next, you'll get to know about Lists, the work horse data type for most functional languages. We show what structural sharing means and how it helps to make immutable data structures efficient and practical. Scala is the primary implementation languages for most of the examples. At times, we also present Clojure snippets to illustrate the underlying fundamental theme. While writing code, we use ADTs (abstract data types). Stacks, Queues, Trees and Graphs are all familiar ADTs. You will see how these ADTs are implemented in a functional setting. We look at implementation techniques like amortization and lazy evaluation to ensure efficiency. By the end of the book, you will be able to write efficient functional data structures and algorithms for your applications. Style and approach Step-by-step topics will help you get started with functional programming. Learn by doing with hands-on code snippets that give you practical experience of the subject.

Function literals, Monads, Lazy evaluation, Currying, and more About This Book Write concise and maintainable code with streams and high-order functions Understand the benefits of currying your Golang functions Learn the most effective design patterns for functional programming and learn when to apply each of them Build distributed MapReduce solutions using Go Who This Book Is For This book is for Golang developers comfortable with OOP and interested in learning how to apply the functional paradigm to create robust and testable apps. Prior programming experience with Go would be helpful, but not mandatory. What You Will Learn Learn how to compose reliable applications using high-order functions Explore techniques to eliminate side-effects using FP techniques such as currying Use first-class functions to implement pure functions Understand how to implement a lambda expression in Go Compose a working application using the decorator pattern Create faster programs using lazy evaluation Use Go concurrency constructs to compose a functionality pipeline Understand category theory and what it has to do with FP In Detail Functional programming is a popular programming paradigm that is used to simplify many tasks and will help you write flexible and succinct code. It allows you to decompose your programs into smaller, highly reusable components, without applying conceptual restraints on how the software should be modularized. This book bridges the language gap for Golang developers by showing you how to create and consume functional constructs in Golang. The book is divided into four modules. The first module explains the functional style of programming; pure functional programming (FP), manipulating collections, and using high-order functions. In the second module, you will learn design patterns that you can use to build FP-style applications. In the next module, you will learn FP techniques that you can use to improve your API signatures, to increase performance, and to build better Cloud-native applications. The last module delves into the underpinnings of FP with an introduction to category theory for software developers to give you a real understanding of what pure functional programming is all about, along with applicable code examples. By the end of the book, you will be adept at building applications the functional way. Style and approach This book takes a pragmatic approach and shows you techniques to write better functional constructs in Golang. We'll also show you how use these concepts to build robust and testable apps.

Increase your productivity by implementing data structures About This Book Gain a complete understanding of data structures using a simple approach Analyze algorithms and learn when you should apply each solution Explore the true potential of functional data structures Who This Book Is For This book is for those who want to learn data structures and algorithms with PHP for better control over application-solution, efficiency, and optimization. A basic understanding of PHP data types, control structures, and other basic features is required What You Will Learn Gain a better understanding of PHP arrays as a basic data structure and their hidden power Grasp how to analyze algorithms and the Big O Notation Implement linked lists, double linked lists, stack, queues, and priority queues using PHP Work with sorting, searching, and recursive algorithms Make use of greedy, dynamic, and pattern matching algorithms Implement tree, heaps, and graph algorithms Apply PHP functional data structures and built-in data structures and algorithms In Detail PHP has always been the the go-to language for web based application development, but there are materials and resources you can refer to to see how it works. Data structures and algorithms help you to code and execute them effectively, cutting down on processing time significantly. If you want to explore data structures and algorithms in a practical way with real-life projects, then this book is for you. The book begins by introducing you to data structures and algorithms and how to solve a problem from beginning to end using them. Once you are well aware of the basics, it covers the core aspects like arrays, listed lists, stacks and queues. It will take you through several methods of finding efficient algorithms and show you which ones you should implement in each scenario. In addition to this, you will explore the possibilities of functional data structures using PHP and go through advanced algorithms and graphs as well as dynamic programming. By the end, you will be confident enough to tackle both basic and advanced data structures, understand how they work, and know when to use them in your day-to-day work Style and approach An easy-to-follow guide full of examples of implementation of data structures and real world examples to solve the problems faced. Each topic is first explained in general terms and then implemented using step by step explanation so that developers can understand each part of the discussion without any problem.
AlgorithmsA Functional Programming ApproachAddison Wesley
Tools for Better Concurrency, Abstraction, and Agility
Algorithms

Programming Scala
Functional programming for the masses
Pearls of Functional Algorithm Design

This book presents latest research developments in the area of functional programming. The contributions in this volume cover a wide range of topics from theory, formal aspects of functional programming, transformational and generic programming to type checking and designing new classes of data types.Not all papers in this book belong to the category of research papers. Also, the categories of project description (at the start of a project) and project evaluation (at the end of a project) papers are represented. Particular trends in this volume are: - software engineering techniques such as metrics and refactoring for high-level programming languages;- generation techniques for data type elements as well as for lambda expressions;- analysis techniques for resource consumption with the use of high-level programming languages for embedded systems;- widening and strengthening of the theoretical foundations.The TFP community (www.tifp.org) is dedicated to promoting new research directions related to the field of functional programming and to investigate the relationships of functional programming with other branches of computer science. It is designed to be a platform for novel and upcoming research

Taming Complex Software with Functional Thinking
Research Directions in Parallel Functional Programming
Eloquent JavaScript, 3rd Edition
The Functional Approach to Programming
Kotlin Cookbook