1. Record Nr. UNINA9911046566003321

Autore	Misra Jayadev
Titolo	Effective Theories in Programming Practice
Pubbl/distr/stampa	New York City : , : Association for Computing Machinery, , 2022 ©2022
ISBN	9781450399746 1450399746
Edizione	[1st ed.]
Descrizione fisica	1 online resource (562 p.)
Collana	ACM Bks.
Soggetti	Computer science Computer logic
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Nota di contenuto	Intro -- Effective Theories in Programming Practice -- Contents -- Preface -- Acknowledgment -- 1 Introduction -- 1.1 Motivation for This Book -- 1.2 Lessons from Programming Theory -- 1.2.1 Terminology: Algorithms versus Programs -- 1.2.2 Domain Knowledge -- 1.3 Formalism: The Good, the Bad and the Ugly -- 1.3.1 Coxeter's Rabbit -- 1.3.2 Why Use Formalism -- 2 Set Theory, Logic and Proofs -- 2.1 Set -- 2.1.1 Basic Concepts -- 2.1.2 Mathematical Objects Used in this Book -- 2.1.2.1 Tuple -- 2.1.2.2 Sequence -- 2.1.2.3 Character -- 2.1.2.4 String -- 2.1.2.5 Integer Interval -- 2.1.2.6 Array, Matrix -- 2.1.2.7 Tree -- 2.1.2.8 Function, Relation -- 2.1.3 Set Comprehension -- 2.1.4 Subset -- 2.1.5 Operations on Sets -- 2.1.5.1 Binary Operators: Commutativity, Associativity, Distributivity -- 2.1.5.2 Union -- 2.1.5.3 Intersection -- 2.1.5.4 Complement -- 2.1.5.5 Cartesian Product -- 2.1.5.6 Difference, Symmetric Difference -- 2.1.5.7 Operations on Sequences -- 2.1.6 Properties of Set Operations -- 2.1.6.1 Laws about Sets -- 2.1.6.2 Laws about Subsets -- 2.1.6.3 Venn Diagram -- 2.2 Function -- 2.2.1 Basic Concepts -- 2.2.1.1 Function Arity -- 2.2.1.2 Function Definition and Computability -- 2.2.1.3 A Taxonomy of Functions -- 2.2.1.4 Cantor-Schröder-Bernstein Theorem -- 2.2.2 Function Composition -- 2.2.3 Function Inverse -- 2.2.4 One-way Function -- 2.2.5 Monotonic Function -- 2.2.6 Higher-order

Function -- 2.2.7 Pigeonhole Principle -- 2.2.7.1 Minimum Length of Monotone Subsequence -- 2.3 Relation -- 2.3.1 Basic Concepts -- 2.3.2 Relational Databases -- 2.3.3 Important Attributes of Binary Relations -- 2.3.4 Equivalence Relation -- 2.3.4.1 Checking for Equivalence -- 2.3.4.2 Bloom Filter -- 2.3.5 Closure of Relation -- 2.4 Order Relations: Total and Partial -- 2.4.1 Total Order -- 2.4.1.1 Sequence Comprehension Using a Total Order.
2.4.1.2 Lexicographic Order -- 2.4.1.3 An Example: Least-significant Digit Sort -- 2.4.2 Partial Order -- 2.4.2.1 Topological Order -- 2.4.2.2 Least Upper Bound, Greatest Lower Bound -- 2.4.2.3 Interval -- 2.4.2.4 Lattice -- 2.4.2.5 Monotonic Function over Partially Ordered Sets -- 2.5 Propositional Logic -- 2.5.1 Basic Concepts -- 2.5.2 Laws of Propositional Logic -- 2.5.3 Additional Aspects of Propositional Logic -- 2.5.3.1 Omitting Parentheses with Transitive and Associative Operators -- 2.5.3.2 Strengthening, Weakening Predicates -- 2.5.3.3 Relationship with Set Algebra -- 2.5.3.4 Boolean Algebra as a Commutative Ring -- 2.5.3.5 Functional Completeness of Boolean Operators -- 2.5.4 Satisfiability, Validity -- 2.5.4.1 Conjunctive and Disjunctive Normal Form -- 2.5.4.2 Resolution Principle -- 2.5.4.3 DPLL Algorithm -- 2.6 Predicate Calculus -- 2.6.1 Free and Bound Variables -- 2.6.2 Syntax of Quantified Expressions -- 2.6.3 Laws of Predicate Calculus -- 2.6.4 Duality Principle -- 2.6.5 Quantified Expressions over Non-Boolean Domains -- 2.7 Formal Proofs -- 2.7.1 Proof Strategies -- 2.7.2 A Style of Writing Proofs -- 2.8 Examples of Proof Construction -- 2.8.1 Proofs by Contradiction, Contrapositive -- 2.8.1.1 2 is Irrational -- 2.8.1.2 Pairing Points with Non-intersecting Lines -- 2.8.1.3 Euclid's Proof of Infinity of Primes, by Contradiction -- 2.8.2 Constructive and Non-constructive Existence Proofs -- 2.8.3 Sum and Product Puzzle -- 2.8.4 Russell's Paradox -- 2.8.5 Cantor's Diagonalization -- 2.8.6 Saddle Point -- 2.8.7 Properties of the Least Upper Bound of Partial Order -- 2.8.8 Knaster-Tarski Theorem -- 2.8.8.1 Proof of Theorem 2.1 (part 1) -- 2.8.8.2 Proof of Theorem 2.1 (part 2) -- 2.9 Exercises -- 3 Induction and Recursion -- 3.1 Introduction -- 3.1.1 Weak Induction Principle Over Natural Numbers.
3.1.2 Strong Induction Principle Over Natural Numbers -- 3.2 Examples of Proof by Induction -- 3.2.1 Examples from Arithmetic -- 3.2.1.1 Sum of Cubes -- 3.2.1.2 A Fundamental Theorem of Number Theory -- 3.2.1.3 Fibonacci Sequence -- 3.2.1.4 Harmonic Numbers -- 3.2.2 Examples About Games and Puzzles -- 3.2.2.1 Trimino Tiling -- 3.2.2.2 A Pebble Movement Game -- 3.3 Methodologies for Applying Induction -- 3.3.1 Applying the Base Step -- 3.3.2 Strengthening -- 3.3.3 Generalization -- 3.3.4 Problem Reformulation, Specialization -- 3.3.5 Proof by Contradiction versus Proof by Induction -- 3.3.6 Misapplication of Induction -- 3.4 Advanced Examples -- 3.4.1 Permutation Using Transposition -- 3.4.2 Arithmetic Mean Is At Least the Geometric Mean -- 3.4.2.1 A Proof Due to Hardy, Littlewood and Pólya -- 3.4.2.2 A Simple Proof Using Reformulation -- 3.4.3 Unique Prime Factorization -- 3.4.4 Termination of a Game -- 3.4.5 Hadamard Matrix -- 3.4.6 McCarthy's 91 Function -- 3.4.7 Topological Order of Partial Orders -- 3.4.7.1 Topological Order Over Finite Sets -- 3.4.7.2 Topological Order Over Countably Infinite Sets -- 3.4.8 König's Lemma -- 3.4.9 Winning Strategy in Finite 2-Player Game -- 3.5 Noetherian or Well-founded Induction -- 3.5.1 Well-founded Relation -- 3.5.1.1 Examples of Well-founded Relations -- 3.5.1.2 Lexicographic Order -- 3.5.1.3 Equivalence of Two Notions of Well-foundedness -- 3.5.1.4 Well-founded Induction Principle -- 3.5.1.5 Examples of Application of Well-founded Induction -- 3.5.2 Multiset or Bag Ordering -- 3.5.2.1 Dershowitz-Manna Order -- 3.5.2.2 Dershowitz-Manna Order is a

Well-founded Order -- 3.6 Structural Induction -- 3.6.1 Defining Recursive Structures -- 3.6.2 Structural Induction Principle -- 3.7 Exercises -- 4 Reasoning About Programs -- 4.1 Overview -- 4.2 Fundamental Ideas -- 4.2.1 Invariant -- 4.2.1.1 The Coffee Can Problem.
4.2.1.2 Chameleons -- 4.2.2 Termination -- 4.3 Formal Treatment of Partial Correctness -- 4.3.1 Specification of Partial Correctness -- 4.3.2 Hoare-triple or Contract -- 4.3.3 Auxiliary and Derived Variables -- 4.4 A Modest Programming Language -- 4.5 Proof Rules -- 4.5.1 Rule of Consequence -- 4.5.2 Simple Commands -- 4.5.3 Sequencing Command -- 4.5.4 Conditional Command -- 4.5.5 Loop Command -- 4.5.6 Non-deterministic Assignment -- 4.5.7 Procedure Call -- 4.5.8 Procedure Implementation -- 4.5.9 Program Annotation -- 4.6 More on Invariant -- 4.6.1 Heuristics for Postulating Invariant -- 4.6.2 Invariant Strength -- 4.6.3 Perpetual Truth -- 4.6.4 Proving Non-termination -- 4.6.4.1 Coloring Cells in a Square Grid -- 4.6.4.2 15-Puzzle -- 4.7 Formal Treatment of Termination -- 4.7.1 A Variation of the Coffee Can Problem -- 4.7.2 Chameleons Problem Revisited -- 4.7.3 Pairing Points with Non-intersecting Lines -- 4.7.4 A Problem on Matrices -- 4.8 Reasoning about Performance of Algorithms -- 4.9 Order of Functions -- 4.9.1 Function Hierarchy -- 4.9.2 Function Order -- 4.10 Recurrence Relations -- 4.10.1 Smaller Examples -- 4.10.1.1 Searching Sorted and Unsorted Lists -- 4.10.1.2 Merge-Sort -- 4.10.2 Solving Recurrence Relations -- 4.10.2.1 Interpolation -- 4.10.2.2 Expansion -- 4.10.2.3 Master Theorem -- 4.10.3 Divide and Conquer -- 4.10.3.1 Long Multiplication -- 4.10.3.2 Batcher Odd-Even Sort -- 4.10.3.3 Median-Finding -- 4.11 Proving Programs in Practice -- 4.12 Exercises -- 4.A Appendix: Proof of Theorem 4.1 -- 4.B Appendix: Termination in Chameleons Problem -- 5 Program Development -- 5.1 Binary Search -- 5.2 Saddleback Search -- 5.3 Dijkstra's Proof of the am-gm Inequality -- 5.4 Quicksort -- 5.4.1 Specification of quicksort -- 5.4.2 Procedure quicksort: Implementation and Proof -- 5.4.3 Procedure partition: Implementation and Proof.
5.4.4 Cost of quicksort -- 5.5 Heapsort -- 5.5.1 Outline of the Main Program -- 5.5.2 Heap Data Structure -- 5.5.3 extendHeap -- 5.5.3.1 Analysis of extendHeap -- 5.5.4 heapify -- 5.5.5 Running Time of Heapsort -- 5.6 Knuth-Morris-Pratt String-matching Algorithm -- 5.6.1 The String-matching Problem and Outline of the Algorithm -- 5.6.1.1 A Naive Algorithm -- 5.6.1.2 KMP Algorithm -- 5.6.2 Underlying Theory of the KMP Algorithm -- 5.6.2.1 Bifix and Core -- 5.6.2.2 A Characterization of Bifixes -- 5.6.3 Core Computation -- 5.6.3.1 Underlying Theorem for Core Computation -- 5.6.3.2 Abstract Program for Core Computation -- 5.6.3.3 Representation of Prefixes -- 5.6.3.4 Running Time -- 5.6.4 KMP Program -- 5.7 A Sieve Algorithm for Prime Numbers -- 5.7.1 Sieve of Eratosthenes -- 5.7.2 Correctness -- 5.7.2.1 Invariant -- 5.7.2.2 Verification Conditions -- 5.7.2.3 Termination -- 5.7.3 Characterization of Composite Numbers -- 5.7.4 Refinement of the Sieve Program -- 5.7.4.1 Initialization -- 5.7.4.2 Loop Iteration Condition -- 5.7.4.3 Removing Multiples of p, Updating p -- 5.7.4.4 Reestablishing Invariant J -- 5.7.5 Discussion -- 5.8 Stable Matching -- 5.8.1 Formal Description of the Problem and an Algorithm -- 5.8.2 Correctness -- 5.8.3 Refinement of the Algorithm -- 5.9 Heavy-hitters: A Streaming Algorithm -- 5.9.1 Problem Description -- 5.9.2 An Abstract Algorithm and its Refinement -- 5.9.2.1 An Abstract Algorithm -- 5.9.2.2 Refinement -- 5.9.3 One-pass Algorithm for Approximate Heavy-hitters -- 5.9.4 The Majority Element of a Bag -- 5.10 Exercises -- 6 Graph Algorithms -- 6.1 Introduction -- 6.2 Background -- 6.2.1 Directed and Undirected Graph -- 6.2.2 Paths and

Cycles -- 6.2.3 Connected Components -- 6.2.4 Labeled Graph -- 6.2.5 Graph Representation -- 6.2.6 Graph Manipulation: Merging, Pruning and Joining -- 6.2.6.1 Adding/Removing Nodes/Edges. 6.2.6.2 Merging Nodes.

Sommario/riassunto

Set theory, logic, discrete mathematics, and fundamental algorithms (along with their correctness and complexity analysis) will always remain useful for computing professionals and need to be understood by students who want to succeed. This textbook explains a number of those fundamental algorithms to programming students in a concise, yet precise, manner. The book includes the background material needed to understand the explanations and to develop such explanations for other algorithms. The author demonstrates that clarity and simplicity are achieved not by avoiding formalism, but by using it properly.The book is self-contained, assuming only a background in high school mathematics and elementary program writing skills. It does not assume familiarity with any specific programming language. Starting with basic concepts of sets, functions, relations, logic, and proof techniques including induction, the necessary mathematical framework for reasoning about the correctness, termination and efficiency of programs is introduced with examples at each stage. The book contains the systematic development, from appropriate theories, of a variety of fundamental algorithms related to search, sorting, matching, graph-related problems, recursive programming methodology and dynamic programming techniques, culminating in parallel recursive structures.