EXND: 02
DATE:

DFS - Depth First Search.

AIM:-

Create a program to solve depth first search Algorithm.

ALGORITHM:-

1. Initialize visited set:-

→ create an empty set 'visited' to keep track of visited node.

2. create the graph:-

→ Ask the user the number of nodes in graph.

→ For each node, ask the user to input the node's neighbour (adjacent nodes) and store the information in directory graph.

3. DFS function:-

→ Define a recursive function dfs (visited, graph, node):

* check if node is visited.
  → If the node is not in the visited then set proceed.
* visit the node.
  → Print the node and add it to visited set
* Recursively visit neighbours.
  → For each neighbour in graph [node] call dfs [visited, graph, neighbour] recursively.

4. start DFS:-

→ Ask the user to input the starting node.

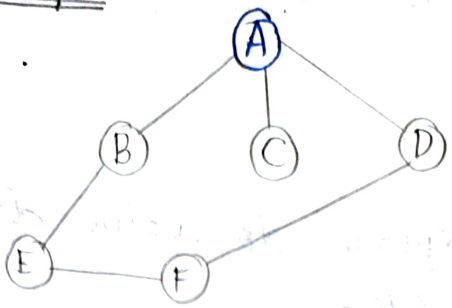
→ call the dfs() function with the visited set, graph and start node.

5. end DFS:-

→ The DFS traversal is complete when all reachable nodes from the start node have been visited.

Example:

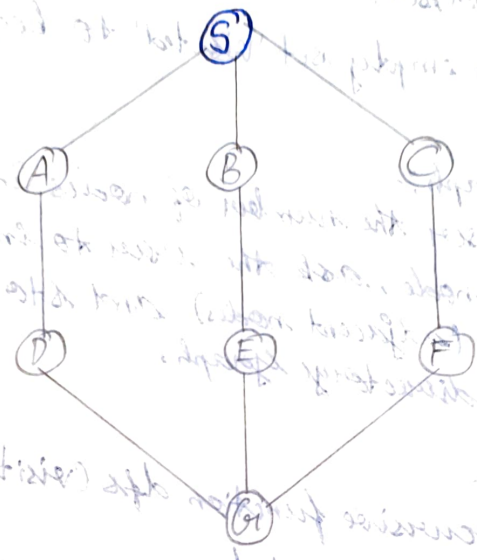
1).



A
A B
A B E
A B E F
A B E F D
A B E F D C

A B E F D C

2)



S						
S	A					
S	A	D				
S	A	D	G1			
S	A	D	G1	E		
S	A	D	G1	E	B	
S	A	D	G1	E	B	F
S	A	D	G1	E	B	F	C

S A D G1 E B F C

code:-

```
def dfs (visited, graph, node):
    if node is not in visited:
        print (node)
        visited. add (node)
        for neighbour in graph [node]
            dfs (visited, graph, neighbour)

def create_graph ():
    graph = {}
    num_nodes = int (input ("Enter no of nodes in graph : "))
    for _ in range (num_nodes):
        node = input ("Enter the node:")
        neighbours = input (f"Enter the neighbours of {node}
                            seperated by space: "). split ()
        graph [node] = neighbours
    return graph

visited = set ()
graph = create_graph ()
start_node = input ("Enter the starting node:"):

Print ("Following is the Depth First Search")
dfs (visited, graph, start_node)
```

Output:-

Enter the number of nodes in graph: 8
Enter the node: S
Enter the neighbour seprate by space: a b. c
Enter the node: a
Enter the neighbour seprate by space: S d
Enter the node: d
Enter the neighbour seprate by space: a g
Enter the node: g
Enter the neighbour seprate by space: D E F
Enter the node: e
Enter the neighbour seprate by space: or B.
Enter the node: B
Enter the neighbour seprate by space: E S
Enter the node: c
Enter the neighbour seprate by space: S F
Enter the node: t
Enter the neighbour seprate by space: c or
Enter the starting node: S
following is the depth first search.
S A D G E B F C.

RESULT:-

Thus, the DFS program is executed and output is verified successfully.