CS420/520: Graph Theory with Applications to CS, Winter 2018

Homework 2

Due: Tue, 2/6/18

Homework Policy:

1. Students should work on homework assignments in groups of preferably three people. Each group submits to TEACH one set of typeset solutions, and hands in a printed hard copy in class or slides the hard copy under my door before the midnight of the due day. The hard copy will be graded.
2. The goal of the homework assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.
3. You are allowed to discuss the problems with other groups, and you are allowed to use other resources, but you *must* cite them. Also, you must write everything in your own words, copying verbatim is plagiarism.
4. *I don't know policy:* you may write "I don't know" *and nothing else* to answer a question and receive 25 percent of the total points for that problem whereas a completely wrong answer will receive zero.
5. Algorithms should be explained in plain english. Of course, you can use pseudocodes if it helps your explanation, but the grader will not try to understand a complicated pseudocode.
6. **Solutions must be typeset.**

Readings:

(A) Jeff lecture notes on graph search: http://jeffe.cs.illinois.edu/teaching/algorithms/notes/19-dfs.pdf.

Problem 1. Given a directed graph $G = (V, E)$ and two nodes s, t, an st-walk is a sequence of nodes $s = v_0, v_1, \ldots, v_k = t$ where (v_i, v_{i+1}) is an edge of G for $0 \leq i < k$. Note that a node may be visited multiple times in a walk ? this is how it differs from a path. Given G, s, t and an integer $k \leq n$, design a linear time algorithm to check if there is an st-walk in G that visits at least k distinct nodes including s and t.

- Solve the problem when G is a DAG.
  - Let $s \to v_1, s \to v_2, \ldots, s \to v_\ell$ be all outgoing edges of s. Describe a recursive solution for this problem.
  - Modify DFS to solve this problem on a DAG.
  - Hint: it is easier to think about this problem if you view the vertices in topological order.
- Solve the problem when G is a an arbitrary directed graph. Hint: If G is strongly connected then there is always such a walk even for k = n (do you see why?).

Problem 2. Let G be a directed acyclic graph with a unique source s and a unique sink t.

(a) A Hamiltonian path in G is a directed path in G that contains every vertex in G. Describe an algorithm to determine whether G has a Hamiltonian path.

(b) Describe an algorithm to compute the number of distinct paths from s to t in G. (Assume that you can add arbitrarily large integers in O(1) time.)

Problem 3. Suppose two players are playing a turn-based game on a directed acyclic graph G with a unique source s. Each vertex v of G is labeled with a real number $\ell(v)$, which could be positive, negative, or zero. The game starts with three tokens at s. In each turn, the current player moves one of the tokens along a directed edge from its current node to another node, and the current player's score is increased by $\ell(u) \cdot \ell(v)$, where u and v are the locations of the two tokens that did not move. At most one token is allowed on any node except s at any time. The game ends when the current player is unable to move (for example, when all three tokens lie on sinks); at that point, the player with the higher score is the winner. Describe an efficient algorithm to determine who wins this game on a given labeled graph, assuming both players play optimally. (Hint: first, try the problem with one token.)