Data Structure and Algorithm Design

COURSE PLAN

13-08-2024

CS6103

Data Structure and Algorithm Design

Data Structure and Algorithm Design

Course details

13-08-2024

CS6103

Course Outcomes

- 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

- 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

- 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

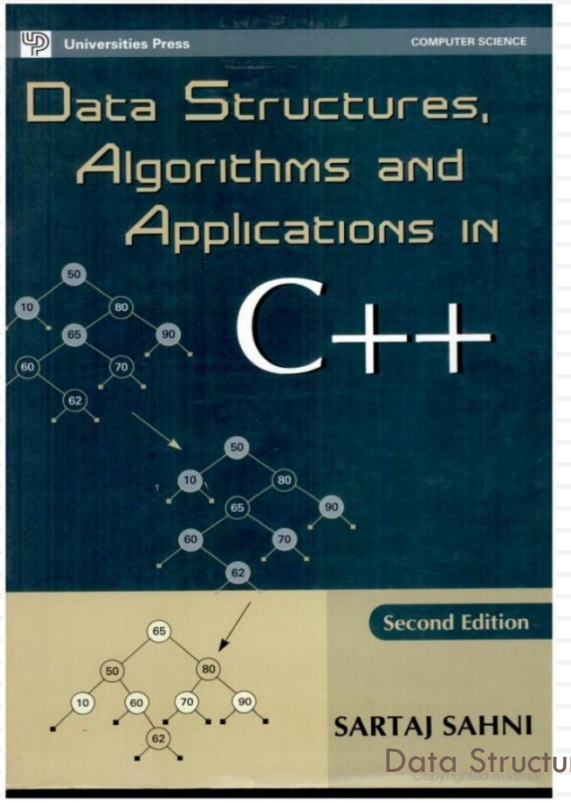
CS6103: Data Structure and Algorithm Design

- **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 Mode 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

- **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

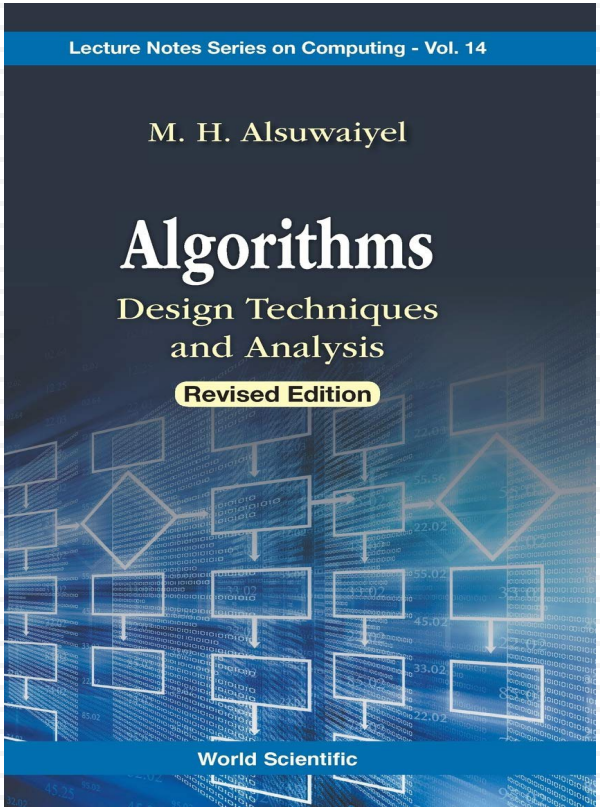
Advanced Data Structures

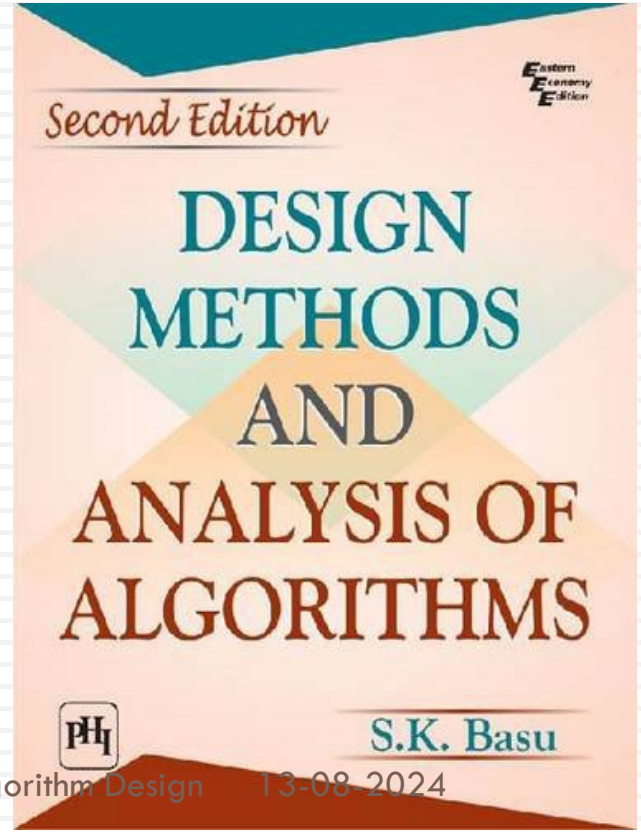


Suggestions

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

Algorithm Design





Data Structure and Algorithm Design

COURSE PLAN

Introduction to Algorithm Design

- **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 Mode: 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

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

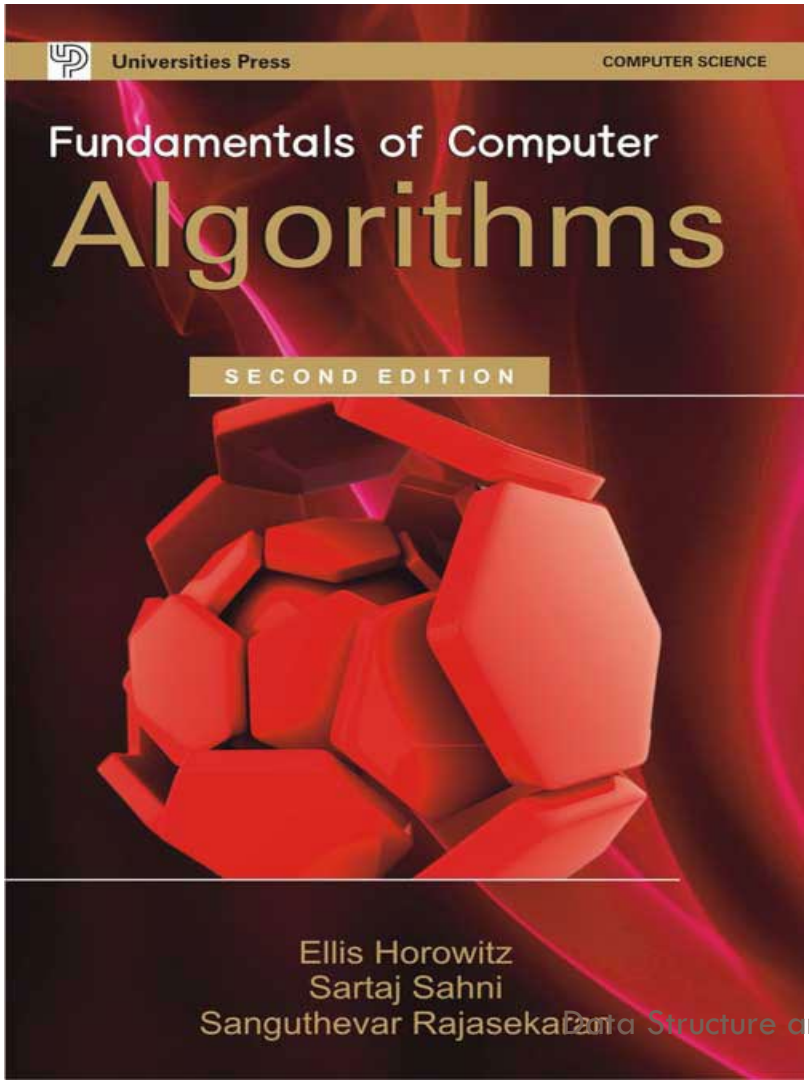
Approximation Algorithms

- 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

- 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



Fundamentals of computer Algorithms

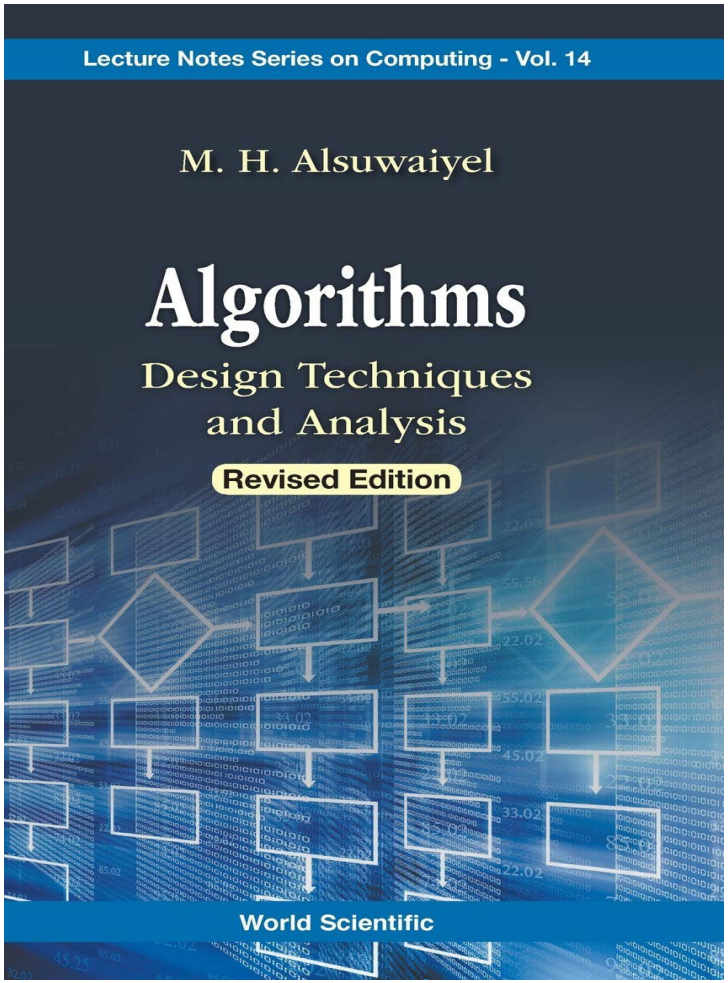
ELLIS HOROWITZ

SARTAJ SAHNI

SANGUTHEVAR RAJASEKARAN.

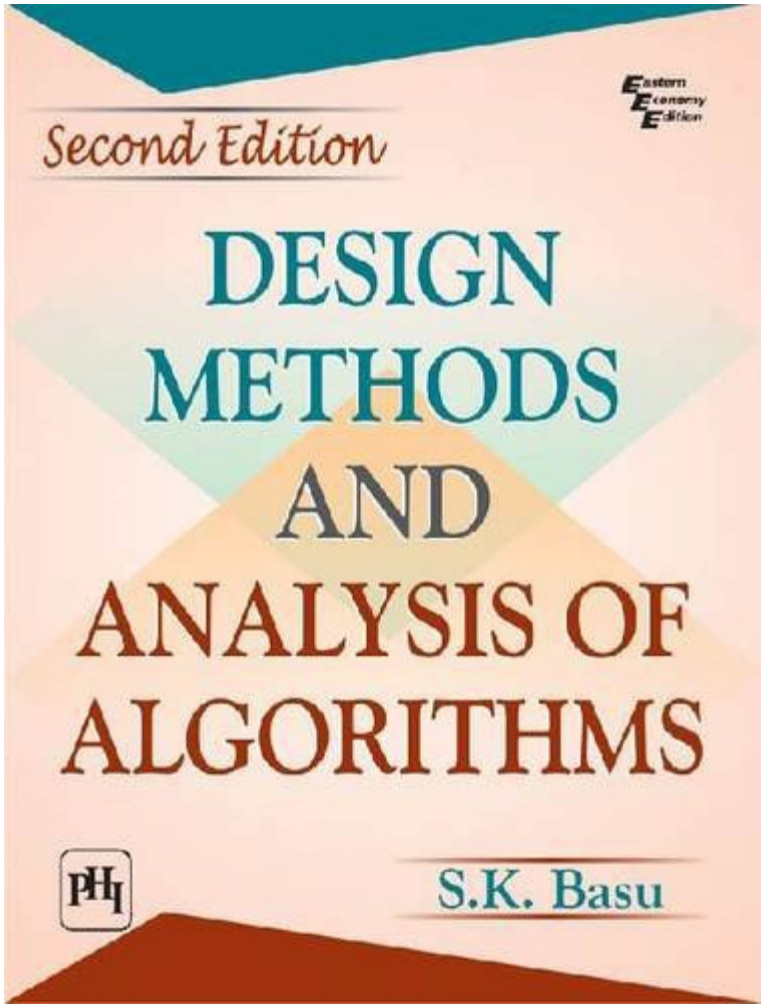
- **University Press**

Reference Book



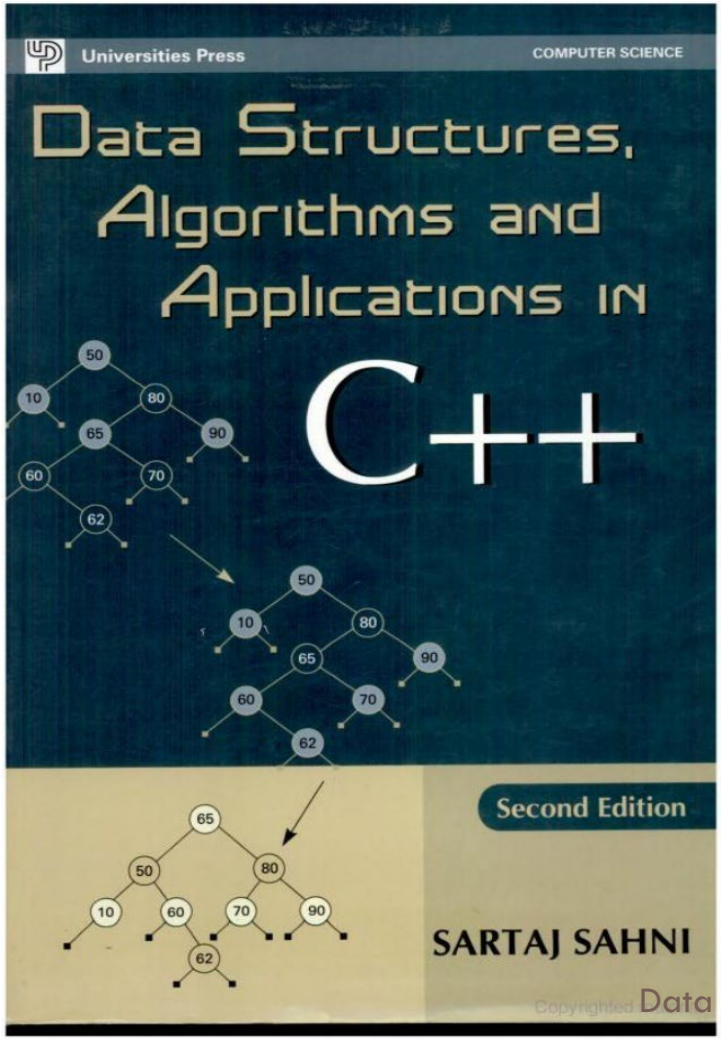
- **Complexity of Problem**
  - NP-complete Problems
  - Introduction to computational complexity
  - Lower Bound
- **Coping with Hardness**
  - Backtracking
  - Randomized Algorithms
  - Approximation Algorithms

Reference Book



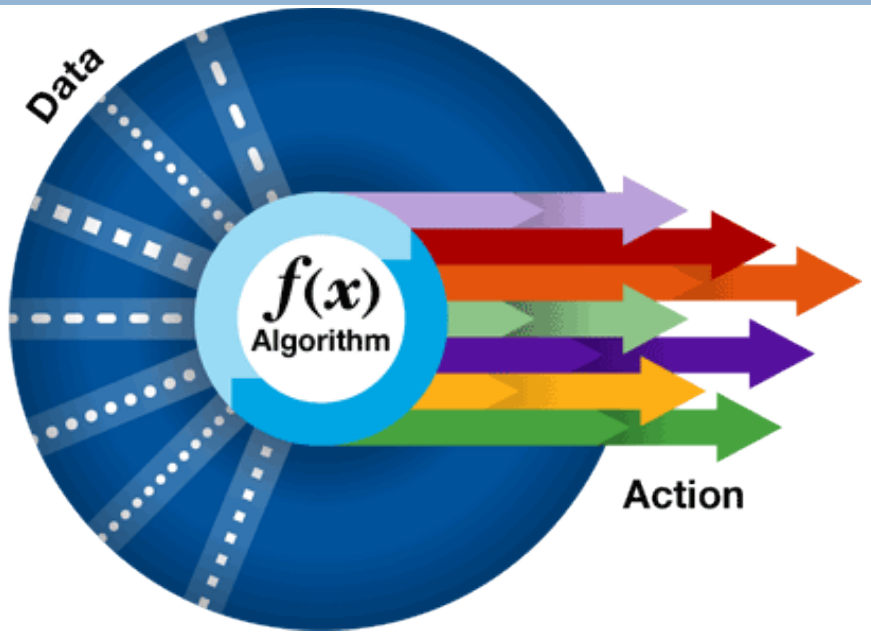
- Randomized Algorithms
- Approximation Algorithms
- Lower bound techniques
- NP completeness
- Genetic Algorithms

Reference Book



- Hashing
- Heap Structures

Algorithmic business



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

Prerequisites :

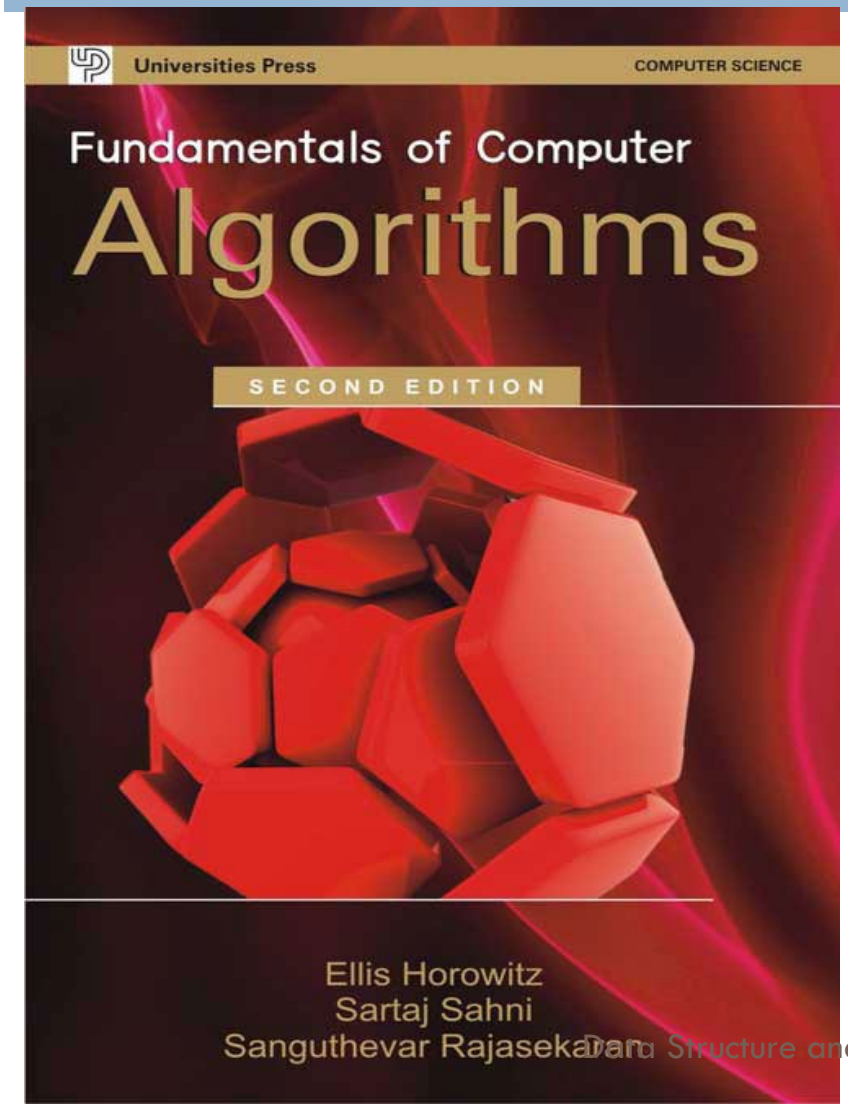
Algorithm Design Paradigm

Prerequisites :Algorithm Design Paradigms

- ***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

- 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


Let the sun light up
your way to Success.
Good Luck.
WishesMsg.com