Problem Set 3
Due: 2021-11-30 20:00

Note: For all problems, if you include pseudocode in your solution, please also include a brief description of what the pseudocode does. (Questions from Textbook will be marked, but not always the original ones.)

---

1. **(Transpose of a Directed Graph. CLRS 22.1-3)** (15 points) The transpose of a directed graph $G = (V, E)$ is the graph $G^T = (V, E^T)$, where $E^T = \{(v, u) \in V \times V : (u, v) \in E\}$. Thus, G^T is G with all its edges reversed. Describe efficient algorithms for computing G^T from G, for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.

   [**We are expecting: pesudo-codes and analyses of the running times.**]

2. **(Cubic of Directed Graph. CLRS 22.1-5)** (15 points) The cubic of a directed graph $G = (V, E)$ is the graph $G^3 = (V, E^3)$ such that $(u, v) \in E^3$ if and only G contains a path with at most three edges between u and v. Describe efficient algorithms for computing G^3 from G for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.

   [**We are expecting: pesudo-codes and analyses of the running times.**]

3. **(Universal Sink. CLRS 22.1-6)** (10 points) Most graph algorithms that take an adjacency-matrix representation as input require time $\Omega(V^2)$, but there are some exceptions. Show how to determine whether a directed graph G contains a universal sink—a vertex with in-degree $|V| - 1$ and out-degree 0—in time $O(V)$, given an adjacency matrix for G. [**We are expecting: a pesudo-code of your algorithm and an analyse of the correctness of your algorithm.**]

4. **(Counting Simple Paths. CLRS 22.4-2)** (10 points) Give a linear-time algorithm that takes as input a directed acyclic graph $G = (V, E)$ and two vertices s and t, and returns the number of simple paths from s to t in G. For example, the directed acyclic graph of Figure 1 contains exactly four simple paths from vertex p to vertex v: pov, $poryv$, $posryv$, and $psryv$. (Your algorithm needs only to count the simple paths, not list them.)

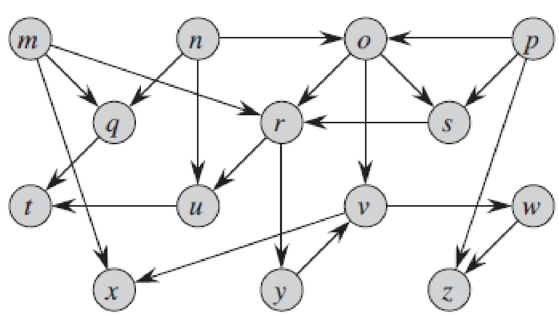

Figure 1: A Directed Graph

5. **(Euler Tour. CLRS 22-3)** (10 points) An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once.

   a. Show that G has an Euler tour if and only if in-degree(v)= out-degree(v) for each vertex $v \in V$.

   b. Describe an $O(E)$-time algorithm to find an Euler tour of G if one exists. (Hint: Merge edge-disjoint cycles.)

6. **(Arbitrage. CLRS 24-3)** (20 points) Arbitrage is the use of discrepancies in currency exchange rates to transform one unit of a currency into more than one unit of the same currency. For example, suppose that 1 U.S. dollar buys 49 Indian rupees, 1 Indian rupee buys 2 Japanese yen, and 1 Japanese yen buys 0:0107 U.S. dollars. Then, by converting currencies, a trader can start with 1 U.S. dollar and buy 49 2 0:0107 D 1:0486 U.S. dollars, thus turning a profit of 4:86 percent. Suppose that we are given n currencies $c_1, c_2, \cdots, c_n$ and an $n \times n$ table R of exchange rates, such that one unit of currency c_i buys $R[i, j]$ units of currency c_j.

   a. Give an efficient algorithm to determine whether or not there exists a sequence of currencies $\langle c_{i1}, c_{i2}, \cdots, c_{ik}$ such that $R[i_1, i_2] \cdot R[i_2, i_3] \cdots R[i_{k-1}, i_k] \cdot R[i_k, i_1] > 1$. Analyze the running time of your algorithm.

   b. Give an efficient algorithm to print out such a sequence if one exists. Analyze the running time of your algorithm.

7. **(Strongly Connected Components Maintaining.)** (20 points) For a directed graph $G = (V, E)$, we say a vertex j is reachable from another vertex i mean that there is at least one path from vertex i to j. A strongly connected component of G is a subgraph $G' = (V', E')$ ($G' \subseteq G$, $V' \subseteq V$, and $E' \subseteq E$) such that for any $u, v \in V'$, they are reachable in G'. For example, in Figure 2, the subgraph of 0,1,2 is a strongly connected component.

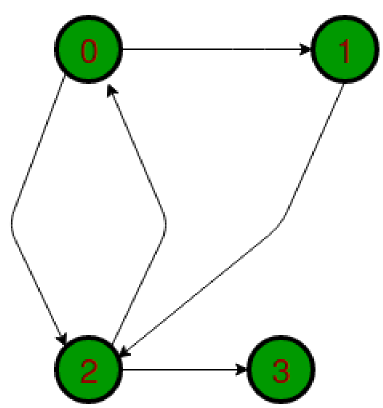

Figure 2: An Example of Transitive Closure

Assuming we have a directed graph $G = (V, E)$, the edges are inserted one-by-one into E. After every edge insertion, we want to know the current strongly connected components. Please answer the following questions:

a. Show how to update the strongly connected components after each edge is inserted. Analyze the running time of your algorithm.

b. Describe an algorithm to maintain the strongly connected components when edges are inserted one-by-one for all m edges. Analyze the total running time of your algorithm (Your analyses should always be correct for any sequence of m insertions of all edges).