CS401: Computer Algorithms I – Course Topics

Fall 2019: Tuesday-Thursday 3:30 pm - 4:45 pm, ARC 136

Abolfazl Asudeh, SEO1131, asudeh@uic.edu

- **Some representative problems**
  - The stable matching problem
  - Five representative problems
- **Basics of Algorithm Analysis (mostly review material)**
  - Comptutational tractability
  - Asymptotic order of growth
  - Common running times
- **Graphs (again, much review material)**
  - Basic definitions and applications
  - Graph traversal
  - Testing bipartiteness
  - Connectivity in directed graphs
  - DAGs and Topological ordering
- **Greedy Algorithms**
  - Interval scheduling
  - Scheduling to minimize lateness
  - Single-source shortest paths (Dijkstra's algorithm)
  - Minimum spanning tree (Prim's algorithm, Kruskal's algorithm, UnionFind data structure)
  - Clustering
- **Divide and Conquer**
  - Mergesort
  - Solving recurrence relations using the following methods: Unrolling the recurrence, substitution (guess and check), and annihilators.
  - Detour: Lower bound for comparison-based sorting algorithms.
  - Counting inversions
  - Closest pair of points
  - Integer multiplication
- **Dynamic Programming**
  - Weighted interval scheduling
  - Coin changing
  - Segmented least squares
  - Subset sum problem
  - Sequence alignment
  - Shortest paths
- **Network Flows**
  - The Maximum-Flow problem and Ford-Fulkerson algorithm
  - Maximum flows and minimum cuts
  - Improving Ford-Fulkerson by choosing good augmenting paths
  - Bipartite matching
  - Disjoint paths
  - Extensions to Max Flow
  - Project selection
- **Computational Intractability**
  - The complexity classes P, NP, EXP, and the importance of the P vs NP question
  - Polynomial-time reductions (Hamiltonian Path, Vertex Cover, Independent Set, Clique, Subset Sum)
  - A review of Turing Machines and the Church-Turing Thesis
  - The Cook-Levin Theorem (SAT is NP-complete) and its proof
  - An introduction to the PCP theorem
- **Independent learning via course project: Tractable special cases of NP-hard problems**
  - Solving NP-hard problems on trees
  - Solving NP-hard problems on graphs with bounded tree-width

subject to change without notice