Bfs recursive

from collections import defaultdict, deque

# defaultdict(set): simplifies graph creation.

deque: double-ended queue — efficient appending/popping from both ends (used for BFS queues).

def bfs\_traverse\_recursive(graph, queue, visited):

"""

Recursively processes one node from the queue at a time,

printing it, enqueuing its unvisited neighbors, then recursing.

"""

if not queue:

return

current = queue.popleft()

print(current) # BFS traversal output

for nbr in graph[current]:

if nbr not in visited:

visited.add(nbr)

queue.append(nbr)

bfs\_traverse\_recursive(graph, queue, visited)

# i f queue is empty → stop.

# Remove the front vertex (current) from queue, print it.

# Add all its unvisited neighbors to the queue, and mark them visited.

# Call itself recursively.

def bfs\_path\_recursive(graph, queue, visited, goal):

"""

Recursively processes (vertex, path) tuples until it finds `goal`.

Returns the path list, or None if not found.

"""

if not queue:

return None

vertex, path = queue.popleft()

if vertex == goal:

return path

for nbr in graph[vertex]:

if nbr not in visited:

visited.add(nbr)

queue.append((nbr, path + [nbr]))

return bfs\_path\_recursive(graph, queue, visited, goal)

#  If queue is empty → no path found.

# Remove the front (vertex, path) tuple.

#If vertex is the goal → return the path.

#Else, for each unvisited neighbor:

#Mark it visited.

#Add (neighbor, updated\_path) to queue.

# Recursively continue until path is found.

def main():

#Build undirected graph: for each neighbor of v, both v to u and u to v edges are created.

# Build an undirected graph from user input

graph = defaultdict(set)

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

for \_ in range(n):

v = input("Vertex name: ")

neighs = input(f"Neighbors of {v} (space-separated): ").split()

for u in neighs:

graph[v].add(u)

graph[u].add(v)

start = input("Start vertex: ")

goal = input("Goal vertex: ")

# --- 1) BFS traversal ---

print("\nBFS traversal (recursive):")

visited = {start}

queue = deque([start])

bfs\_traverse\_recursive(graph, queue, visited)

# --- 2) BFS path to goal ---

visited = {start}

queue = deque([(start, [start])])

path = bfs\_path\_recursive(graph, queue, visited, goal)

print("\nBFS path to goal:")

if path:

print(path)

else:

print(f"No path found from {start} to {goal}.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

# shravanbobade@Shravans-Laptop BFS % python bfs\_recursive.py

# Enter number of vertices: 5

# Vertex name: a

# Neighbors of a (space-separated): b c e

# Vertex name: b

# Neighbors of b (space-separated): a d

# Vertex name: c

# Neighbors of c (space-separated): a d

# Vertex name: d

# Neighbors of d (space-separated): a b c

# Vertex name: e

# Neighbors of e (space-separated): a d

# Start vertex: a

# Goal vertex: c

# BFS traversal (recursive):

# a

# b

# d

# c

# e

# BFS path to goal:

# ['a', 'c']