

The second midterm will test material covered in lectures 11 through 21.

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.
  - 3.E. Solving dynamic-programming problems using problems on DAGs.
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, topological sort for DAGs, knowledge of a linear-time algorithm to create the meta-graph, finding a cycle in a graph etc.
  - 4.E. Algorithms for DFAs/NFAs using graph algorithms.
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. Floyd-Warshall algorithm.
  - 5.F. Single-source shortest paths in DAGs — linear time algorithm for arbitrary edge lengths.
  - 5.G. Shortest path trees and their basic properties.
  - 5.H. 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
- 7. Designing greedy algorithms, and proving their correctness.