GRAPH

A Graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph. More formally a Graph can be defined as,

*A Graph consists of a finite set of vertices(or nodes) and set of Edges which connect a pair of nodes.*



In the above Graph, the set of vertices V = {0,1,2,3,4} and the set of edges E = {01, 12, 23, 34, 04, 14, 13}.

Graphs are used to solve many real-life problems. Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like linkedIn, Facebook. For example, in Facebook, each person is represented with a vertex(or node). Each node is a structure and contains information like person id, name, gender, locale etc.

GRAPH TRAVERSAL

Graph traversal is a technique used for searching vertex in a graph. The graph traversal is also used to decide the order of vertices to be visited in the search process. A graph traversal finds the egdes to be used in the search process without creating loops. That means using graph traversal we visit all the vertices of graph without getting into looping path.  
  
There are two graph traversal techniques and they are as follows...

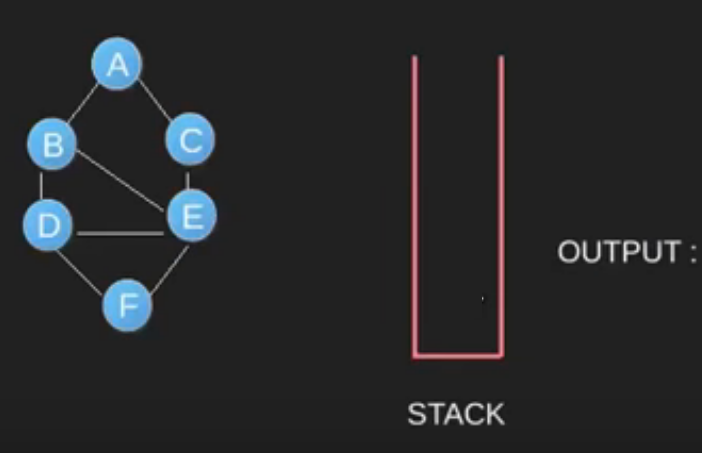
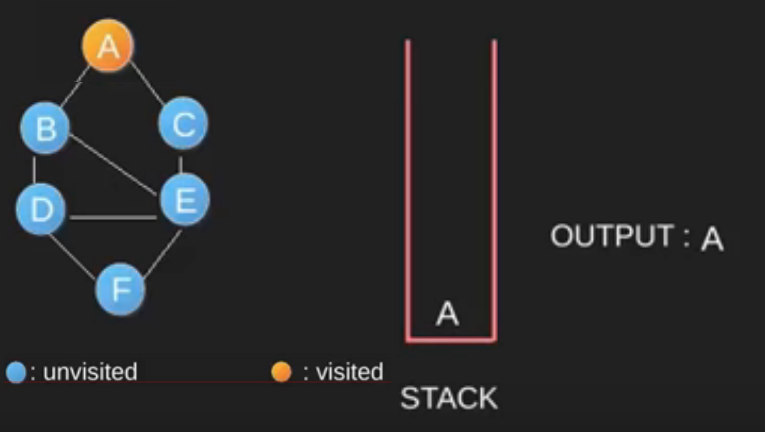
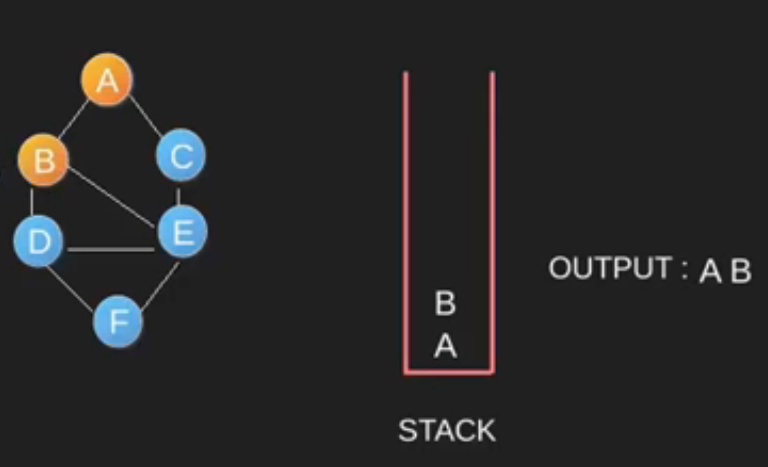
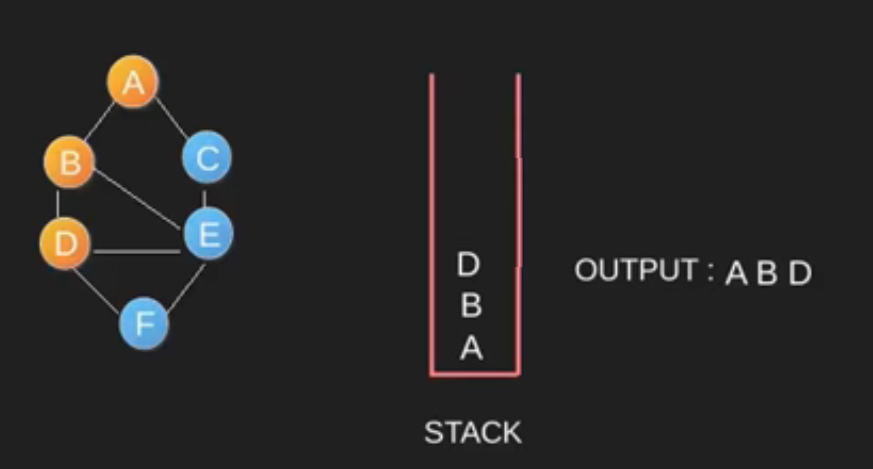
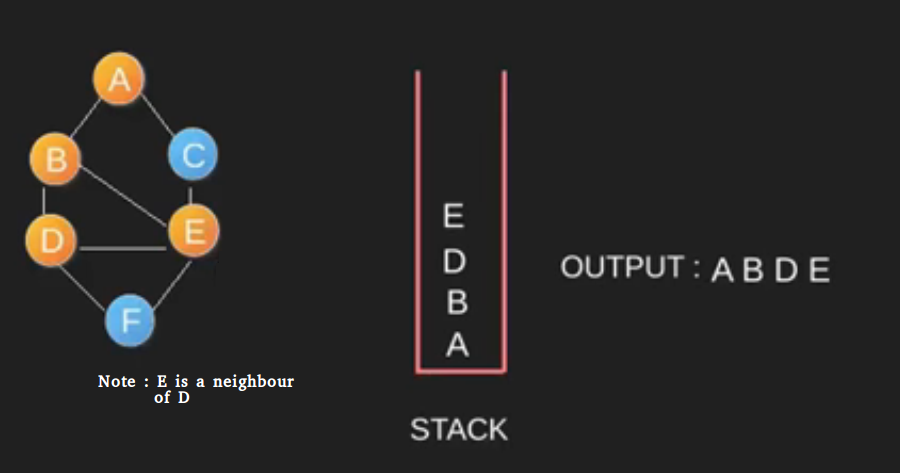
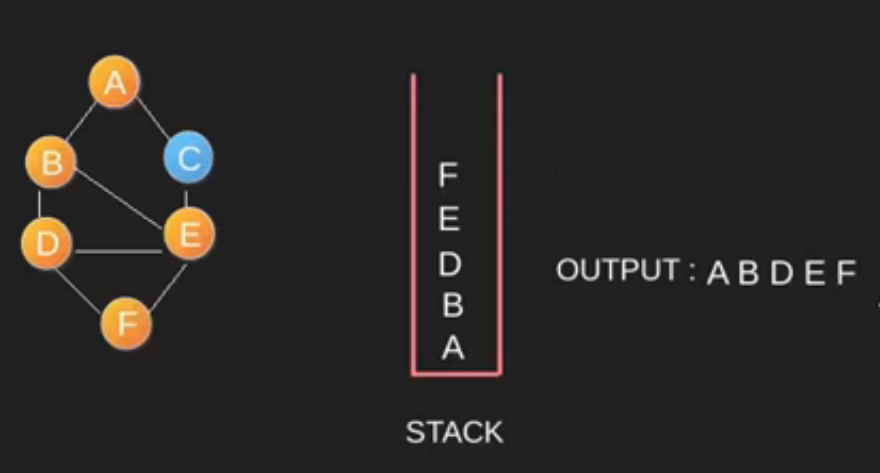
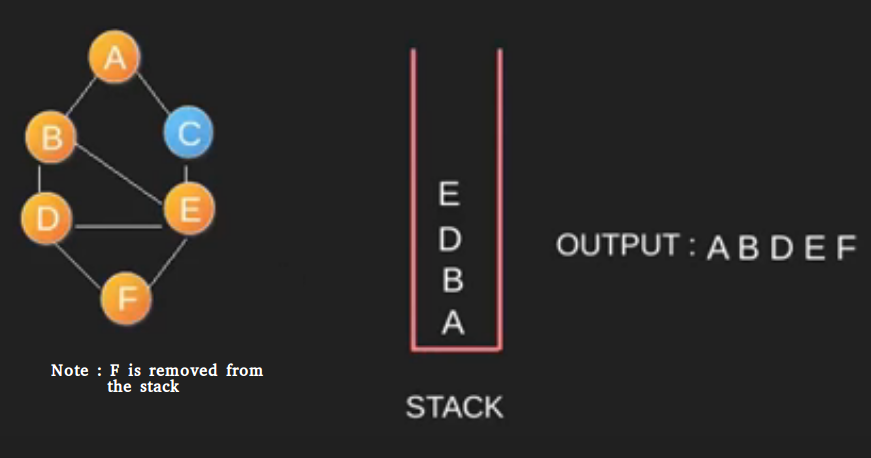
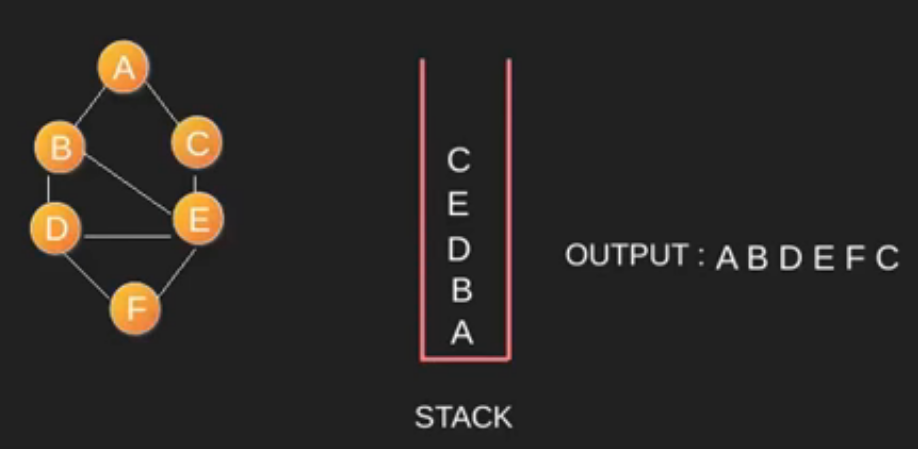
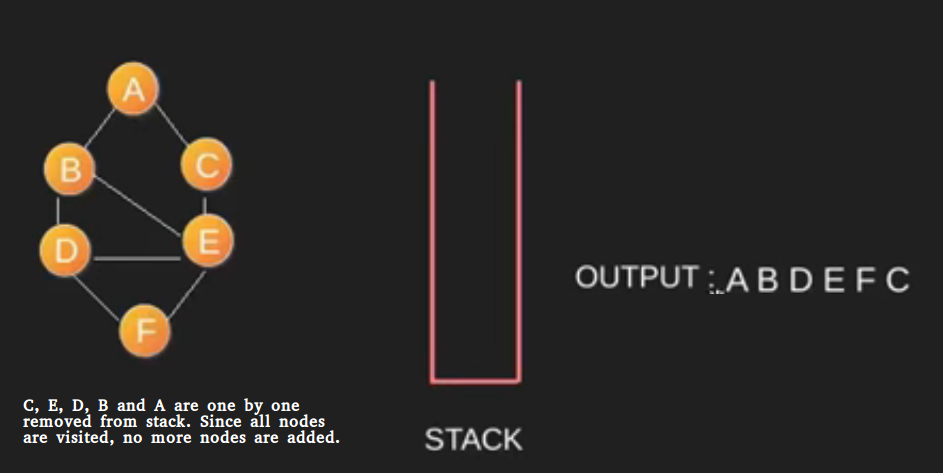
1. **DFS (Depth First Search)**
2. **BFS (Breadth First Search)**

* **DFS (Depth First Search) of graph**

DFS traversal of a graph produces a **spanning tree** as final result. **Spanning Tree** is a graph without loops. We use **Stack data structure**with maximum size of total number of vertices in the graph to implement DFS traversal.  
  
We use the following steps to implement DFS traversal...

* **Step 1 -**Define a Stack of size total number of vertices in the graph.
* **Step 2 -**Select any vertex as **starting point** for traversal. Visit that vertex and push it on to the Stack.
* **Step 3 -**Visit any one of the non-visited **adjacent** vertices of a vertex which is at the top of stack and push it on to the stack.
* **Step 4 -**Repeat step 3 until there is no new vertex to be visited from the vertex which is at the top of the stack.
* **Step 5 -**When there is no new vertex to visit then use **back tracking** and pop one vertex from the stack.
* **Step 6 -**Repeat steps 3, 4 and 5 until stack becomes Empty.
* **Step 7 -**When stack becomes Empty, then produce final spanning tree by removing unused edges from the graph

**Example of DFS**

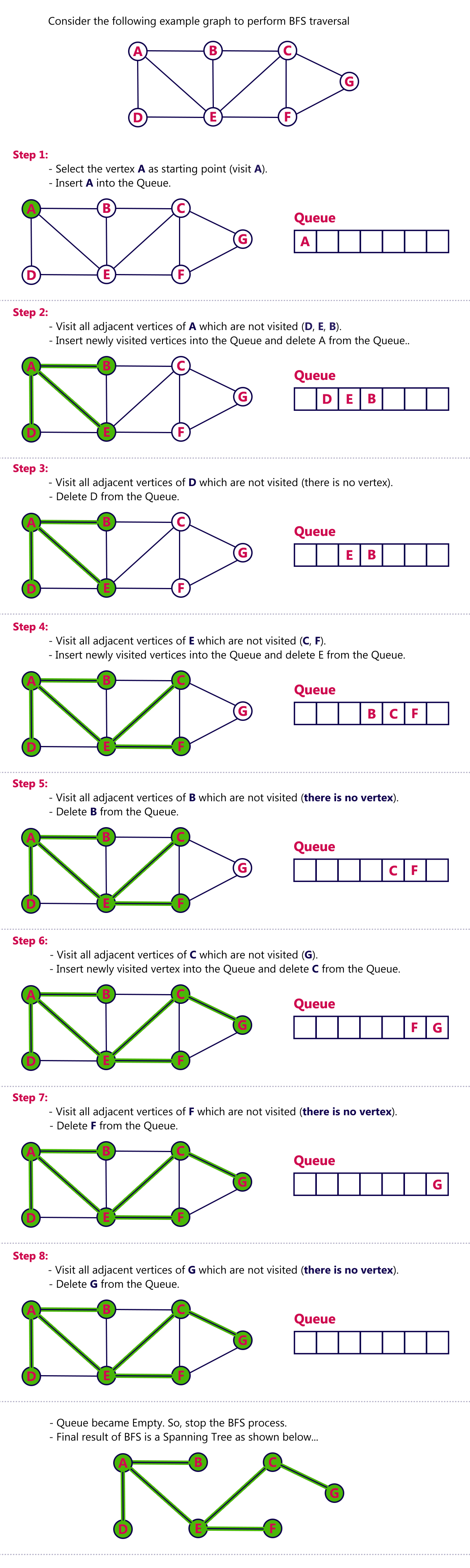
[](https://www.geeksforgeeks.org/wp-content/uploads/dfs1.png)  
[](https://www.geeksforgeeks.org/wp-content/uploads/dfs2.png)  
[](https://www.geeksforgeeks.org/wp-content/uploads/dfs3.png)  
[](https://www.geeksforgeeks.org/wp-content/uploads/dfs4.png)[](https://www.geeksforgeeks.org/wp-content/uploads/dfs5.png)  
[](https://www.geeksforgeeks.org/wp-content/uploads/DFS6.png)  
[](https://www.geeksforgeeks.org/wp-content/uploads/DFS7.png)  
[](https://www.geeksforgeeks.org/wp-content/uploads/DFS8.png)  
[](https://www.geeksforgeeks.org/wp-content/uploads/DFS9.png)

# BFS (Breadth First Search) of graph

BFS traversal of a graph produces a **spanning tree** as final result. **Spanning Tree** is a graph without loops. We use **Queue data structure**with maximum size of total number of vertices in the graph to implement BFS traversal.  
  
We use the following steps to implement BFS traversal...

* **Step 1 -**Define a Queue of size total number of vertices in the graph.
* **Step 2 -**Select any vertex as **starting point** for traversal. Visit that vertex and insert it into the Queue.
* **Step 3 -**Visit all the non-visited **adjacent** vertices of the vertex which is at front of the Queue and insert them into the Queue.
* **Step 4 -**When there is no new vertex to be visited from the vertex which is at front of the Queue then delete that vertex.
* **Step 5 -**Repeat steps 3 and 4 until queue becomes empty.
* **Step 6 -**When queue becomes empty, then produce final spanning tree by removing unused edges from the graph

**Example of BFS**

 **Final result is ADEBCFG**