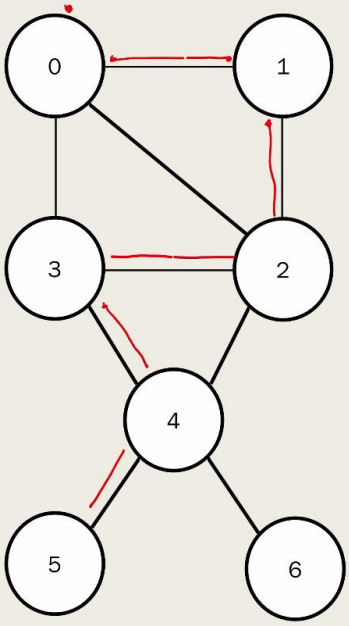
Depth First Search

- **Graph traversal** refers to the process of visiting (checking and/or updating) each vertex(node) in a graph.
- Two Algorithms of Graph Traversal are:
  - *Breadth First Search (BFS)*
  - *Depth First Search (DFS)*
- In DFS, we start with a node and start exploring its connected nodes keeping on suspending the exploration of previous nodes

DFS Procedure

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.

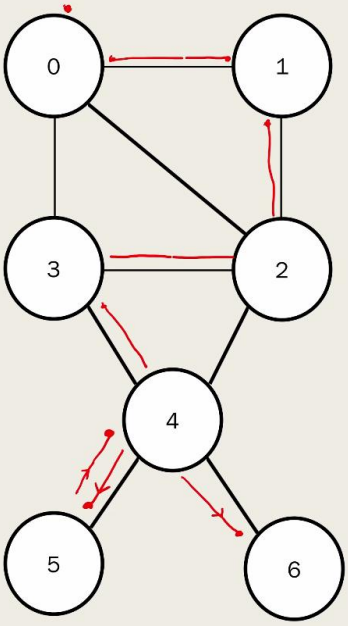
Visited : 0 1 2 3 4 5



DFS Procedure

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.

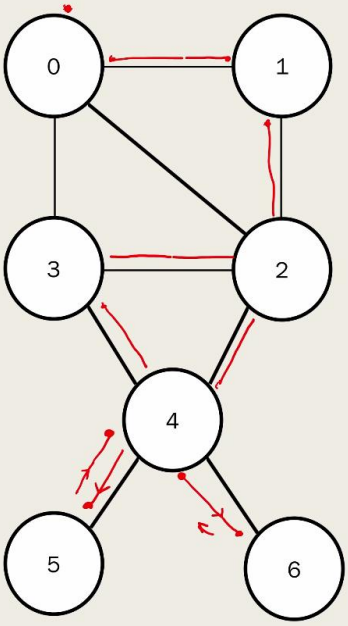
Visited : 0 1 2 3 4 5 6



DFS Procedure

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.

Visited : 0 1 2 3 4 5 6 2

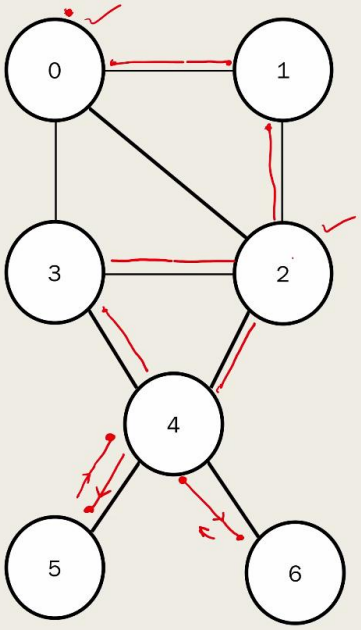


V: 0 1 2

DFS Procedure

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.

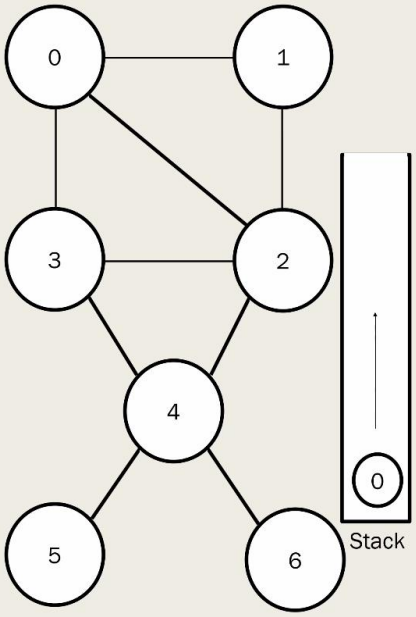
Visited : 0 1 2 3 4 5 6 2



DFS Procedure – Step 2

1. Start by putting any one of the graph's vertices on top of a stack – Lets put 0 into the stack!
2. Take the top item of the stack and add it to the visited list – 0 is now visited!

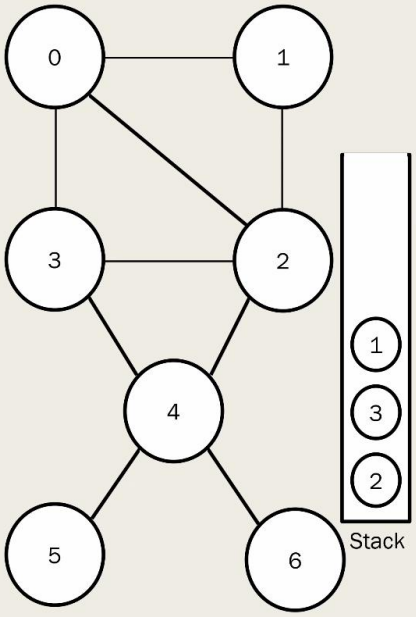
Visited: 0



DFS Procedure – Step 3

1. Start by putting any one of the graph's vertices on top of a stack – Lets put 0 into the 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.

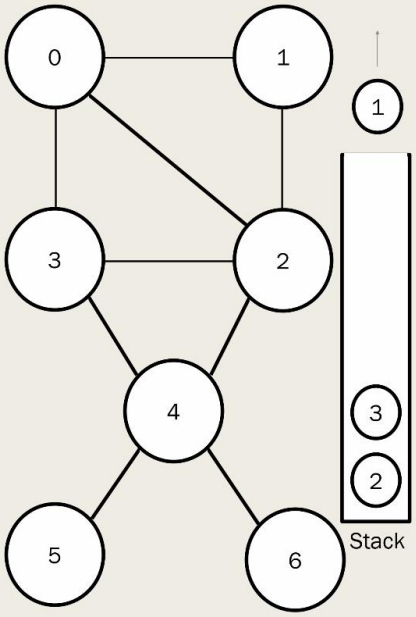
Visited: 0,



DFS Procedure – Step 2

1. Take the top item of the stack and add it to the visited list – 1 is now visited!

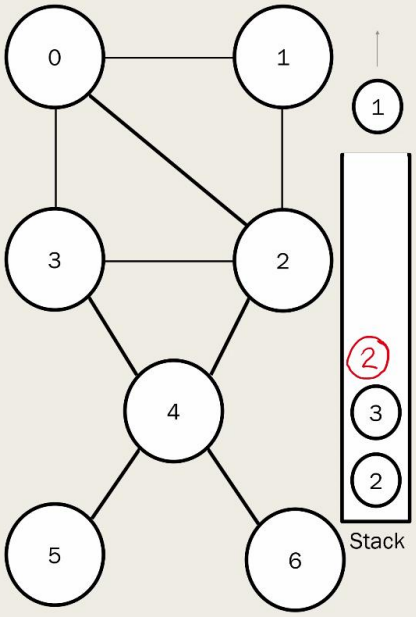
Visited: 0, 1



DFS Procedure – Step 2

1. Take the top item of the stack and add it to the visited list – 1 is now visited!

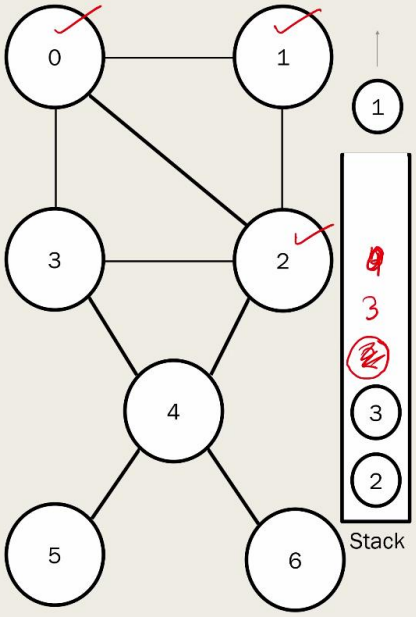
Visited: 0, 1



DFS Procedure – Step 2

1. Take the top item of the stack and add it to the visited list – 1 is now visited!

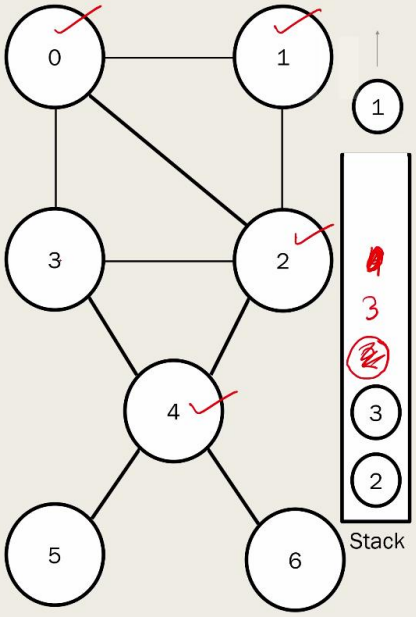
Visited: 0, 1, 2



DFS Procedure – Step 2

1. Take the top item of the stack and add it to the visited list – 1 is now visited!

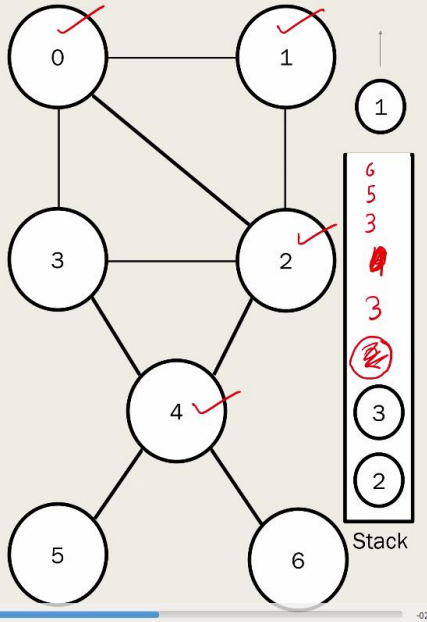
Visited: 0, 1, 2, 4



DFS Procedure – Step 2

1. Take the top item of the stack and add it to the visited list – 1 is now visited!

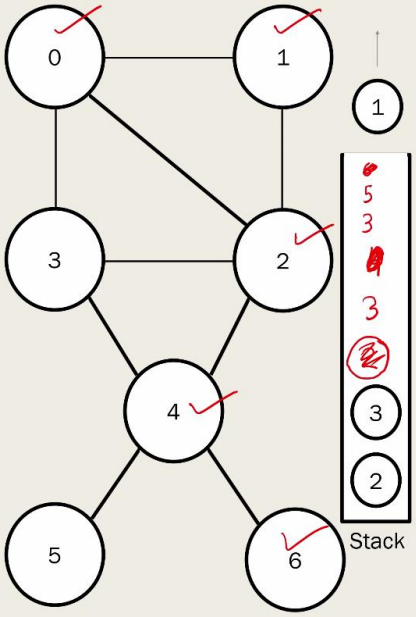
Visited: 0, 1, 2, 4



DFS Procedure – Step 2

1. Take the top item of the stack and add it to the visited list – 1 is now visited!

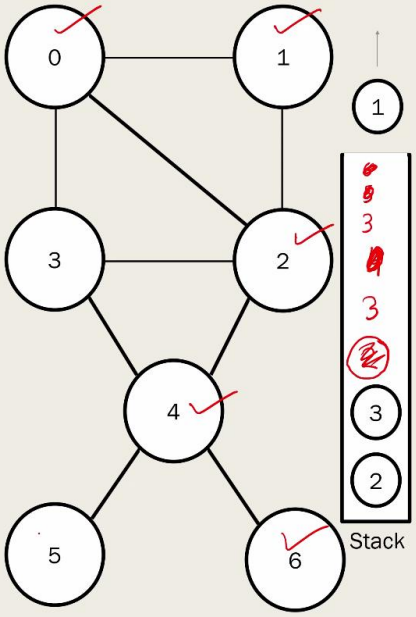
Visited: 0, 1, 2, 4, 6



DFS Procedure – Step 2

1. Take the top item of the stack and add it to the visited list – 1 is now visited!

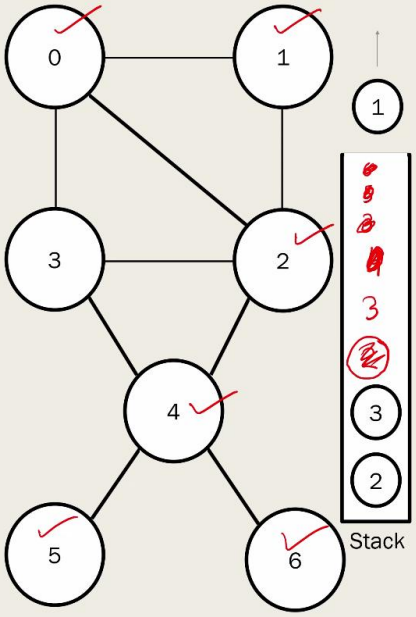
Visited: 0, 1, 2, 4, 6, 5



DFS Procedure – Step 2

1. Take the top item of the stack and add it to the visited list – 1 is now visited!

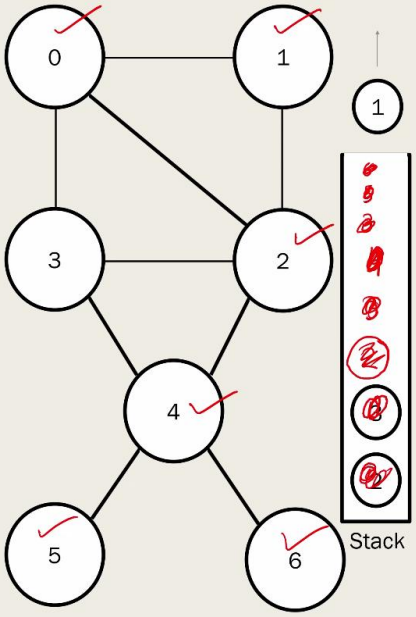
Visited: 0, 1, 2, 4, 6, 5, 3



DFS Procedure – Step 2

1. Take the top item of the stack and add it to the visited list – 1 is now visited!

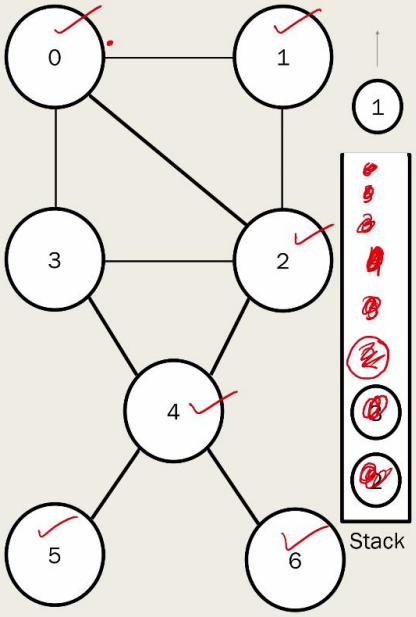
Visited: 0, 1, 2, 4, 6, 5, 3



DFS Procedure – Step 2

1. Take the top item of the stack and add it to the visited list – 1 is now visited!

Visited: 0, 1, 2, 4, 6, 5, 3 → DFS



DFS Pseudocode

```
DFS(G, u)
   u.visited = true
   for each v ∈ G.Adj[u]
      if v.visited == false
         DFS(G,v)

init() {
   For each u ∈ G
      u.visited = false
   For each u ∈ G
      DFS(G, u)
```