Algorithms Final Examination
Jan. 12, 2022 (10:10 ~ 12:20)

(Note that, if you design an algorithm, you must have pseudo-code to represent your algorithm. You can put comments after your pseudo-code to clarify your presentation. You need to prove or explain the time complexity of your algorithm. Your algorithms may involve some procedures such as DFS and BFS. You can clearly state the information that could be found using DFS and BFS instead of writing out all the details of DFS and BFS. Keeping the pseudo-code as simple as possible is also recommended, and the algorithm's main idea is clearly expressed.)

1. (Chapter 23: 10%) Suppose that all edge weights in a graph are integers ranging from 1 to $|V|$. How fast can you make Prim's algorithm run? What if the edge weights are integers ranging from 1 to W for some constant W?

2. (Chapter 24: 10%) Describe Dijkstra's single-source shortest path algorithm. Explain why Dijkstra's algorithm does not allow negative edges.

3. (Chapter 24: 10%) Given a weighted, directed graph $G = (V, E)$ with no negative-weight cycles, let m be the maximum over all vertices $v \in V$ of the minimum number of edges in a shortest path from the source s to v. (Here, the shortest path is by weight, not the number of edges.) Suggest a simple change to the Bellman-Ford algorithm that allows it to terminate in $m + 1$ passes, even if m is not known in advance.

4. (Chapter 25: 10%) How can we use the output of the Floyd-Warshall algorithm to detect the presence of a negative-weight cycle? How to modify the Faster-All-Pairs-Shortest-Paths so that it can determine whether the graphs contain a negative-weight cycle.

5. (Chapter 25: 10%) Answer the following questions according to Johnson's algorithm.
   a) (6%) Professor Greenstreet claims that there is no need to create a new source vertex. He claims that instead, we can use any vertex in G as the source. Give an example of a connected, weighted, and directed graph for which the professor's idea does not work. You should briefly explain why the method does not work on this graph.
   b) (4%) Following question a), under what condition will the method work?

6. (6%) Given a directed graph $G = (V, E)$, we want to find the shortest paths of every pair of vertices $u, v \in V$. Please find the algorithm and its corresponding time complexity for the all-pair shortest-paths problem with each of the following assumptions. (You don't need to present the algorithm. But you need to describe what algorithms and data structures are used to solve the all-pair shortest-paths problem with the following assumptions.)
   a) Each edge of graph G has a unit weight.
   b) Assume the graph G are acyclic.

7. (Chapter 26 10 %) Let $G = (V, E)$ be a bipartite graph with vertex partition $V = L \cup R$, and let G' be its corresponding flow network. Give a good upper bound on the length of any augmenting path found in G' during the execution of Ford-Fulkerson.

8. (Chapter 34 10%) Please prove that the Traveling-salesman problem is NP-complete if you know that the Hamiltonian-cycle problem is NP-Complete.

9. (Chapter 35 10%) Give an efficient greedy algorithm that finds an optimal vertex cover for a tree in linear time.

10. (16%) For each of the following statements, determine whether it is true or false. Give brief explanations if the answer is false. (答錯倒扣一分)

    (1) Let e be a maximum-weight edge on some cycle of connected graph $G = (V, E)$. The minimum spanning tree of G does not include e.

    (2) If all edge weights of a graph are positive or nonpositive, then any subset of edges that connects all vertices and has minimum total weight must be a tree.

    (3) The time complexity of the Ford-Fulkerson algorithm is $O(|E| f^*)$, where $|E|$ is the number of edges in a directed graph and f^* is the maximum flow in the graph. So, the Ford-Fulkerson algorithm is a polynomial-time algorithm.

    (4) The 3-CNF satisfiability problem is polynomial-time reducible to the 2-CNF problem.

    (5) If a problem is in NP, the problem is polynomial-time verifiable.

    (6) If NP = P, any NP-hard problem can be solved in polynomial time.

    (7) If we can find a polynomial-time approximation algorithm for the Euclidean travelling-salesman problem, then NP = P.

    (8) If G is an undirected bipartite graph with an odd number of vertices, then G is nonhamiltonian.

Happy New Year!