CSE 373: Topics Covered (post-midterm: July 24 – August 16, 2017)

(Note that this is only a big-picture overview – it leaves out a lot of detail)

Graphs

- General knowledge & terminology
  - Mathematical representation (G = (V, E), etc.)
  - Undirected & Directed Graphs
  - Self Edges
  - Weights
  - Paths
  - Cycles
  - Connectedness
  - Trees as graphs
  - DAGs
  - Density & Sparsity
- Graph data structures
  - Adjacency Matrix
  - Adjacency List
  - When to use which and why

Graph algorithms

- Topological Sort
  - What it is
  - Necessary conditions
  - Two algorithms for topological sort
- Traversals
  - Depth First Search (DFS)
  - Breadth First Search (BFS)
  - When to use which
- Shortest path
  - For unweighted graphs
  - For weighted graphs (Dijkstra's algorithm)
  - Two approaches to Dijkstra's, when to use which
- Spanning Trees
  - Approach #1: DFS
  - Approach #2: Add acyclic edges
- Minimum Spanning Tree (MST)
  - Prim's Algorithm
  - Kruskal's Algorithm

Sorting Algorithms

- Terminology
  - Stable sort
  - In-place sort
  - External sort
- Comparison Sort
  - Insertion Sort
  - Selection Sort
  - Heapsort (including in-place version)
  - Merge Sort (including time- & space-saving versions)
  - Quicksort (including different pivot rules and in-place quicksort)
  - Using cutoffs
- Other Sorts
  - Conditions that let you use them
  - Bucket Sort (a.k.a. Bin Sort)
  - Radix Sort
- How to sort massive data
  - What algorithms make the most sense and why
  - How to sort
- For each algorithm:
  - Worst- best-case scenarios & run times
  - Other pro's/con's of each
  - When to use which

General Algorithms Knowledge

- Analyzing algorithms
  - Correctness (less emphasis here)
  - Efficiency
- Several algorithm types
  - Greedy algorithms
  - Dynamic programming
  - Divide-and-conquer
- P vs NP

Software Design: Preserving Abstractions

- Abstraction (what it is, why it's important)
- Memory representation (call stack, heap space, program counter, etc.)
- Aliasing and mutations, how they're problematic
- Copy-in
- Copy-out
- Immutability (e.g. using the 'final' keyword)
- Deep copies & deep immutability (and why)

Parallelism

- Terminology
- Parallelism vs Concurrency
- Shared memory & race conditions
- Threads / Fork-join programming
    - How to use in Java (subclass, create 'thread' object, start(), join())
    - What happens under the hood
- Divide-and-conquer approach and why
- Map & Reduce
- Analysis (including Amdahl's Law)

Design decisions

- Ability to ask questions about problem to inform solution
- How to analyze/justify a decision
    - Time efficiency
    - Space efficiency
    - How parallelizable (in a few cases)
- Fluency with data structures & algorithms concepts/knowledge
    - Purposes a data structure is well-suited for and why
        - Available operations
        - Efficiency of basic operations
        - Space usage (conceptually)
    - Pro's and con's of different algorithms

CSE 373: Topics Covered (pre-midterm: June 19 – July 19, 2017)

(Note that this is only a big-picture overview – it leaves out a lot of detail)

- **Abstract Data Types (ADTs)** and **Data Structures**
- **Stacks** and **Queues**
  - Linked list implementation
  - Array implementations (including circular arrays)
- **Asymptotic Analysis**
  - Big-O of code snippets
  - Inductive Proofs
  - Recurrence Relations (and when to apply them)
  - Formal definition of Big-O
  - Big-O and -Omega, Theta, little-o and -omega
  - Amortized Analysis
- **Dictionary ADT**
- **Hash Tables**
  - Hash functions, hash values, and indexing
  - insert, find, remove
  - Collisions
  - Separate chaining
  - Open addressing / probing
  - Linear probing
  - Quadratic probing
  - Double hashing
  - Rehashing
- **Generic trees**
  - Terminology
- **Binary trees**
  - Terminology
  - Representation
  - Calculating the height
  - Traversals
- **Binary Search Tree (BST)**
  - find
  - insert
  - delete (3 cases)
  - buildTree
  - Terminology (e.g. successor, predeccessor)
  - Balanced vs unbalanced trees
- **AVL Trees**
  - Balance conditions
  - AVL balance condition
  - Rotations
  - insert (4 cases)
- **Priority Queue ADT**
- **Heaps**
  - insert & delete
  - Percolations
  - Array representation/implementation
  - buildTree (client version and Floyd's Method /heapify)
  - d-heaps
- **For each data structure**
  - Ways to implement
  - Pros, Cons, and other reasons to choose one over the other