CSCI2100: Regular Exercise Set 12

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:

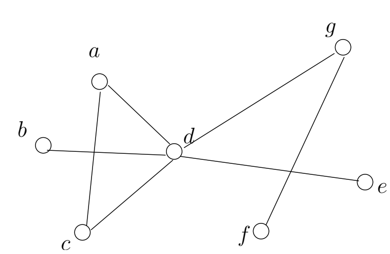
- Suppose that when a vertex u is discovered, there is still a white path from u to a vertex v (namely, we can hop from u to v while stepping on only white vertices). Then, v must be a descendant of u in the DFS forest.
- Conversely, if v is a descendant of u in the DFS forest, then there must be a white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

Solution.
Proof of the First Statement. Let π be the path from u to v. We will prove that all the vertices on π must be descendants of u in the DFS forest. Suppose that this is not true. Let v' be the first vertex on π—in the order from u to v—that is not a descendant of u. Clearly, $v' \neq u$. Let u' be the vertex that precedes v' on π.

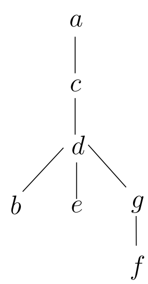
Consider the moment before u' turns black. As u' is a descendant of u in the DFS forest, we know that u is in the stack currently. The color of v' cannot be white—otherwise, DFS must now push v' into the stack, which is a contradiction of the fact that u' is turning black. On the other hand, if v' is either gray or black, it means that v must have been pushed into the stack while u still remains in the stack. This contradicts the fact that v is not a descendant of u.

Proof of the Second Statement. As v is a descendant of u, there is a moment in DFS when u and v were both in the stack with v being the top of the stack. It thus follows that there is a white path from u to v 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.



Solution.



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:

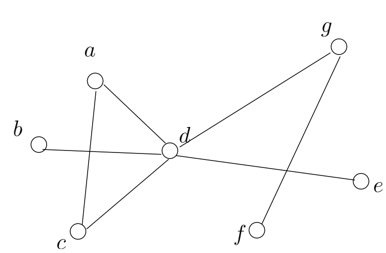
- *Tree edge*: if u is the parent of v or v is the parent of u.
- *Back edge*: otherwise.

Solution. The white path theorem—as stated in Problem 1—still holds for undirected DFS (the same proof applies here as well). Between u and v, let u be the vertex discovered first. Then, the white path theorem says that v must be a descendant of u.

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.

Solution. Perform DFS on G. Declare cycle presence if and only if a back edge is found. For example, in the Solution of Problem 2, there is such an edge $\{a, d\}$, which implies a cycle.

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.

Solution.
Proof of the First Bullet.
Suppose that u is an articulation vertex. Let s be the starting vertex of the DFS. Then there must be a vertex u' such that all the paths from s to u' must go by way of u. This implies that, when v is discovered by DFS, there must be a white path from u to u'. The white path theorem then says that u' must be a descendant of u, contradicting the fact that u is a leaf.

Proof of the Second Bullet.
Only-if direction. Imagine removing u from G, which should disconnect G. Let $C_1, C_2, ..., C_t$ for some $t \geq 2$ be the connected components (CCs) of the resulting graph (recall that a CC is a set of vertices that are reachable from each other). Without loss of generality, assume that s belongs to C_1. Consider the moment right before the first vertex v in C_2 is discovered. It must be a child of u in the DFS tree (because any path from s to u must cross the edge $\{u, v\}$). At this moment, all the vertices in C_2 must be white; and they are the only vertices that v can reach via white paths. Hence, all the vertices of C_2 must be the *only* descendants of v. It thus follows that there can be no back edge connecting a descendant of v to a proper ancestor of u.

If direction. We will prove that, after u is removed from G, s can no longer reach v, which thus indicates that u is an articulation vertex. Suppose, on the contrary, that u can still access v by a path π (that does not contain u). Denote the vertices on π as $v_1, v_2, ..., v_x$ with $v_1 = s$ and $v_x = v$. Let v_i (for some $i \in [1, x]$) be the *last* vertex on π that is an ancestor of u. We will prove that v_{i+1} must be a descendant of v, making $\{v_i, v_{i+1}\}$ a back edge that connects a descendant of v to a proper ancestor of u, which contradicts the fact that no such back edges exist.

Consider the moment right before the discovery of v. We argue that the colors of $v_{i+1}, v_{i+2}, ..., v_{x-1}$ must all be white at this moment:

- First, none of them can be gray—otherwise, such a vertex must be an ancestor of u (because u is the parent of v), contradicting the definition of v_i.

- If v_{i+1} is black, it means that v_{i+1} was discovered before v. Furthermore, when v_{i+1} turned black, v_{i+2} cannot be white (otherwise, DFS would have crossed the edge $\{v_{i+1}, v_{i+2}\}$ to push v_{i+2} into the stack). Thus, at this moment, v_{i+2} must be black (as mentioned, v_{i+2} cannot be gray currently). Following the same argument, we obtain that $v_{i+3}, v_{i+4}, ..., v_x$ must all be black at the moment. However, this contradicts the fact that $v_x = v$ is white.

- The same argument proves that none of $v_{i+2}, v_{i+3}, ..., v_{x-1}$ can be black.

Therefore, all of $v_{i+1}, v_{i+2}, ..., v_{x-1}$ must be descendants of v.

Proof of the Third Bullet.
Only-if direction. Vertex u is the starting vertex of DFS. Imagine removing u from G, which should disconnect G into CCs $C_1, C_2, ..., C_t$ for some $t \geq 2$. Let v be the second vertex discovered by DFS (i.e., right after u). Without loss of generality, suppose that $v \in C_1$. Then, when v is discovered,

there is no white path from v to any vertex in C_2. Hence, none of the vertices in C_2 can be descendants of v, implying that u must have another child.

If direction. Let v be the second vertex discovered by DFS (i.e., right after u). Let v' any other child of u in the DFS tree. We will prove that any path from v to v' must go through u, which indicates that u is an articulation vertex.

Assume that there is a path π from v to v' that does not go through u. Then, when v is discovered, there is a white path from v to v', which means that v' must be a descendant of v in the DFS tree. This contradicts the fact that v' and v are siblings.

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.

Solution. First grow a DFS-tree T, but make sure that at each node u we record its level (the root is at level 0), denoted as *level*(u). We now process the vertices of T in a bottom-up manner (i.e., descending order of level). Let u be a vertex to be processed next. We do the following:

- *Case 1: u is a leaf node*: We inspect all the edges $\{u, v\}$ of u, and obtain:

$$\textit{highest-back-level}(u) = \min_{\text{all } \{u,v\}} \textit{level}(v).$$

- *Case 2: u is an internal node but not the root*: Let $v_1, v_2, ..., v_t$ be its children (which have already been processed). If

$$\max_{i=1}^{t} \textit{highest-back-level}(v_i) \geq \textit{level}(u)$$

  we report u as an articulation vertex, and finish.

  Otherwise, inspect all the edges $\{u, v\}$ of u, and obtain:

$$\textit{highest-back-level}(u) = \min_{\text{all } \{u,v\}} \textit{level}(v).$$

  Then, update *highest-back-level*(u) to be:

$$\min \left\{ \textit{highest-back-level}(u), \min_{i=1}^{t} \textit{highest-back-level}(v_i) \right\}.$$

- *Case 3: u is the root*: Report u as an articulation vertex if it has at least 2 child nodes.