#DFS

from collections import defaultdict

def dfsRec(adj, visited, s, res, goal=None):

visited[s] = True

res.append(s)

if s == goal:

return True

for neighbor in adj[s]:

if not visited[neighbor]:

if dfsRec(adj, visited, neighbor, res, goal):

return True

return False

def DFS(adj, V, start, goal=None):

visited = [False] \* V ## initialyy the visited list is false

res = []

found = dfsRec(adj, visited, start, res, goal)

return res, found

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

adj[u].append(v)

adj[v].append(u) # undirected

# -------- Driver code --------

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

adj = defaultdict(list)

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

print("Enter edges in the format: from to")

for \_ in range(E):

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

add\_edge(adj, u, v)

start = int(input("Enter start node: "))

goal\_input = input("Enter goal node (or press Enter to skip): ")

goal = int(goal\_input) if goal\_input.strip() else None

res, found = DFS(adj, V, start, goal)

print("DFS traversal:", " ".join(map(str, res)))

if goal is not None:

if found:

print("Goal node reached!")

else:

print("Goal node not reachable.")