CS 374: Algorithm and Models of Computation (Spring 2018)
Syllabus for Midterm 2

The second midterm will test material covered in lectures 9 through 17. This corresponds to material in Jeff's notes that is Section 36.4, 36.6, 36.7, 36.8 (Church-Turing Thesis), Lecture 1 (Divide and Conquer paradigm), Appendix II (solving recurrences), Lecture 3 (Backtracking), Lecture 5 (Dynamic Programming), Lecture 7 (Greedy Algorithms), Lecture 18 (Basic Graph Algorithms), and Lecture 19 (Depth First Search). Additional material covered in the lectures is in notes titled "The Church-Turing Thesis". Also, Chandra Chekuri's lecture slides offers an alternate presentation of all the algoithms material. The algorithms material can also be accessed from the textbook by Dasgupta, Papdimitriou, and Vazirani (Chapters 2, 3, 5, and 6) available from `http://beust.com/algorithms.pdf`.

Specific skills that may be test include (the following list may not be exhasutive)

Turing machines and Church-Turing thesis

- Understanding nondeterministic Turing machines and how they work.
- Equivalence between nondeterministic and deterministic Turing machines.
- Understanding the statement of the Church Turing thesis, and the Invariance thesis

Measuring time on deterministic and nondeterministic Turing machines

- Definitions of complexity classes P and NP.
- Understanding the order notation, and being able to compare functions with respect to the order notation.

Divide and Conquer Paradigm

- Solving recurrences characterizing the running time of divide and conquer algorithms.
- Familiarity with specific Divide and Conquer Algorithms and the running times: Merge Sort, Quick Sort, Karatsuba's Algorithm, Linear Selection.
- Ability to design and analyze divide and conquer algorithms for new problems.

Backtracking and Dynamic Programming Algorithms

- Using the dynamic programming methodology to design algorithms for new problems.
- Ability to analyze the running time of dynamic programming algorithms.

Greedy Algorithms

- Ability to design and prove correctness of greedy algorithms for example problems.
- Ability to construct examples to show non-optimality of greedy solutions.

Graphs

- Basic definitions of undirected and directed graphs, DAGs, paths, cycles.
- Definitions of reachable states, connected components, and strongly connected components.
- Understand the structure of directed graphs in terms of the meta-graph of strongly connected components.

Graph Search

- Understand properties of the basic search algorithm and its running time.
- Understand properties of depth first search traversal on directed and undirected graph.
- Understand properties of the depth first search tree.