CS 331: Algorithms and Complexity (Spring 2024)
Unique Number: 50930, 50935 50940, 50945

Assignment 2

Due on Thursday, 1 Febrauary, by 11.59pm

Problem 1: Short Answers Section

For this section, restrict answers to no more than a few sentences. Most answers can be expressed in a single sentence. Unless otherwise stated, briefly justify. No proofs are necessary for this section.

(a) (1 pt) Every graph with n vertices and $n-1$ edges is a tree.

(b) (2 pts) There is topological ordering for a connected and directed graph.

(c) (2 pts) If we represent a graph with $|V|$ vertices and $\Theta(|V|)$ edges as an adjacency matrix, the worst-case running time of breadth-first search is $\Theta(|V|^2)$.

(d) (2 pts) In a dense and undirected and connected graph, the time to scan all edges if using an adjacency matrix is $\Theta(|V|)$.

(e) (1 pt) The DFS tree always has a greater depth than that of the BFS when applied to the same undirected, connected graph.

(f) (2 pts) In a tree, every two nodes are connected by a unique path.

Problem 2

For undirected and unweighted graphs, answer the following:

(a) (3 pts) A tree can always be represented as a bipartite graph. If this true, describe an algorithm in words to 2-color a tree. If not, give a counterexample.

(b) (7 pts) Building upon your solution to part a, propose an algorithm to find the maximum number of edges that can be added to a tree (in the same set of vertices) such that the resultant graph is a bipartite graph. Prove its correctness. Your algorithm should run in $O(n)$ time.

Problem 3

(10 pts)

Sometimes, it's good to have alternatives.

Supposed you've just moved to a new city. You want to select an apartment that has the larget number of distinct shortest paths to your workplace. Since you're a computer scientist, in accordance with Maslow's law of the hammer, you decide to solve this by representing the ciy as a graph: each road becomes a weighted edge (weighted according to how long it takes to travel that section of road), and places where the roads meet (i.e. intersections) become nodes.

After completing this exercise, you notice something peculiar: every edge in this graph has the same weight! For convenience, you decide to declare that the weight of each edge to be 1, turning this into an unweighted graph. Now you just need to figure out how to calculate the number of different shortest paths.

Given an unweighted, undirected graph $G = (V, E)$ representing the road network and two fixed nodes s and t within this graph (representing a potential apartment and your new workplace, respectively), devise an algorithm to find the number of different shortest paths between s and t. Prove the correctness of your algorithm.

Your algorithm should run in $O(|V| + |E|)$ time. You do not need to prove the time complexity