Advanced Algorithms (CS6L093)

Books

1. **[DPV]** Algorithms - Sanjay Dasgupta, Christos Papadimitriou, Umesh Vazirani
2. **[KT]** Algorithm Design - Jon Kleinberg and Eva Tardos
3. **[CLRS]** Introduction to Algorithms (Third Edition) - Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein

---

Learning Outcomes

After completing the course, you will be able to:

1. Define formally a problem statement for a query in the real life scenario
2. Determine mathematical techniques required to solve the problem
3. Design algorithms for well defined problems
4. Analyze the correctness and efficiency of the algorithms
5. Apply different algorithm design approaches in problem specific manner

---

Evaluation (tentative plan)

Component 1: 4 problem sets : 4 X 5 → 20 marks

Component 2: Mid-Semester Examination → 30 marks

Component 3: End-Semester Examination → 50 marks

Week 1:

a) Introduction to Algorithms, Correctness and Efficiency of an algorithm, case study: Fibonacci Numbers and Insertion Sort.
b) Worst-case analysis of an algorithm
c) Models of computation - Word-RAM model
d) Asymptotic Analysis: Big O, Omega, Theta
e) Some examples of different upper bounds

Reference: Chapter 1 in DPV, Chapter 2 in KT, Chapter 2 (Section 2.1 and 2.2) in CLRS

Week 2:

Algorithm Design Technique 1: Divide and Conquer

a) Multiplying two n-bit numbers (Karatsuba and Ofman)
b) Proof of Master Theorem
c) Matrix Multiplication (Strassen)
d) A problem in computational geometry : Finding closest pair of points
e) Counting Inversions
f) Finding Maximum Sum Subarray

Reference: Chapter 2 in DPV, Chapter 4 in KT, Section 4.1 in CLRS

Week 3:

Proving Lower Bounds

a) Lower bound using Decision Tree model (Information Theoretic Argument) :
Case study: Comparison Sorting Algorithms, Searching a sorted sequence, Merging two sorted sequence
b) Limitation of Information Theoretic Argument and need for Adversary Argument
c) Lower bound using Reduction

Reference: Toniann Pitassi's **Notes** and Section 8.1 in CLRS

Week 4-5:

Graph Algorithms

a) s-t connectivity problem
b) Correctness proof of Breadth-First Search (BFS), BFS tree properties
c) Application of BFS: Testing Bipartiteness and Finding diameter in a tree
d) Depth-First Search (DFS) in directed graphs and DFS tree properties
e) Application of DFS in directed graphs: Detecting Directed Acyclic Graphs, Topological Sort, Finding Strongly Connected Components

Reference: Chapter 3 in KT, Chapter 22 in CLRS

Week 5-6:

Amortized Analysis

a) Aggregate Method
b) Accounting Method
c) Potential Method
d) Dynamic Tables (Expansion)
e) Dynamic Tables (Expansion and Contraction)

Reference: Chapter 17 in CLRS

Week 6-7:

Algorithm Design Technique 2: Greedy

a) Interval Scheduling Problem: optimality proof using " Greedy stays ahead" strategy
b) Interval Coloring Problem: optimality proof using structural lower bound argument
c) Interval Scheduling to minimize lateness: optimality proof using exchange argument
d) Recap of Kruskal's and Prim's algorithms - Cut-Property for proof of correctness
e) Reverse Delete algorithm - Cycle-Property for proof of correctness

Reference: Chapter 5 in KT

Week 7-8:

Matroid Theory

a) Introduction to Matroid Theory : Vector Matroid, Partition Matroid
b) Graphic Matroid for an Undirected Graph
c) Weighted Matroid and Minimum Spanning Tree problem as weighted graphic matroid
d) A GREEDY algorithm for Weighted Matroid
e) Proof that GREEDY is optimal

Reference: Section 16.4 in CLRS

Week 9-10: Mid-Semester exam and Autumn Semester Break

Week 11-12:

Algorithm Design Technique 3: Dynamic Programming

a) Weighted Interval Scheduling Problem :
   Top-Down approach: Recursion and Smart Recursion (Memoization)
   Bottom-Up approach: Iterative method
b) Principles of Dynamic Programming design
c) Longest Increasing Subsequence Problem
d) Subset Sums Problem
e) Knapsack Problem: With repetition and Without repetition
f) Rod Cutting Problem
g) Edit Distance Problem
h) Symmetric Traveling Salesman Problem
i) Maximum Independent Set in Trees

Reference: Sections 6.1, 6.2, 6.4 in KT and Section 6.2, 6.3, 6.4, 6.6, 6.7 in DPV, and 15.1 in CLRS

Week 13-14:

Algorithm Design Technique 4: Network Flow Algorithms

a) The Maximum Flow Problem, Residual Graphs, Augmenting Paths
b) Ford-Fulkerson algorithm, Running time analysis
c) Correctness of Ford-Fulkerson algorithm: Max-Flow Min-Cut theorem
d) Scaling Max Flow algorithm
e) Dinitz Edmond and Karp algorithm

Reference: Sections 7.1, 7.2, 7.3 in KT, Theorem 26.8 in CLRS

Week 15-16:

NP-completeness

a) Polynomial time reductions
    - Independent Set to Vertex Cover
    - Vertex Cover to Independent Set
    - Vertex Cover to Set Cover
    - Independent Set to Set Packing
    - 3-SAT to Independent Set
b) The class P and NP
c) NP-Complete problems

Reference: Sections 8.1, 8.2, 8.3 , and 8.4 in KT

Week 16-17:

Approximation Algorithms: How to cope with NP-completeness

a) 2-approximation algorithm for vertex cover
b) 2-approximation algorithm for Traveling Salesman problem with Triangle Inequality
c) 4 \sqrt{n} approximation for coloring 3-colorable graphs
d) If P is not equal to NP, then for any constant c >=1, there is no polynomial time approximation algorithm with approximation ratio c for general Traveling Salesman Problem.
e) 2-approximation algorithm for k-center problem
f) If P is not equal to NP, then for any constant 1 <= c < 2, there is no polynomial time approximation algorithm with approximation ratio c for k-center problem.

Reference: Sections 35.1, 35.2 in CLRS,
Section 6.5 and Section 2.2 from the book **The Design of Approximation Algorithms by David P. Williams and David B. Shmoys** (the book available online for free).