|  |  |
| --- | --- |
| Breadth-First-Search(Graph, root):  2  3 for each node n in Graph:  4 n.distance = INFINITY  5 n.parent = NIL  6  7 create empty queue Q  8  9 root.distance = 0  10 Q.enqueue(root)  11  12 while Q is not empty:  13  14 current = Q.dequeue()  15  16 for each node n that is adjacent to current:  17 if n.distance == INFINITY:  18 n.distance = current.distance + 1  19 n.parent = current  20 Q.enqueue(n)  create a queue Q  mark v as visited and put v into Q  while Q is non-empty  remove the head u of Q  mark and enqueue all (unvisited)  neighbours of u | 1 **procedure** DFS-iterative(*G*,*v*):  2 let *S* be a stack  3 *S*.push(*v*)  4 **while** *S* is not empty  5 *v* = *S*.pop()  6 **if** *v* is not labeled as discovered:  7 label *v* as discovered  8 **for all** edges from *v* to *w* **in** *G*.adjacentEdges(*v*) **do**  9 *S*.push(*w*) |

***Push* (Stk, VStart)**

**While Stk Is Not *Empty***

***Vtemp := Pop* (Stk)**

**If VTemp is *Not* *Marked***

***Process* VTemp**

***Mark* VTemp**

**For Each VAdj Adjacent to VTemp**

**Push (Stk, VAdj)**

**End For**

**End IF**

**End While**

***Enqueue* (Q, VStart)**

**While Q Is Not *Empty***

***Dequeue* (Q, VTemp)**

**If VTemp is Not *Processed***

***Process* VTemp**

***Mark* VTemp**

**For Each VAdj Adjacent to VTemp**

***Enqueue* (Q, VAdj)**

**End For**

**End IF**

**End While**