

Global initialization : $v.\text{dfsnum} := -1$ for all v .

```

ArticPointDFS (Vertex  $v$ ) {
     $v.\text{dfsnum} := \text{dfsCounter}++$ 
     $v.\text{low} := v.\text{dfsnum}$ 
    for each edge  $(v, x)$  do {
        if  $(x.\text{dfsnum} == -1)$  then do { //  $x$  is undiscovered
             $x.\text{dfslevel} := v.\text{dfslevel}++$ 
             $v.\text{numChildren} := v.\text{numChildren}++$ 
            stack.push edge  $(v, x)$  // add this edge to the stack
            ArticPointDFS  $(x)$  // recursively perform DFS at children nodes
             $v.\text{low} := \min(v.\text{low}, x.\text{low})$ 
            if  $(v.\text{dfsnum} == 1)$  then do {
                // Special Case for root :
                // Root is an artic. point iff there are two or more children
                if  $(v.\text{numChildren} \geq 2)$  then do articPointList.add  $(v)$ 
                // Retrieve all edges in a biconnected component
                while  $(\text{stack.top} \neq (v, x))$  do bccEdgeList.add  $(\text{stack.pop})$ 
            }
            else {
                if  $(x.\text{low} \geq v.\text{dfsnum})$  then do {
                    //  $v$  is artic. point separating  $x$ .
                    // Children of  $v$  cannot climb higher than  $v$  without passing through  $v$ .
                    articPointList.add  $(v)$ 
                    // Retrieve all edges in a biconnected component
                    while  $(\text{stack.top} \neq (v, x))$  do bccEdgeList.add  $(\text{stack.pop})$ 
                }
            }
        }
    }
    else if  $(x.\text{dfslevel} < v.\text{dfslevel} - 1)$  then do {
        //  $x$  is at a lower level than the level of  $v$ 's parent,
        // equiv.  $(v, x)$  is a back edge
         $v.\text{low} := \min(v.\text{low}, x.\text{dfsnum})$ 
        stack.push edge  $(v, x)$  // add the back edge to the stack
    }
}

```