COSC 240, Fall 2018: Problem Set #5

Assigned: Monday, 11/5.
Due: Monday 11/12, at the beginning of class (hand in hard copy).
Lectures Covered: 15 to 18
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. Rewrite the pseudocode for the BFS algorithm studied in class (and presented in the textbook) to work for an adjacency matrix representation of the graph instead of an adjacency list representation.

2. Analyze the time complexity of your answer from the previous problem. (To receive credit, you must not only give the time complexity for this algorithm, but also have a clear and correct argument for why this complexity is correct.)

3. Consider the following conjecture about the depth-first search (DFS) algorithm we studied in class:

   > Assume there is a directed path from u to v in graph G. It follows that in any depth-first search on G, $v.d \leq u.f$. That is, v has to start before u finishes.

   Prove this conjecture is false by providing a counterexample (e.g., demonstrate a graph and a particular execution of the DFS algorithm we studied on this graph that produces values for d and f for which this conjecture does not hold).

4. Let (u, v) be the edge in a weighted undirected graph with the minimum weight. Prove that there exists at least one MST of this graph that includes (u, v).

5. In class, we studied Prim's MST algorithm. This algorithm stores graph nodes in a priority queue. We analyzed the time complexity under the assumption that this queue was implemented with an efficient Fibonacci heap. This problem asks you to analyze the time complexity that results if the priority queue used by the algorithm is implemented less efficiently.

   In more detail, assume you implement with priority queue with a sorted linked list. With this implementation, to insert an element into the queue, you traverse the list from left to right (i.e., in increasing order) to find the first element with a key that is greater than or equal to the new element, and insert the new element before it in the list. If you change the key value of an element in the queue, you remove it from the underlying list, then re-insert it using the above insert procedure. Assume you can test for membership in $O(1)$ steps by keeping an auxiliary array with a bit for each node that indicates whether or not it is in the list.

   Analyze the time complexity of Prim's MST algorithm under the assumption that it uses this list-based priority queue implementation. Your analysis should touch on all the ways that the algorithm uses the queue (see the analysis of Prim's from the textbook to make sure you are not missing any way that the algorithm implicitly uses the queue).