**Name-**Pawar Aadesh

**Roll no-**33

**Problem Statement**

Implement depth first search algorithm and Breadth First Search algorithm. Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure

**Input** **from collections import deque**

**class Graph:**

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

self.vertices = vertices

self.adjacency\_list = [[] for \_ in range(vertices)]

def add\_edge(self, u, v):

self.adjacency\_list[u].append(v)

self.adjacency\_list[v].append(u)

def dfs\_recursive(self, vertex, visited):

visited[vertex] = True

print(vertex, end=" ")

for neighbor in self.adjacency\_list[vertex]:

if not visited[neighbor]:

self.dfs\_recursive(neighbor, visited)

def dfs(self, start\_vertex):

visited = [False] \* self.vertices

self.dfs\_recursive(start\_vertex, visited)

def bfs(self, start\_vertex):

visited = [False] \* self.vertices

queue = deque([start\_vertex])

visited[start\_vertex] = True

while queue:

vertex = queue.popleft()

print(vertex, end=" ")

for neighbor in self.adjacency\_list[vertex]:

if not visited[neighbor]:

queue.append(neighbor)

visited[neighbor] = True

# Example usage

g = Graph(6)

g.add\_edge(0, 1)

g.add\_edge(0, 2)

g.add\_edge(1, 3)

g.add\_edge(2, 4)

g.add\_edge(2, 5)

start\_vertex = 0

print("Depth First Traversal:")

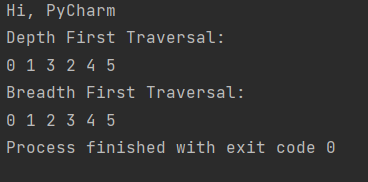
g.dfs(start\_vertex)

print()

print("Breadth First Traversal:")

g.bfs(start\_vertex)

**Ouput**

****