Depth First Search (DFS)

Depth first Search or Depth first traversal is a recursive algorithm for searching all the vertices of a graph or tree data structure. Traversal means visiting all the nodes of a graph.

Depth First Search Algorithm

A standard DFS implementation puts each vertex of the graph into one of two categories:
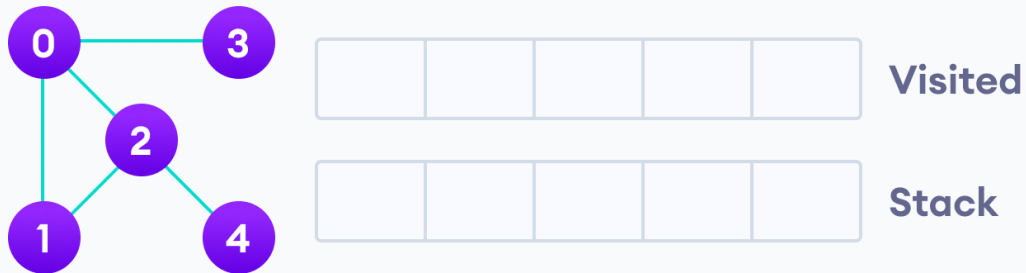
1. Visited

2. Not Visited

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

The DFS algorithm works as follows:

1. Start by putting any one of the graph's vertices on top of a stack.

2. Take the top item of the stack and add it to the visited list.

3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.

4. Keep repeating steps 2 and 3 until the stack is empty.

Depth First Search Example

Let's see how the Depth First Search algorithm works with an example. We use an undirected graph with 5 vertices.
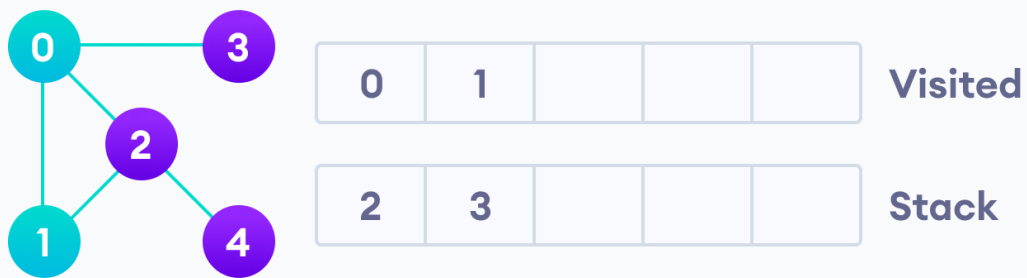

Undirected graph with 5 vertices

We start from vertex 0, the DFS algorithm starts by putting it in the Visited list and putting all its adjacent vertices in the stack.
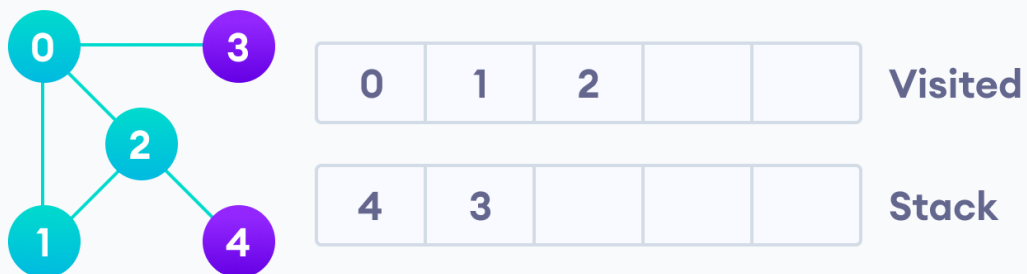

Visit the element and put it in the visited list

Next, we visit the element at the top of stack i.e. 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.
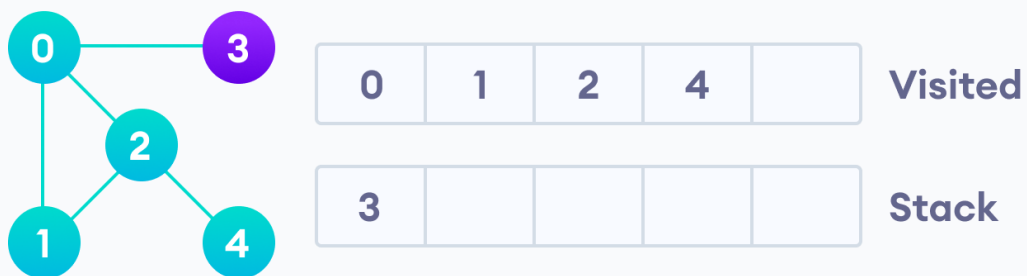
Visit the element at the top of stack

Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Traversal of the graph.

| 0 | 1 | 2 | 4 | 3 | Visited |
| | | | | | Stack |

After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Traversal of the graph.

Complexity of Depth First Search

The time complexity of the DFS algorithm is represented in the form of $O(V + E)$, where V is the number of nodes and E is the number of edges.
The space complexity of the algorithm is $O(V)$

Application of DFS Algorithm

1. For finding the path

2. For finding the strongly connected components of a graph

3. For detecting cycles in a graph

```python
# DFS algorithm in Python


# DFS algorithm
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)

    print(start)

    for next in graph[start] - visited:
        dfs(graph, next, visited)
    return visited


graph = {'0': set(['1', '2']),
         '1': set(['0', '3', '4']),
         '2': set(['0']),
         '3': set(['1']),
         '4': set(['2', '3'])}

dfs(graph, '0')
```