10 Review

CS216 Algorithm Design and Analysis (H)

Instructor: Shan Chen

chens3@sustech.edu.cn

Course Chapters

- Algorithm Analysis
- Stable Matching
- Greedy Algorithms
- Divide and Conquer
- Dynamic Programming
- Network Flow
- Computational Intractability
- Randomized Algorithms

Algorithm Analysis

- **Computational tractability:**
  - Worst-case/average-case analysis
  - Efficient = worst-case polynomial-time
- **Asymptotic order of growth:**
  - O, Ω, Θ definitions and their properties
  - Common running times: logarithmic, linear, linearithmic, quadratic, cubic, polynomial, exponential, factorial, etc.
- **Five representative problems on independent set:**
  - Interval scheduling, weighted interval scheduling, bipartite matching, independent set, competitive facility location.

Stable Matching

- **One-to-one stable matching:**
  - Example: marriage
  - Gale-Sharpley algorithm
  - Perfect stable matching
  - Man optimality vs woman optimality
- **One-to-many stable matching:**
  - Example: medical students applying for hospitals
  - Extended Gale-Sharpley algorithm
  - Stable matching
  - Student optimality vs hospital optimality

Greedy Algorithms

- **Greedy.** Build up solution myopically to optimize some underlying criterion.
- **Scheduling:**
  - Interval scheduling: greedy algorithm stays ahead
  - Interval partitioning: "structural" bound
  - Scheduling to minimize lateness: exchange argument
  - Optimal caching: LRU, LFU for online caching and FF for offline caching
- **Graphs and trees:**
  - Single-source/destination shortest paths: Dijkstra
  - Single-pair shortest path: *A** search algorithm
  - Minimum spanning trees and *k*-clustering: Prim, Kruskal, Borůvka, etc.
  - Min-cost arborescences: Chu-Liu's algorithm and its $O(m \log n)$ implementation
  - Optimal prefix codes (represented as binary trees): Huffman codes

Divide and Conquer

- **Divide and conquer:**
  - Divide up problem into several independent subproblems (of the same kind).
  - Solve (conquer) each subproblem recursively.
  - Combine solutions to subproblems into overall solution.
- **Applications:**
  - Merge sort and counting inversions
  - Randomized quick sort and randomized quick select
  - Closest pair of points
  - Integer and matrix multiplication
  - Convolution and Fast Fourier Transform (FFT)

Dynamic Programming

- **Dynamic Programming:**
  - Divide up problem into several overlapping subproblems and combine solutions to subproblems into overall solution.
  - Strategy: define subproblems, memorize intermediate results for later use, and order subproblems from "smallest" to "largest".
- **Techniques and applications:**
  - Binary choice: weighted interval scheduling
  - Multiway choice: segmented least squares
  - Adding a new variable: knapsack problem
  - Intervals: RNA secondary structure
  - DP + divide and conquer: Hirschberg's algorithm for sequence alignment
  - Graphs: SPFA, distance vector, negative cycle detection (and Tarjan's trick)

Network Flow

- **Theory and algorithms:**
  - Duality: max-flow value = min-cut capacity
  - Ford-Fulkerson algorithm: improve flow value with augmenting paths
  - More advanced algorithms: capacity-scaling, Edmonds-Karp, Dinitz.
  - Adding costs to max flow: min-cost max-flow algorithms
- **Applications and extensions:**
  - Bipartite matching (and min-cost perfect bipartite matching)
  - Disjoint paths
  - Circulation with supplies, demands, and lower bounds
  - Survey design
  - Airline scheduling
  - Image segmentation

Computational Intractability

- **Basic reduction strategies:**
  - Simple equivalence: INDEPENDENT-SET $\equiv_P$ VERTEX-COVER.
  - Special case to general case: VERTEX-COVER $\leq_P$ SET-COVER.
  - Encoding with gadgets: 3-SAT $\leq_P$ INDEPENDENT-SET.
- **Three types of problems:**
  - Decision problems vs search problems vs optimization problems
- **Important complexity classes and examples:**
  - **P**, **NP**, **NP**-complete, **NP**-hard
  - The first **NP**-complete problem: CIRCUIT-SAT
  - 3-SAT is **NP**-complete
  - Exploiting Intractability, e.g., RSA in cryptography

Randomized Algorithms

- **Why randomize?** Can lead to simplest, fastest, or only known algorithm for a particular problem.
- **Applications:**
  - Content resolution
  - Global min cut
  - Load balancing
  - MAX 3-SAT
- **Important math bounds for analysis:**
  - Union bound
  - Chernoff bounds
- **Two types of randomized algorithms.** Monte Carlo vs Las Vegas.

Good Luck!