Depth First Search  (undirected graphs)

DFSmain(G) {
    for each u in V do // Initialize
        color[u] = white;
        parent[u] = NULL;
        time = 0;
    for each u in V do
        if (color[u] == white) DFSVisit(u); // start a new tree
    }

DFSVisit(u) {
    color[u] = red; // u is discovered
    d[u] = ++time; // set u’s discover time
    for each v in Adj(u) do
        if (color[v] == white) {
            parent[v] = u; // edge (u,v) is a tree edge
            DFSVisit(v); // visit unvisited vertices
        }
    color[u] = black; // u has finished
    f[u] = ++time; // set u’s finish time
    }

Theorem 26.1 (Properties of DFS on Undirected Graphs)

Let $G$ be an undirected graph with $n$ vertices and $m$ edges. Then $\text{DFS}(G)$ runs in linear time, i.e., $O(n + m)$. DFS computes connected components of $G$ and determines which of these components is cyclic.

A component is cyclic iff DFS discovers a backedge.

**Proof:** To see that DFS runs in linear time, i.e., $O(n + m)$, note that DFSmain performs a constant number of steps per vertex, i.e., a total of $O(n)$ steps. Furthermore, DFSVisit is called exactly once for each vertex either from DFSmain or from DFSVisit. DFSVisit($v$) performs a bounded number of steps except for walking down $v$’s adjacency list, which is done once for each vertex $v$. Thus each edge is examined once in each direction, for a total of $O(m)$ steps.

We claim that the trees of the $\text{DFS}$ forest are exactly the connected components of $G$. To see this, it suffices to show that all the vertices reachable from $r$ are included in the DFS tree whose root is $r$. We prove this by induction on the number of vertices in $r$’s connected component.

**base case:** If $r$’s connect component has size 1, then the whole connected component is visited at the first step of DFSVisit($r$).

**inductive case:** Assume the indHyp which says that for all connected undirected graphs, $H$, with at most $n_0$ vertices, and all vertices $v$ from $H$, if we start with all vertices white, then DFSVisit($v$) visits all of $H$.

Assume that $G$ is an arbitrary connected undirected graph with $n_0 + 1$ vertices. With all vertices white, call DFSVisit($r$), for some vertex $r$ of $G$. Let $a$ be the first neighbor of $r$. Let $H_1$ be the set of vertices reachable from $a$ without going through $r$. Then $H_1$ is connected and has at most $n_0$ vertices, so by the indHyp the call to DFSVisit($a$) visits all of $H_1$. When we return to $r$, let $H_2 = G - H_1$. Then the remainder of DFSVisit($r$) is the same as if $H_1$ didn’t exist and we are just doing the DFS of $H_2$. By the indHyp, this remaining DFS visits all of $H_2$. Thus the initial call of DFSVisit($r$) visits all of $G = H_1 \cup H_2$.

Finally, we show that $G$ is cyclic iff $\text{DFS}(G)$ finds a backedge. One direction is obvious: if there is a backedge from $b$ to $a$, then there is a path in the tree from $a$ down to $b$, so this path plus the backedge forms a cycle.

Conversely, suppose that $G$ contains a cycle $C = a_1, a_2, \ldots, a_{k-1}, a_1$ and let $a_1$ be the first vertex of the cycle that is visited in $\text{DFS}(G)$. By the previous argument, we know that $a_{k-1}$ is visited during the call of DFSVisit($a_1$). But when $a_{k-1}$ is visited, $a_1$ is red, and thus the edge $(a_{k-1}, a_1)$ is a backedge. $\square$