OLD CS 473: Fundamental Algorithms, Spring 2015

Discussion 3

February 5, 2015

3.1. 2SAT.

You are given a boolean formula that is a 2CNF. That is, every clause is the OR of two boolean variables, and the formula is the conjunction of the clauses. For an example, consider the following formula:

$$F = (x_1 \vee \overline{x_2}) \wedge (\overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee \overline{x_4}) \wedge (\overline{x_2} \vee \overline{x_3}).$$

(A) What is a satisfying assignment for the above formula?

(B) Describe a linear time algorithm that computes a satisfying assignment if it exists (hint: think about numbers $i/-i$).

3.2. Reductions.

Show that the following problems can be reduced to the standard shortest path problems. No proof required.

(A) Given directed graph $G = (V, E)$ and two disjoint sets of nodes S, T. Find the shortest path from some node in S to some node in T.

(B) G is a directed graph and nodes and edges have non-negative lengths. Find s-t shortest path where the length of a path is equal to the sum of the lengths of the nodes and edges on the path.

(C) Given a directed graph G with node lengths (no edge lengths), is there a negative length cycle? Here the length of a cycle is the sum of the lengths of nodes on the cycle.

(D) G is a DAG and each node has a non-negative length. Given two nodes s, t in G, find the s-t longest simple path in linear time.

3.3. Quick Fix.

Your "friend" suggests that the easiest algorithm for finding shortest paths in a directed graph with negative-weighted edges is to make all the weights positive by adding a sufficiently large constant to each weight and then running Dijkstra's algorithm. Give an example that you can show your friend to prove that his or her method is incorrect.

3.4. Almost Positive.

We are given a directed graph $G = (V, E)$ with potentially negative edge lengths. Your friend ran Dijkstra's algorithm and came up with a shortest path tree T for distances from a node s. You realize that Dijkstra's algorithm may not output distances correctly when a graph has negative edge lengths. However, before you run the more expensive Bellman-Ford algorithm, you wish to check whether T is a correct shortest path tree or not. Describe an

$O(m + n)$ time algorithm to do this check. Don't forget to prove that your algorithm is correct!

3.5. LIMITED SHORTEST PATHS.
We are given a directed graph in which the shortest path between any two vertices u and v is guaranteed to have at most k edges. Give an algorithm that finds the shortest path between two vertices u and v in $O(k(n + m))$ time. Remember, edges can have negative weights.

3.6. AVERAGE CYCLE.
You are given a directed weighted graph G (the weights are positive), and a number x. Design an algorithm that decides if G has a cycle with average cost strictly smaller than x. The average cost of a cycle is the total weight of its edges divided by the number of edges. How fast is your algorithm?

3.7. BEST EDGE TO ADD, FAST.
Suppose you are given a directed graph $G = (V, E)$ with non-negative edge lengths; $\ell(e)$ is the length of $e \in E$. You are interested in the shortest path distance between two given locations/nodes s and t. It has been noticed that the existing shortest path distance between s and t in G is not satisfactory and there is a proposal to add exactly one edge to the graph to improve the situation. The candidate edges from which one has to be chosen is given by $E' = \{e_1, e_2, \ldots, e_k\}$ and you can assume that $E \cap E' = \emptyset$. The length of the e_i is $\alpha_i \geq 0$. Your goal is figure out which of these k edges will result in the most reduction in the shortest path distance from s to t. Describe an algorithm for this problem that runs in time $O(n \log n + m + k)$ where $m = |E|$ and $n = |V|$. Note that one can easily solve this problem in $O(k(m + n) \log n)$ by running Dijkstra's algorithm k times, one for each G_i where G_i is the graph obtained by adding e_i to G.