William Fiset (Depth first Search) (4).

DFS overview → Most fundamental search algorithm used to explore nodes and edges of a graph.
Runs with O(V+E) complexity.

→ Not very useful by itself, but combined with other algorithms, it shines.

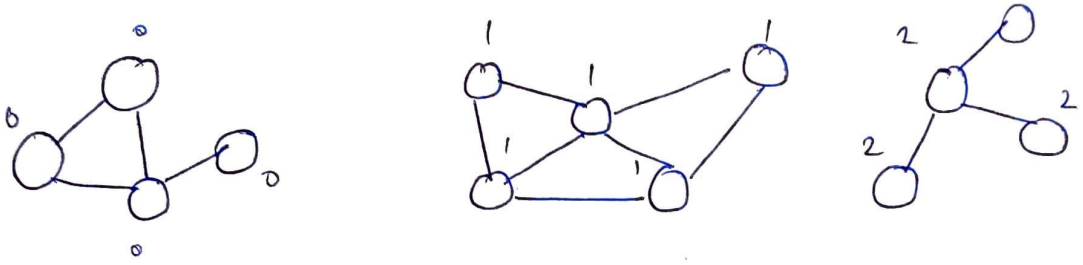
→ DFS plunges depth first without regard for which node it takes next until it cannot go any further. at which point it backtracks and continues.

```
# Global class scope variables
n = number of nodes in graph.
g = adjacency list representing graph.
visited = [false, false .... false]  # size n.

function dfs(at):
    if visited[at]: return
    visited[at] = true.

    neighbours = graph[at]
    for next in neighbours:
        dfs(next)

# start dfs at node zero
start_node = 0
dfs(start_node).
```

→ Finding connected components in a graph.

Sometimes, a graph is split into multiple components. It's useful to be able to identify and count these components.

Assign an integer value to each group to be able to tell them apart.

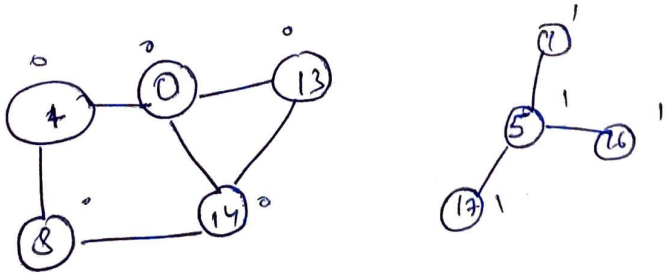


We can use DFS to identify components.

First, make sure all the nodes are labeled from [0, n) where n is the number of nodes.

For every node, do a DFS if a value is not yet assigned. Mark all the components in DFS with same integer.

ex =



```
# global variables
n : number of nodes in the graph
g : adjacency list.
count = 0
components = empty integer array   # size n.
visited = [false, false .... false]   # size n.

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

function dfs(at):
    visited[at] = true
    components[at] = count
    for (next : g[at])
        if (!visited[next]):
            dfs(next).
```

What else can DFS do?

(i) Compute a graph's minimum spanning tree

(ii) Detect and find cycles in a graph.

(iii) Check if a graph is bipartite.

(iv) Find strongly connected components

(v) Topologically sort the nodes of a graph.

(vi) Find bridges and articulation points

(vii) Find augmenting paths in a flow network.

(viii) Generate mazes.