**Study Guide for 5008 Quiz/Exams**

I am interested in evaluating how you reason through a Data Structure, Algorithms, and Systems problems. You need to be able to discuss how a system, data structure, or algorithm works, and analyze trade-offs by defending these tradeoffs. This is a suggested study guide, or rather a list of major topics discussed in the course. Anything from homeworks, labs, and lecture is fair game.

The exam will be taken online. This study guide will be updated closer to the exam date, consider a supplement to help you study.

**Spring 2021 - The exam format will be 100% multiple choice**

# First Half of Semester Exam Topics of Emphasis

* Operating System Overview
  + What exactly is the kernel?
  + Be familiar with the terminal and the shell.
* C Programming
  + What is a pointer? Know the Syntax
  + What is Pass by Value?
  + Know how to dynamically allocate and free memory.
    - i.e. malloc and free
* Assembly and Machine Representation
  + What is assembly language a representation of?
  + Know about registers
  + Understand how the stack and heap work.
* Data Structures (How they’re used, Big O complexity of operations, trade-offs)
  + Linked List
  + Stack
  + Queue
  + Doubly-Linked List
  + Hashmap
* Compilers and Linkers
  + Know the compilation process
    - What does a compiler do?
      * Why do we use them?
      * What is the frontend responsible for? The backend?
      * Have some understanding of Compilers(clang)/LLVM and gcc
  + Know about Compile-Time and Link-Time Interpositioning.
    - What is its purpose, what could it be used for.
  + Understand the link process
    - What is a linker?
      * Why do we use them?
* Processor and Processes
  + What is a context switch? What information gets saved?
  + What are signals?
  + How do you write a signal handler? What goes on when a signal is found?
  + What does a fork() do? Why is it a *strange* function?
  + What does execve(...) do? What about execvp?
  + What is a context switch?
* The Memory Hierarchy
  + What is pipelining?
  + Why is branching bad for performance?
  + What is caching?
  + What is a directly mapped cache?
* ~~Virtual Memory~~ **~~(We did not really cover this in this course)~~**
  + ~~What is virtual memory?~~
* Concurrency
  + What is a thread? What is a process? What is the difference?
    - What is fork(), wait(), a child process, zombie processes, orphan processes,
  + Be familiar with posix threads
    - How to spawn threads, What does it mean to join a thread, etc.
  + Understand the problems with concurrency and sharing memory

# Second Half of Semester Exam Topics of Emphasis

1. Topics in C
   1. Dynamic memory allocation using malloc
   2. Arrays vs. Linked List
2. Searching
   1. Linear Search vs. Binary Search vs. Bogo/Bozo Search
      1. What are the trade-offs of each approach
   2. Depth-First Search
   3. Breadth-First Search
   4. Dijkstra’s Shortest Path
      1. What kinds of graphs does it work on?
      2. Do negative weights present a challenge?
   5. ~~String Matching~~
      1. ~~Know why the brute-force approach is bad (~~**~~not on the exam~~**~~)~~
3. Sorting
   1. Know the complexity of each sort and the trade-off of each
      1. Bubble Sort
      2. Selection Sort
      3. Insertion Sort
      4. Heap Sort
      5. Merge Sort
      6. Topological Sort
      7. Quick Sort
4. Data Structures
   1. Adjacency List vs Adjacency Matrix representation on graphs
   2. Difference between queue and stack data structures (FIFO vs LIFO)
   3. Trade-off of Linked List vs Doubly Linked List
   4. Heaps
      1. Min-queue and max-queue
   5. For Graphs, know their properties
      1. Notation: Nodes, edges, weighted vs unweighted, directed vs undirected, degree (in-degree, out-degree)
      2. How to detect a cycle (what algorithms could we use, how would we keep track?)
   6. Hash map
      1. What are the average-case complexities
      2. What is a hash function? Does a bad hash function hurt our performance?
   7. ~~Tries (~~**~~not on the exam~~**~~)~~
      1. ~~Understand how they work, time complexity for insert/lookup, and why their space complexity is not so good.~~
5. Analysis
   1. Given a piece of code, be able to identify it’s Big-O complexity
   2. Know the complexity of operations such as:
      1. i.e. Add/Insert, Update, Search, Delete for each data structure
   3. Know why we use worst-case analysis in computer science
   4. What is an amortized analysis?
      1. What would an example algorithm or data structure be?
   5. Why is the Traveling Salesman problem intractable? (i.e. why does it take so long?)
   6. Given a recurrence tree, be able to identify the run-time of an algorithm.
6. Dynamic Programming
   1. What is it?
   2. What is memoization?
   3. How did it help reduce the fibonacci number generation from exponential to polynomial?
7. Strategies
   1. Comparison Sort-based
   2. Greedy Algorithms
   3. Divide and Conquer
   4. Randomized Algorithms
   5. Dynamic Programming