CSI 604 – Spring 2016
Applications of Depth-First Search

(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 \( u \).

\[
\text{CONNECTED-COMP}(G)
\]

1. \textbf{for} each vertex \( u \in V \) \textbf{do}
   \begin{itemize}
   \item \text{Color}[u] = \text{white}.
   \end{itemize}
2. \( t = 0 \). (At the end, \( t \) has the number of connected components.)
3. \textbf{for} each vertex \( u \in V \) \textbf{do}
   \begin{itemize}
   \item if (\text{Color}[u] = \text{white}) \textbf{then}
   \item \( t = t + 1; \ \text{CC}[u] = t \).
   \item \text{DFS-Visit}(u).
   \end{itemize}

\[
\text{DFS-Visit}(u)
\]

1. \text{Color}[u] = \text{gray}.
2. \textbf{for} each vertex \( v \in \text{Adj}[u] \) \textbf{do}
   \begin{itemize}
   \item if (\text{Color}[v] = \text{white}) \textbf{then}
   \item \text{CC}[v] = t;
   \item \text{DFS-Visit}(v).
   \end{itemize}
3. \text{Color}[u] = \text{black}. (Vertex \( u \) is finished.)

(b) Pseudocode for topological sort of a dag:

Note: The algorithm is based on the fact that the topological sort of a dag 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 \( \text{DFS}(G) \) and compute \( f[u] \) for each vertex \( u \in V \).
3. As each vertex is finished, insert it at the \textit{front} of \( L \).
4. Return \( L \).