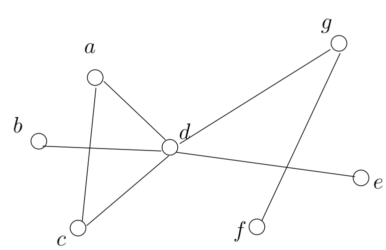
COMP3506/7505: Regular Exercise Set 11

Prepared by Yufei Tao

Problem 1 (Correctness of the White Path Theorem) Consider performing DFS on a directed graph $G = (V, E)$. Then, both of the following statements are true:

- Suppose that when a vertex u is discovered, there is still a white path from u to a vertex v (namely, we can hop from u to v while stepping on only white vertices). Then, v must be a descendant of u in the DFS forest.
- Conversely, if v is a descendant of u in the DFS forest, then there must be a white path from u to v at the moment when u is discovered.

Problem 2 (DFS on Undirected Graphs). Let $G = (V, E)$ be an undirected graph. Consider the execution of DFS on G. The algorithm runs in exactly the same way as DFS on a directed graph. The only difference is that, a vertex u is popped out of the stack, only if none of its neighbors (instead of out-neighbors) is still white. Give a possible DFS tree produced if we (i) start DFS on a in the following graph, and (ii) follow the convention that we explore the neighbors of a vertex in alphabetic order.



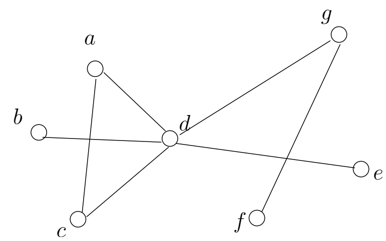
Problem 3 (No Cross Edges in Undirected DFS). Let $G = (V, E)$ be an undirected graph. Consider the DFS forest produced by running DFS on G (assuming arbitrary starting and re-starting vertices). Let $\{u, v\}$ be an edge in G (note that we use the notation $\{u, v\}$, instead of (u, v), to emphasize that the edge has no directions). Prove: either u is an ancestor of v, or v is an ancestor of u.

Remark: Because of this lemma, we can classify each edge $\{u, v\}$ in G as follows:

- *Tree edge*: if u is the parent of v or v is the parent of u.
- *Back edge*: otherwise.

Problem 4 (Undirected Cycle Detection). Let $G = (V, E)$ be an undirected graph. A *cycle* is a sequence of edges $\{v_1, v_2\}, \{v_2, v_3\}, ..., \{v_{t-1}, v_t\}$ where $v_t = v_1$. Adapt DFS to design an algorithm to detect whether G has a cycle in $O(|V| + |E|)$ time.

Problem 5 (Articulation Vertex).** Let $G = (V, E)$ be an undirected graph that is connected (i.e., there is a path between any two distinct vertices). A vertex $u \in V$ is called an *articulation vertex* if the following is true: G becomes disconnected after removing u and all the edges of u. For example, in the figure below, vertex g is an articulation, and so is d. No other vertices are articulation vertices.



Consider any DFS tree on G. Prove:

- If a vertex u is a leaf in the DFS tree, it cannot be an articulation vertex.
- Let u a vertex that is neither a leaf in the DFS tree nor the root. It is an articulation vertex if and only if the following is true:
  - There is at least one child v of u, such that no back edge connects a descendant of v to a proper ancestor of u.
- Let u be the root of a DFS tree. It is an articulation vertex if and only if it has at least two child nodes in the DFS tree.

Problem 6* (Finding an Articulation Vertex). Let $G = (V, E)$ be an undirected graph that is connected. Design an algorithm to determine whether G has any articulation vertex. Your algorithm must finish in $O(|V| + |E|)$ time.

Problem 7 (The L-Ordering Lemma of the SCC Algorithm). Prove the lemma on Slide 28 of the lecture notes about strongly connected components (SCCs). Let S_1, S_2 be SCCs such that there is a path from S_1 to S_2 in G^{SCC}. In the ordering of L, the earliest vertex in S_2 must come before the earliest vertex in S_1.

Problem 8. Prove that for any directed graph $G = (V, E)$, the SCC decomposition is unique. Namely, there is only one way to decompose V into disjoint subsets, each of which is an SCC; and furthermore, such a decomposition always exists.