Graph Traversal

Anoop S Babu

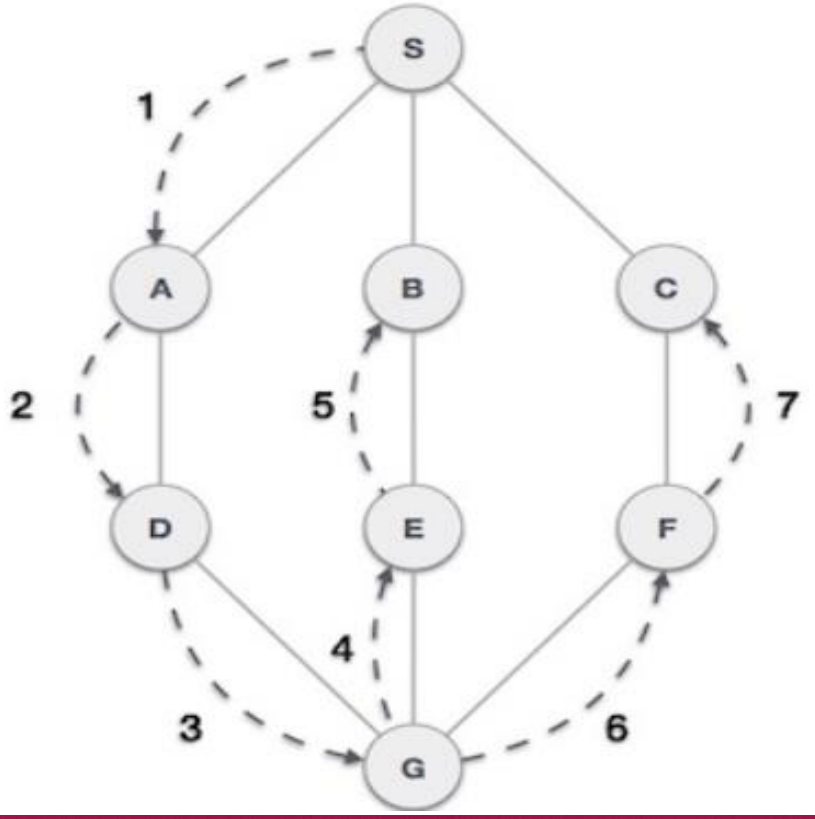
Faculty Associate

Dept. of Computer Science & Engineering

bsanoop@am.amrita.edu

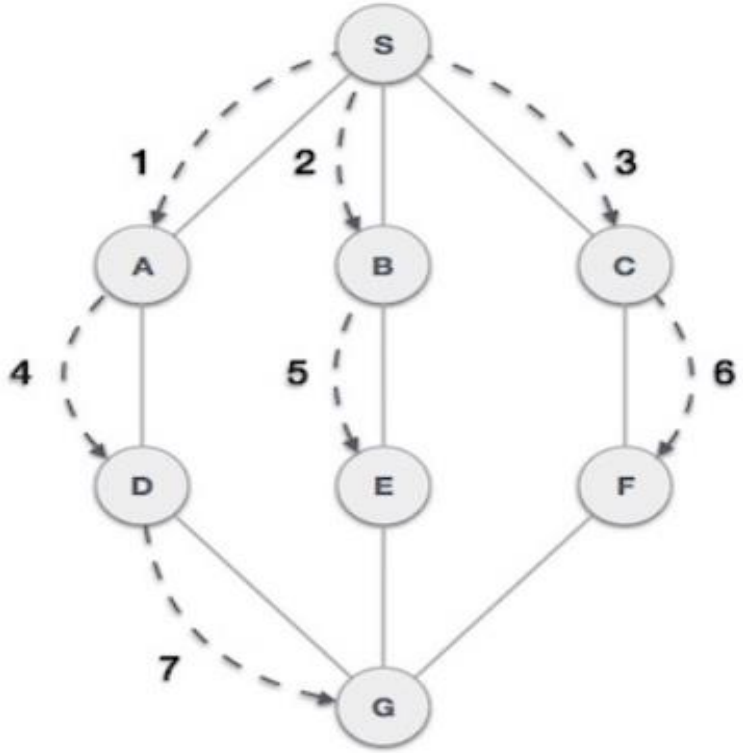
Graph Traversal

- Graph traversal (also known as graph search) refers to the **process of visiting each vertex in a graph.**
- Graph traversal is a technique used for a searching vertex in a graph.
- There are two standard methods
  - BFS (Breadth First Search)
  - DFS (Depth First Search)



Breadth First Search (BFS)

- Breadth First Search (BFS) algorithm traverses a graph in a breadthward motion.
- It starts from an arbitrary vertex of a graph, and explores all of the neighbor vertices at the present depth prior to moving on to the vertices at the next depth level.
- Also called as level order search.
- Uses a **queue** to remember to get the next vertex to start a search.



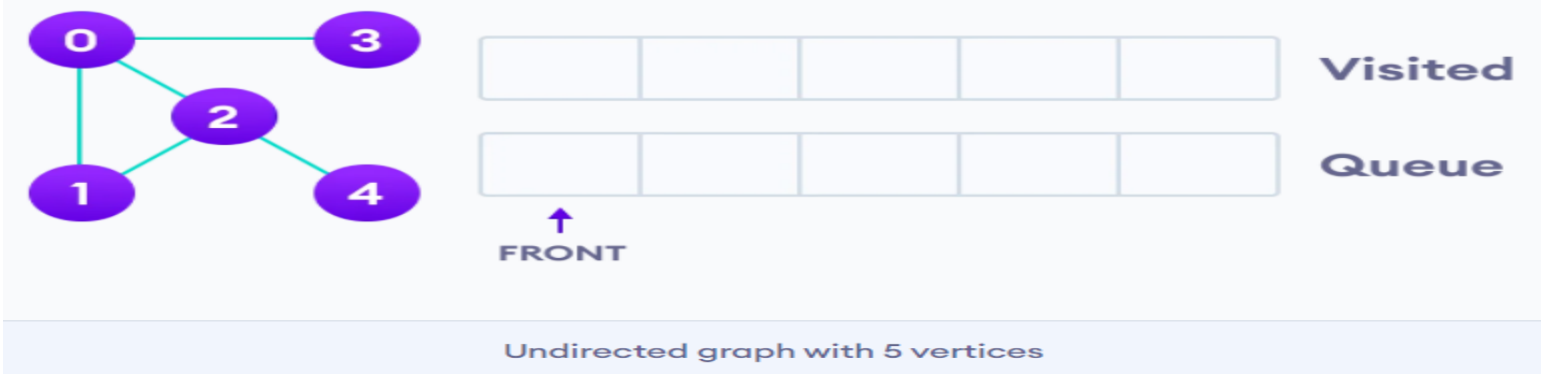
BSF order: S,A,B,C,D,E,F,G

Breadth First Search (BFS)

The BFS algorithm works as follows:

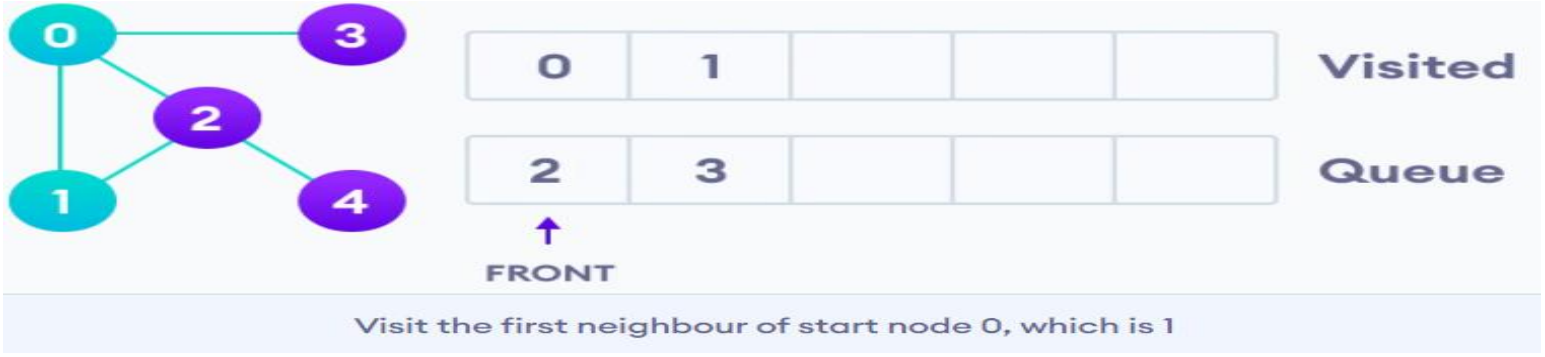
1. Start by putting any one of the graph's vertices at the back of a queue.
2. Take the front item of the queue and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.
4. Keep repeating steps 2 and 3 until the queue is empty.

BFS Algorithm Working



Undirected graph with 5 vertices

Visit start vertex and add its adjacent vertices to queue

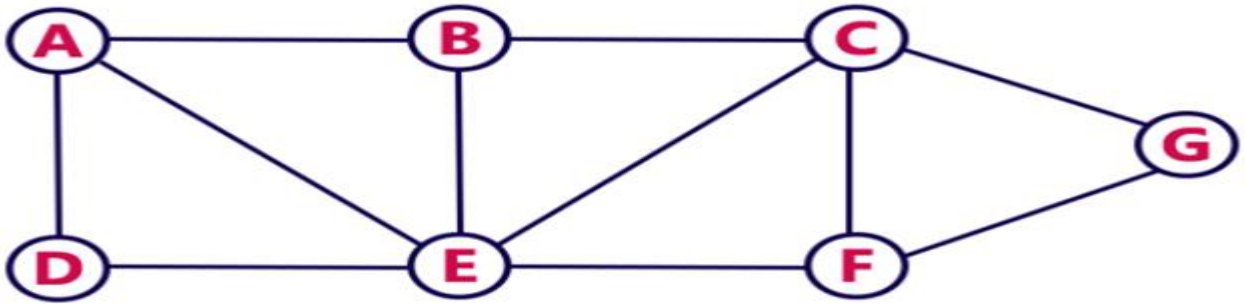


Visit the first neighbour of start node 0, which is 1

BFS Algorithm Working

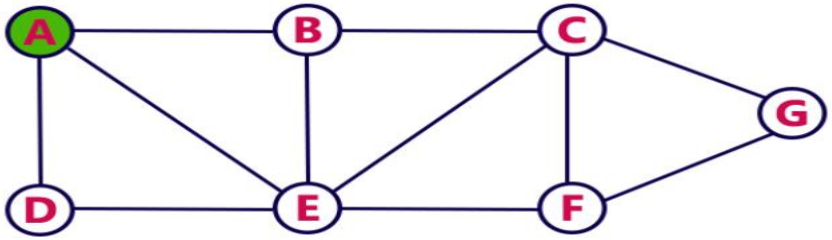
Since the queue is empty, we have completed the BFS Traversal of the graph.

BSF - Example



Step 1:

- Select the vertex **A** as starting point (visit **A**).
- Insert **A** into the Queue.

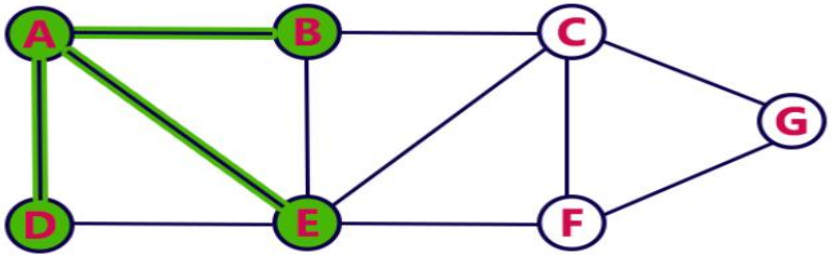


Queue

A						

Step 2:

- Visit all adjacent vertices of **A** which are not visited (**D**, **E**, **B**).
- Insert newly visited vertices into the Queue and delete A from the Queue..



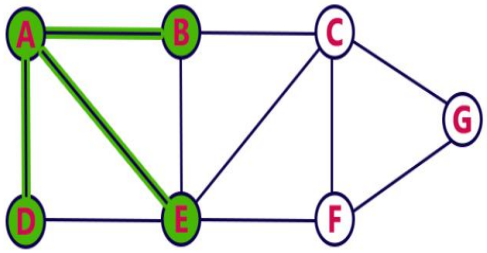
Queue

	D	E	B			

BSF - Example

Step 3:

- Visit all adjacent vertices of **D** which are not visited (there is no vertex).
- Delete D from the Queue.

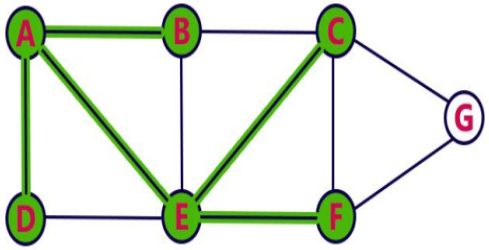


Queue

		E	B			

Step 4:

- Visit all adjacent vertices of **E** which are not visited (**C**, **F**).
- Insert newly visited vertices into the Queue and delete E from the Queue.

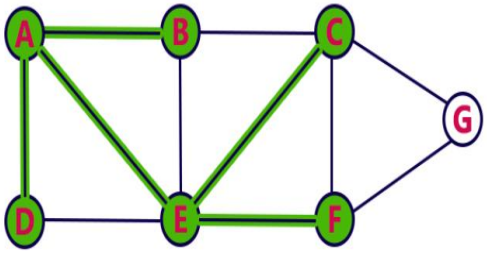


Queue

			B	C	F	

Step 5:

- Visit all adjacent vertices of **B** which are not visited (**there is no vertex**).
- Delete **B** from the Queue.

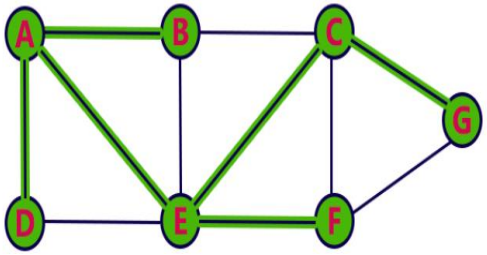


Queue

				C	F	

Step 6:

- Visit all adjacent vertices of **C** which are not visited (**G**).
- Insert newly visited vertex into the Queue and delete **C** from the Queue.

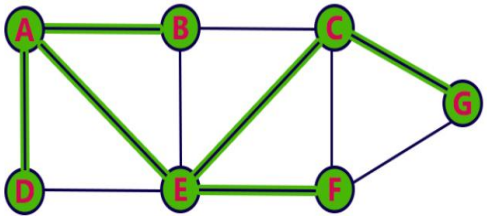


Queue

					F	G

Step 7:

- Visit all adjacent vertices of **F** which are not visited (**there is no vertex**).
- Delete **F** from the Queue.

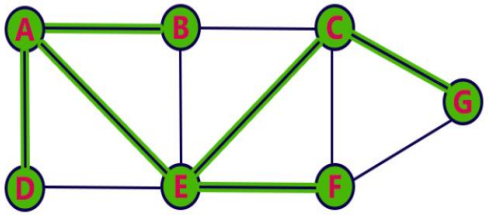


Queue

						G

Step 8:

- Visit all adjacent vertices of **G** which are not visited (**there is no vertex**).
- Delete **G** from the Queue.

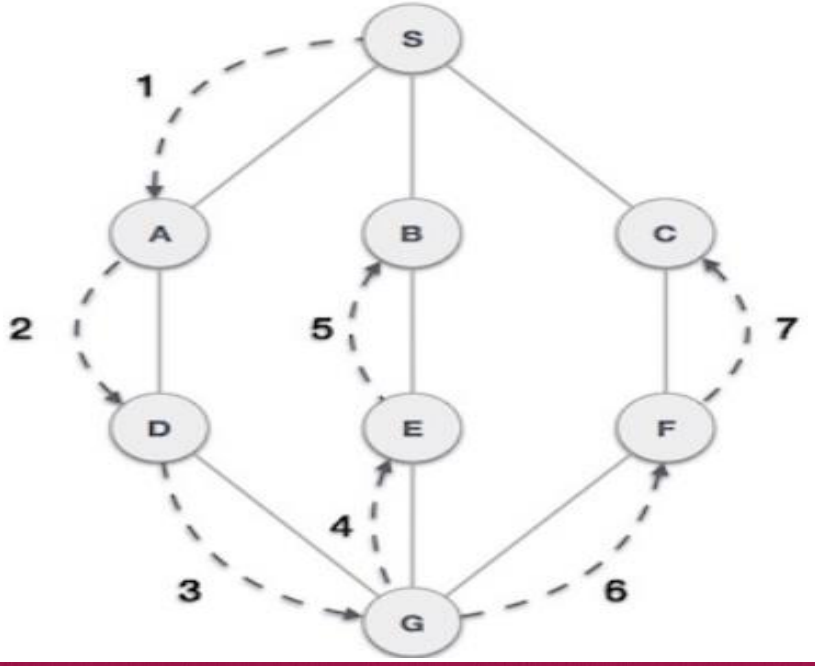


Queue

BFS Order: A D E B C F G

Depth First Search (DFS)

- Depth First Search (DFS) algorithm traverses a graph in a depthward motion.
- It starts from arbitrary vertex a graph, and explores as far as possible along each branch before backtracking.
- Uses a stack to get the next vertex to start a search, when a dead end occurs.



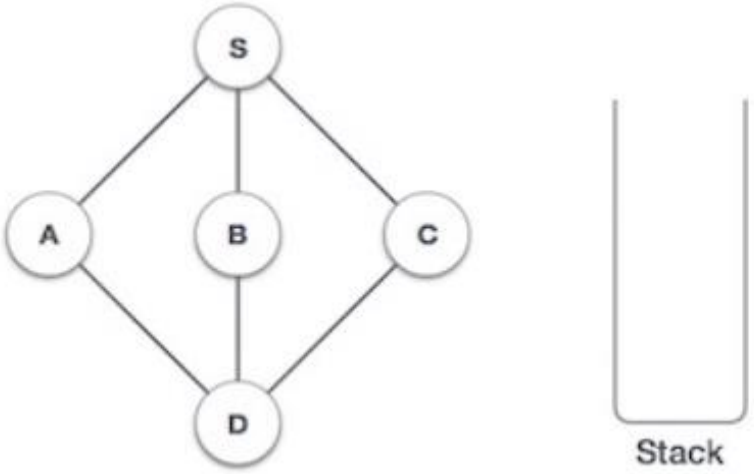
DFS order: S,A,D,G,E,B,F,C

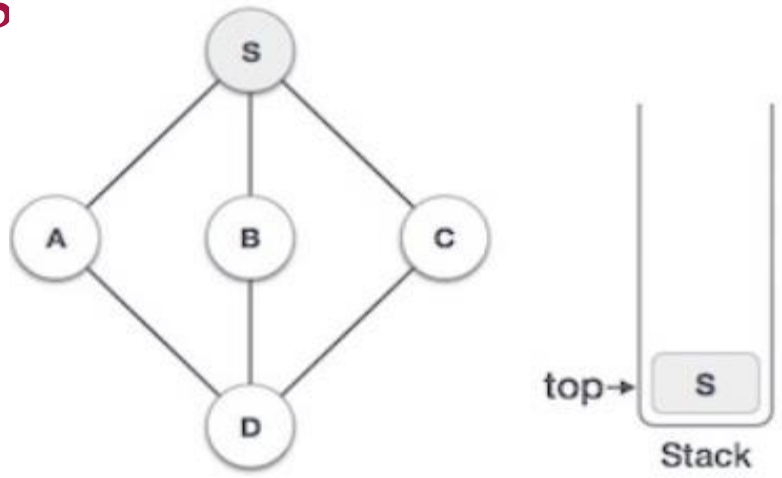
Depth First Search (DFS)

The DFS algorithm works as follows:

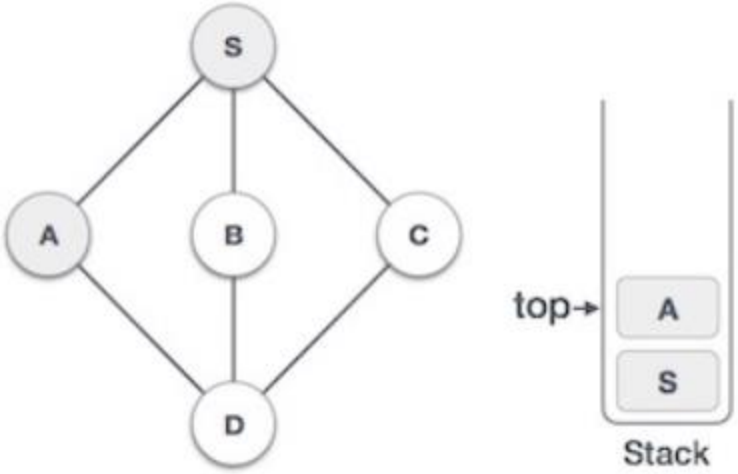
1. Start from any one of the graph's vertices. Mark it as visited. Push it in a stack
2. Visit the adjacent unvisited vertex. Mark it as visited. Push it in a stack.
3. If no adjacent vertex is found, pop up a vertex from the stack.
4. Repeat step 2 and step 3 until the stack is empty.

DFS Algorithm Working

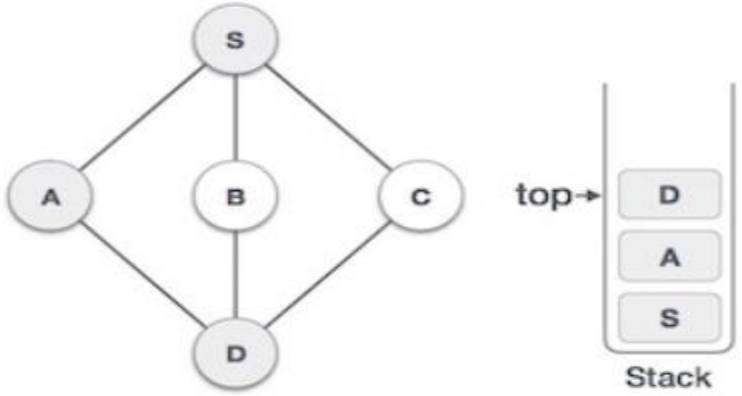




Mark **S** as visited and put it onto the stack.
Explore any unvisited adjacent node from **S**.

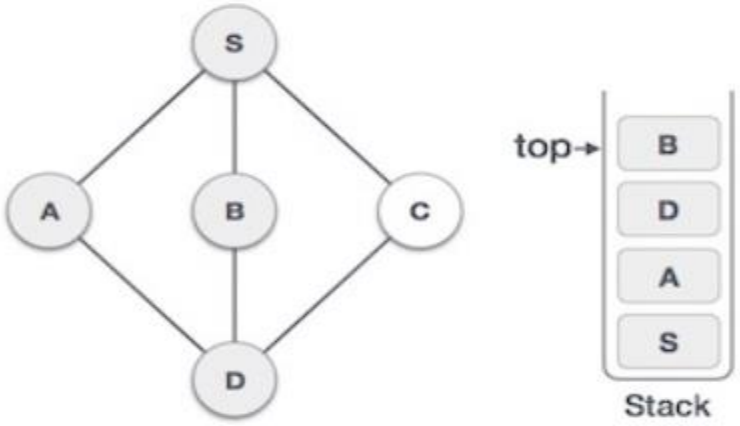


Mark **A** as visited and put it onto the stack.
Explore any unvisited adjacent node from **A**

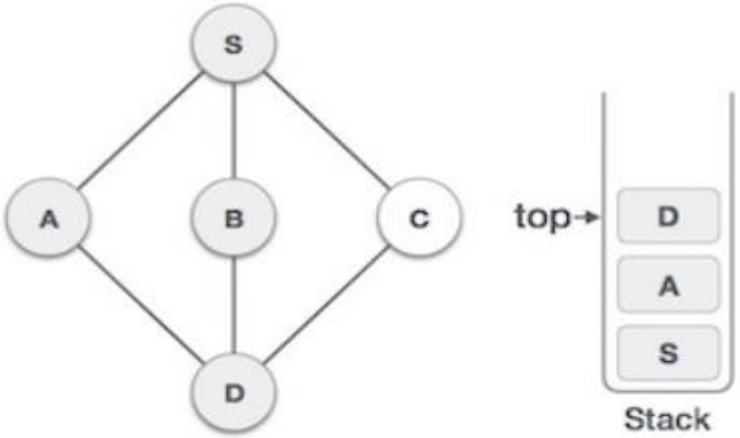


Mark **D** as visited and put it onto the stack.
Explore any unvisited adjacent node from **D**

DFS Algorithm Working

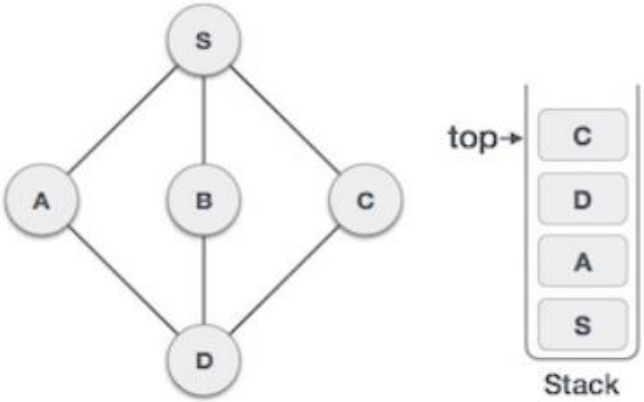


We choose **B**, mark it as visited and put onto the stack.



B does not have any unvisited adjacent node. So, pop **B** from the stack.

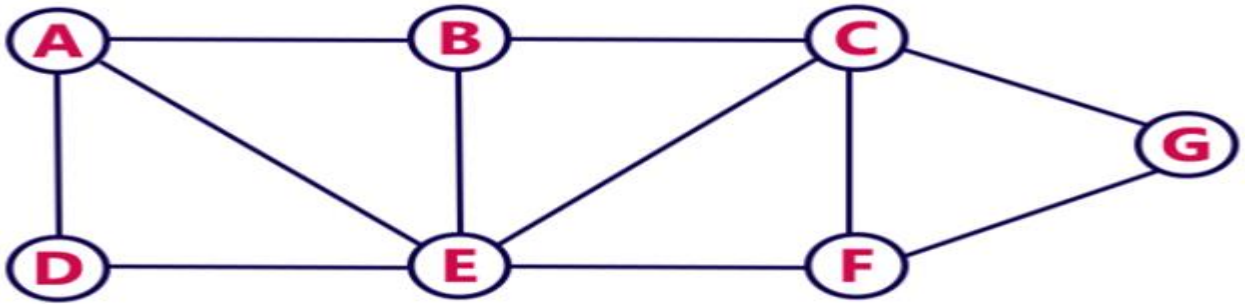
We check the stack top for return to the previous node and check if it has any unvisited nodes.
D to be on the top of the stack.



Only unvisited adjacent node from **D** is **C**. So mark **C** as visited and put it onto the stack.

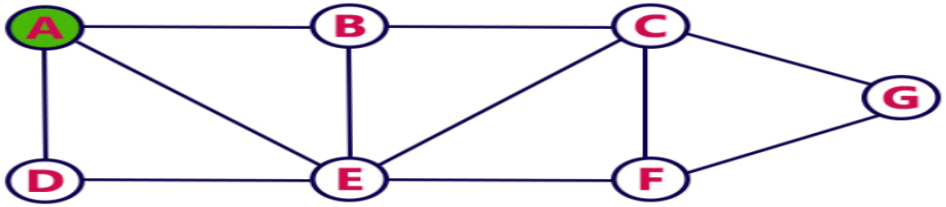
As **C** does not have any unvisited adjacent node. So we keep popping the stack until we find a node that has an unvisited adjacent node.
In this case, we until the stack is empty

DSF - Example



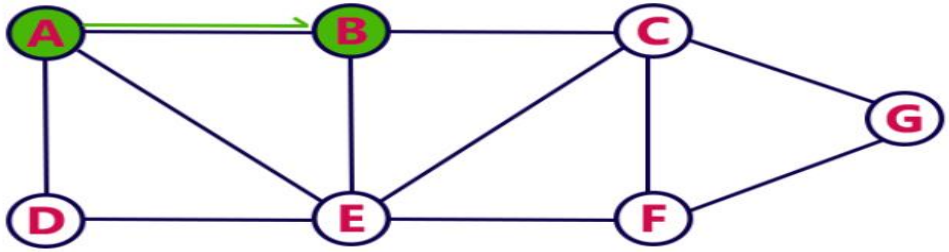
Step 1:

- Select the vertex **A** as starting point (visit **A**).
- Push **A** on to the Stack.



Step 2:

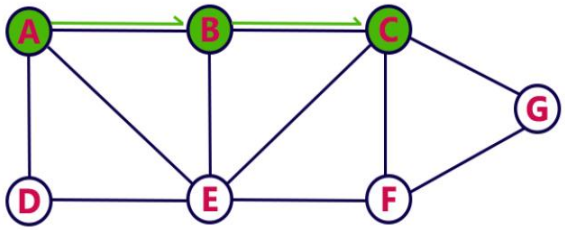
- Visit any adjacent vertex of **A** which is not visited (**B**).
- Push newly visited vertex B on to the Stack.

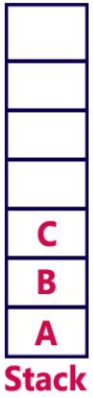


DSF - Example

Step 3:

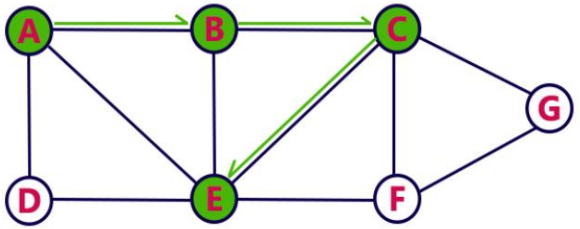
- Visit any adjacent vertext of **B** which is not visited (**C**).
- Push C on to the Stack.





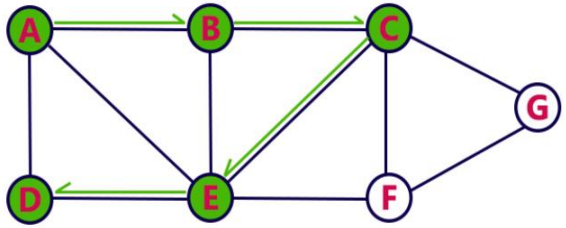
Step 4:

- Visit any adjacent vertext of **C** which is not visited (**E**).
- Push E on to the Stack



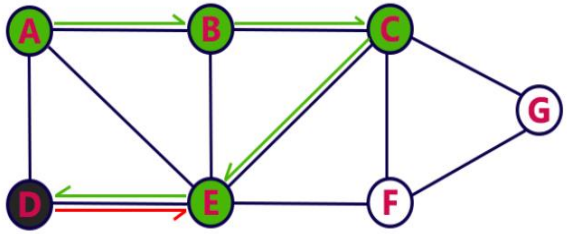
Step 5:

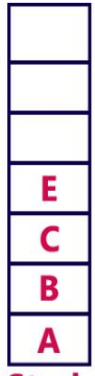
- Visit any adjacent vertext of **E** which is not visited (**D**).
- Push D on to the Stack



Step 6:

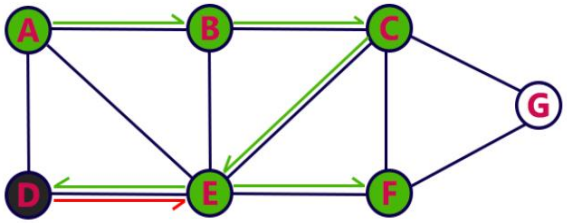
- There is no new vertiex to be visited from D. So use back track.
- Pop D from the Stack.

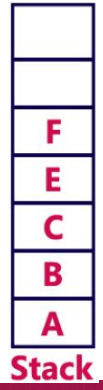




Step 7:

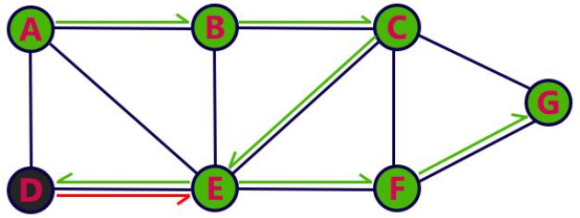
- Visit any adjacent vertex of **E** which is not visited (**F**).
- Push **F** on to the Stack.





Step 8:

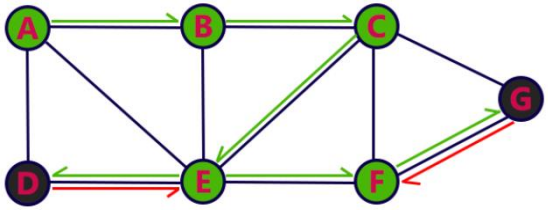
- Visit any adjacent vertex of **F** which is not visited (**G**).
- Push **G** on to the Stack.



DSF - Example

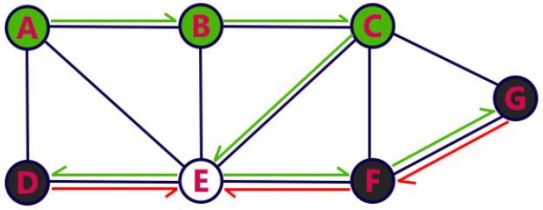
Step 9:

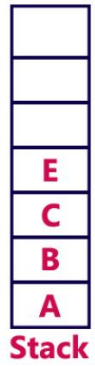
- There is no new vertiex to be visited from G. So use back track.
- Pop G from the Stack.



Step 10:

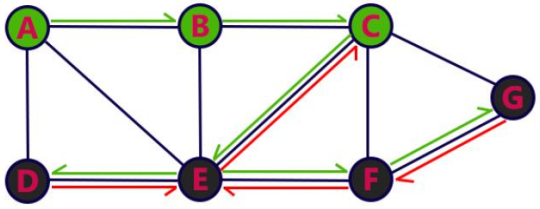
- There is no new vertiex to be visited from F. So use back track.
- Pop F from the Stack.

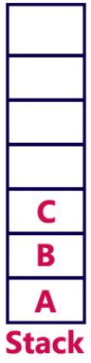




Step 11:

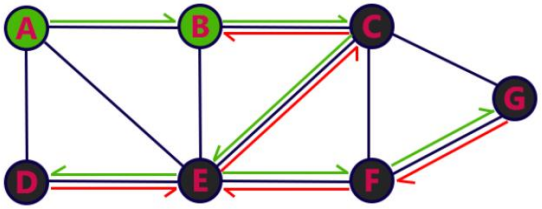
- There is no new vertiex to be visited from E. So use back track.
- Pop E from the Stack.





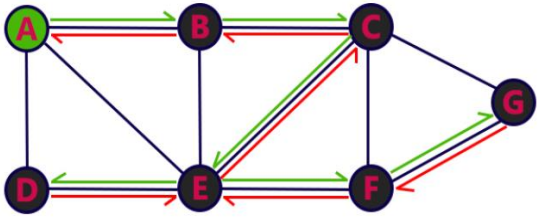
Step 12:

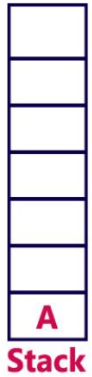
- There is no new vertiex to be visited from C. So use back track.
- Pop C from the Stack.



Step 13:

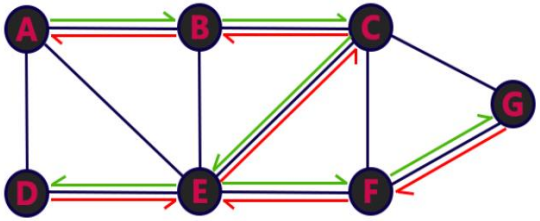
- There is no new vertiex to be visited from B. So use back track.
- Pop B from the Stack.





Step 14:

- There is no new vertiex to be visited from A. So use back track.
- Pop A from the Stack.



DFS Order: A B C E D F G

Practice Problem

Find the BFS and DFS traversal of the graph.