(a) Pseudocode for Breadth-First Search

BFS(G, s) // G is the graph and s is the source vertex.
1. for each u ∈ V − {s} do
   Color[u] = white;  d[u] = ∞;  π[u] = NULL.
2. Color[s] = gray;  d[s] = 0;  π[s] = NULL.
3. Initialize Q to contain only vertex s.
4. while (Q is not empty) do
   (a) u = head[Q].
   (b) for each v ∈ Adj[u] do
       if (Color[v] = white) then
           Enqueue(Q, v).
       (c) Dequeue(Q).  // This removes u from Q.
   (d) Color[u] = black.

Print-Path(G, s, v) // Prints the path in the BFS tree from s to v.
Idea: Follow the parent pointers from v to s; recursion allows us to print the path in the right order.
1. if (v = s)
   then print s
   else
       if (π[v] = NULL)
           then Print “No path from s to v”.
           else
               Print-Path(G, s, π[v]).
               Print v.
(b) Pseudocode for Depth-First Search

DFS(G) // Depth-first search of graph G(V,E).

Note: Procedure DFS uses procedure DFS-Visit described below.

1. for each vertex \( u \in V \) do
   \[ \text{Color}[u] = \text{white}; \ \pi[u] = \text{NULL}. \]
2. time = 0. // Note: time is a global variable.
3. for each vertex \( u \in V \) do
   if (Color\[u\] = white) then
      DFS-Visit\( (u) \).

DFS-Visit\( (u) \)

1. Color\[u\] = gray; time = time+1; \( d[u] = \text{time}. \) // \( u \) has just been discovered.
2. for each vertex \( v \in \text{Adj}[u] \) do (Edge \( \{u, v\} \) is explored.)
   if (Color\[v\] = white) then
      \( \pi[v] = u; \)
      DFS-Visit\( (v) \).
3. Color\[u\] = black. // Vertex \( u \) is finished.
4. time = time + 1; \( f[u] = \text{time}. \)