G – graph, adjacency list of vertices

```
DFS_Recurs(G)
      DFS_Clear(G)
      for each vertex u∈G
            if color[u]==WHITE
                  DFS_Visit(u)
            end-if
      end-for
end-DFS_Recurs


DFS_Visit(u)
      color[u] = GRAY
      time = time + 1
      first[u] = time
      for each vertex v∈Adj(u)
            if color[v]==GRAY and pred[u]!=v and v!=u
                  hasCycle = true;
                  startCycle = u
                  endCycle = v
            end-if
            if color[v]==WHITE
                  pred[u] = v
                  DFS_Visit(v)

            end-if
      end-for
      color[u] = BLACK
      time = time + 1
      last[u] = time
end-DFS_Visit


DFS_Clear(G)
      hasCycle = false
      time = 0
      for each vertex u∈G
            color[u] = WHITE
            pred[u] = NIL
            first[u] = 0
            last[u] = 0
      end-for
end-DFS_Clear
```

```
DFS_Path(G, u, v)
      DFS_Clear(G)
      path = ""
      DFS_Visit(u)
      if  color[v] != WHITE
            path = path + v
            v = pred[v]
            while (v != NIL)
                  path = v + "->" + path
                  v = pred[v]
            end-while
      end-if
      return path;
end-DFS_Path
```

```
DFS_Cycle(G)
      DFS_Recurs(G)
      cycle = ""
      if hasCycle
            DFS_Clear(G)
            DFS_Visit(startCycle)
            cycle = dfsPath(startCycle, endCycle)
      end-if
      cycle = cycle + "->" + endCycle
      return cycle
end-DFS_Cycle
```

```
DFS_Loop(G)
      for each vertex u∈G
            color[u] = WHITE
            pred[u] = NIL
            first[u] = 0
            last[u] = 0
      end-for
      time = 0
      stack<-empty
      for each vertex u∈G
            dfs_loop(G, u)
      end-for
end-DFS_Loop


dfs_loop(G, u)
      hasCycle = false
      color[u] = GRAY
      pred[u] = NIL
      time = time + 1
      first[u] = time
      stack.push(u);
      v = u.next
      while (stack not empty)
            if color[v]==GRAY and pred[u]!=v and v!=u
                  hasCycle = true;
            end-if
            if color[v]==WHITE
                  color[v] = GRAY
                  pred[v] = u;
                  u = v;
                  time = time + 1
                  first[v] = time
                  stack.push(v)
                  u = G[v]
                  v = u.next
            else
                  if v.next ≠ null then v = v.next
                  else
                        u = stack.pop();
                        color[u] = BLACK
                        time = time + 1
                        last[u] = time
                        u = pred[u]
                        if u ≠ NIL
                              u = G[u];
                              v = u.next;
                        end-if
                  end-if
            end-if
      end-while
end-dfs_loop
```