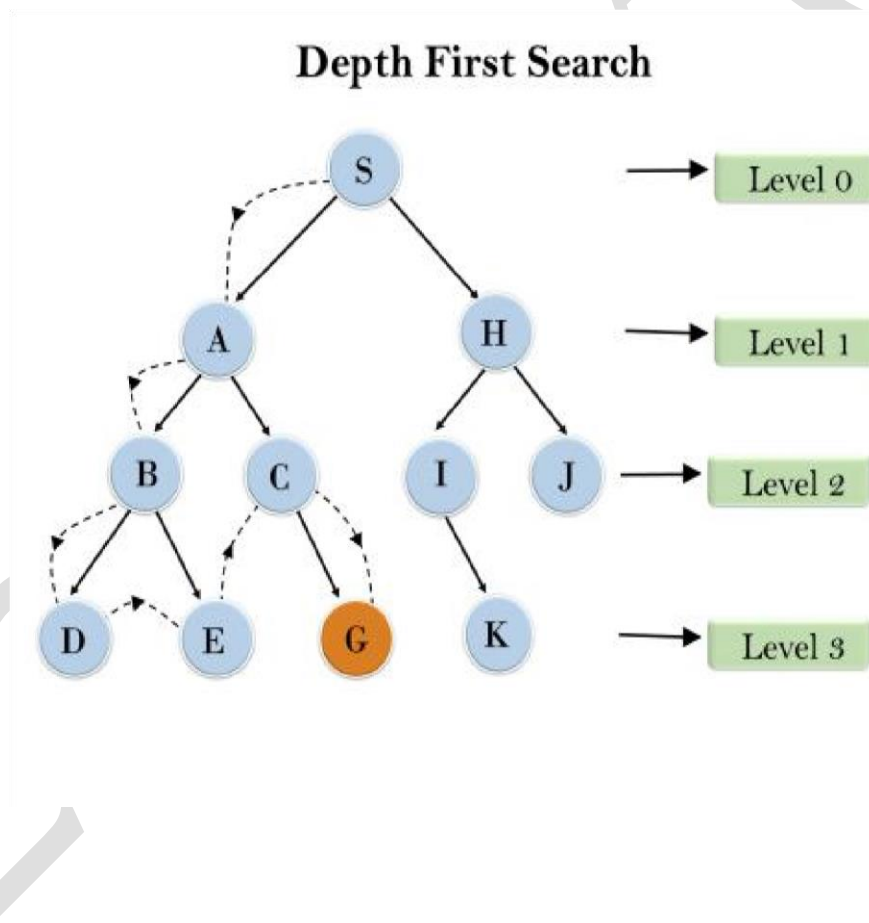


## EX.NO: 2

### DEPTH FIRST SEARCH

- Depth first search (DFS) algorithm or searching technique starts with the root node of the graph G, and then travel to deeper and deeper until we find the goal node or the node which has no children by visiting different node of the tree.
- The algorithm, then backtracks or return back from the dead end or last node towards the most recent node that is yet to be completely unexplored.
- The data structure(DS) which is being used in DFS Depth first search is stack. The process is quite similar to BFS algorithm.
- In DFS, the edges that goes to an unvisited node are called discovery edges while the edges that goes to an already visited node are called block edges.



## AIM:

To implement a depth first search problem using python.

### **SOURCE CODE :**

```
import networkx as nx
```

```
#FUNCTION TO SOLVE DFS
```

```
def solveDFS(graph, v, visited) :  
    visited.add(v)    print(v, end=' ')    for  
    neighbour in graph[v] :    if neighbour  
    not in visited :        solveDFS(graph,  
    neighbour, visited)    g = nx.DiGraph()
```

```
#CREATE A GRAPH USING NETWORKX
```

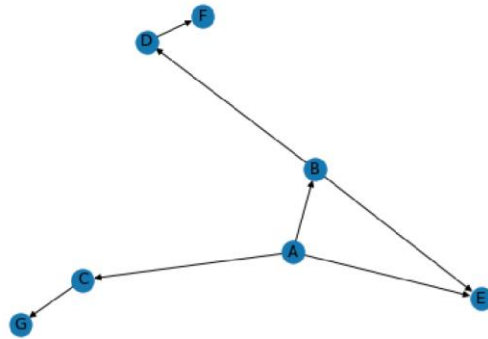
```
g.add_edges_from([('A','B'),('A','C'),('C','G'),('B','D'),('B','E'),('D','F'),('A','E')]) # Add edges for th  
at graph  
nx.draw(g, with_labels=True) # Graph Visualization
```

```
#SOLVE DFS FOR THAT GRAPH
```

```
print("Following is DFS from (starting from vertex A)") visited  
= set()  
solveDFS(g, 'A', visited)
```

### **OUTPUT :**

Following is DFS from (starting from vertex A)  
A B D F E C G



**RESULT :** Thus the above python code is executed successfully and output is verified.

**EX.NO: 3**

V KEERTHISH