0.1 Depth-first Search

The following search algorithm is how we will determine the connected components of a disconnected graph. We do this by selecting an arbitrary vertex of the graph as the "root". And then exploring along the branches from the vertex as far as possible until we need to backtrack, at which point we choose a new undiscovered arbitrary root and repeat. Labelling each vertex discovered along that exploration as discovered. The aim is to discover all the vertices of the graph.

Algorithm 1 DFS

Let G be a graph with vertex set $V = \{1, ..., n\}$

```
1: procedure DFS(G, V)
2:     label v as discovered
3:     for all edges from v to w in G.adjacentEdges(V) do
4:         if (vertex w is not labelled as discovered) then
5:             recursively call DFS(G, w)
```

This algorithm allows you to find the connected components of a disconnected graph. Then using the following algorithm we can check if our forest at each step of our algorithm is acyclic. We do this by counting the number of edges that each component has since we know a tree can have at most $n-1$ edges by *lemma 3.7*. This procedure is invoked in *algorithm 4*.

Algorithm 2 Acyclic Check

Let G be a graph with the set of connected componenets C as found by DFS(G,v) where v is an arbitrary vertex in G.

```
1: procedure ACYCLIC(G, C)
2:     for all i in C do
3:         if i.edgeCount() > n − 1 then return False
       return True
```