Practice Questions for Midterm Exam

Warning and Disclaimer: These are practice problems for the upcoming midterm exam. It does not necessarily reflect the length or coverage of the actual exam.

Problem 1. Consider the following directed graph:

(a) Label each vertex \( v \) with the values \( d[v] \) and \( f[v] \) which would be computed by a DFS starting at vertex \( A \). Assume that the DFS starts at vertex \( a \) and, given a choice, chooses to visit vertices in alphabetical order.

(b) Label each edge as a tree edge, back edge, forward edge, or cross edge.

(c) Circle each strongly connected component.

Problem 2. Consider an undirected graph whose \( n \) vertices are “colored” with at most \( n \) (not necessarily distinct) integers from \( 1, \ldots, n \). Give a polynomial time algorithm to determine if there are two distinct vertices within a distance of two of each other that have the same color. Make your algorithm as efficient as possible: \( O(n + e) \) is better than \( O(n^2) \) is better than \( O(ne) \) is better than \( O(n^3) \), etc.

Problem 3. Let \( G = (V, E) \) be a directed, acyclic graph (DAG), where each vertex is colored red, blue, or green.

(a) Give an efficient algorithm using depth first search to find the set of all green vertices that are reachable (by a path) from both a red vertex and a blue vertex. Write the Psuedo-Code and briefly explain your algorithm in English.

(b) Analyze its running time.

Problem 4. Do Exercise 10 on pages 110-111 of Kleinberg and Tardos.

Problem 5. Consider the following undirected, weighted graph:

(a) Show the order that the edges would be included using Kruskal’s algorithm.

(b) Show the order that the edges would be included using Prim's algorithm.
Problem 6. In the knapsack problem you are given a set of $n$ items with values $v_i$ and weights $w_i$, and a weight bound $W$. The goal is to find a subset of the items whose total weight is at most $W$, and whose total value is as large as possible.

In the fractional knapsack problem you may cut an item to make it fit into your knapsack. (Its value is proportional to the original value, so, for example, one third an item has one third of its value.)

(a) Give an efficient algorithm to solve the fractional knapsack problem.
(b) Prove your algorithm finds an optimal solution.
(c) Analyze its run time.