**Tên: Trương Hữu Nghĩa**

**MSSV: 3122410263**

**Bài tập trong file Jupyter Notebook**

**Bài 1. Biểu diễn đồ thị**

A diagram of a network

AI-generated content may be incorrect.

**1.Biểu diễn bằng danh sách / ma trận kề**

Code

ADJ ={}

ADJ['A'] = {'C','E','F','D'}

ADJ['B'] = {}

ADJ['C'] = {'H'}

ADJ['D'] = {'H','E'}

ADJ['E'] = {'K','I'}

ADJ['F'] = {'I','G'}

ADJ['G'] = {}

ADJ['H'] = {'K'}

ADJ['I'] = {'K','B'}

ADJ['K'] = {'B'}

Kết quả:

A black rectangular object with a white border

AI-generated content may be incorrect.

**2. Thao tác duyệt đồ thị**

(a) Liệt kê các đỉnh trong đồ thị

(b) Liệt kê tất cả các cạnh đồ thị hiển thị dạng danh sách kề

(c) Cho hai đỉnh A,B. Hỏi 2 đỉnh A, B có kề nhau không?

(d) Cho 1 đỉnh A , hỏi danh sách các đỉnh kề với A

(a) Liệt kê các đỉnh trong đồ thị

Code:

print("(a) Liệt kê các đỉnh trong đồ thị")

print("Các đỉnh của đồ thị: ", end = " ")

""" CÁC BẠN LÀM BÀI Ở ĐÂY """

# ...

for v in ADJ.keys():

    print(v, end = " ")

Kết quả:

A black background with white text

AI-generated content may be incorrect.

(b) Liệt kê tất cả các cạnh đồ thị hiển thị dạng danh sách kề

Code:

print("(b) Liệt kê tất cả các cạnh đồ thị hiển thị dạng danh sách kề")

print("Danh sách kề của đồ thị: ")

def HienThiDoThi(G):

    """ CÁC BẠN LÀM BÀI Ở ĐÂY """

    # ...

    for u in G.keys():

        tmp = ""

        for x in G[u]:

            tmp +=x+", "

        print("+ Đỉnh", u,":",tmp)

    pass

HienThiDoThi(ADJ)

Kết quả:

A computer screen with white text

AI-generated content may be incorrect.

(c) Cho hai đỉnh A,B. Hỏi 2 đỉnh A, B có kề nhau không?

Code:

print("(c) Cho hai đỉnh A, B. Hỏi hai đỉnh A, B có kề nhau không?")

def LaKe(G, a, b):

    """

    input: G, a, b

    return:

    + -1: dinh a, hay b khong ton tai

    +  0: dinh a, b ton tai va a khong ke b

    +  1: dinh a, b ton tai va a, b ke nhau

    """

    result = None

    """ CÁC BẠN LÀM BÀI Ở ĐÂY """

    # ...

    a\_adj = G.get(a)

    b\_adj = G.get(b)

    # Nếu a hoặc b không tồn tại trong đồ thị

    if a\_adj is None or b\_adj is None:

        return -1

    for i in a\_adj:

        if i == b:

            return 1

    result = 0

    return result

Gọi hàm:

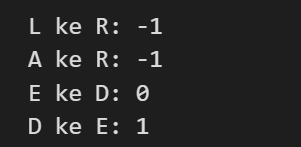
a = "L"; b = "R"; print(f"{a} ke {b}: {LaKe(ADJ, a, b)}")

a = "A"; b = "R"; print(f"{a} ke {b}: {LaKe(ADJ, a, b)}")

a = "E"; b = "D"; print(f"{a} ke {b}: {LaKe(ADJ, a, b)}")

a = "D"; b = "E"; print(f"{a} ke {b}: {LaKe(ADJ, a, b)}")

kết quả:



(d) Cho một đỉnh A. Hỏi danh sách các đỉnh kề với A

Code:

print("(d) Cho một đỉnh A. Hỏi danh sách các đỉnh kề với A")

def LayKe(G, a):

    """

    input: G, a

    return:

    + None: neu a khong ton tai

    + []  : a khong ke voi bat ky dinh nao

    + [x, y, ...]: mang cac dinh ke voi a

    """

    result = None

    """ CÁC BẠN LÀM BÀI Ở ĐÂY """

    # ...

    a\_adj = G.get(a)

    if a\_adj is None:

        result = None

    else:

        result = list(a\_adj)

    return result

a = "P"; print(f"Danh sach ke voi dinh {a}: {LayKe(ADJ, a)}")

a = "A"; print(f"Danh sach ke voi dinh {a}: {LayKe(ADJ, a)}")

a = "B"; print(f"Danh sach ke voi dinh {a}: {LayKe(ADJ, a)}")

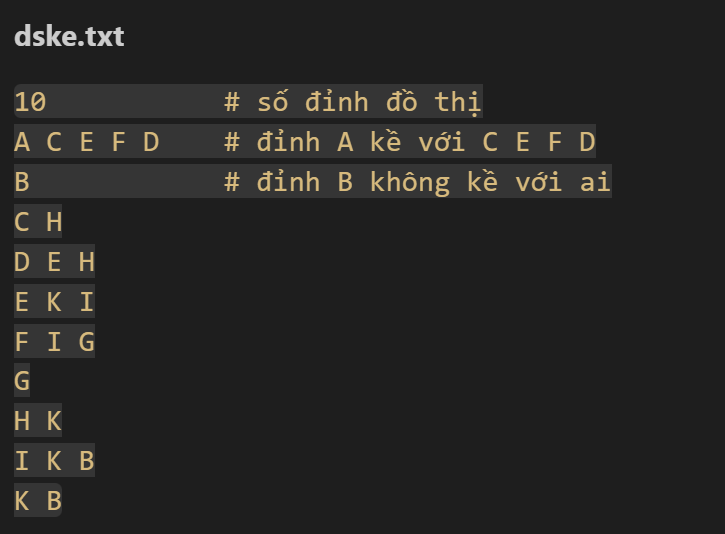
a = "D"; print(f"Danh sach ke voi dinh {a}: {LayKe(ADJ, a)}")

Kết quả

A black background with white text

AI-generated content may be incorrect.

3. Đọc và lưu đồ thị



1. Lưu đồ tị xuống tập tin:

Code:

print("1. Lưu đồ thị xuống tập tin")

def LuuDoThi(G, file\_path, verbose = True):

    import os

    """ CÁC BẠN LÀM BÀI Ở ĐÂY """

    # Tạo thư mục chứa file\_path

    file\_dir = os.path.dirname(file\_path)

    if file\_dir != "" and os.path.exists(file\_path) == False:

        os.makedirs(file\_dir)

        if verbose: print(f"+ Tao thu muc: {file\_dir}")

    # Lưu đồ thị

    with open(file\_path, "wt") as file:

        """ CÁC BẠN LÀM BÀI Ở ĐÂY """

        # ...

        file.write(f"{len(G)}\n")

        for u in G.keys():

            file.write(f"{u} ")

            for v in G[u]:

                file.write(f"{v} ")

            file.write("\n")

        if verbose: print(f"Luu do thi thanh cong xuong tap tin: {file\_path}")

    pass

LuuDoThi(ADJ, "dske1.txt", verbose = True)

with open("dske1.txt", "rt") as file:

    lines = file.readlines()

    for line in lines: print(line, end="")

kết quả của file “dske1.txt”

A black rectangular object with white text

AI-generated content may be incorrect.

1. Đọc đồ thị từ tập tin

Code:

print("2. Đọc đồ thị từ tập tin")

import pprint

def DocDoThi(file\_path, verbose = True):

    """

    return:

    + None: doc that bai

    + <>None: tra ve do thi

    """

    import os

    result = None

    if os.path.exists(file\_path) == False:

        result = None

    else:

        G = {}

        with open(file\_path, "rt") as file:

            n = int(file.readline())

            """ CÁC BẠN LÀM BÀI Ở ĐÂY """

            # ...

            for line in file:

                parts = line.strip().split()

                node = parts[0]

                neighbors = set(parts[1:]) if len(parts) > 1 else set()

                G[node] = neighbors

            pass

        result = G

    return result

G1 = DocDoThi("dske1.txt", verbose = True)

pprint.pprint(G1)

Kết quả:

A computer screen shot of a black screen

AI-generated content may be incorrect.

**Bài 2. Tim kiếm đường đi trên đồ thị**

1. Tìm kiếm theo chiều rộng

Code:

import pprint

def BFS(G, start, goal):

    """

    return:

    + mang chua duong di tu a --> b, neu rong la ko co duong di

    + None: dinh a, b khong hop le

    """

    result = None

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

        result = None

    else:

        path = {}     # path[a]=b nghia la muon di toi dinh b phai qua a

        s\_open   = []

        s\_closed = []

        # đưa start vào open

        s\_open.append(start)

        path[start] = None

        while len(s\_open)>0:

            """ CÁC BẠN LÀM BÀI Ở ĐÂY """

            # ...

            u = s\_open.pop(0)

            for v in G[u]:

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

                    s\_open.append(v)

                    path[v] = u

                    if v == goal:

                        return path

            s\_closed.append(u)

            pass

    return path

path = BFS(ADJ, "A", "B")

pprint.pprint(path)

Kết quả:

A black background with white text

AI-generated content may be incorrect.

Tìm kiếm đường đi từ start đến goal theo theo chiều rộng

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

    """

    Cho mang truy hoi duong, tim ds dinh tu start --> goal

    """

    result = []

    """ CÁC BẠN LÀM BÀI Ở ĐÂY """

    # ...

    x = goal

    while(x is not None):

        result.insert(0,x)

        x = path.get(x)

    return result

    pass

find\_path(path, 'A', 'B')

Kết quả

A black and white text

AI-generated content may be incorrect.

1. Tìm kiếm theo chiều sâu

Code:

import pprint

def DFS(G, start, goal):

    """

    return:

    + mang chua duong di tu a --> b, neu rong la ko co duong di

    + None: dinh a, b khong hop le

    """

    result = None

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

        result = None

    else:

        path = {}     # path[a]=b nghia la muon di toi dinh b phai qua a

        s\_open   = []

        s\_closed = []

        # đưa start vào open

        s\_open.append(start)

        path[start] = None

        while len(s\_open)>0:

            """ CÁC BẠN LÀM BÀI Ở ĐÂY """

            # ...

            u = s\_open.pop()

            for v in G[u]:

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

                    s\_open.append(v)

                    path[v] = u

                    if v == goal:

                        return path

            pass

    return path

path = DFS(ADJ, "A", "B")

pprint.pprint(path)

Kết quả

A screenshot of a computer

AI-generated content may be incorrect.