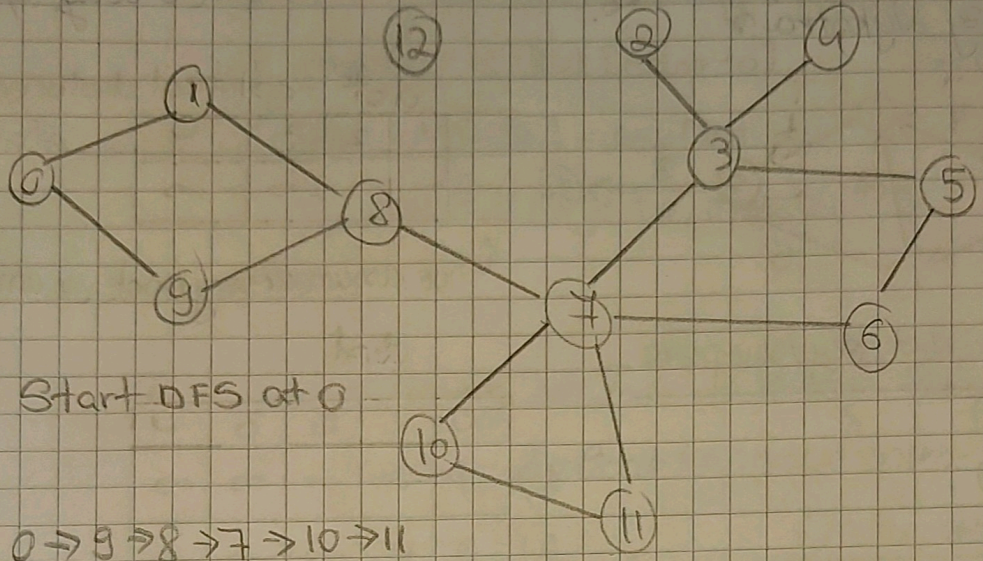
DEPTH-FIRST SEARCH TRAVERSAL

Complexity: O(V+E)

↳ DFS plunges depth-first into a graph without regard for which edge it takes next, until it cannot go any further at which point it backtracks and continues



Start DFS at 0

0 → 9 → 8 → 7 → 10 → 11

↳ we do not want to revisit 7 so we backtrack

7 → 3 → 2 → backtrack

3 → 4

3 → 5 → 6

8 → 1

Backtrack all the way to 0

Pseudocode

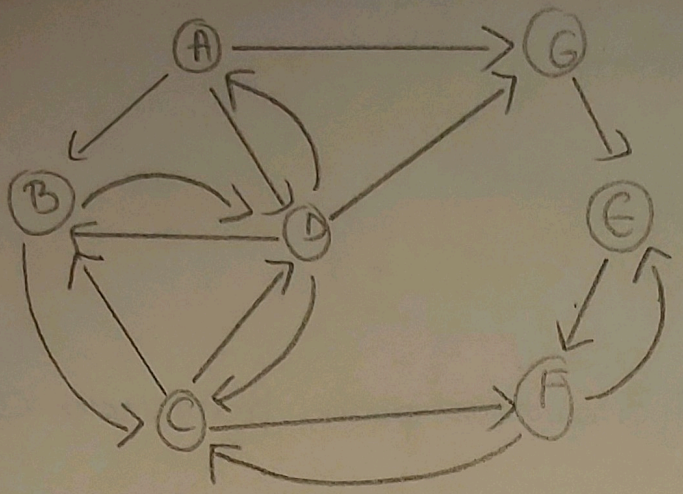
```
n = no. of nodes in the graph
g = adjacency list
visited = [false, ..., false] #size n
function dfs(at):
    if visited[at]: return
    visited[at] = true
    neighbours = graph[at]
    for next in neighbours:
        dfs(next)
```

Connected components using DFS

↳ start DFS at every unvisited node and mark all reachable nodes as being part of the same component

```
function findComponents():
    for (i=0; i<n; i++):
        if ! visited[i]:
            count++
            dfs(i)
    return count, components
```

DFS



A
↳G B
G → E → F → C → D
↳~~B~~

A B G E F C D

A G E F C D B

~~A B~~

~~=A~~
~~↳B~~

	x	y	visited
			A B C D E F G 0 0 0 0 0 0 0
DFS(A)	x=A	y=B	1 0 0 0 0 0 0
DFS(B)	x=B	y=C	1 1 0 0 0 0 0
DFS(C)	x=C	y=B x y=D	1 1 1 0 0 0 0
DFS(D)	x=D	y=A x y=G	1 1 1 1 0 0 0
DFS(G)	x=G	y=E	1 1 1 1 0 0 1
DFS(E)	x=E	y=F	1 1 1 1 1 0 x
DFS(F)	x=F		x 1 1 1 1 1 1

↳ A B C D G E F