

① Algorithm connectedComponents(G)
 return DFS(G)

Algorithm initResult(G)
 $S :=$ new Sequence

Algorithm postComponentVPset(G, v)
 $S.$ InsertLast(v)

Algorithm result(G)
 return S

Algorithm find Path (G, s, L)

Start $\leftarrow s$

dest $\leftarrow L$

return BFS(G)

Algorithm initResult (G)

S new stack

Path new list stack with

Algorithm isNextComponent (G, v)

return start == v

Algorithm postDisc EdgeVisit (G, v, e, w)

if $S.top \neq dest$ then
 $S.push(e)$
 $S.push(w)$

Algorithm finishBFScomponent (G, s)

top := $S.Top()$

if top = dest then

~~path.insertLast~~

while ! $S.IsEmpty$ do

vertex := $S.pop()$

edge := $S.pop()$

if getLabel(edge) = 'Discovery' then

path.push(vertex)

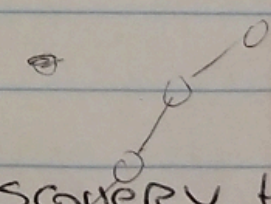
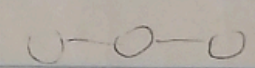
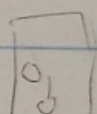
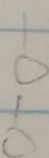
path.push(edge)

path.push(start)

Algorithm result (G)

if path.size = 0 then return NO_SUCH_PATH

else path.elements()



CYCLE BFS

Algorithm find Cycle (G)

return BFS (G)

Algorithm Init Result (G)

cycle Found := false

cycle := new Sequence

Algorithm result (G)

If cycle Found then

return cycle

return NO-SUCH-CYCLE

Algorithm preDiscEdge Visit (G, v, e, w)

set Parent (w, e)

Algorithm cross Edge Visit (G, v, e, w)

If !cycle Found then

build Cycle (G, v, e, w)

Algorithm build Cycle (G, v, e, w)

cycle.InsertLast (w)

left := v

right := w

cycle.InsertFirst (e)

cycle.InsertFirst (v)

while left \neq right do

e1 := getParent (left)

If e1 $\neq \emptyset$ then

left := G.opposite (e1, left)

cycle.InsertFirst (e1)

cycle.InsertFirst (left)

e2 := getParent (right)

If e2 $\neq \emptyset$ then

right := G.opposite (e2, right)

cycle.InsertLast (e2)

cycle.InsertLast (right)

cycle Found := true

Algorithm counter Met(G)

return BFS(G)

Algorithm initResult(G)

counter $\leftarrow 0$

Algorithm pos Component Visit(G, v)

counter++

Algorithm startBFSComponent(G, s)

setLabel(s , counter)

Algorithm postDiscEdgeVisit(G, v, e, w)

setLabel(w , counter)

Algorithm result(G)

return G