**ASSIGNMENT NO : 1**

**Name :** Sanika Anand Jadav

**Class :** TE **Div :** A

**Roll No. :** 49

**Subject :** Artificial Intellience

**Code For BFS**

from collections import defaultdict

class Graph:

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

self.graph = defaultdict(list)

def addEdge(self, u, v):

self.graph[u].append(v)

def BFS(self, s):

visited = [False] \* (max(self.graph) + 1)

queue = []

queue.append(s)

visited[s] = True

while queue:

s = queue.pop(0)

print(s, end=" ")

for i in self.graph[s]:

if not visited[i]:

queue.append(i)

visited[i] = True

g = Graph()

g.addEdge(0, 1)

g.addEdge(0, 2)

g.addEdge(1, 2)

g.addEdge(2, 0)

g.addEdge(2, 3)

g.addEdge(3, 3)

print("Breadth-First Search starting from vertex 2:")

g.BFS(2)

**OUTPUT:**

Breadth-First Search starting from vertex 2 : 2 0 3 1

**Code For DFS**

from collections import defaultdict

class Graph:

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

self.graph = defaultdict(list)

def addEdge(self, u, v):

self.graph[u].append(v)

def DFSUtil(self, v, visited):

visited.add(v)

print(v, end=' ')

for neighbour in self.graph[v]:

if neighbour not in visited:

self.DFSUtil(neighbour, visited)

def DFS(self, v):

visited = set()

self.DFSUtil(v, visited)

g = Graph()

g.addEdge(0, 1)

g.addEdge(0, 2)

g.addEdge(1, 2)

g.addEdge(2, 0)

g.addEdge(2, 3)

g.addEdge(3, 3)

print("Following is DFS from (starting from vertex 2)")

g.DFS(2)

**Output:**

Following is DFS from (starting from vertex 2) : 2 0 1 3