Design and Analysis of Algorithms (CS345A)

Practice-sheet 4 : **Topological ordering and DFS tree**

1. (**Counting the number of shortest paths in DAG**)

   There is a directed acyclic graph $G = (V, E)$ on $n = |V|$ vertices and $m = |E|$ edges. Each edge has a length which is a positive real number. There is a unique vertex $s \in V$ such that every vertex in G is reachable from s. Let $N(v)$ be the number of the shortest paths from s to v. Design an $O(m+n)$ time algorithm to compute $N(v)$ for all $v \in V$. Assume that the number of such paths are only polynomial in n. (Can you see why this assumption is important ?)

2. (**The classification of edges**)

   We discussed the classification of edges by a DFS forest. Let G be a graph and let (u, v) be an edge. Try to answer the following questions.

   (a) Does there always exist some DFS forest of G such that (u, v) is present as a tree edge in it ? If not, what must be the necessary condition that G must satisfy for this to happen ?

   (b) Does there always exist some DFS forest of G such that (u, v) is present as a forward edge in it ? If not, what must be the necessary condition that G must satisfy for this to happen ?

   (c) Does there always exist some DFS forest of G such that (u, v) is present as a backward edge in it ? If not, what must be the necessary condition that G must satisfy for this to happen ?

   (d) Does there always exist some DFS forest of G such that (u, v) is present as a cross edge in it ? If not, what must be the necessary condition that G must satisfy for this to happen ?

3. (**More on DFS**)

   Let $G = (V, E)$ be a directed graph and $u, v \in V$ be any two vertices. Prove or give a counterexample for each of the following statements.

   (a) If u and v are strongly connected, then either u is ancestor of v or v is ancestor of u in every DFS tree.

   (b) If G is a DAG and there is a path from u to v in G, then u is surely going to be an ancestor of v in every DFS tree.

   (c) Suppose u and w are strongly connected whereas u and v are not strongly connected. If (u, v) is an edge, then $F(w) > F(v)$ must hold always.

   (d) We can construct G such that one DFS traversal may produce a DFS tree with degree $n - 1$ whereas another DFS traversal may produce a DFS tree which is just a chain of n vertices.

4. (**About SCC in a graph**)

   Prove the following statement or give a counterexample: If $S \subseteq V$ is any strongly connected component in a given directed graph $G = (V, E)$, then there must exist a cycle that passes through all the vertices of S.

5. (**SCC graph**) SCC graph is obtained by compressing each strongly connected component of a directed graph into a single vertex. If there are multiple edges between one strongly connected component s to another strongly connected component s', then in the SCC graph we keep only one edge from vertex corresponding to s to the vertex corresponding to s'. It can be observed that the SCC graph of a given graph G is always a directed acyclic graph. Moreover, using the algorithm for computing strongly connected components, we can compute SCC graph in $O(m + n)$ time only.

   The concept of SCC graph is very useful in solving various problems. Think over it. For a better understanding of SCC graph and to know its importance, the instructor has appended a few slides at the end of Lecture 14. Please go through these slides carefully. Using the concept of SCC graph or otherwise, solve the following problem

   You are given a weighted directed graph $G = (V, E)$ that models the road network of a country. Each node corresponds to a city and an edge (u, v) represents a direct road from city u to city v. The label on each road is a positive real number that represents the toll tax that a vehicle has to pay if the vehicle travels along that road. After the election in the country, the new government decided to remove the toll tax for all those roads that connect cities of the same state. You are provided with a list of pairs $\{(u, s(u)) | u \in V\}$ such that $s(u)$ is the state to which the city u belongs. Design an $O(m + n)$ time algorithm to compute the shortest path from a given source s to a destination t in this graph.

Only for fun (not for exams)

(**Eulerian tour**)

A strongly connected directed graph is said to be Eulerian if there exists a tour which starts from a node, traverses each edge exactly once, and returns to the same node. It is well known that a directed graph is Eulerian if and only if for each vertex the number of incoming edges is the same as the number of outgoing edges. Design an $O(m+n)$ time algorithm to determine if a graph is Eulerian. If the graph is found to be Eulerian, you must print an Euler tour of the graph.