**Exam 1 Review Outline**

**Chapter 0**

- Big-O Compare \( f(n) \) vs \( g(n) \) for dominance
- An \( f(n) \) algorithm takes \( X \) seconds to run on \( n \) items. How long will it take to run on \( m \) items?
- An algorithm has the following run times for sizes: \( (\text{size}, \text{time}) = \{ (n, x), (2n, y), (4n, z), ... \} \). What is the approximate complexity of the algorithm? Probably one of: \( \log(n), n, n^2, n^3, 2^n \).

**Chapter 2**

- Recurrence Relations: \( T(n) = a \cdot T(f(n)) + g(n) \)
- Solve via Master Theorem if \( f(n) = \frac{n}{b} \) and \( g(n) = n^d \)
- Solve via Substitution otherwise.
- Divide-and-Conquer as an algorithm strategy.
  - Break problem into smaller pieces
  - Solve smaller problems
  - Assemble smaller solutions into larger solution
- Given a problem, create a Divide-and-Conquer algorithm to solve the problem.
- Given a Divide-and-Conquer algorithm, find its complexity.

**Chapter 3**

- Graphs: undirected, directed, DAG
- \( \text{explore}(G, u) \)
- \( \text{dfs}(G) \)
- pre/post numbers
- edge types: [ tree, forward, back, cross ]
- connected components for undirected graphs
- linearize a DAG
- Strongly Connected Components for directed graphs
- Given graph \( G = (V,E) \), run algorithm \( X \) on it, show the process and results
- Given a problem with a graph \( G = (V,E) \), create a graph algorithm to solve the problem.
- Given a problem, give an algorithm to convert it to a graph, such that a graph algorithm can solve it.
- Given a graph algorithm, find the complexity of the algorithm.