DSA through C++

Graph traversing

Saurabh Shukla (MySirG)

Agenda

1. Traversing Graph
2. BFS
3. Logic for BFS
4. Example
5. DFS
6. Logic for DFS
7. Example

Traversing

There are two standard way of traversing a graph.

1. BFS — Breadth First Search
2. DFS — Depth First Search

BFS

Traversing graph has only issue that graph may have cycle. You may revisit a node.

To avoid processing a node more than once, we divide the vertices into two categories:

1. visited
2. Not visited

A boolean visited array is used to mark the visited vertices

BFS (Breadth First Search) uses a queue data structure for traversal.

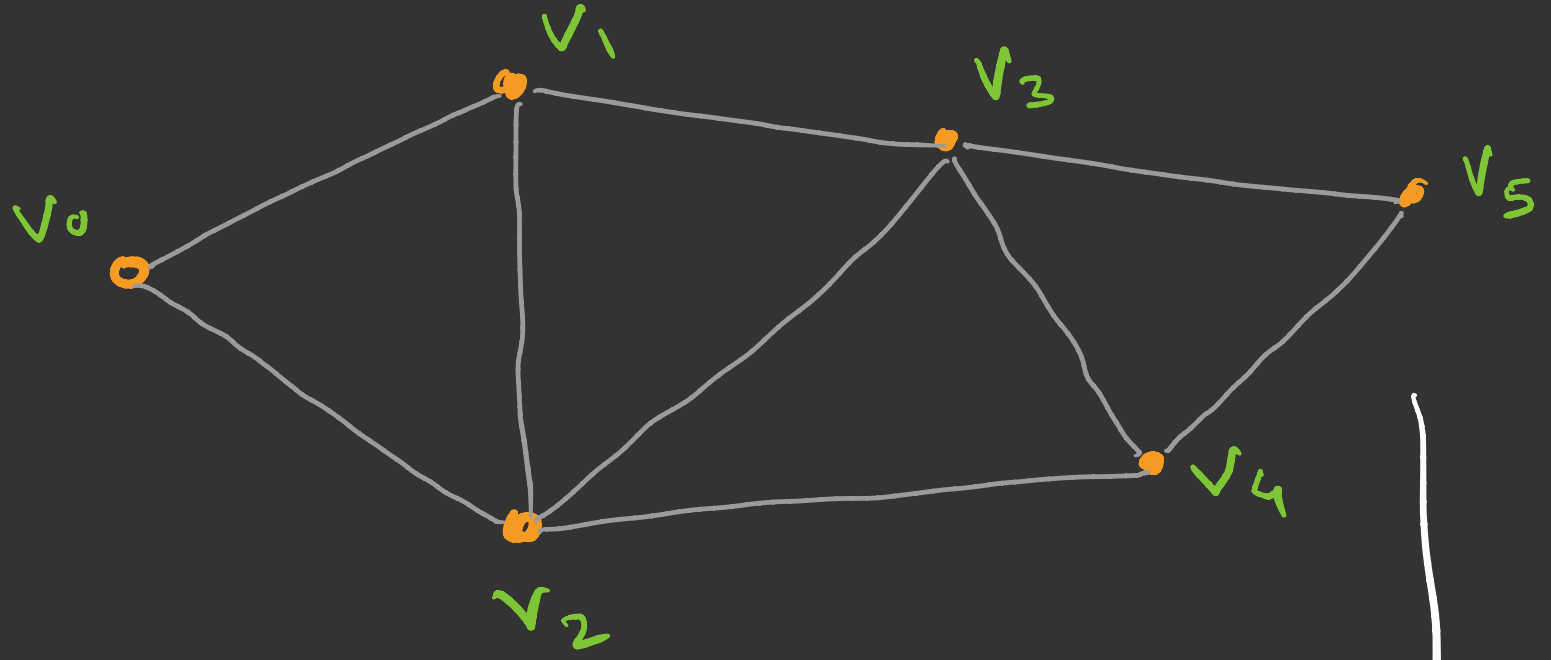
Traversing begin from a node called source node.

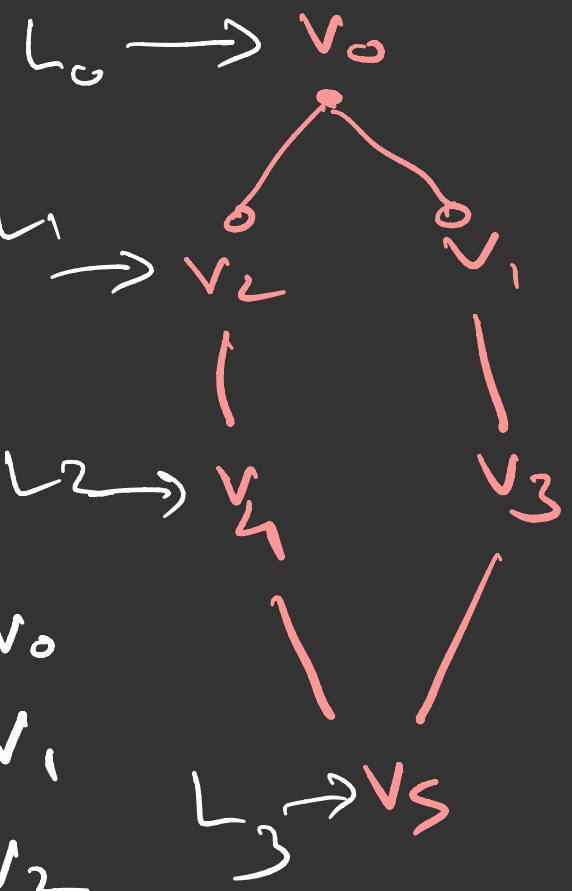
Logic for BFS

```
BFS(G, s)
    Let Q be the queue
    Q.insert(s);
    v[s] = true;

    while (!Q.isEmpty())
    {
        n= Q.getFront();
        Q.del();
        for all the neighbors u of n
            if v[u] == false
                Q.insert(u);
                v[u] = true;

    }
```

Example

Source = v_0





Q

n =

0	1	2	3	4	5
T	T	T	T	T	T

v_0
v_1
v_2
v_3
v_4
v_5

DFS

DFS is Depth first Search.

The main defference between BFS and Dfs is that the DFS uses Stack in place of Queue.

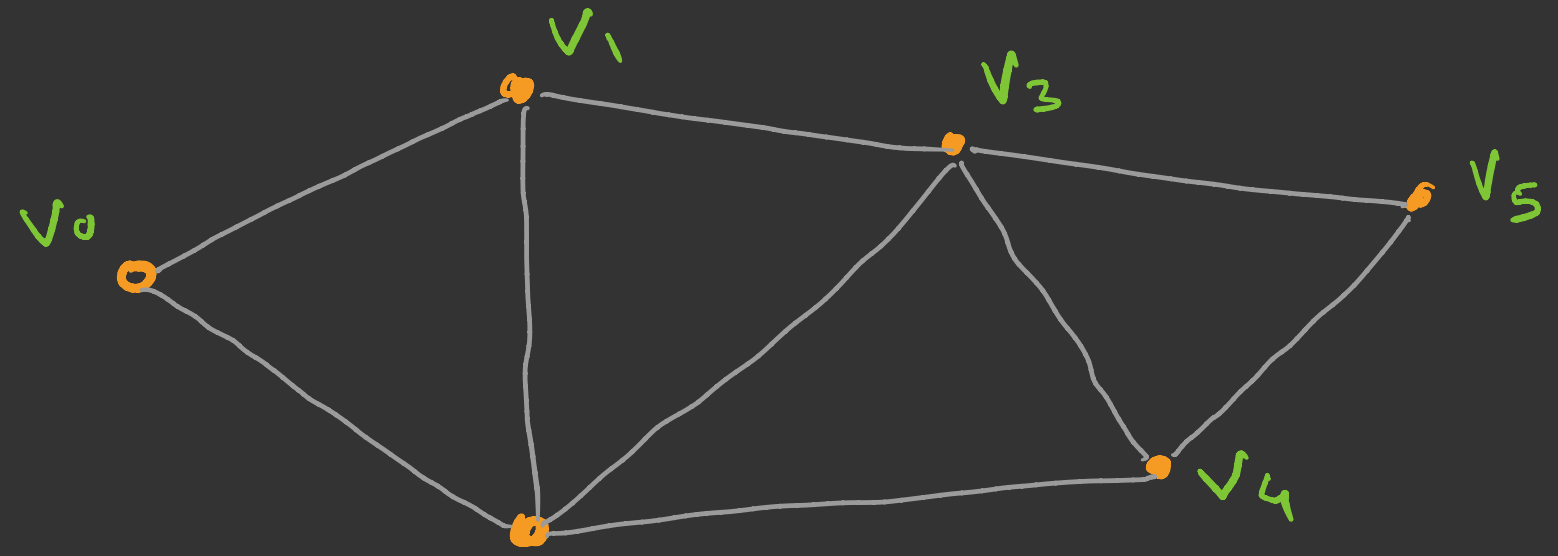
Logic for DFS

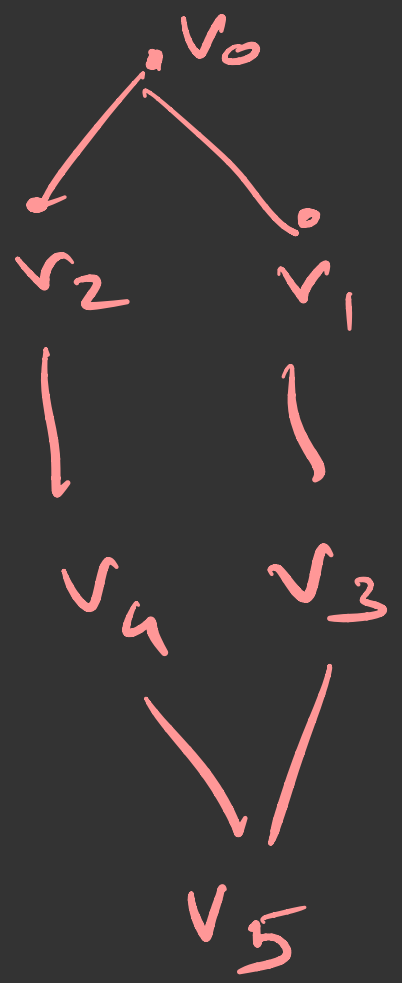
DFS (G, S)

```
Let stack be the stack
stack.push(S)
v[S] = true;
while ( !stack.isEmpty() )
{
    n= stack.peek();
    stack.pop();
    for all the neighbors u of n
        if v[u] == false
            stack.push(u);
            v[u] = true;
}
```

Example

Source = V_0





0	1	2	3	4	5
T	T	T	T	T	T

DFS

V_0
V_2
V_4
V_5
V_3
V_1