Algorithm DFS

* **Step 1:** SET STATUS = 1 (ready state) for each node in G
* **Step 2:** Push the starting node A on the stack and set its STATUS = 2 (waiting state)
* **Step 3:** Repeat Steps 4 and 5 until STACK is empty
* **Step 4:** Pop the top node N. Process it and set its STATUS = 3 (processed state)
* **Step 5:** Push on the stack all the neighbours of N that are in the ready state (whose STATUS = 1) and set their  
  STATUS = 2 (waiting state)  
  [END OF LOOP]
* **Step 6:** EXIT
* **Pseudocode**
* DFS-iterative (G, s): //Where G is graph and s is source vertex
* let S be stack
* S.push( s ) //Inserting s in stack
* mark s as visited.
* while ( S is not empty):
* //Pop a vertex from stack to visit next
* v = S.top( )
* S.pop( )
* //Push all the neighbours of v in stack that are not visited
* for all neighbours w of v in Graph G:
* if w is not visited :
* S.push( w )
* mark w as visited
* DFS-recursive(G, s):
* mark s as visited
* for all neighbours w of s in Graph G:
* if w is not visited:
* DFS-recursive(G, w)
* Algorithm BFS
* **Step 1:** SET STATUS = 1 (ready state)  
  for each node in G
* **Step 2:** Enqueue the starting node A  
  and set its STATUS = 2  
  (waiting state)
* **Step 3:** Repeat Steps 4 and 5 until  
  QUEUE is empty
* **Step 4:** Dequeue a node N. Process it  
  and set its STATUS = 3  
  (processed state).
* **Step 5:** Enqueue all the neighbours of  
  N that are in the ready state  
  (whose STATUS = 1) and set  
  their STATUS = 2  
  (waiting state)  
  [END OF LOOP]
* **Step 6:** EXIT
* **Pseudocode**
* BFS (G, s) //Where G is the graph and s is the source node
* let Q be queue.
* Q.enqueue( s ) //Inserting s in queue until all its neighbour vertices are marked.
* mark s as visited.
* while ( Q is not empty)
* //Removing that vertex from queue,whose neighbour will be visited now
* v = Q.dequeue( )
* //processing all the neighbours of v
* for all neighbours w of v in Graph G
* if w is not visited
* Q.enqueue( w ) //Stores w in Q to further visit its neighbour
* mark w as visited.