





Code:

# Shubham - 31118

# undirected graph

class Graph:

    def \_\_init\_\_(self, n) -> None:

        self.n = n

        self.m = 0

        self.adj\_list = [[] for \_ in range(n+7)]

    def add\_adge(self, u, v):

        self.m += 1

        self.adj\_list[u].append(v)

        self.adj\_list[v].append(u)

    def print\_graph(self):

        print("Graph (Adjacency List Representation):")

        for u in range(self.n):

            print(u, end="-> ")

            for v in self.adj\_list[u]:

                print(v, end=" ")

            print()

    def dfs\_ut(self, u, vis):

        vis.append(u)

        print(u, end=" ")

        for v in self.adj\_list[u]:

            if v not in vis:

                self.dfs\_ut(v, vis)

    def dfs(self):

        print("Depth-first-traversal: ")

        vis = []

        for i in range(1, self.n+1):

            if i not in vis:

                self.dfs\_ut(i, vis)

                print()

    def bfs\_ut(self, vis, que):

        if not que:

            return

        u = que[0]

        que.pop(0)

        print(u, end=" ")

        for v in self.adj\_list[u]:

            if v not in vis:

                vis.append(v)

                que.append(v)

        self.bfs\_ut(vis, que)

    def bfs(self):

        print("Breadth-first-traversal: ")

        vis = []

        for i in range(1, self.n+1):

            if i not in vis:

                vis.append(i)

                self.bfs\_ut(vis, [i])

                print()

def main():

    # 1 - based indexing

    n = int(input("Enter number of vertices: "))

    m = int(input("Enter number of edges: "))

    g = Graph(n)

    print("Enter the edges details")

    for i in range(m):

        inp = input()

        [u\_, v\_] = inp.split(' ')

        u = int(u\_)

        v = int(v\_)

        g.add\_adge(u, v)

    g.print\_graph()

    g.dfs()

    g.bfs()

main()











