Topological Sort

```
# topSort() and dfsLabelling() both refer
# to global counter dfsCounter which will be
# incremented during the topological sort.
# It will be used to assign an orderNumber to
# each node in the graph.
dfsCounter = 1

# topSort() performs a topological sort on
# a directed, possibly cyclic graph.

def topSort(graph):

  # Mark all nodes in the graph as un-visited
  for node in graph:
    node.visited = FALSE

  # Some topSort algorithms start from a "root" node
  # (defined as a node with in-degree = 0).
  # As we need to use topSort() on cyclic graphs (in our strongComp
  # algorithm), there may not exist such a "root" node.
  # We will keep starting a dfsLabelling() from any node in
  # the graph until all nodes have been visited.
  for node in graph:
    if not node.visited:
      dfsLabelling(node)

# dfsLabelling() does a depth-first traversal of a possibly
# cyclic directed graph. By marking nodes visited upon first
# encounter, we avoid infinite looping.

def dfsLabelling(node, graph):
  #  if the node has already been visited, the recursion stops here
  if not node.visited:

    # avoid infinite loops
    node.visited = TRUE

    # visit all neigbours first (depth first)
    for neigbour in node.out_neigbours:
      dfsLabelling(neighbour, graph)

    # label the node with the counter and
    # subsequently increment it
    node.orderNumber = dfsCounter
    dfsCounter += 1
```