**Name: Usman Ul Haq**

**Roll# BSAIM-035**

**AI LAB TASKs**

**LAB TASK 6:**

**BFS without Queue & without Node**

class Graph:

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

self.vertices = vertices

self.adj\_list = {i: [] for i in range(vertices)}

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

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

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

def bfs\_recursive(self, visited, level, current\_level):

if not current\_level:

return

next\_level = []

for node in current\_level:

if not visited[node]:

print(node, end=" ")

visited[node] = True

next\_level.extend([neighbor for neighbor in self.adj\_list[node] if not visited[neighbor]])

self.bfs\_recursive(visited, level + 1, next\_level)

def bfs(self, start):

visited = [False] \* self.vertices

self.bfs\_recursive(visited, 0, [start])

g = Graph(5)

g.add\_edge(0, 1)

g.add\_edge(0, 2)

g.add\_edge(1, 3)

g.add\_edge(2, 4)

print("BFS without Queue & without Node:")

g.bfs(0)

**Output**

**BFS without Queue & without Node:**

**0 1 2 3 4**

**2. BFS with Queue & Node**

class Node:

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

self.value = value

self.neighbors = []

class Graph:

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

self.nodes = {}

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

if u not in self.nodes:

self.nodes[u] = Node(u)

if v not in self.nodes:

self.nodes[v] = Node(v)

self.nodes[u].neighbors.append(self.nodes[v])

self.nodes[v].neighbors.append(self.nodes[u])

def bfs(self, start):

if start not in self.nodes:

return

visited = set()

queue = deque([self.nodes[start]])

while queue:

node = queue.popleft()

if node.value not in visited:

print(node.value, end=" ")

visited.add(node.value)

for neighbor in node.neighbors:

if neighbor.value not in visited:

queue.append(neighbor)

g = Graph()

g.add\_edge(0, 1)

g.add\_edge(0, 2)

g.add\_edge(1, 3)

g.add\_edge(2, 4)

print("\nBFS with Queue & Node:")

g.bfs(0)

**Output**

**BFS with Queue & Node:**

**0 1 2 3 4**