Review topics for final exam (Sections B and D)

Below is a list of topics covered in our section to help you review. You are responsible for material covered in the lectures in our section and for the material covered in the homeworks (excluding the material marked as being for Prof. Aronov's section only).

If there is material in the book that we did not cover in class or in the homework, you are NOT responsible for it.

You should read the solution sets even if you received full credit for your homework.

- Questions to ask about an algorithm:
  1. Does it work? (correctness)
  2. What is its running time?
  3. Can we do better?
- logarithms
- repeated doubling and repeated halving
- Fibonacci numbers
- proof by induction
- Asymptotic notation: big-Oh, theta, omega
- bit-cost model and unit cost model
- algorithms for standard arithmetic operations
  - addition, subtraction, multiplication, exponentiation in the bit-cost model
  - Multiplication a la francais
- equivalence modulo x
- modular arithmetic operations in the bit cost model
- Euclid's algorithm for GCD, Extended-Euclid
- modular division
- inverse modulo x
- Randomized algorithms
  - expected running time
  - probability of error

- density of primes
- Fermat's Little Theorem
- primality testing
- generating random prime numbers
- RSA

- Solving recurrences (various methods)
- Master Theorem

- Divide and conquer algorithms
- Binary search
- Mergesort
- Randomized Select
- Recursive division algorithm
- Linear-time deterministic selection
- Divide and Conquer algorithm for multiplying two n-bit numbers
- Strassen's matrix multiplication algorithm (you do NOT have to know the details of how XY is broken into the 7 subproblems)
- FFT

- Graph Types
  - Directed, Undirected
  - DAGs (Directed Acyclic Graphs)
  - Weighted (weights on edges)
- Adjacency List and Adjacency Matrix representations

- DFS
  - calls to Explore
  - Previsit and Postvisit numbers
  - Cross, Back, Forward, and Tree edges

  - DFS tree and DFS forest
  - Using DFS to count number of connected components in unweighted graph
  - Detecting a cycle in a directed graph using DFS (check for a back edge)
- DAGs
  - Sinks/Sources
  - DFS-based Algorithm for Topological sort of DAG
- Strongly Connected Components
  - Definition of a Strongly Connected Component
  - Strongly Connected Components algorithm
- Breadth First Search
  - Calculating single-source shortest path in unweighted graph
  - Breadth-first search tree
- Dijkstra's algorithm
  - Known region
  - Implementation with Min-Heap
  - Runtime Analysis
  - Update/Relax an edge
- Bellman Ford for graphs with negative weights, but no negative cycles
- DAG Shortest Path algorithm (single source)
- Minimum Spanning Tree
- Kruskal's algorithm
  - Disjoint Sets Data Structure
  - Union by Rank
- Prim's algorithm
  - Implementation of Prim's algorithm with Min-Heap
- The cut property
- Analysis of runtime of Prim's and Kruskal's

- Set Cover Problem
  - Greedy Approximation algorithm

- Dynamic Programming
- Edit Distance algorithm
  - alignment of two strings
- Knapsack without repetition
- All-pairs shortest paths:
- Floyd Warshall algorithm

- Linear Programming
- Objective Function
- Linear Constraints
- Feasible Region
- Vertices of feasible region

- NP-completeness
- Optimization problems
- What is a decision problem?
- Definition of NP
- SAT (satisfiability)
- Reduction
- Integer Linear Programming
- Definition of NP-completeness
- The million dollar question: Is P=NP?
- NP-Complete problems
  - Intuitively, hardest problems in NP
  - Finding a polynomial-time algorithm for just one NP-complete problem would yield polynomial-time algorithms for all of them and prove P=NP