CS420/520: Graph Theory with Applications to CS, Winter 2020

Homework 2

Due: Thr, 2/6/20

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 single source shortest paths: http://jeffe.cs.illinois.edu/teaching/algorithms/book/08-sssp.pdf.

(B) Jeff lecture notes on all pairs shortest paths: http://jeffe.cs.illinois.edu/teaching/algorithms/book/09-apsp.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. You are given a directed graph $G = (V, E)$, where every edge e has an independent safety probability $p(e)$. The safety of a path is the product of the safety probabilities of its edges. Design and analyze an algorithm to determine the safest path from a given start vertex s to a given target vertex t. You may assume that all necessary arithmetic operations can be performed in $O(1)$ time.

Problem 3. Let $G = (V, E)$ be a connected directed graph with non-negative edge weights, let s and t be vertices of G, and let H be a subgraph of G obtained by deleting some edges. Suppose we want to reinsert exactly one edge from G back into H, so that the shortest path from s to t in the resulting graph is as short as possible. Describe and analyze an algorithm that chooses the best edge to reinsert, in $O(E \log V)$ time.

Problem 4. Suppose we are given a directed graph G with weighted edges and two vertices s and t.

(a) Describe and analyze an algorithm to find the shortest path from s to t when exactly one edge in G has negative weight.

(b) Describe and analyze an algorithm to find the shortest path from s to t when exactly k edges in G have negative weight. How does the running time of your algorithm depend on k?

Problem 5. Describe and analyze an efficient algorithm to compute the number of shortest paths between two specified vertices s and t in a directed graph G whose edges have positive weights. [Hint: Which edges of G can lie on a shortest path from s to t?]

Problem 6. Prove that Ford's generic relaxation algorithm (and therefore Dijkstra's algorithm) halts after at most $O(2^V)$ relaxations, unless the input graph contains a negative cycle.