Design and Analysis of Algorithms (CS345A)

Practice Sheet on DFS traversal in Directed Graphs

The hints to selected problems are given on the 3rd page.

1. (**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 ?

2. (**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.

3. (**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.

4. (**Least toll tax path**)

   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. The country consists of various states. Two cities belong to the same state if each of them is reachable from the other by a directed path. 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. Design an $O(m+n)$ time algorithm to compute the least toll tax path from a given source s to a destination t in this graph.

5. (**Least Label Vertex**)

   There is a directed graph $G = (V, E)$ where each vertex has a weight which is a real number. For each vertex v, you wish to compute the least label vertex reachable from it. Design an $O(m + n \log n)$ time algorithm which outputs n pairs $\{(v, L(v))|v \in V\}$, where $L(v)$ denotes the least label vertex reachable from v. Can you design an algorithm for this problem which runs in $O(m + n)$ time ?

6. (**Semi-connected graphs**)

   A directed graph $G = (V, E)$ is called semi-connected if for each $u, v \in V$, either there is a path from u to v or there is a path from v to u. Design an $O(|E|)$ time algorithm to determine if a directed graph is semi-connected.

Hints

Problem 3. The answer is NO. For the counterexample, try to draw 2 cycles suitably so that they overlap at some edges.

Problem 4. Make use of the SCC graph.

Problem 5. For $O(m + n \log n)$ time algorithm, sort the vertices in decreasing order of labels, reverse the edges of the graph, and then ... For the $O(m + n)$ time algorithm, make use of the SCC graph.

Problem 6. Make use of the SCC 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.