CS 3510: Homework 3

Due on Tuesday, June 22, 11:59pm via Gradescope.
Late submission with no penalty until Wednesday, June 23, 11:59am.

Professor Brito

CS 3510 Staff

Instructions and black-box list.

For the graded problems, you are allow to use the algorithms from class as black-boxes without further explanation. These include

- DFS (outputs connected components,a topological sorting on a DAG. You also have access to the pre and post arrays!), BFS and the Explore subroutine (outputs a path between the input vertex v and any other vertex in the same connected component).
- Dijkstra's algorithm to find the shortest distance from a source vertex to all other vertices and a path can be recovered backtracking over the pre labels.
- Bellman-Ford and Floyd-Warshall to compute the shortest path when weights are allowed to be negative.
- SCCs which outputs the strongly connected components, and the metagraph of connected components.

When using a black-box, make sure you clearly describe which input you are passing to it and how you use the output from or take advantage of the data structures created by the algorithm. To receive full credit, your solution must:

- Include the description of your algorithm in words (no pseudocode!).
- Explain the correctness of your design.
- State and analyse the running time of your design (you can cite and use the running time of black-boxes without further explanations).

Unless otherwise indicated, black-box graph algorithms should be used without modification.

Suggested reading.

1. Chapter 3 (for DFS, topological sorting and SCCs).
2. Chapter 4 (for distances and paths on graphs, BFS and Dijkstra's)

Suggested problems.

`The numeration of the problems corresponds to the 2006 edition of the book.`
`You do not need to turn in these problems.`

1. Chapter 3: $3.1 - 3.5, 3.7, 3.16, 3.21, 3.22$.
2. Chapter 4: $4.1, 4.2, 4.5, 4.13, 4.14, 4.20$.

Practice problem.

1. **problem** 3.11 **(determine if a given edge is on a cycle).**
2. Given an undirected graph $G = (V, E)$ with positive edge weights and two nodes $s, t \in V$, design an efficient algorithm to determine the set of all edges that lie on at least one shortest path from s to t.

DO NOT USE PSEUDOCODE
A solution using pseudocode will receive a zero, even if it is correct.

Problem 1

The police department in the city of Computopia has made all streets one-way. The mayor contends that there is still a way to drive legally from any intersection in the city to any other intersection, but the opposition is not convinced. A computer program is needed to determine whether the mayor is right. However, the city elections are coming up soon, and there is just enough time to run a linear-time algorithm.

1. Formulate this problem graph-theoretically, and explain why it can indeed be solved in linear time.
2. Suppose it now turns out that the mayor's original claim is false. She next claims something weaker: if you start driving from town hall, navigating one-way streets, then no matter where you reach, there is always a way to drive legally back to the town hall. Formulate this weaker property as a graph-theoretic problem, and carefully show how it too can be checked in linear time.

Problem 2

Let $G = (V, E)$ be an undirected graph with the additional property that every edge also has a color, either red or blue. Let u and v be distinct vertices in $G = (V, E)$.

(a) Design an efficient algorithm that decides whether or not there exists a path from u to v such that the path contains only red edges. Justify correctness and running time.

(b) Design an efficient algorithm that decides whether or not there exists a path from u to v such that within the path, all blue edges appear after all red edges . Justify correctness and running time. Fastest (and correct) solutions worth more credit.

.