COSC 240, Fall 2017: Problem Set #6

Assigned: Wednesday, 11/15.
Due: Monday 11/27, at the beginning of class (hand in hard copy).
Lectures Covered: 19 to 21
Academic Integrity: You can work with other people in the class but you must write up your own answers in your own words. You can also use the textbook and talk to the professor. You may not use any other resources (e.g., material found online) or talk to people outside the class about these problems. See the syllabus for details on the academic integrity policy for problem sets.

Problems

1. Draw a connected undirected weighted graph G that contains an MST T and pair of nodes (u, v) such that the shortest path between u and v in T has a larger weight than the shortest path between u and v in G. To receive credit you must identify the MST T and pair (u, v) in G for which this property holds.

2. Assume $p = \langle u_1, u_2, ..., u_i \rangle$ is a shortest path from u_1 to u_i in some weighted graph G that contains only positive weights. Prove that p cannot contain a cycle.

   (Hint: start by assuming for contradiction that p *does* contain a cycle, then build from this assumption to a logical contradiction.)

3. Consider the class of connected undirected weighted graphs where each edge is weighted with a value from $\{1, 2, ..., k\}$, for some integer $k > 1$. Describe clearly and concisely in words (i.e., no pseudocode needed) an algorithm that solves the single source shortest path problem on graphs from this class in $O(k|V| + |E|)$ time.

   (Hint: You can use the BFS algorithm we studied in class as a subroutine.)

4. Describe a flow network G and flow f such that G has 5 nodes and every edge in G_f has residual capacity 5. To receive credit you must draw both the flow network (with flow values) and its corresponding residual network (with residual capacities).

5. This question has two parts, both of which concern the time complexity analysis of the Ford-Fulkerson algorithm (with integer capacities) that we studied in class.

   (a) In the time complexity analysis, I argued that each iteration of the main *while* loop must increase the value of the flow f by at least 1. Why is this?

   (b) We proved that the algorithm we studied requires $O(|f^*| \cdot |E|)$ time, where f^* is a maximum flow in the network. Consider the following "optimization." Starting with integer capacities, we reduce each capacity by a factor of $1/k$. That is, we replace each $c(u, v)$ with $c(u, v)/k$. We then calculate the max flow g^* on this reduced-capacity graph. It is clear to see that $|g^*| = |f^*|/k$, so to get the final answer we multiply all resulting flow values by k.

   At first glance, it might seem like this optimized version of the algorithm will run in $O(|g^*| \cdot |E|) = O(|f^*|/k \cdot |E|)$ time—potentially a big improvement for large k. But this first glance analysis is wrong. Explain why this is wrong and provide a more accurate time complexity upper bound.