CSI 503 – Data Structures and Algorithms
Applications of Depth-First Search

Handout 15.2

(a) Pseudocode for finding connected components

**Note:** The algorithm is almost same as DFS itself. The algorithm does not use discovery times or finish times of nodes. For each node \( u \), it computes \( CC[u] \), the number of the connected component containing \( u \).

**CONNECTED-COMP**(\( G \))

1. **for** each vertex \( u \in V \) **do**
   
   Color\([u]\) = white.

2. \( t = 0 \). (At the end, \( t \) has the number of connected components.)

3. **for** each vertex \( u \in V \) **do**
   
   if \( (\text{Color}[u] = \text{white}) \) **then**
   
   \( t = t + 1; \ CC[u] = t. \)
   
   **DFS-Visit**(\( u \)).

**DFS-Visit**(\( u \))

1. Color\([u]\) = gray.

2. **for** each vertex \( v \in \text{Adj}[u] \) **do**

   if \( (\text{Color}[v] = \text{white}) \) **then**

   \( CC[v] = t; \)
   
   **DFS-Visit**(\( v \)).

3. Color\([u]\) = black. (Vertex \( u \) is finished.)

(b) Pseudocode for topological sort

**Note:** The algorithm is based on the fact that the topological sort of an acyclic graph is obtained by decreasing order of finish times of vertices when a DFS is carried out.

1. Initialize linked list \( L \) to empty.

2. Call **DFS**(\( G \)) and compute \( f[u] \) for each vertex \( u \in V \).

3. As each vertex is finished, insert it at the front of \( L \).

4. Return \( L \).