```
def DFS(g, u, discovered):
  """Perform DFS of the undiscovered portion of Graph g starting at Vertex u.

  discovered is a dictionary mapping each vertex to the edge that was used to
  discover it during the DFS. (u should be "discovered" prior to the call.)
  Newly discovered vertices will be added to the dictionary as a result.
  """
  for e in g.incident_edges(u):    # for every outgoing edge from u
    v = e.opposite(u)
    if v not in discovered:        # v is an unvisited vertex
      discovered[v] = e            # e is the tree edge that discovered v
      DFS(g, v, discovered)        # recursively explore from v

def construct_path(u, v, discovered):
  """
  Return a list of vertices comprising the directed path from u to v,
  or an empty list if v is not reachable from u.

  discovered is a dictionary resulting from a previous call to DFS started at u.
  """
  path = []                        # empty path by default
  if v in discovered:
    # we build list from v to u and then reverse it at the end
    path.append(v)
    walk = v
    while walk is not u:
      e = discovered[walk]         # find edge leading to walk
      parent = e.opposite(walk)
      path.append(parent)
      walk = parent
    path.reverse()                 # reorient path from u to v
  return path

def DFS_complete(g):
  """Perform DFS for entire graph and return forest as a dictionary.

  Result maps each vertex v to the edge that was used to discover it.
  (Vertices that are roots of a DFS tree are mapped to None.)
  """
  forest = {}
  for u in g.vertices():
    if u not in forest:
      forest[u] = None             # u will be the root of a tree
      DFS(g, u, forest)
  return forest
```