**Practicle 1**

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

from collections import defaultdict, deque

class Graph:

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

# Default dictionary to store graph

self.graph = defaultdict(list)

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

self.graph[u].append(v)

self.graph[v].append(u) # Since it's undirected, add the reverse edge as well.

def dfs\_recursive(self, vertex, visited=None):

if visited is None:

visited = set()

visited.add(vertex)

print(vertex, end=' ')

for neighbor in self.graph[vertex]:

if neighbor not in visited:

self.dfs\_recursive(neighbor, visited)

def bfs(self, start):

visited = set() # To keep track of visited nodes

queue = deque([start]) # Use deque for an efficient queue implementation

visited.add(start)

while queue:

vertex = queue.popleft() # Pop the front of the queue

print(vertex, end=' ')

# Add all unvisited neighbors to the queue

for neighbor in self.graph[vertex]:

if neighbor not in visited:

queue.append(neighbor)

visited.add(neighbor)

# Example usage:

g = Graph()

g.add\_edge(0, 1)

g.add\_edge(0, 2)

g.add\_edge(1, 2)

g.add\_edge(2, 3)

g.add\_edge(3, 4)

print("Depth First Search (starting from vertex 0):")

g.dfs\_recursive(0)

print("\nBreadth First Search (starting from vertex 0):")

g.bfs(0)