Syllabus for Midterm 2

CS/ECE 374 A: Algorithms & Models of Computation (Spring 2024) Ver: 1.0

The second midterm will test material covered in the lectures (and labs) from week 6 through week 10. See lecture scribbles and the relevant chapters from Jeff's book. There are a few differences in how we describe/approach the topics.

Specific skills that may be tested include (the following list may not be exhaustive):

1 Divide and Conquer Paradigm

1.A. Solving recurrences characterizing the running time of divide and conquer algorithms.

1.B. Familiarity with specific Divide and Conquer Algorithms and the running times: Binary Search, Merge Sort, Quick Sort, Karatsuba's Algorithm, Linear Selection.

1.C. Ability to design and analyze divide and conquer algorithms for new problems.

2 Dynamic Programming Algorithms

2.A. Using the dynamic programming methodology to design algorithms for new problems.

2.B. Ability to analyze the running time of dynamic programming algorithms.

3 Graphs

3.A. Basic definitions of undirected and directed graphs, DAGs, paths, cycles.

3.B. Definitions of reachable nodes, connected components, and strongly connected components.

3.C. Understand the structure of directed graphs in terms of the meta-graph of strongly connected components.

3.D. Understand the structure of DAGs: sources, sinks and topological sort.

4 Graph Search

4.A. Understand properties of the basic search algorithm and its running time.

4.B. Understand properties of **DFS** traversal on directed and undirected graph.

4.C. Understand properties of the **DFS** tree.

4.D. Algorithms based on search for finding connected components in undirected graphs, checking whether a graph is a DAG, computing topological sort for DAGs, finding a cycle in a graph etc. The existence of a linear-time algorithm to compute strongly connected components and create the meta-graph.

5 Shortest Paths in Graphs

5.A. Understand properties of the **BFS** trees.

5.B. Understand properties of **BFS** traversal on directed and undirected graph to find distances in unweighted graphs.

5.C. Dijkstra's algorithm for finding single-source shortest paths in undirected and directed graphs with non-negative edge lengths.

5.D. Negative length edges and Bellman-Ford algorithm to check for negative length cycles or find shortest paths if there is none.

5.E. Single-source shortest paths in DAGs — linear time algorithm for arbitrary edge lengths.

5.F. Shortest path trees and their basic properties.

5.G. Dynamic programming for shortest path problems in graphs.

6 Graph reductions and tricks

6.A. Modeling problems via graphs and solving them using graph structure, reachability and shortest path algorithms.

6.B. Adding sources, sinks, splitting edges, nodes

6.C. Creating layered graphs