

# Data Structure and Algorithm Design

## COURSE PLAN

13-08-2024

# Data Structure and Algorithm Design

## Course details

13-08-2024

CS6103

# Course Outcomes

3

- ❑ Argue the correctness of algorithms using inductive proofs and invariants.
- ❑ Analyze worst-case running times of algorithms using asymptotic analysis.
- ❑ Describe the divide-and-conquer paradigm and explain when an algorithmic design situation calls for it. Recite algorithms that employ this paradigm. Synthesize divide-and-conquer algorithms. Derive and solve recurrences describing the performance of divide-and-conquer algorithms.
- ❑ Describe the dynamic-programming paradigm and explain when an algorithmic design situation calls for it. Recite algorithms that employ this paradigm. Synthesize dynamic-programming algorithms, and analyze them.
- ❑ Describe the greedy paradigm and explain when an algorithmic design situation calls for it. Recite algorithms that employ this paradigm. Synthesize greedy algorithms, and analyze them.

# Course Outcomes

4

- Explain the major graph algorithms and their analyses. Employ graphs to model engineering problems, when appropriate. Synthesize new graph algorithms and algorithms that employ graph computations as key components, and analyze them.
- Explain the different ways to analyze randomized algorithms (expected running time, probability of error). Recite algorithms that employ randomization. Explain the difference between a randomized algorithm and an algorithm with probabilistic inputs.
- Analyze randomized algorithms. Employ indicator random variables and linearity of expectation to perform the analyses. Recite analyses of algorithms that employ this method of analysis.
- Explain what amortized running time is and what it is good for. Describe the different methods of amortized analysis (aggregate analysis, accounting, potential method). Perform amortized analysis.

# Course Outcomes

5

- Explain what competitive analysis is and to which situations it applies. Perform competitive analysis.
- Compare between different data structures. Pick an appropriate data structure for a design situation.
- Explain what an approximation algorithm is, and the benefit of using approximation algorithms. Be familiar with some approximation algorithms, including algorithms that are PTAS or FPTAS. Analyze the approximation factor of an algorithm.
- Application of heuristic and metaheuristic techniques to design algorithm
- Introduction to Machine learning algorithms

# CS6103: Data Structure and Algorithm Design

6

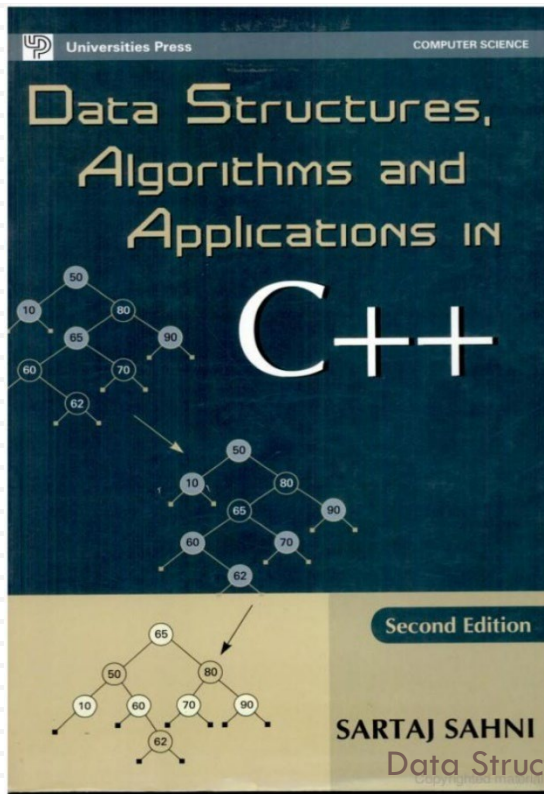
- **Introduction:** introduction to the ideas of specification, correctness, and analysis of algorithms, proving algorithm correctness, analyzing algorithms asymptotic analysis and amortized analysis, analyzing the worst-case performance of algorithms.
- **Problem-solving:** The art of problem-solving, problem solving, and decision making, Basic algorithmic structures as an approach to "automatic" problem solving, towards good algorithm design, the role of data structure in algorithm design. The general structure of an optimization algorithm, constraints, solution space, and feasible reasons, and representation of solution space.
- **NP-completeness:** The Class P, The Class NP, NP-complete Problems The satisfiability problem, vertex cover, independent set and clique problems, More NP-complete Problems, The Class co-NP, The Class NPI, The Relationships Between the Four Classes.
- **Lower bound techniques:** Introduction, Trivial Lower Bounds, The Decision Tree Model The search problem, The sorting problem, The Algebraic Decision Tree Model The element uniqueness problem, Linear Time Reductions The convex hull problem, The closest pair problem, The Euclidean minimum spanning tree problem.
- **Hashing:** The symbol table abstract data type Static Hashing, Dynamic Hashing, Hashing algorithms and application of hashing algorithm.
- **Heap Structures:** Min-max heaps, Deaps, Leftist Trees, Binomial Heaps, Fibonacci Heaps.

# CS6103: Data Structure and Algorithm Design

7

- **Randomized Algorithms:** Introduction, Las Vegas and Monte Carlo Algorithms, Randomized Quicksort, Randomized Selection, Testing String Equality, Pattern matching, Random Sampling, Primality Testing.
- **String matching:** String matching problem, solving real world problem with string matching, String matching algorithm: Naive Algorithm, KMP (Knuth Morris Pratt) Algorithm, Boyer Moore Algorithm, multiple pattern matching algorithm: Rabin–Karp algorithm, application of string matching algorithm.
- **Approximation Algorithms:** Introduction, Basic Definitions, Difference Bounds Planar graph coloring, Hardness results in the knapsack problem, Relative Performance Bounds The bin packing problem, The Euclidean traveling salesman problem, the vertex cover problem, Hardness result in the traveling salesman problem, Polynomial Approximation Schemes The knapsack problem, Fully Polynomial Approximation Schemes, the subset-sum problem.
- **Heuristics Algorithms:** Differentiate between Heuristics and metaheuristic algorithms, Optimization algorithms like Genetic optimization, Particle Swarm Optimization, and Ant colony optimization

# Advanced Data Structures





<b>CHAPTER 8</b>	<b>HASHING</b>	<b>464</b>
8.1	The Symbol Table Abstract Data Type	464
8.2	Static Hashing	466
8.2.1	Hash Tables	466
8.2.2	Hashing Functions	468
8.2.3	Overflow Handling	471
8.2.4	Theoretical Evaluation of Overflow Techniques	478
8.3	Dynamic Hashing	482
8.3.1	Motivation for Dynamic Hashing	482
8.3.2	Dynamic Hashing using Directories	483
8.3.3	Analysis of Directory-Based Dynamic Hashing	489
8.3.4	Directoryless Dynamic Hashing	491
8.4	References and Selected Readings	496

<b>CHAPTER 9</b>	<b>HEAP STRUCTURES</b>	<b>497</b>
9.1	Min-Max Heaps	497
9.1.1	Definitions	497
9.1.2	Insertion into a Min-Max Heap	499
9.1.3	Deletion of the Min Element	502
9.2	Deaps	507
9.2.1	Definition	507
9.2.2	Insertion into a Deap	508
9.2.3	Deletion of the Min Element	511
9.3	Leftist Trees	515
9.4	Binomial Heaps	522
9.4.1	Cost Amortization	522
9.4.2	Definition of Binomial Heaps	523

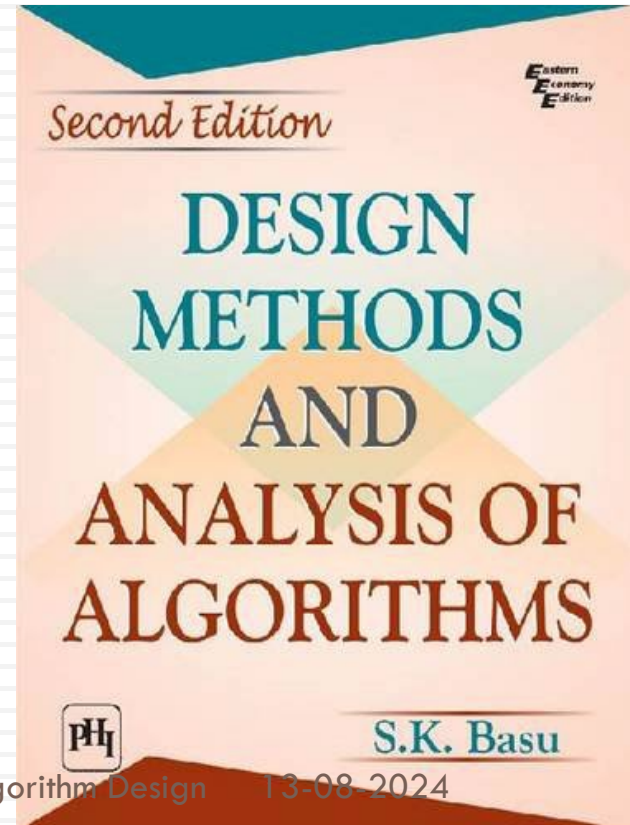
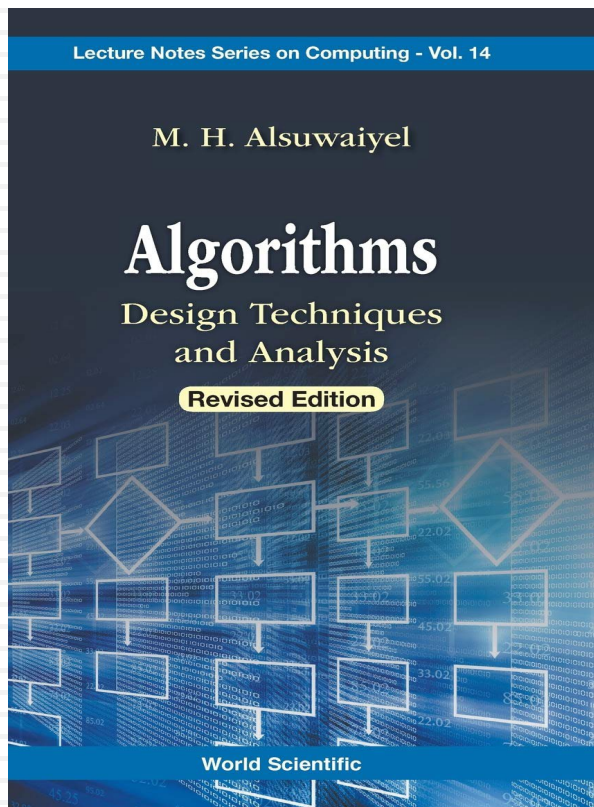
9.4.3	Insertion into a Binomial Heap	524
9.4.4	Combining Two Binomial Heaps	524
9.4.5	Deletion of Min Element	525
9.4.6	Analysis	529
9.5	Fibonacci Heaps	531
9.5.1	Definition	531
9.5.2	Deletion from an F-heap	532
9.5.3	Decrease Key	532
9.5.4	Cascading Cut	533
9.5.5	Analysis	534
9.5.6	Application to The Shortest Paths Problem	536
9.6	References and Selected Readings	539
9.7	Additional Exercise	539

# Suggestions

12

- Implementation of all the Data structure
- C++, C++(STL) and python

# Algorithm Design



# Data Structure and Algorithm Design

## COURSE PLAN

# Introduction to Algorithm Design

15

- **Introduction:** introduction to the ideas of specification, correctness, and analysis of algorithms, proving algorithm correctness, analyzing algorithms asymptotic analysis and amortized analysis, analyzing the worst-case performance of algorithms.
- **Problem-solving:** The art of problem-solving, problem solving, and decision making, Basic algorithmic structures as an approach to "automatic" problem solving, towards good algorithm design, the role of data structure in algorithm design. The general structure of an optimization algorithm, constraints, solution space, and feasible reasons, and representation of solution space.

# NP completeness

16

- The Class P
- The Class NP
- NP-complete Problems: The satisfiability problem, vertex cover, independent set and clique problems, More NP-complete Problems
- The Class co-NP
- The Class NPI
- The Relationships Between the Four Classes



# Lower bound techniques

17

- Introduction
- Trivial Lower Bounds
- The Decision Tree Model: The search problem, The sorting problem
- The Algebraic Decision Tree Model: The element uniqueness problem
- Linear Time Reductions: The convex hull problem, The closest pair problem, The Euclidean minimum spanning tree problem

# Randomized Algorithms

18

- Introduction
- Las Vegas and Monte Carlo Algorithms
- Randomized Quicksort
- Randomized Selection
- Testing String Equality
- Pattern matching
- Random Sampling
- Primality Testing

# Approximation Algorithms

19

- Introduction
- Basic Definitions
- Difference Bounds: Planar graph coloring, Hardness result: the knapsack problem
- Relative Performance Bounds: The bin packing problem, The Euclidean traveling salesman problem, the vertex cover problem, Hardness result: the traveling salesman problem
- Polynomial Approximation Schemes: The knapsack problem
- Fully Polynomial Approximation Schemes: The subset-sum problem

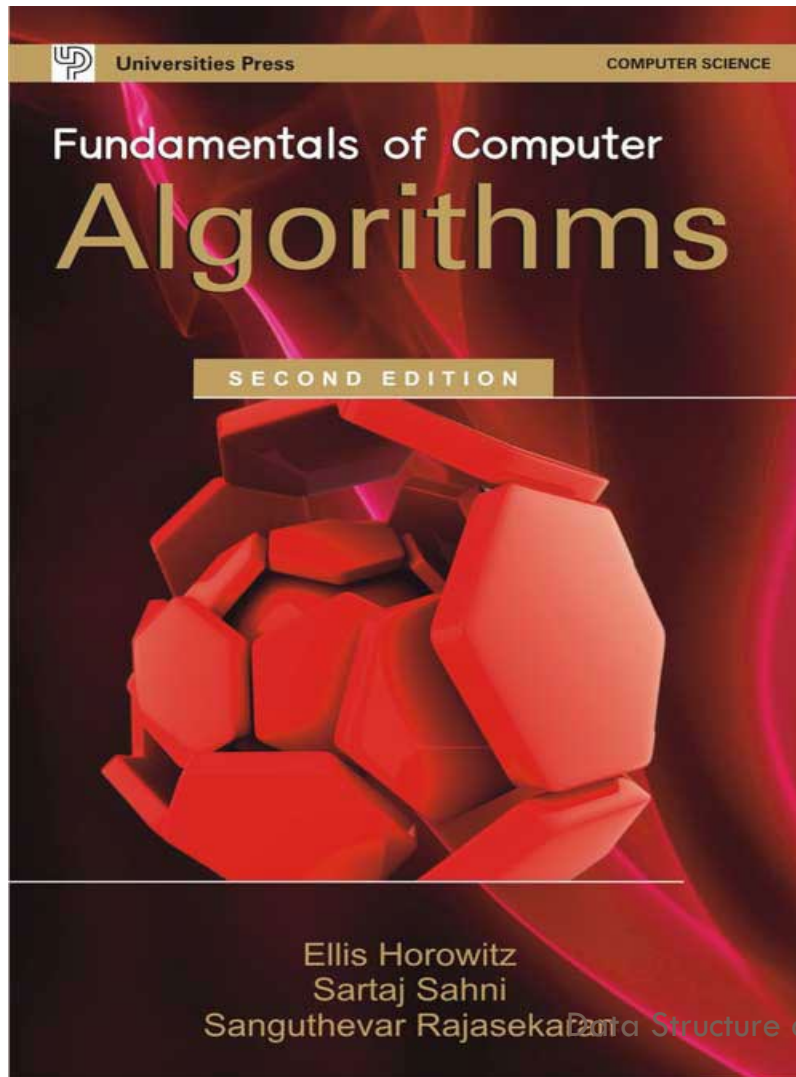
# Heuristics Algorithms

20

- Differentiate between Heuristics and metaheuristic algorithms.
- General structure of an optimization algorithms, constraints, solution space and the feasible reasons.
- Optimization algorithms like Genetic optimization, Particle Swarm optimization and Ant colony optimization

# Reference Book

21



## Fundamentals of computer Algorithms

ELLIS HOROWITZ

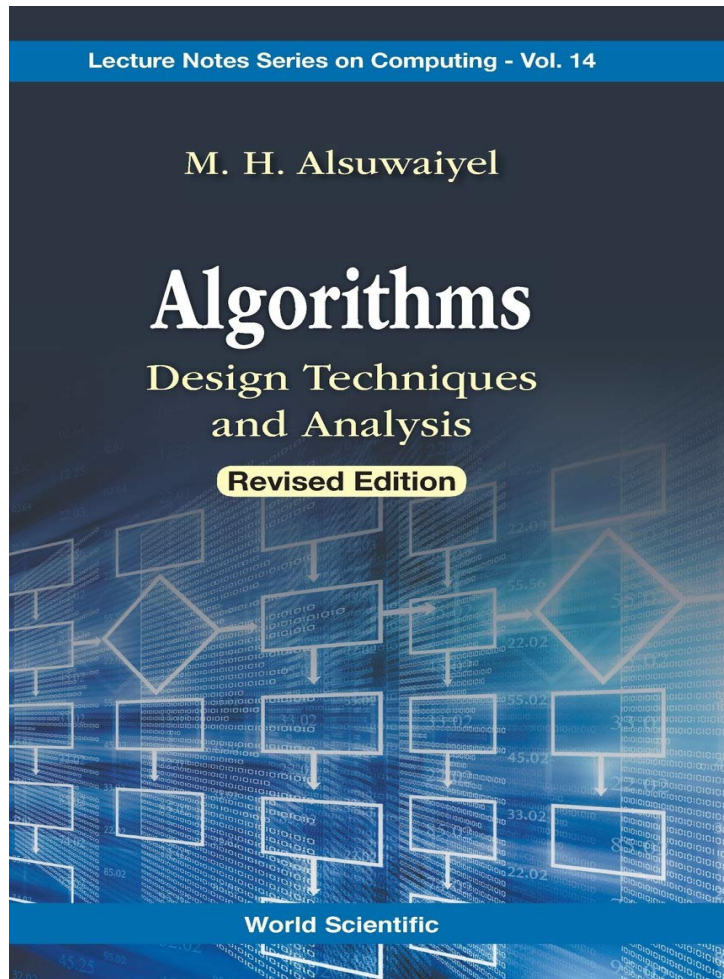
SARTAJ SAHNI

SANGUTHEVAR RAJASEKARAN.

□ **University Press**

# Reference Book

22



## □ Complexity of Problem

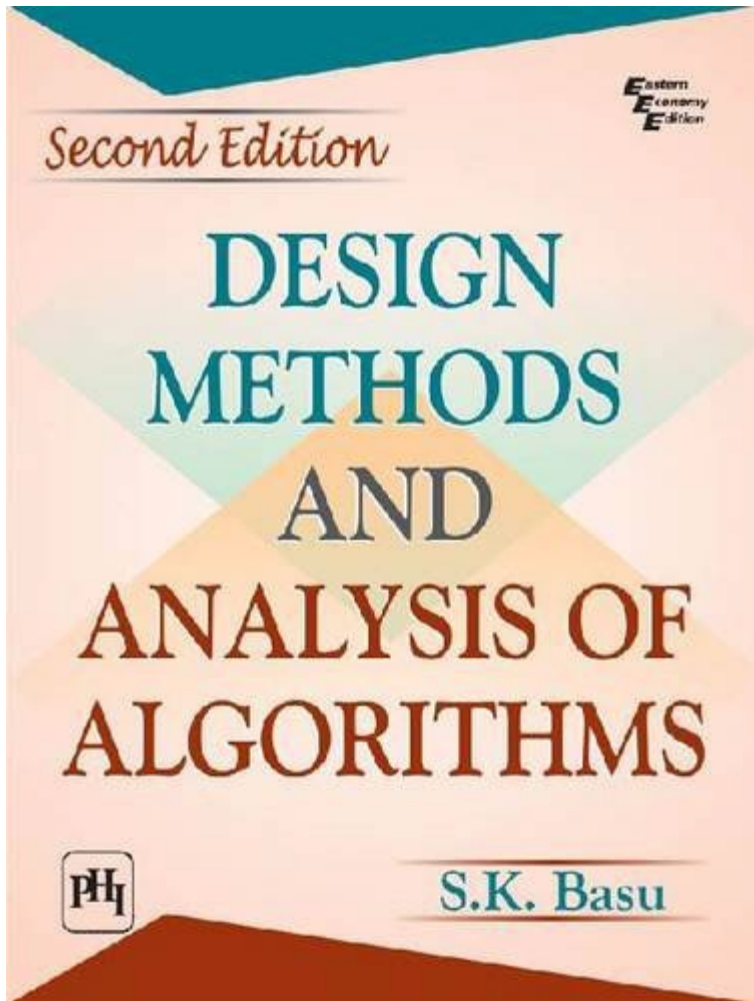
- ❖ NP-complete Problems
- ❖ Introduction to computational complexity
- ❖ Lower Bound

## □ Coping with Hardness

- ❖ Backtracking
- ❖ Randomized Algorithms
- ❖ Approximation Algorithms

# Reference Book

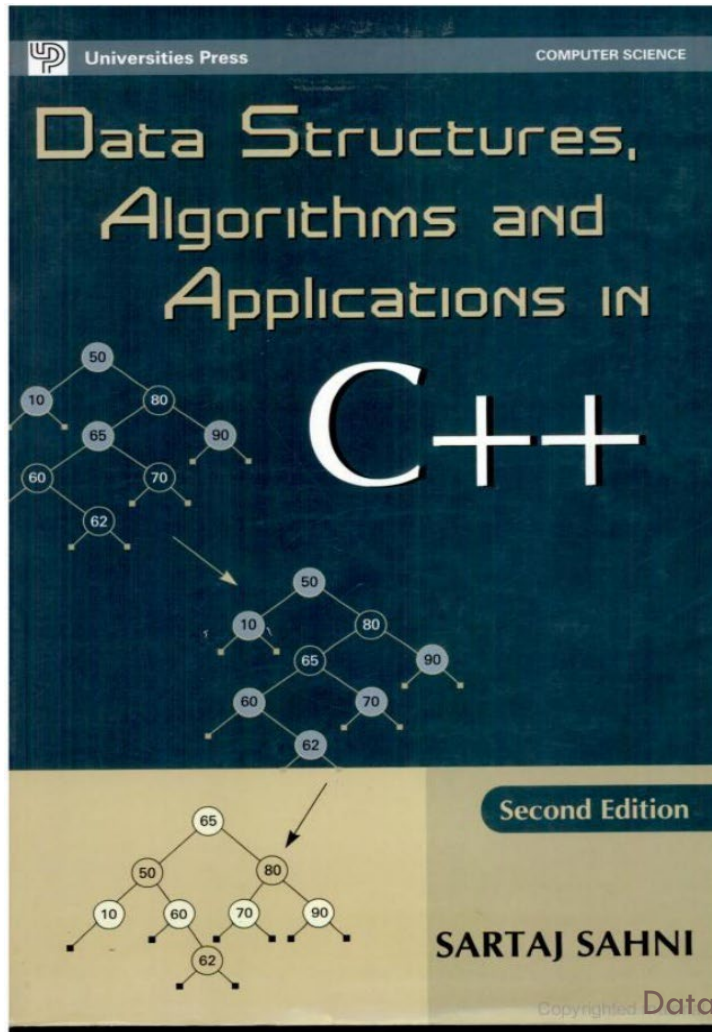
23



- ❖ Randomized Algorithms
- ❖ Approximation Algorithms
- ❖ Lower bound techniques
- ❖ NP completeness
- ❖ Genetic Algorithms

# Reference Book

24

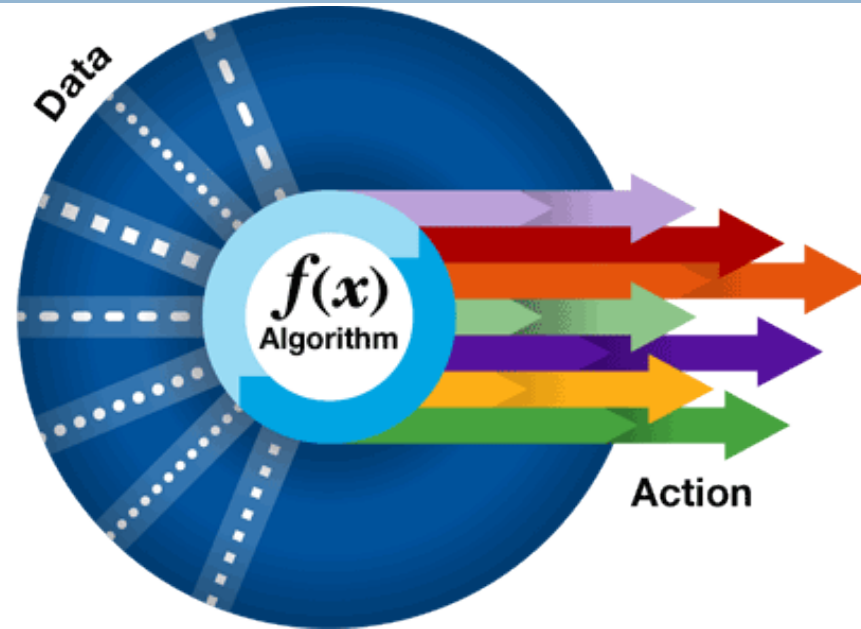


- Hashing
- Heap Structures



# Algorithmic business

25



**Algorithmic business** is the industrialized use of complex mathematical algorithms pivotal to driving improved business decisions or process automation for competitive differentiation.

<https://www.gartner.com/smarterwithgartner/five-keys-to-understanding-algorithmic-business>

26

**Prerequisites :**

**Algorithm Design Paradigm**

# Prerequisites :Algorithm Design Paradigms

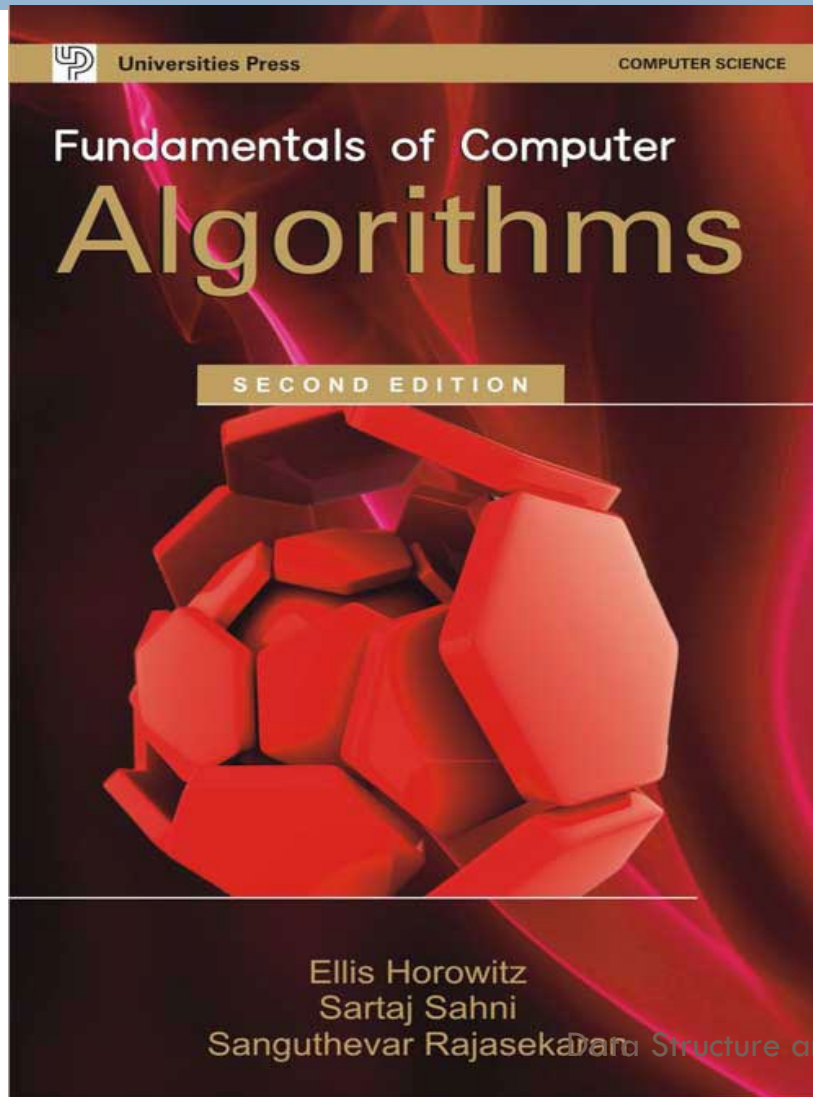
27

- **Algorithm Design Paradigms:** *General approaches to the construction of efficient solutions to problems.*
- *Such methods are of interest because:*
  - ▣ *They provide templates suited to solving a broad range of diverse problems.*
  - ▣ *They can be translated into common control and data structures provided by most high-level languages.*
  - ▣ *The temporal and spatial requirements of the algorithms which result can be precisely analyzed.*
- *Although more than one technique may be applicable to a specific problem, it is often the case that an algorithm constructed by one approach is clearly superior to equivalent solutions built using alternative techniques.*

# Algorithm Design Paradigms

28

- **Introduction**
- **Brute force**
- **Divide-and-conquer,**
  - ▣ **Decrease-and-conquer**
  - ▣ **Transform-and-conquer**
- **Greedy algorithms**
- **Dynamic programming**
- **Backtracking**
- **Branch-and-bound**
- **Genetic algorithms**
- **Conclusion**



- Chapter 1
- Chapter 3
- Chapter 4
- Chapter 5
- Chapter 6
- Chapter 7
- Chapter 8
- Chapter 10

# Divide and Conquer

30

<b>3</b>	<b>DIVIDE-AND-CONQUER</b>	<b>127</b>
3.1	GENERAL METHOD . . . . .	127
3.2	BINARY SEARCH . . . . .	131
3.3	FINDING THE MAXIMUM AND MINIMUM . . . . .	139
3.4	MERGE SORT . . . . .	145
3.5	QUICKSORT . . . . .	154
3.5.1	Performance Measurement . . . . .	159
3.5.2	Randomized Sorting Algorithms . . . . .	159
3.6	SELECTION . . . . .	165
3.6.1	A Worst-Case Optimal Algorithm . . . . .	169
3.6.2	Implementation of <code>Select2</code> . . . . .	172
3.7	STRASSEN'S MATRIX MULTIPLICATION . . . . .	179
3.8	CONVEX HULL . . . . .	183
3.8.1	Some Geometric Primitives . . . . .	184
3.8.2	The <code>QuickHull</code> Algorithm . . . . .	185
3.8.3	Graham's Scan . . . . .	187
3.8.4	An $O(n \log n)$ Divide-and-Conquer Algorithm . . . . .	188
3.9	REFERENCES AND READINGS . . . . .	193
3.10	ADDITIONAL EXERCISES . . . . .	194

<b>4</b>	<b>THE GREEDY METHOD</b>	<b>197</b>
4.1	THE GENERAL METHOD . . . . .	197
4.2	KNAPSACK PROBLEM . . . . .	198
4.3	TREE VERTEX SPLITTING . . . . .	203
4.4	JOB SEQUENCING WITH DEADLINES . . . . .	208
4.5	MINIMUM-COST SPANNING TREES . . . . .	216
4.5.1	Prim's Algorithm . . . . .	218
4.5.2	Kruskal's Algorithm . . . . .	220
4.5.3	An Optimal Randomized Algorithm (*) . . . . .	225
4.6	OPTIMAL STORAGE ON TAPES . . . . .	229
4.7	OPTIMAL MERGE PATTERNS . . . . .	234
4.8	SINGLE-SOURCE SHORTEST PATHS . . . . .	241
4.9	REFERENCES AND READINGS . . . . .	249
4.10	ADDITIONAL EXERCISES . . . . .	250

<b>5</b>	<b>DYNAMIC PROGRAMMING</b>	<b>253</b>
5.1	THE GENERAL METHOD . . . . .	253
5.2	MULTISTAGE GRAPHS . . . . .	257
5.3	ALL PAIRS SHORTEST PATHS . . . . .	265
5.4	SINGLE-SOURCE SHORTEST PATHS: . . . . .	
	GENERAL WEIGHTS . . . . .	270
5.5	OPTIMAL BINARY SEARCH TREES (*) . . . . .	275
5.6	STRING EDITING . . . . .	284
5.7	0/1-KNAPSACK . . . . .	287
5.8	RELIABILITY DESIGN . . . . .	295
5.9	THE TRAVELING SALESPERSON PROBLEM . . . . .	298
5.10	FLOW SHOP SCHEDULING . . . . .	301
5.11	REFERENCES AND READINGS . . . . .	307
5.12	ADDITIONAL EXERCISES . . . . .	308



<b>6</b>	<b>BASIC TRAVERSAL AND SEARCH TECHNIQUES</b>	<b>313</b>
6.1	TECHNIQUES FOR BINARY TREES . . . . .	313
6.2	TECHNIQUES FOR GRAPHS . . . . .	318
6.2.1	Breadth First Search and Traversal . . . . .	320
6.2.2	Depth First Search and Traversal . . . . .	323
6.3	CONNECTED COMPONENTS AND SPANNING TREES .	325
6.4	BICONNECTED COMPONENTS AND DFS . . . . .	329
6.5	REFERENCES AND READINGS . . . . .	338

<b>7</b>	<b>BACKTRACKING</b>	<b>339</b>
7.1	THE GENERAL METHOD . . . . .	339
7.2	THE 8-QUEENS PROBLEM . . . . .	353
7.3	SUM OF SUBSETS . . . . .	357
7.4	GRAPH COLORING . . . . .	360
7.5	HAMILTONIAN CYCLES . . . . .	364
7.6	KNAPSACK PROBLEM . . . . .	368
7.7	REFERENCES AND READINGS . . . . .	374
7.8	ADDITIONAL EXERCISES . . . . .	375

<b>8</b>	<b>BRANCH-AND-BOUND</b>	<b>379</b>
8.1	THE METHOD . . . . .	379
8.1.1	Least Cost (LC) Search . . . . .	380
8.1.2	The 15-puzzle: An Example . . . . .	382
8.1.3	Control Abstractions for LC-Search . . . . .	386
8.1.4	Bounding . . . . .	388
8.1.5	FIFO Branch-and-Bound . . . . .	391
8.1.6	LC Branch-and-Bound . . . . .	392
8.2	0/1 KNAPSACK PROBLEM . . . . .	393
8.2.1	LC Branch-and-Bound Solution . . . . .	394
8.2.2	FIFO Branch-and-Bound Solution . . . . .	397
8.3	TRAVELING SALESPERSON (*) . . . . .	403
8.4	EFFICIENCY CONSIDERATIONS . . . . .	412
8.5	REFERENCES AND READINGS . . . . .	416

<b>11 <math>\mathcal{NP}</math>-HARD AND <math>\mathcal{NP}</math>-COMPLETE PROBLEMS</b>	<b>495</b>
11.1 BASIC CONCEPTS . . . . .	495
11.1.1 Nondeterministic Algorithms . . . . .	496
11.1.2 The classes $\mathcal{NP}$ -hard and $\mathcal{NP}$ -complete . . . . .	504
11.2 COOK'S THEOREM (*) . . . . .	508
11.3 $\mathcal{NP}$ -HARD GRAPH PROBLEMS . . . . .	517
11.3.1 Clique Decision Problem (CDP) . . . . .	518
11.3.2 Node Cover Decision Problem . . . . .	519
11.3.3 Chromatic Number Decision Problem (CNDP) . . . . .	521
11.3.4 Directed Hamiltonian Cycle (DHC) (*) . . . . .	522
11.3.5 Traveling Salesperson Decision Problem (TSP) . . . . .	525
11.3.6 AND/OR Graph Decision Problem (AOG) . . . . .	526
11.4 $\mathcal{NP}$ -HARD SCHEDULING PROBLEMS . . . . .	533
11.4.1 Scheduling Identical Processors . . . . .	534
11.4.2 Flow Shop Scheduling . . . . .	536
11.4.3 Job Shop Scheduling . . . . .	538
11.5 $\mathcal{NP}$ -HARD CODE GENERATION PROBLEMS . . . . .	540
11.5.1 Code Generation With Common Subexpressions . . . . .	542
11.5.2 Implementing Parallel Assignment Instructions . . . . .	546
11.6 SOME SIMPLIFIED $\mathcal{NP}$ -HARD PROBLEMS . . . . .	550
11.7 REFERENCES AND READINGS . . . . .	553
11.8 ADDITIONAL EXERCISES . . . . .	553

**Let the sun light up  
your way to Success.**

*Good Luck.*

