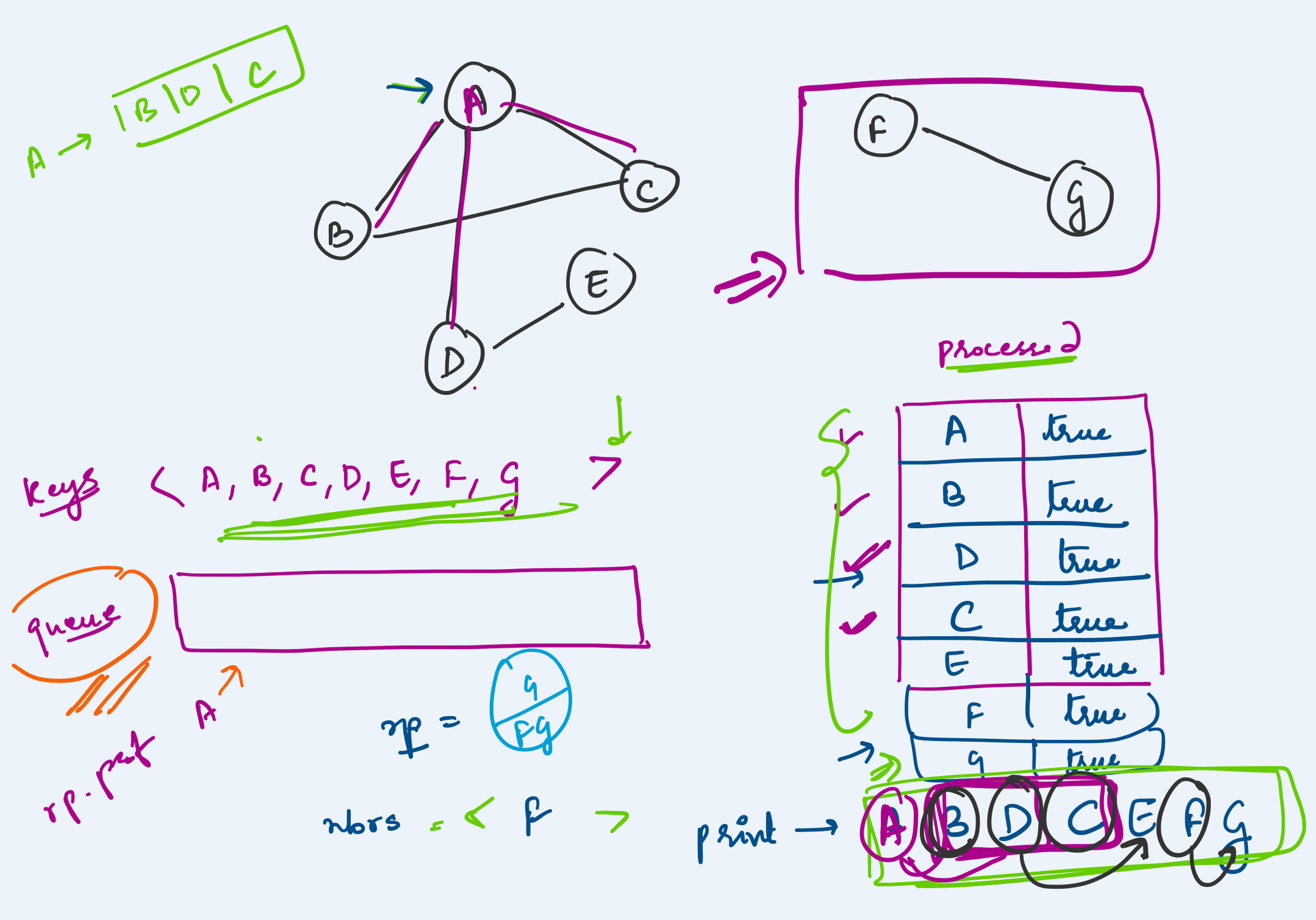
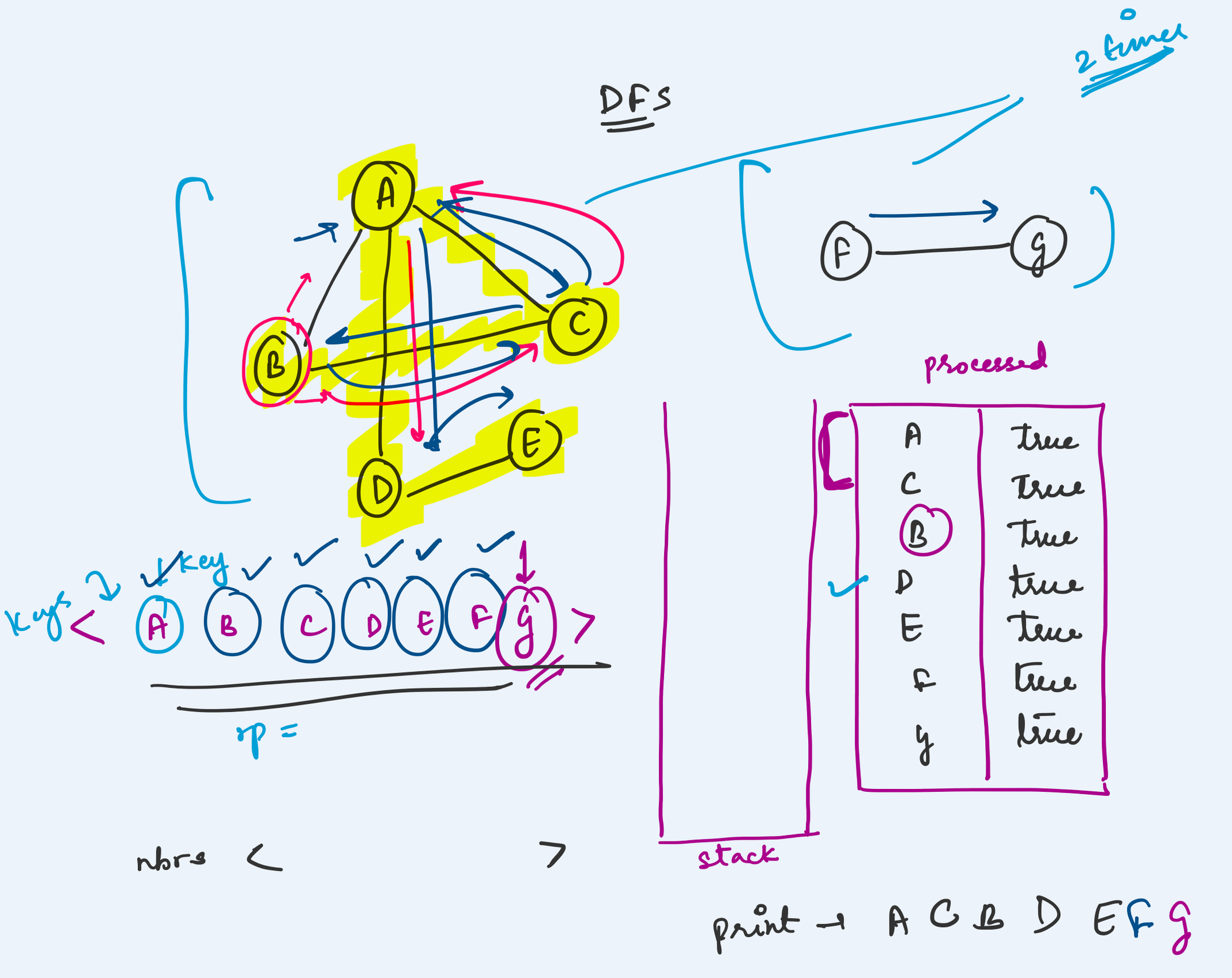
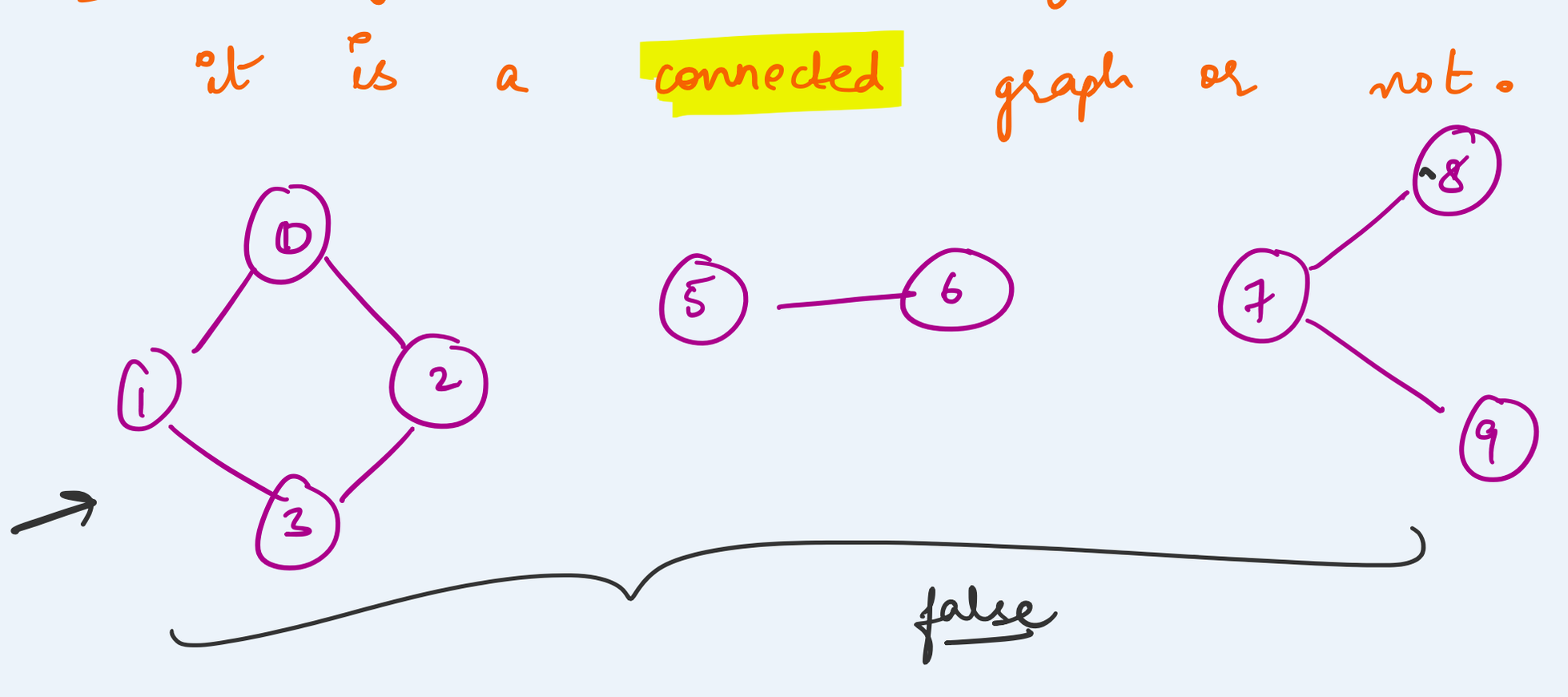
Thursday, 31 December 2020 12:37 PM

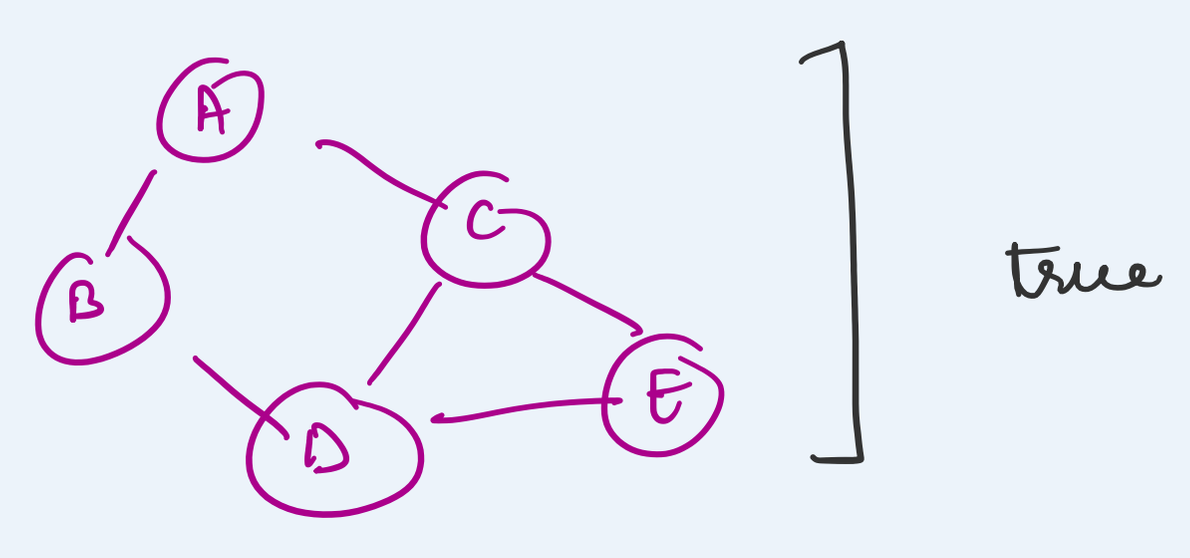


DFS



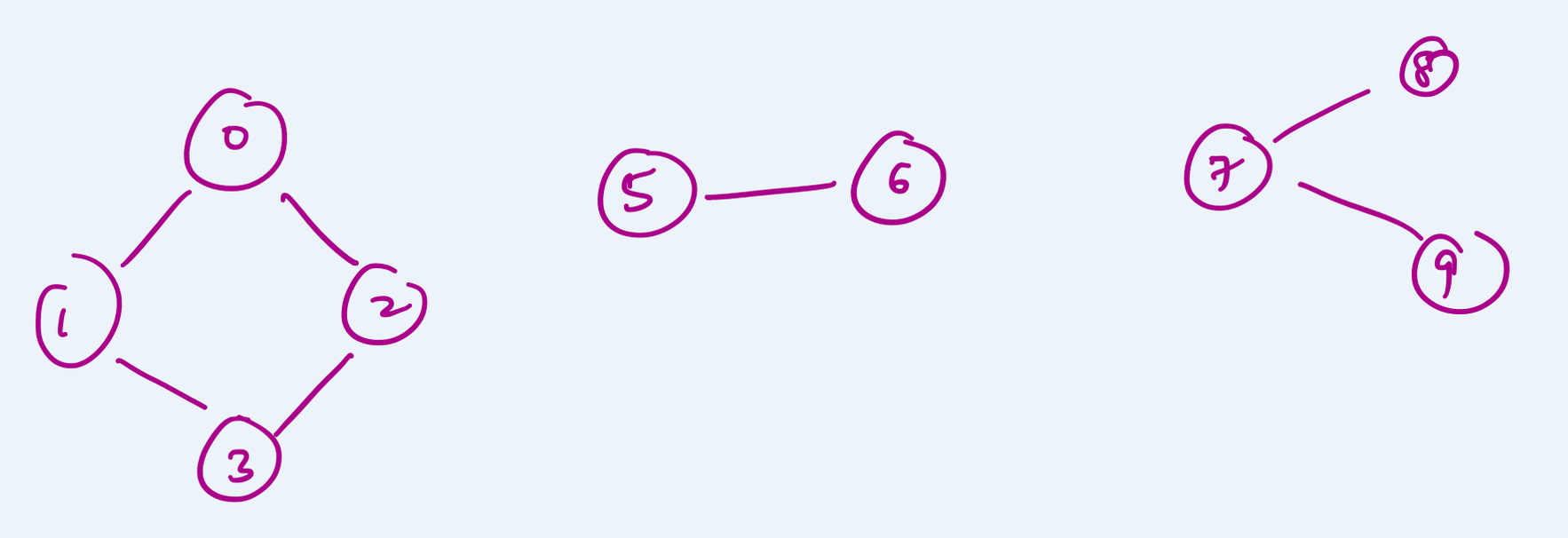
Ques 1: given a undirected graph, tell whether it is a connected graph or not.





1. Pick any random node
2. Trigger BFS / DFS from that node.
3. → a) check processed Hash map, if any node is not present there, then graph is not connected
   b) If the no. of times BFS / DFS is done is more than one 1, that means graph is not connected.

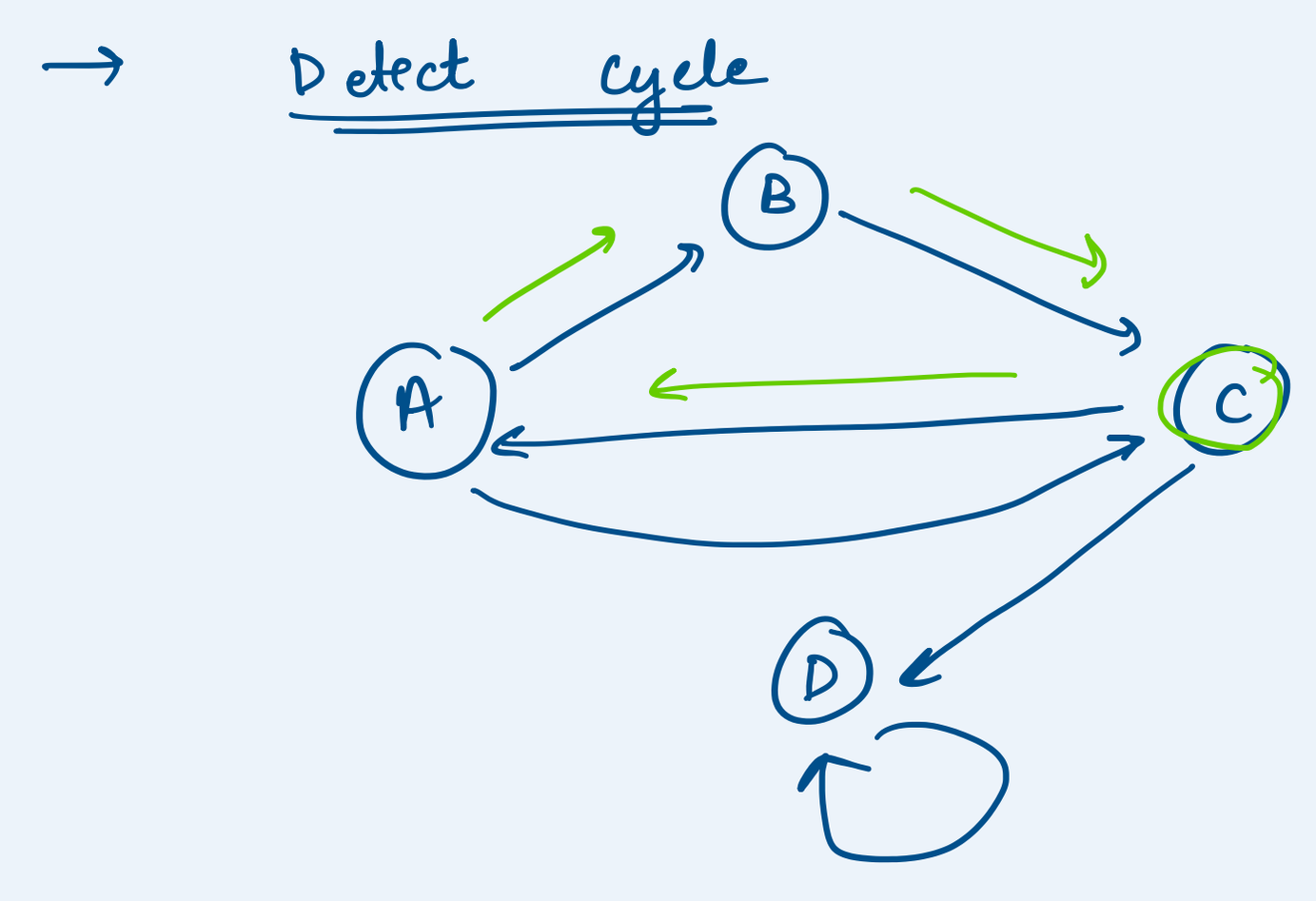
Ques given a graph tell all the connected components.



output : → [[0, 1, 2, 3] , [5, 6], [7, 8, 9]]

Given a directed graph

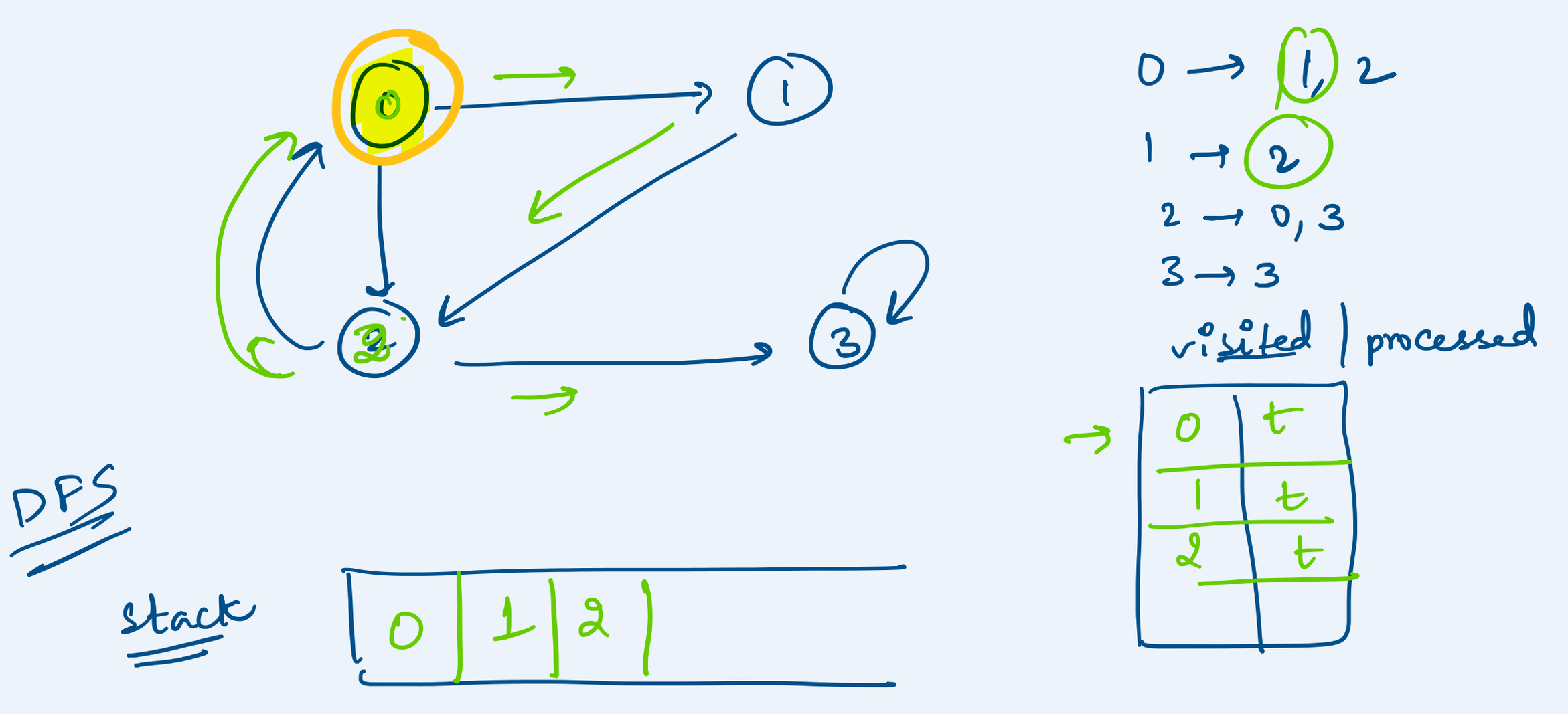
→ Detect cycle



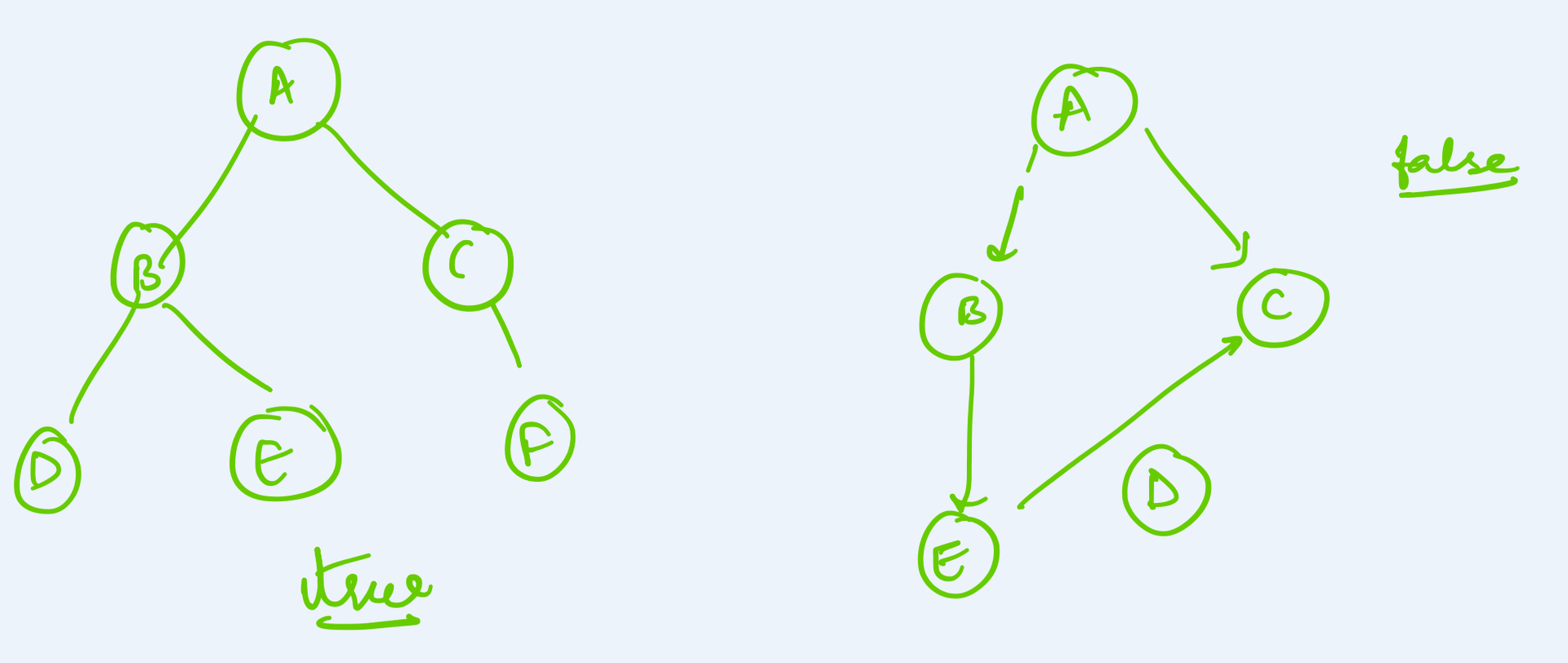
Q given a DG, detect cycle.

There is a cycle in a graph if and only if there is a back edge.

A backedge is an edge that is from a node to itself or to any of its ancestors.



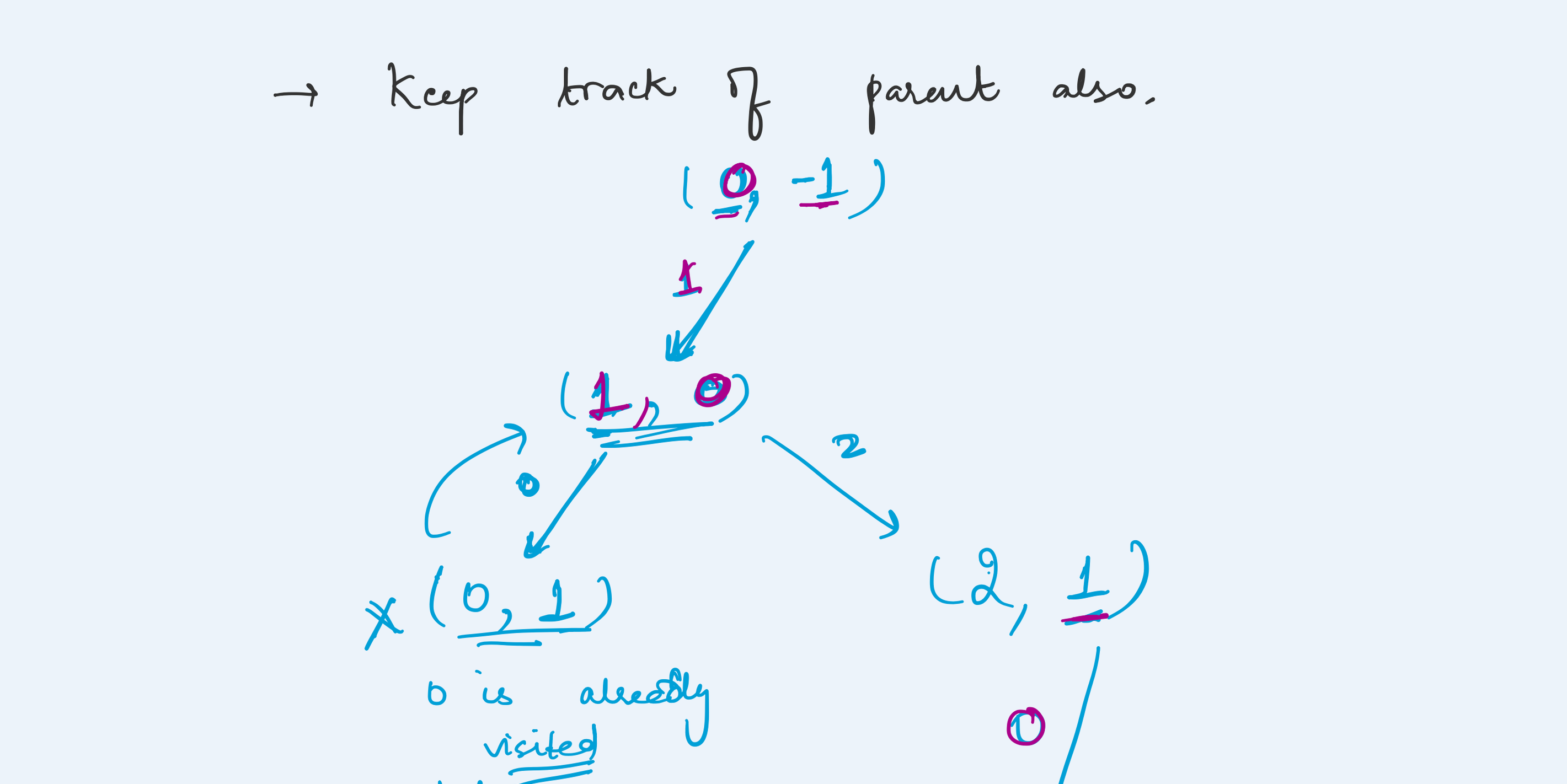
Ques given a graph, tell if the graph is a tree.



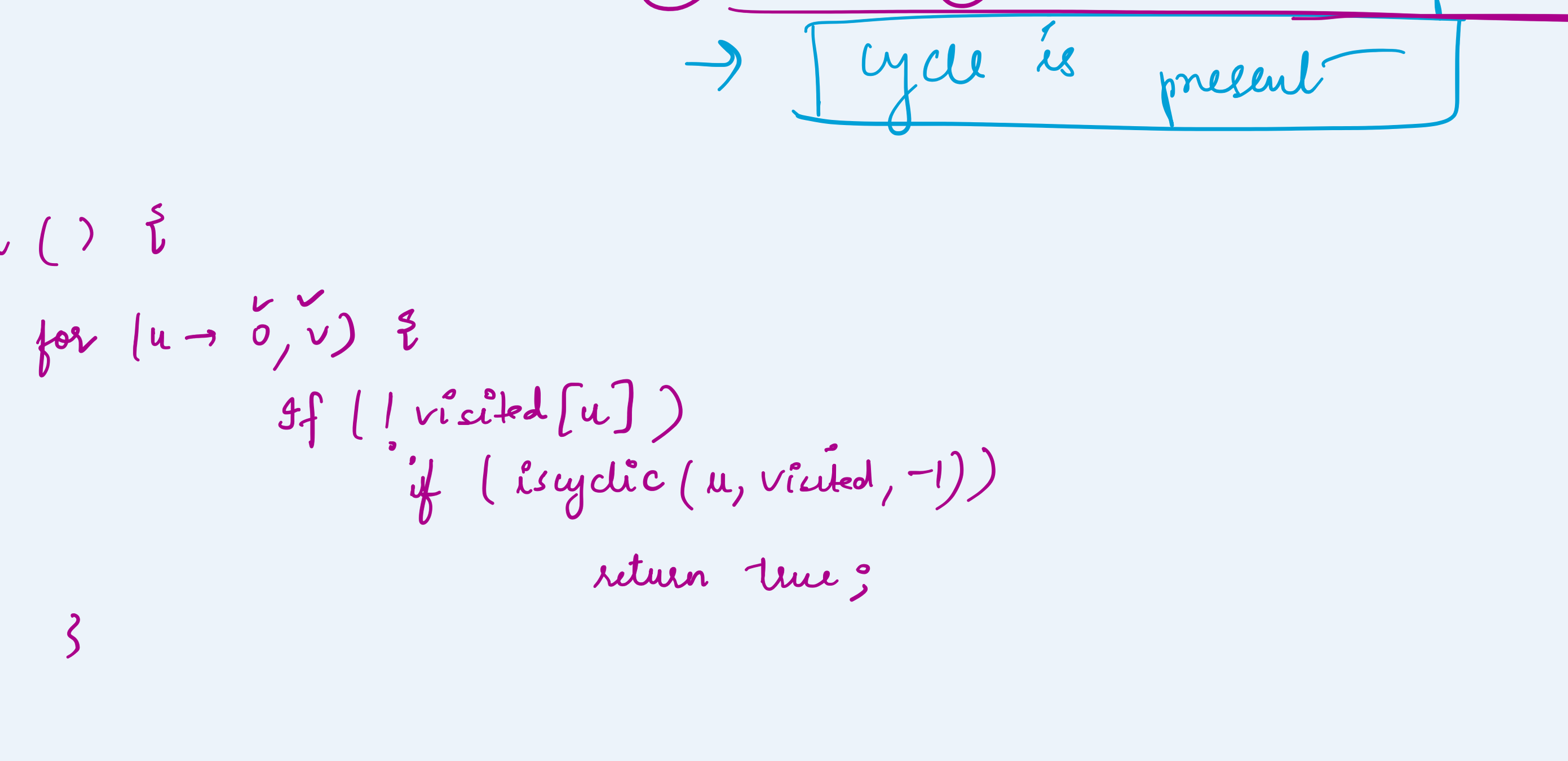
We can say that a graph is a tree if and only if the graph is connected and it does not contain a cycle.

Detect cycle in undirected graph

Is the same logic valid here also ??



→ Keep track of parent also.



1. 0 is already visited
2. and 0 is not a parent of 2

→ cycle is present

```
main () {
    for (u → 0, v) {
        if ( ! visited[u] )
            if ( iscyclic (u, visited, -1))
                return true;
    }
}

boolean iscyclic (v, visited [], parent) {
    visited [v] = true;
    for all w in adj [v] {
        if ( ! visited [w] ) {
            if ( isCyclic (w, visited , v))
                return true;
        }
        else if ( w ! = parent )
            return true;
    }
    return false;
}
```