CSCI 3110 Review Topics

• Definitions
  o little and big O, Omega, Theta
  o Design patterns
  o Data structures and basic algorithms
    • Arrays, linked lists, stacks, queues

• Fundamentals
  o Order of function growth
  o Using limits to prove small o
  o Using constants to Prove O, Omega, Theta

• Algorithm Analysis
  o What does an algorithm do?
  o Analyze the running time of an algorithm
    • Counting loops
    • Bounding the number of function calls
    • Recurrence relations
    • Amortized analysis
  o Prove correctness
    • Termination
    • Correctness
      ▪ Contradiction
      ▪ Induction
      ▪ Stay-ahead arguments
      ▪ Loop invariants

• Algorithm Design
  o Graph Algorithms
    • Graph exploration
    • Undirected/directed, adjacency list, etc
    • Proofs: Contradiction
    • BFS/DFS as building blocks
    • Problems
      ▪ Connected components
      ▪ Bipartiteness testing
      ▪ Topological Sorting
      ▪ Strongly Connected Components
  o Greedy Algorithms
    • Make progress toward a globally optimal solution by making locally optimal choices
    • Problems
      ▪ Interval Scheduling
      ▪ Minimum Spanning Tree
        • Kruskal
        • Prim
      ▪ Shortest Paths
        • Dijkstra
      ▪ Minimum-length codes
  • Techniques
- Induction
- Stay-ahead arguments
- Exchange arguments

- Data Structures
  - Priority Queue
    - Thin heap
  - Union-find data structure

- Divide and Conquer
  - Divide the problem into subproblems, recurse, and combine the solutions
  - Techniques
    - Induction
    - Recurrence Relations
  - Problems
    - Sorting
      - Merge Sort, Quick Sort
    - Selection
    - Matrix multiplication
    - Finding the closest pair

- Dynamic Programming
  - Recursively break the problem into smaller subproblems
  - Avoid repeatedly solving the same subproblems by caching their solutions
    - Memoization
    - Table
  - Techniques
    - Recurrence relations
  - Problems
    - Matrix chain multiplication
    - Weighted interval scheduling
    - Sequence alignment
    - Shortest paths

- Data Structures
  - Use data structures to implement non-trivial steps in algorithms
  - Augmenting data structures
    - add information to existing data structures so they support additional queries
  - Specific Data structures
    - (a,b)-trees
      - nodes have (a,b) children (root has (2,b) children)
      - leaves at same depth
      - insert, delete, find, rangefind, predecessor, successor, minimum, maximum
      - rebalance with node fusions and splits
      - logn time or logn +k time operations
    - Rank-select trees
      - rank and select queries
        - logn time
        - store number of descendant leaves at each node
    - priority search trees
      - 3-sided range reporting
      - (a,b) tree on x-coordinates
- add y-coordinate using max heap property
  - range trees
    - nested (a,b)-trees on x, y, z, etc coordinates
- Problems
  - orthogonal and general line segment intersection reporting and counting
    - sweep line
  - range reporting and counting
- Randomization
  - do the easy thing and hope it works for most inputs
  - make random choices and hope they are good
  - complicated analysis using statistics
  - expected running time - average running time over all possible inputs
- Problems
  - Sorting (quick sort)
    - randomize the input and use simple quicksort
    - randomize the pivot using randomized quicksort
  - Permuting
  - Selection
  - Game tree evaluation
- NP-hardness
  - Computational (in)tractability
  - Decision problems and optimization problems
  - Decision problems and formal languages
  - The class P
  - Decision and verification
  - The class NP
  - NP hardness and NP completeness
  - Polynomial-time reductions
  - NP-complete problems
    - Satisfiability
      - naturally NP-hard
    - Vertex Cover
      - 3-SAT -> vertex cover
    - Hamiltonian Cycle
      - Vertex Cover -> Hamiltonian Cycle
    - Subset sum
      - 3-SAT -> Subset sum