Fall 2018
Final Exam Study Guide

The final exam is cumulative, so material from throughout the courses may appear, however, the emphasis of the final will be on material covered since the midterm.

Searching and sorting

- What is searching? What is sorting? Why are they important? How are they relevant to each other?
- Be familiar with the following algorithms and their characteristics to the extent that you could implement them in code:
  - Linear search
  - Binary search
  - Selection sort
  - Bubble sort
- How do the search algorithms differ in terms of time taken? Best case? Worst case?
- How do the sort algorithms differ? Best case? Worst case?
- How does sorting pertain to the effectiveness/speed of search algorithms?

Time complexity

- Understand the basics of big O notation, to the extent that they have been covered in class.
- Understand linear ($O(n)$) and logarithmic ($O(\log(n))$) complexity and how they relate to each other.
- Understand base 2 logarithms.
- Why is a "fixed coefficient" generally disregarded when discussing complexity?

Data structures/trees

- Understand the basic structure of trees to the extent that you could implement a simple tree class.
  - How are trees like/unlike linked lists?
  - How might binary-branching trees be implemented?
  - How might trees with arbitrary branchings be implemented?
- Binary search trees: what are their characteristics? How do they relate to search?
- Insertion/deletion in binary search trees.
- Insertion/deletion in arrays.

Recursion

- What is recursion?
- What is mutual recursion?
- Why is recursion important/worth studying?
- What is a base case? What is a stopping condition? Why are they important?
- What is head recursion and tail recursion? Why might this distinction be important?

Python functionality

- What are static methods? How do they differ from instance methods? When might they be useful?

Code writing

- Be able to write simple recursive functions.
- Be able to implement search & sort algorithms.
- Be able to write classes & methods, and work with strings, lists, dictionaries, sets, stacks, queues.