**Bài 3 Tìm Kiếm trong không gian trạng thái**

**Bài tập 1. Tạo tập tin biểu diễn đồ thị**

10

A C D E F

B

C H

D E H

E I K

F G I

G

H K

I B K

K B

- Dòng đầu: số lượng đỉnh.

- Mỗi dòng sau: một đỉnh và các đỉnh kề của nó.

**Bài tập 2. Cài đặt chương trình đọc tập tin và đưa vào cấu trúc**

import pprint

# --- Đọc đồ thị từ file ---

def read\_graph(filename):

    G = {}

    with open(filename, "rt", encoding="utf-8") as f:

        n = int(f.readline().strip())  # số đỉnh

        for line in f:

            parts = line.strip().split()

            if parts:

                G[parts[0]] = set(parts[1:])

    return G

# --- BFS ---

def BFS(G, start, goal):

    if G.get(start) is None or G.get(goal) is None:

        return None

    path = {start: None}

    s\_open = [start]   # queue

    s\_closed = []

    while s\_open:

        u = s\_open.pop(0)

        s\_closed.append(u)

        if u == goal:

            break

        for v in G[u]:

            if v not in s\_open and v not in s\_closed:

                s\_open.append(v)

                path[v] = u

    return path

# --- DFS ---

def DFS(G, start, goal):

    if G.get(start) is None or G.get(goal) is None:

        return None

    path = {start: None}

    s\_open = [start]   # stack

    s\_closed = []

    while s\_open:

        u = s\_open.pop()

        s\_closed.append(u)

        if u == goal:

            break

        for v in G[u]:

            if v not in s\_open and v not in s\_closed:

                s\_open.append(v)

                path[v] = u

    return path

# --- Hàm dựng đường đi từ path ---

def find\_path(path, start, goal):

    if path is None or goal not in path:

        return []

    result = []

    cur = goal

    while cur is not None:

        result.insert(0, cur)

        cur = path[cur]

    if not result or result[0] != start:

        return []

    return result

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

    G = read\_graph("graph.txt")

    print("Đồ thị (danh sách kề):")

    pprint.pprint(G)

    # Tìm đường từ A đến B

    bfs\_path = BFS(G, "A", "B")

    dfs\_path = DFS(G, "A", "B")

    print("\nBFS path map:")

    pprint.pprint(bfs\_path)

    print("Đường đi BFS A→B:", find\_path(bfs\_path, "A", "B"))

    print("\nDFS path map:")

    pprint.pprint(dfs\_path)

    print("Đường đi DFS A→B:", find\_path(dfs\_path, "A", "B"))

**Bài 3. Kết quả**

A white background with a blue and yellow border

AI-generated content may be incorrect.

A screenshot of a computer code

AI-generated content may be incorrect.