#Bản nhập tay có sẵn

#Bài 1

import numpy as np

from collections import deque

# 1. Sử dụng ma trận kề để biểu diễn đồ thị

# Ma trận kề

adj\_matrix = np.zeros((11, 11), dtype=int)

edges = [

    (1, 5), (1, 6), (2, 9), (2, 10), (3, 8), (3, 4),

    (4, 8), (4, 10), (5, 9), (6, 9), (6, 7), (8, 10)

]

for u, v in edges:

    adj\_matrix[u][v] = 1

    adj\_matrix[v][u] = 1

print("Ma trận kề:")

print(adj\_matrix)

# 2. Sử dụng danh sách cạnh để biểu diễn đồ thị

# Danh sách cạnh

adj\_list = {i: [] for i in range(1, 11)}

for u, v in edges:

    adj\_list[u].append(v)

    adj\_list[v].append(u)

print("\nDanh sách cạnh:")

for key, value in adj\_list.items():

    print(f"{key}: {value}")

# 3. Hiện thực lại giải thuật BFS – Duyệt đồ thị từ đỉnh 5

def bfs(start):

    visited = [False] \* 11

    queue = deque([start])

    visited[start] = True

    order = []

    while queue:

        vertex = queue.popleft()

        order.append(vertex)

        for neighbor in adj\_list[vertex]:

            if not visited[neighbor]:

                queue.append(neighbor)

                visited[neighbor] = True

    return order

print("\nBFS từ đỉnh 5:")

print(bfs(5))

# 4. Hiện thực lại giải thuật DFS – Duyệt đồ thị từ đỉnh 7

def dfs(start):

    visited = [False] \* 11

    stack = [start]

    order = []

    while stack:

        vertex = stack.pop()

        if not visited[vertex]:

            visited[vertex] = True

            order.append(vertex)

            for neighbor in adj\_list[vertex]:

                if not visited[neighbor]:

                    stack.append(neighbor)

    return order

print("\nDFS từ đỉnh 7:")

print(dfs(7))

#Bài 2

import numpy as np

from collections import deque

# Định nghĩa các cạnh của đồ thị

edges = [

    (1, 2), (1, 3), (2, 5), (3, 4), (4, 2), (4, 5), (4, 6), (4, 7), (5, 7), (6, 7)

]

# 5. Sử dụng ma trận kề để biểu diễn đồ thị

num\_vertices = 7  # Số đỉnh của đồ thị

adj\_matrix = np.zeros((num\_vertices + 1, num\_vertices + 1), dtype=int)

for u, v in edges:

    adj\_matrix[u][v] = 1

print("Ma trận kề:")

print(adj\_matrix)

# 6. Sử dụng danh sách cạnh để biểu diễn đồ thị

adj\_list = {i: [] for i in range(1, num\_vertices + 1)}

for u, v in edges:

    adj\_list[u].append(v)

print("\nDanh sách cạnh:")

for key, value in adj\_list.items():

    print(f"{key}: {value}")

# 7. Hiện thực lại giải thuật BFS – Duyệt đồ thị từ đỉnh 1

def bfs(start):

    visited = [False] \* (num\_vertices + 1)

    queue = deque([start])

    visited[start] = True

    order = []

    while queue:

        vertex = queue.popleft()

        order.append(vertex)

        for neighbor in adj\_list[vertex]:

            if not visited[neighbor]:

                queue.append(neighbor)

                visited[neighbor] = True

    return order

print("\nBFS từ đỉnh 1:")

print(bfs(1))

# 8. Hiện thực lại giải thuật DFS – Duyệt đồ thị từ đỉnh 4

def dfs(start):

    visited = [False] \* (num\_vertices + 1)

    stack = [start]

    order = []

    while stack:

        vertex = stack.pop()

        if not visited[vertex]:

            visited[vertex] = True

            order.append(vertex)

            for neighbor in adj\_list[vertex]:

                if not visited[neighbor]:

                    stack.append(neighbor)

    return order

print("\nDFS từ đỉnh 4:")

print(dfs(4))

#Bản nhập từ file txt

#Bài 1

import numpy as np

from collections import deque

# Đọc danh sách cạnh từ file txt

def read\_edges\_from\_file(file\_path):

    edges = []

    with open(file\_path, 'r') as file:

        for line in file:

            u, v = map(int, line.strip().split())

            edges.append((u, v))

    return edges

# 1. Sử dụng ma trận kề để biểu diễn đồ thị

def create\_adj\_matrix(edges, num\_vertices):

    adj\_matrix = np.zeros((num\_vertices + 1, num\_vertices + 1), dtype=int)

    for u, v in edges:

        adj\_matrix[u][v] = 1

        adj\_matrix[v][u] = 1

    return adj\_matrix

# 2. Sử dụng danh sách cạnh để biểu diễn đồ thị

def create\_adj\_list(edges, num\_vertices):

    adj\_list = {i: [] for i in range(1, num\_vertices + 1)}

    for u, v in edges:

        adj\_list[u].append(v)

        adj\_list[v].append(u)

    return adj\_list

# 3. Hiện thực lại giải thuật BFS – Duyệt đồ thị từ đỉnh 5

def bfs(adj\_list, start):

    visited = [False] \* (len(adj\_list) + 1)

    queue = deque([start])

    visited[start] = True

    order = []

    while queue:

        vertex = queue.popleft()

        order.append(vertex)

        for neighbor in adj\_list[vertex]:

            if not visited[neighbor]:

                queue.append(neighbor)

                visited[neighbor] = True

    return order

# 4. Hiện thực lại giải thuật DFS – Duyệt đồ thị từ đỉnh 7

def dfs(adj\_list, start):

    visited = [False] \* (len(adj\_list) + 1)

    stack = [start]

    order = []

    while stack:

        vertex = stack.pop()

        if not visited[vertex]:

            visited[vertex] = True

            order.append(vertex)

            for neighbor in adj\_list[vertex]:

                if not visited[neighbor]:

                    stack.append(neighbor)

    return order

# Định nghĩa file chứa danh sách cạnh

file\_path = 'edges.txt'

# Đọc danh sách cạnh từ file

edges = read\_edges\_from\_file(file\_path)

num\_vertices = 10

# Tạo ma trận kề

adj\_matrix = create\_adj\_matrix(edges, num\_vertices)

print("Ma trận kề:")

print(adj\_matrix)

# Tạo danh sách cạnh

adj\_list = create\_adj\_list(edges, num\_vertices)

print("\nDanh sách cạnh:")

for key, value in adj\_list.items():

    print(f"{key}: {value}")

# Duyệt đồ thị bằng BFS từ đỉnh 5

print("\nBFS từ đỉnh 5:")

print(bfs(adj\_list, 5))

# Duyệt đồ thị bằng DFS từ đỉnh 7

print("\nDFS từ đỉnh 7:")

print(dfs(adj\_list, 7))

#Bài 2

import numpy as np

from collections import deque

# Đọc danh sách cạnh từ file txt

def read\_edges\_from\_file(file\_path):

    edges = []

    with open(file\_path, 'r') as file:

        for line in file:

            u, v = map(int, line.strip().split())

            edges.append((u, v))

    return edges

# 5. Sử dụng ma trận kề để biểu diễn đồ thị

def create\_adj\_matrix(edges, num\_vertices):

    adj\_matrix = np.zeros((num\_vertices + 1, num\_vertices + 1), dtype=int)

    for u, v in edges:

        adj\_matrix[u][v] = 1

        adj\_matrix[v][u] = 1

    return adj\_matrix

# 6. Sử dụng danh sách cạnh để biểu diễn đồ thị

def create\_adj\_list(edges, num\_vertices):

    adj\_list = {i: [] for i in range(1, num\_vertices + 1)}

    for u, v in edges:

        adj\_list[u].append(v)

        adj\_list[v].append(u)

    return adj\_list

# 7. Hiện thực lại giải thuật BFS – Duyệt đồ thị từ đỉnh 1

def bfs(adj\_list, start):

    visited = [False] \* (len(adj\_list) + 1)

    queue = deque([start])

    visited[start] = True

    order = []

    while queue:

        vertex = queue.popleft()

        order.append(vertex)

        for neighbor in adj\_list[vertex]:

            if not visited[neighbor]:

                queue.append(neighbor)

                visited[neighbor] = True

    return order

# 8. Hiện thực lại giải thuật DFS – Duyệt đồ thị từ đỉnh 4

def dfs(adj\_list, start):

    visited = [False] \* (len(adj\_list) + 1)

    stack = [start]

    order = []

    while stack:

        vertex = stack.pop()

        if not visited[vertex]:

            visited[vertex] = True

            order.append(vertex)

            for neighbor in adj\_list[vertex]:

                if not visited[neighbor]:

                    stack.append(neighbor)

    return order

# Định nghĩa file chứa danh sách cạnh

file\_path = 'edges1.txt'

# Đọc danh sách cạnh từ file

edges = read\_edges\_from\_file(file\_path)

num\_vertices = 7

# Tạo ma trận kề

adj\_matrix = create\_adj\_matrix(edges, num\_vertices)

print("Ma trận kề:")

print(adj\_matrix)

# Tạo danh sách cạnh

adj\_list = create\_adj\_list(edges, num\_vertices)

print("\nDanh sách cạnh:")

for key, value in adj\_list.items():

    print(f"{key}: {value}")

# Duyệt đồ thị bằng BFS từ đỉnh 1

print("\nBFS từ đỉnh 1:")

print(bfs(adj\_list, 1))

# Duyệt đồ thị bằng DFS từ đỉnh 4

print("\nDFS từ đỉnh 4:")

print(dfs(adj\_list, 4))