CISC320 algorithms, 10S, midterm exam review notes

Chapter 0:
Definitions of estimation tools: big-O, Ω, Θ.
Key points:

- the inequalities between functions defining these must hold up to a positive constant factor. For instance, $f(n)$ is $O(g(n))$ requires $f(n) \leq c * g(n)$, for some positive c. Often the first challenge in proving a big-O relationship is guessing c, i.e. finding a c that will work.
- The inequalities may be wrong for a few small values of n. It is only required that they be true for all values of n *after some threshold*. For instance, n is $O(n * \log(n))$ even though when $n = 1$, we see that $n > c * n * \log(n)$, regardless of c. On the other hand – and what counts – for all $n \geq 2$, we see that $n \leq n * \log(n)$ is true because $\log(n) \geq 1$ for $n \geq 2$.

Chapter 1:
Topics:

- integer multiplication and division with remainder
- modular arithmetic
- testing for primality
- RSA encription

Theme: public key encryption (RSA in particular) is based on the fact that some things are hard (factoring the product of two large prime numbers) and some are tractable (basic integer arithmetic, modular arithmetic including exponentiation (modexp), determining if a number is prime) // //
Algorithms:

- multiply (p15)
- divide (quotient and remainder) (p15)
- modexp (p19)
- Euclid (p20)
- extended-Euclid (p21)
- primality (1.1) (p25)
- primality2 (1.1) (p27)
- RSA (build keys, encrypt, decrypt) (p34)

Chapter 2:

- Master theorem (applies to recurrences diving down to n/b) and Muster theorem (applies to recurrences stepping down to $n - b$)
- divide and conquer integer (also polynomial) multiplication (Karatsuba algorithm) (p47)
- divide and conquer sorting (merge sort) (p50)
- lower bound for sorting (p51)

- divide and conquer selection (randomized median) (section 2.4)
- divide and conquer matrix multiplication (Strassen's algorithm) (section 2.5)
- divide and conquer polynomial (also integer) multiplication (fast Fourier transform) (p68)

Themes: divide and conquer algorithms may be organized around applicable case of Master theorem (see algorithms list below).
Also notice that the tractable problems of chapter 1 can be done even faster than we first thought: integer, polynomial, and matrix multiplication, thus also modexp and primality are faster.

Algorithms:

- Case 1 of Master theorem: **binary search** (p50), **selection** (randomized median) (section 2.4).
- Case 2 of Master theorem: **merge sort** (p50), **FFT**, use of FFT (p68) in **polynomial multiplication** (p60)
- Case 3 of Master theorem: **integer multiplication** (Karatsuba algorithm) (p47), **Strassen's matrix multiplication** algorithm (section 2.5),

Chapter 3:

- Depth first search in (undirected) graphs and digraphs (directed graphs). Tree, forward, back, and cross edge categorization with respect to a depth first search.
- DAGs (directed acyclic graphs) and linearization
- strongly connected components and the DAG of strongly connected components.

Themes: basic graph representation and terminology, dfs is basis of solving some problems.

Chapter 4: Dijkstra's algorithm (assuming a priority queue) for single source shortest paths.

Overall: Things to know about each algorithm:

- Why is it correct (vis a vis it's input/output specification)? What are the theorems and properties used to explain it's workings?
- What measure, n, of it's input is used as basis for analysis (for instance, n = bound on number of bits in numbers, or n = size of an array)?
- What formula (function of that n) estimates it's runtime? Usually the formula is a recurrence relation.
- What is the solution of that formula/recurrence?

Kinds of questions: multiple choice, short answer, write algorithm, track algorithm.