Design and Analysis of Algorithms - Midterm Overview

You should be able to sufficiently justify why any algorithm or data structure operation has the time complexity it does.

Ch 1. Analysis of Algorithms

- Definition of O, Θ, Ω . I won't ask you about little-oh, little-omega.
- Show that $f(n)$ is $O(g(n))$
- Express and justify the runtime of an algorithm in Big-Oh notation
- Amortized analysis

Ch 2. Elementary Data Structures & Ch. 3 Search Trees

How each data structure is **implemented**, any characteristic **properties** or **definitions** of the data structure, what **operations** can be performed on it, and understand the **complexity** of those operations.

Data structures include:

- Stacks (including resizing and amortized analysis for stack operations)
- Queues
- Trees
- Binary Trees
- Priority Queues
- Heaps (including definition, height)
- Dictionaries (hash tables, hash functions, universal hashing, how to avoid collisions, collision handling strategies, performance in relation to load factor)
- Arbitrary binary search trees (including definition, height)
- Red-black trees (including definition, color properties, height). I won't ask you to demonstrate the removal of an item from a red-black tree.

Algorithms include:

- Execution of any data structure operation (e.g., insert into a red-black tree, remove an item from a binary search tree, find an item in a hash table using linear probing or double hashing, etc.)
- Tree traversals (preorder, postorder, inorder, Euler tour)
- Binary search on an array
- Selection-sort, insertion-sort, heap-sort
- Bottom-up heap construction

Ch. 4 Sorting and Selection

Definitions of stable sort, in-place algorithm, divide and conquer paradigm, and lexicographic order. Lower bound on comparison-based sorting. A comparison of sorting algorithms based on time-complexity, if it is stable, and if it is in-place.

Data structures include:

- Set (including operations using generic-merge).

Algorithms include:

- Merge-sort (including sub-routine of merging two sorted lists)
- Quick-sort (including pivot selection, its affect on performance, and sub-routine of partitioning two lists)
- Bucket-sort and radix-sort
- Quick-select