1. Given an undirected graph \( G = (V, E) \), the components of \( G \) partition \( V \) into nonempty, nonintersecting subsets \( C_1, C_2, \ldots, C_k \) of mutually connected vertices. That is,

- \( C_i \neq \emptyset \) for \( 1 \leq i \leq k \);
- \( u, v \in C_i \Rightarrow \exists \) path from \( u \) to \( v \) via edges in \( E \);
- \( u \in C_i, v \in C_j, i \neq j \Rightarrow \exists \) path from \( u \) to \( v \) via edges in \( E \);
- \( V = \bigcup_{i=1}^{k} C_i \).

Assume that each vertex contains an attribute \( c \) that will, after a depth-first search scan, contain the number of the component containing the vertex. Consequently when the algorithm terminates, each vertex \( u \) has \( u.c = i \) for some \( 1 \leq i \leq k \). The number of components, \( k \), is not known when the algorithm starts. Modify the depth-first search of the text such that each vertex contains its component mark on termination. That is, all vertices within a component will have the same component mark, and no two vertices in different components will have the same mark.

2. Give an example of a directed graph, \( G = (V, E) \), containing a path from \( u \) to \( v \), but for which the DFS algorithm assigns \( u.f < v.d \). That is, the algorithm exhausts the adjacency list of \( u \) before discovering \( v \). To make this happen, you will have to specify the order of edges on each adjacency list, and also the starting vertex of the for-loop in lines 5–7 of the textbook DFS algorithm.

3. Give an example of a directed graph, \( G = (V, E) \), containing a path from \( u \) to \( v \), for which DFS assigns \( u.d < v.d \), but for which \( v \) is not a \( G_\pi \)-descendant of \( u \). Again, you will need to specify the order of edges on the adjacency list and the starting vertex to force DFS to behave in the desired manner.

Solutions.

1. When DFS chooses a white vertex, that choice starts a new \( G_\pi \) tree and consequently starts a new component in an undirected graph which can contain no forward or cross edges. So, we simply put a counter in the main loop to keep track of the number of new \( G_\pi \) trees discovered so far. The code appears on the next page.

2. The graph appear on the next page.

3. An example is provided by the same graph used in Problem 2.
DFS(G)
    for u in G.V loop
        u.color = white;
        u.pi = nil;
        u.c = 0; // component attribute
    end loop;
    time = 0;
    counter = 0;
    for u in G.V loop
        if u.color == white then
            counter = counter + 1;
            DFS-Visit(G, u);
        end if;
    end loop;
end DFS;

DFS-Visit(G, u)
    time = time + 1;
    u.d = time;
    u.color = gray;
    u.c = counter;
    for v in G.Adj(u) loop
        if v.color == white then
            v.pi = u;
            DFS-Visit(G, v);
        end if;
    end loop;
    u.color = black;
    time = time + 1;
    u.f = time;
end DFS-Visit;

Graph for problems 2 and 3.