Design and Analysis of Algorithms

Practice sheet : **DFS traversal of directed graphs**

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 u and v are strongly connected, then there must be a cycle passing through them in G.

   (c) 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 forest.

   (d) 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.

   (e) We can construct a directed graph 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. (**Constructing DFS tree using start and finish times of the DFS traversal**)

   Let $G = (V, E)$ be a directed graph on $n = |V|$ vertices and $m = |E|$ edges. We perform a DFS traversal on G and obtain arrays D and F for storing start time and finish time of DFS traversal. Let $\mathcal{F}$ be the DFS Forest. Unfortunately $\mathcal{F}$ is lost. Can you re-construct it using arrays D and F ? If so, design the most efficient algorithm for this task. Can you reconstruct $\mathcal{F}$ even after G is lost ? If so, design the most

efficient algorithm for this task.
Hint: You need to determine, for each vertex v, its parent, if exists, in the DFS tree containing it. To achieve this objective, focus on how D and F arrays are populated during the DFS traversal. Note that you may sort vertices based on start time and/or finish time in $O(n)$ time. You might like to use an elementary data structure you must have learnt in ESO207 if you wish to design an elegant algorithm.

4. (**Least label vertex**)

   There is a directed graph $G = (V, E)$. There is also an array L such that $L[v]$, called label of v, stores a real number. For each vertex v, you wish to compute the label of the least label vertex reachable from it.

   (a) Design an $O(m+n \log n)$ time algorithm which outputs n pairs $\{(v, minL(v)) | v \in V\}$, where $\min L(v)$ denotes the least label vertex reachable from v.
   Hint: You might like to (1) sort the vertices according to their label, (2) make some changes on G, and (3) use DFS or BFS traversal.

   (b) Can you design an algorithm for this problem which runs in $O(m + n)$ time ?
   Hint: You should use a tool (discussed in the lecture) which is based on the strongly connected components.

5. (**Singly-connected graph**)

   Let $G = (V, E)$ be a directed graph. A pair of vertices $u, v \in V$ are said to be singly connected if the following condition holds true. Either there is a path from u to v or there is a path from v to u. Obviously, a pair of strongly connected vertices are always semi-connected. A graph is said to be semi-connected if each pair of vertices in G are semi-connected. Design an $O(|E|)$ time algorithm to determine if a directed graph is semi-connected.
   Hint: You should use a tool (discussed in the lecture) which is based on the strongly connected components.

6. (**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. The algorithm must take $O(m + n)$ time.