class Graph:

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

self.graph = {}

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

if u not in self.graph:

self.graph[u] = []

if v not in self.graph:

self.graph[v] = []

self.graph[u].append(v)

self.graph[v].append(u) # Remove this line for a directed graph

def dfs(self, node, visited=None):

if visited is None:

visited = set()

visited.add(node)

print(node, end=" ")

for neighbor in self.graph.get(node, []):

if neighbor not in visited:

self.dfs(neighbor, visited)

def bfs(self, start):

visited = set()

queue = [start]

while queue:

node = queue.pop(0)

if node not in visited:

print(node, end=" ")

visited.add(node)

for neighbor in self.graph.get(node, []):

if neighbor not in visited:

queue.append(neighbor)

# Take user input

g = Graph()

num\_edges = int(input("Enter the number of edges: "))

print("Enter each edge as 'u v' (space-separated nodes):")

for \_ in range(num\_edges):

u, v = map(int, input().split())

g.add\_edge(u, v)

start\_node = int(input("Enter the starting node for traversal: "))

print("\nDFS Traversal:")

g.dfs(start\_node)

print("\nBFS Traversal:")

g.bfs(start\_node)

Enter the number of edges: 5

Enter each edge as 'u v' (space-separated nodes):

1 2

1 3

2 4

3 5

4 5

Enter the starting node for traversal: 1