Soumen Samanta

Batch 4

16010420133

Exp 2 – AI LAB

**Algorithm:**

* Create a recursive function that takes the index of the node and a visited array.
* Mark the current node as visited and print the node
* Traverse all the adjacent and unmarked nodes and call the recursive function with the index of the adjacent node

**Code:**

from collections import defaultdict

class Graph:

    def \_\_init\_\_(self):  #constructor

        self.graph = defaultdict(list)  #default dict to store graph

    def addEdge(self,u,v): # function to add edge to the graph

        self.graph[u].append(v)

    def DFSUtil(self,v,visited):

        visited.add(v)    # Mark the current node as visited and print it

        print(v,end=' ')

        for neighbour in self.graph[v]: # Recur for all the vertices

            if neighbour not in visited:

                self.DFSUtil(neighbour,visited)

    def DFS(self,v):

        visited = set() #Create a set to store all the visited vertices

        self.DFSUtil(v,visited)

#Driver Code

g = Graph()

g.addEdge(0, 10)

g.addEdge(0, 12)

g.addEdge(12, 4)

g.addEdge(4, 0)

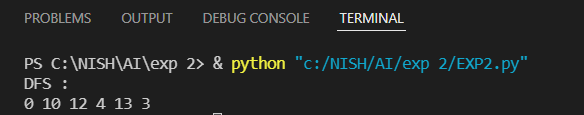
g.addEdge(4, 13)

g.addEdge(13, 3)

print("DFS : ")

g.DFS(0)

**Output:**



**Problem:**

