

CHƯƠNG 3. ĐỒ THỊ

Handout Buổi 6: Giới thiệu đồ thị và các khái niệm cơ bản

Mục tiêu:

- Nhận biết và phân biệt các loại đồ thị cơ bản: đơn, đa, có hướng, đa có hướng.
- Hiểu các khái niệm nền tảng: đỉnh, cạnh, đỉnh liền kề, bậc, bậc vào/ra.
- Biểu diễn đồ thị bằng ma trận kề và ma trận liên thuộc.
- Tính và kiểm tra bậc đỉnh, áp dụng các định lý tổng bậc trong đồ thị.
- Vận dụng kiến thức để mô hình hóa các hệ thống thực tế bằng đồ thị.

PHẦN 1: LÝ THUYẾT

1. GIỚI THIỆU

Trong kỷ nguyên của dữ liệu và kết nối, khái niệm **đồ thị** không còn xa lạ với ngành **Công nghệ Thông tin**. Từ các mạng xã hội như **Facebook, Instagram, Zalo**, đến hệ thống định tuyến Internet, cơ sở dữ liệu quan hệ, hay thậm chí các mô hình trí tuệ nhân tạo – tất cả đều ẩn chứa bên dưới một cấu trúc chung: **một tập hợp các thực thể (đỉnh) và mối quan hệ giữa chúng (cạnh)**.

Đồ thị là một công cụ **mạnh mẽ và phổ quát**, không chỉ trong **Toán học rời rạc**, mà còn là **nền tảng của nhiều thuật toán và hệ thống** trong ngành CNTT: từ thuật toán tìm đường đi, cấu trúc mạng, đồ thị tri thức, đến công nghệ blockchain và phân tích mạng xã hội.

Để mở đầu buổi học, sinh viên được đặt vào một tình huống quen thuộc:

"Hãy tưởng tượng mỗi sinh viên là một điểm. Nếu hai người là bạn trên mạng xã hội, thì điều đó tương ứng với gì trong ngôn ngữ kỹ thuật?"

Câu hỏi này không chỉ kích hoạt tư duy mà còn tạo **cầu nối giữa trải nghiệm số hàng ngày và khái niệm trừu tượng** trong đồ thị học. Qua thảo luận lớp học, sinh viên nhanh chóng nhận ra rằng:

- Mỗi người là một **đỉnh (vertex)**.
- Mỗi mối quan hệ kết nối là một **cạnh (edge)**.
- Tổng số bạn bè có thể diễn đạt bằng **bậc của đỉnh (degree)**.
- Các quan hệ không đối xứng như “theo dõi” hay “gửi email một chiều” chính là các ví dụ sinh động của **đồ thị có hướng, đa đồ thị**, hoặc **cạnh khuếch**.

Từ đó, sinh viên nhận thấy rằng **đồ thị không chỉ là lý thuyết**, mà là mô hình của chính **các hệ thống công nghệ mà họ đang tương tác hàng ngày**.

Buổi học hôm nay sẽ cung cấp một nền tảng vững chắc về:

- Các loại đồ thị cơ bản: đơn, đa, có hướng...
- Cách biểu diễn đồ thị bằng ma trận.
- Khái niệm bậc của đỉnh – nền tảng của nhiều thuật toán quan trọng.

Đồ thị không chỉ là công cụ toán học – đó là ngôn ngữ để hiểu, mô hình hóa và điều khiển thế giới số.

2. ĐỊNH NGHĨA CƠ BẢN VỀ ĐỒ THỊ

3.1.1. Đơn đồ thị (Simple Graph)

Định nghĩa

Một **đơn đồ thị** là một cấu trúc toán học được ký hiệu là $G = (V, E)$, trong đó:

- V là tập hợp hữu hạn, không rỗng các **đỉnh** (*vertices*).
- E là tập hợp các **cạnh** (*edges*), trong đó mỗi phần tử của E là một **tập con gồm đúng hai phần tử phân biệt của V** .

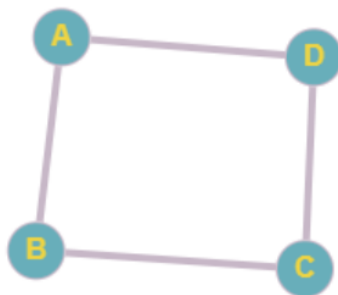
Hay nói cách khác, mỗi cạnh $e \in E$ là một cặp không thứ tự $\{u, v\}$ với $u, v \in V$ và $u \neq v$.

Tính chất của đơn đồ thị

- **Không có cạnh khuyết (loop):** Không tồn tại cạnh nối một đỉnh với chính nó, tức là không có cạnh $\{v, v\}$.
- **Không có đa cạnh:** Giữa hai đỉnh bất kỳ chỉ tồn tại tối đa một cạnh nối.
- **Không có hướng:** Cạnh $\{u, v\}$ không phân biệt thứ tự, tức là $\{u, v\} = \{v, u\}$.

Ví dụ

Xét đồ thị $G = (V, E)$ với:



- $V = \{A, B, C, D\}$
- $E = \{\{A, B\}, \{B, C\}, \{C, D\}, \{A, D\}\}$

Đồ thị này có 4 đỉnh và 4 cạnh. Mỗi cạnh nối hai đỉnh phân biệt và không có lặp, nên đây là một đơn đồ thị.

Biểu diễn trực quan

Trong hình học, đơn đồ thị thường được biểu diễn bằng sơ đồ gồm:

- Các điểm (hoặc vòng tròn) đại diện cho các đỉnh.
- Các đoạn thẳng nối hai điểm, đại diện cho các cạnh.

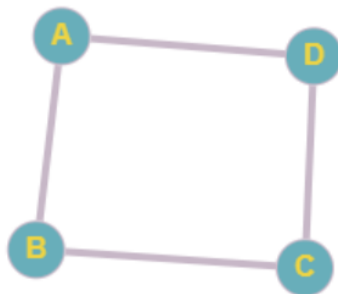
Biểu diễn trên máy tính

Các cách lưu trữ đơn đồ thị trong lập trình bao gồm:

- **Danh sách cạnh (edge list):** Mỗi cạnh là một cặp đỉnh.
- **Danh sách kề (adjacency list):** Với mỗi đỉnh, lưu danh sách các đỉnh liền kề.
- **Ma trận kề (adjacency matrix):** Ma trận nhị phân $n \times n$, với phần tử ở hàng i , cột j là 1 nếu tồn tại cạnh nối giữa đỉnh i và j , ngược lại là 0.

Ví dụ

Xét đồ thị $G = (V, E)$ với:



- $V = \{A, B, C, D\}$
- $E = \{\{A, B\}, \{B, C\}, \{C, D\}, \{A, D\}\}$
 1. Danh sách cạnh (Edge list): $\{\{A, B\}, \{B, C\}, \{C, D\}, \{A, D\}\}$
 2. Danh sách kề (Adjacency list): Vì đồ thị vô hướng, mỗi cạnh được tính cho cả hai đỉnh:
 - A: B, D
 - B: A, C
 - C: B, D

D: C, A

3. Ma trận kề (Adjacency matrix) (Thứ tự đỉnh: A, B, C, D)

	A	B	C	D
A	0	1	0	1
B	1	0	1	0
C	0	1	0	1
D	1	0	1	0

Vai trò trong ứng dụng

Đơn đồ thị là mô hình nền tảng trong nhiều bài toán thực tế và kỹ thuật, bao gồm:

- Mạng xã hội (người dùng kết bạn)
- Mạng giao thông (các nút giao thông và tuyến đường)
- Mạng máy tính (các thiết bị kết nối)

Lưu ý

- Không nhầm lẫn giữa đơn đồ thị và đa đồ thị (multigraph) – trong đó cho phép nhiều cạnh giữa hai đỉnh.
- Cần phân biệt đơn đồ thị với đồ thị có hướng – nơi mỗi cạnh mang thứ tự.

3.1.2. Đa đồ thị (Multigraph)

Định nghĩa

Một **đa đồ thị** là một đồ thị có thể có:

- **Nhiều hơn một cạnh** giữa cùng một cặp đỉnh, và/hoặc
- **Cạnh khuyên** (cạnh nối một đỉnh với chính nó).

Một đa đồ thị được ký hiệu là $G = (V, E)$, trong đó:

- V là tập hợp hữu hạn các đỉnh (*vertices*).
- E là tập hợp các cạnh (*edges*), mỗi cạnh là một **cặp (không nhất thiết phân biệt)** hai đỉnh trong V .
Khác với đơn đồ thị, trong đa đồ thị, hai đỉnh có thể được nối bởi **nhiều cạnh khác nhau**.

Tùy theo bối cảnh, mỗi cạnh $e \in E$ có thể:

- Được định nghĩa là một tập có thứ tự hoặc không có thứ tự (với đồ thị có hoặc không có hướng).
- Gắn kèm nhãn phân biệt (để nhận diện giữa các cạnh trùng nhau).

Tính chất

- Cho phép **đa cạnh**: tồn tại nhiều cạnh cùng nối một cặp đỉnh u và v .
- Có thể cho phép **cạnh khuyết** (loop): cạnh có dạng $\{v, v\}$, nối một đỉnh với chính nó.
- Các cạnh không cần phải là duy nhất trong tập E , tức là E có thể là một **đa tập** (multiset).

Ví dụ

Xét đồ thị $G = (V, E)$ với:

- $V = \{A, B\}$
- $E = \{\{A, B\}, \{A, B\}, \{A, A\}\}$



Trong ví dụ này:

- Có hai cạnh nối giữa A và B \Rightarrow đây là **đa cạnh**.
- Có một cạnh nối A với chính nó \Rightarrow **cạnh khuyết**.

Do đó, đây là một **đa đồ thị**.

Biểu diễn trực quan

Trong sơ đồ hình học, các cạnh trùng nhau có thể được vẽ bằng:

- Nhiều đường cong song song giữa hai đỉnh.
- Cạnh khuyết thường được vẽ như một vòng tròn nhỏ nối từ đỉnh quay về chính nó.

Biểu diễn trên máy tính

Vì đa đồ thị cho phép đa cạnh, nên cần cấu trúc dữ liệu linh hoạt hơn:

- **Danh sách cạnh**: cho phép lưu nhiều bản ghi có cùng cặp đỉnh.

- **Danh sách kề có lặp:** mỗi đỉnh liên kết với một danh sách có thể chứa nhiều bản sao của cùng một đỉnh kề.
- **Danh sách cạnh có nhãn:** mỗi cạnh có thể gắn một chỉ số phân biệt hoặc giá trị riêng.

Ứng dụng thực tế

- Mô hình hóa **giao thông:** nhiều tuyến xe buýt đi cùng một cặp trạm.
- Mạng viễn thông: nhiều kênh truyền tin giữa hai thiết bị.
- Biểu diễn hệ thống **luồng dữ liệu**, trong đó một đỉnh có thể nhận dữ liệu từ chính nó hoặc qua nhiều đường truyền khác nhau.

Phân biệt với đơn đồ thị

Tiêu chí	Đơn đồ thị	Đa đồ thị
Số cạnh giữa hai đỉnh	Tối đa 1	Có thể lớn hơn 1
Cạnh khuyết	Không cho phép	Có thể cho phép
Dạng tập cạnh E	Tập hợp	Đa tập (multiset)

Lưu ý: Khi thiết kế thuật toán xử lý đồ thị, cần xác định rõ đồ thị thuộc loại nào (đơn đồ thị, đa đồ thị, có hướng, không hướng...) vì điều này ảnh hưởng trực tiếp đến cách lưu trữ, thuật toán tìm kiếm, duyệt, phân tích đường đi, v.v.

3.1.3. Đồ thị có hướng (Directed Graph)

Định nghĩa

Một **đồ thị có hướng** (hay còn gọi là **digraph**) là một cấu trúc toán học ký hiệu là $G = (V, E)$

Trong đó:

- V là tập hợp hữu hạn các **đỉnh** (*vertices*).
- E là tập hợp các **cạnh có hướng** (*directed edges*, còn gọi là **cung**), mỗi phần tử của E là một **cặp có thứ tự** (u, v) , với $u, v \in V$.

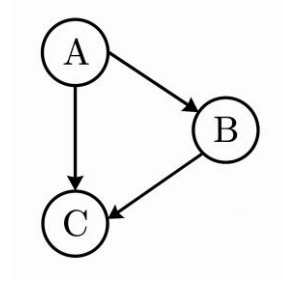
Cặp $(u, v) \in E$ biểu thị một cung đi từ đỉnh u đến đỉnh v , và u được gọi là **đỉnh gốc** (tail), còn v là **đỉnh đích** (head).

Tính chất

- Cạnh có thứ tự: $(u, v) \neq (v, u)$
- Có thể tồn tại hai cung đối chiều giữa cùng một cặp đỉnh.
- Có thể tồn tại **cạnh khuyết có hướng**: (v, v) – cung từ một đỉnh về chính nó.

- Không phân biệt giữa các cung cùng hướng nếu không cho phép đa cung (trong đồ thị có hướng đơn).

Ví dụ



Xét đồ thị có hướng $G = (V, E)$ với:

- $V = \{A, B, C\}$
- $E = \{(A, B), (B, C), (C, A)\}$

Đây là một đồ thị có hướng trong đó:

- Có cung từ A đến B
- Có cung từ B đến C
- Có cung từ C đến A

Mỗi cung mang thông tin về chiều di chuyển, không thể hoán vị như trong đồ thị vô hướng.

Biểu diễn trực quan

Đồ thị có hướng được biểu diễn bằng:

- Các mũi tên thay cho đoạn thẳng.
- Mỗi cung được vẽ từ đỉnh gốc đến đỉnh đích, có mũi tên chỉ chiều.

Biểu diễn trên máy tính

- **Danh sách cung:** Mỗi cung là một cặp có thứ tự (u, v) .
- **Danh sách kề có hướng:** Với mỗi đỉnh u , lưu danh sách các đỉnh v sao cho $(u, v) \in E$.
- **Ma trận kề:** Ma trận nhị phân $n \times n$, trong đó phần tử $A[i][j] = 1$ nếu có cung từ đỉnh i đến đỉnh j .

Khái niệm liên quan

- **Bậc vào (in-degree)** của một đỉnh: Số cung đi đến đỉnh đó.
- **Bậc ra (out-degree)** của một đỉnh: Số cung đi ra từ đỉnh đó.
- Tổng bậc vào = Tổng bậc ra = Số cung trong đồ thị.

Ứng dụng

Đồ thị có hướng được dùng rộng rãi trong các mô hình và hệ thống có chiều, bao gồm:

- **Luồng dữ liệu** trong mạng máy tính.
- **Biểu đồ phụ thuộc** trong lập trình và quản lý dự án.
- **Cây thư mục, biểu đồ quy trình.**
- **Liên kết giữa các trang web** (web graph), trong đó mỗi liên kết từ trang A đến trang B là một cung (A , B).

Phân biệt với đồ thị vô hướng

Tiêu chí	Đồ thị vô hướng	Đồ thị có hướng
Cạnh	Cặp không thứ tự {u , v}	Cặp có thứ tự (u , v)
Biểu diễn	Đoạn thẳng	Mũi tên
Quan hệ giữa hai đỉnh	Hai chiều (song phương)	Một chiều (đơn phương)
Phân tích bậc	Tổng bậc	Bậc vào, bậc ra

Lưu ý: Đồ thị có hướng là nền tảng cho nhiều bài toán quan trọng trong Khoa học Máy tính như tìm đường đi, phân tích chu trình, kiểm tra thứ tự phụ thuộc, lập lịch, và xác định thành phần liên thông mạnh.

3.1.5. Đa đồ thị có hướng (Directed Multigraph)

Định nghĩa

Một **đa đồ thị có hướng** (hay còn gọi là **directed multigraph**) là một mở rộng của đồ thị có hướng, trong đó cho phép:

- Tồn tại **nhiều cung cùng hướng** giữa một cặp đỉnh.
- Tồn tại **cạnh khuyết có hướng** (cung đi từ một đỉnh về chính nó).

Một đa đồ thị có hướng được mô tả dưới dạng:

$$G = (V , E)$$

Trong đó:

- V: Tập hữu hạn các đỉnh (*vertices*).
- E: Tập hợp các cung (*directed edges*), **mỗi cung là một cặp có thứ tự** $(u , v) \in V \times V$.
Tập E là một **đa tập** (*multiset*), nghĩa là có thể có nhiều bản sao của cùng một cung (u , v) .

Tính chất

- Cho phép **đa cung có hướng** giữa hai đỉnh.
- Cho phép **cung khuyết có hướng**: (v, v) .
- Phân biệt rõ giữa các cung cùng hướng: (u, v) xuất hiện nhiều lần với ý nghĩa riêng biệt (ví dụ: luồng dữ liệu song song).

Ví dụ

Xét đồ thị $G = (V, E)$ với:

- $V = \{P, Q\}$
- $E = \{(P, Q), (P, Q), (Q, P), (Q, Q)\}$



Đồ thị này bao gồm:

- Hai cung từ P đến Q (đa cung).
- Một cung từ Q đến P.
- Một cung khuyết tại Q.

Do đó, đây là một **đa đồ thị có hướng**.

Biểu diễn trực quan

- Mỗi cung được vẽ bằng **mũi tên**, có thể có nhiều mũi tên song song cùng chiều giữa hai đỉnh.
- Cung khuyết thường được vẽ như một **mũi tên cong quay về chính đỉnh**.

Biểu diễn trên máy tính

Để mô tả chính xác các cung lặp, cần sử dụng cấu trúc linh hoạt:

- **Danh sách cung có nhãn (labeled edge list)**: Mỗi cung có thể được gắn mã định danh hoặc trọng số riêng.
- **Danh sách kề lặp**: Cho phép nhiều lần xuất hiện của cùng một đỉnh trong danh sách kề.

- **Ma trận kề không nhị phân:** Thay vì chỉ lưu 0/1, phần tử $A[i][j]$ lưu số lượng cung từ i đến j .

Ứng dụng thực tế

Đồ thị có hướng rất hữu dụng trong các mô hình hệ thống có nhiều kênh tương tác cùng chiều:

- **Hệ thống mạng truyền thông:** Có thể có nhiều kênh truyền từ một thiết bị này sang thiết bị khác.
- **Hệ thống luồng dữ liệu:** Một bước xử lý có thể gửi nhiều kết quả theo cùng một hướng.
- **Giao thông đô thị:** Có thể có nhiều làn đường một chiều giữa hai giao lộ.

So sánh tổng quan

Tiêu chí	Đồ thị có hướng	Đồ thị vô hướng
Số lượng cung giữa hai đỉnh	Tối đa 1	Không giới hạn
Cạnh khuyết (cung về chính nó)	Có thể có	Có thể có
Phân biệt giữa các cung giống nhau	Không (nếu không gán nhãn)	Có (có thể gán nhãn, định danh)

Lưu ý:

Khi triển khai thuật toán trên đồ thị có hướng, cần xử lý đặc biệt để:

- Không bỏ sót các cung trùng lặp.
- Xác định đúng số lượng và chiều các luồng.
- Bảo toàn các thuộc tính riêng của từng cung (trọng số, nhãn, thời gian, v.v.).

3. BIỂU DIỄN ĐỒ THỊ BẰNG MA TRẬN

Trong Tin học và Toán học rời rạc, việc biểu diễn đồ thị dưới dạng ma trận là phương pháp phổ biến nhằm hỗ trợ lập trình, xử lý thuật toán và phân tích lý thuyết. Có hai dạng ma trận thông dụng:

- **Ma trận kề (Adjacency Matrix)**
- **Ma trận liên thuộc (Incidence Matrix)**

3.1. Ma trận kề (Adjacency Matrix)

Định nghĩa

Với một đồ thị $G = (V, E)$, trong đó $|V| = n$, ma trận kề là ma trận vuông kích thước $n \times n$, ký hiệu là $A = [a_{ij}]$, trong đó:

- Với đồ thị vô hướng:

$$a_{ij} = \begin{cases} 1 & \text{nếu có cạnh nối giữa đỉnh } v_i \text{ và } v_j \\ 0 & \text{ngược lại} \end{cases}$$

- Với **đồ thị có hướng**:

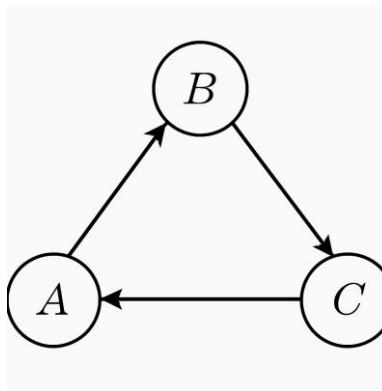
$$a_{ij} = \begin{cases} 1 & \text{nếu có cung đi từ } v_i \text{ đến } v_j \\ 0 & \text{ngược lại} \end{cases}$$

- Với **đa đồ thị**: $a_{ij}a$

là số lượng cạnh hoặc cung giữa v_i và v_j

Ví dụ

Đồ thị có hướng G với $V = \{A, B, C\}$, $E = \{(A,B), (B,C), (C,A)\}$



Ma trận kề:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Đặc điểm

- **Ưu điểm:**
 - Dễ dàng kiểm tra tồn tại cạnh giữa hai đỉnh: chỉ cần truy cập phần tử ma trận.
 - Phù hợp cho đồ thị **dày đặc** (dense graph).
- **Nhược điểm:**
 - Tốn bộ nhớ: yêu cầu $O(n^2)$ không gian lưu trữ, kể cả khi đồ thị thưa.

3.2. Ma trận liên thuộc (Incidence Matrix)

Định nghĩa

Cho đồ thị $G = (V, E)$ với $|V| = n$, $|E| = m$, ma trận liên thuộc là ma trận kích thước $n \times m$, ký hiệu $M = [m_{ij}]$, trong đó:

- Với đồ thị vô hướng:

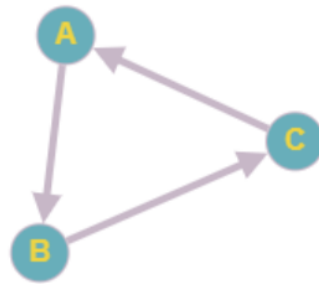
$$m_{ij} = \begin{cases} 1 & \text{nếu đỉnh } v_i \text{ là một đầu mút của cạnh } e_j \\ 0 & \text{ngược lại} \end{cases}$$

- Với đồ thị có hướng:

$$m_{ij} = \begin{cases} -1 & \text{nếu } v_i \text{ là đỉnh gốc của cung } e_j \\ 1 & \text{nếu } v_i \text{ là đỉnh đích của cung } e_j \\ 0 & \text{nếu } v_i \text{ không liên quan đến cung } e_j \end{cases}$$

Ví dụ

Đồ thị có hướng với $V = \{A, B, C\}$, $E = \{(A,B), (B,C), (C,A)\}$



Ma trận liên thuộc:

$$M = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

Mỗi cột tương ứng với một cung; mỗi hàng tương ứng với một đỉnh.

Đặc điểm

- Ưu điểm:
 - Giúp phân tích cấu trúc nội tại giữa đỉnh và cạnh.
 - Phù hợp cho xử lý các bài toán về **luồng**, **định tuyến**, hoặc **đồ thị thưa**.
- Nhược điểm:
 - Truy vấn thông tin giữa hai đỉnh không trực tiếp bằng chỉ số ma trận.

So sánh hai biểu diễn

Tiêu chí	Ma trận kề	Ma trận liên thuộc
Kích thước	$n \times n$	$n \times m$
Truy vấn cạnh	Trực tiếp ($O(1)$)	Phức tạp hơn

Tốt cho đồ thị	Dày đặc	Thưa
Lưu trữ thông tin cạnh	Có	Chi tiết hơn (hướng, vai trò)
Hỗ trợ trọng số	Có thể mở rộng dễ dàng	Có thể gắn trọng số vào cột

Lưu ý:

Tùy thuộc vào đặc điểm bài toán và kích thước đồ thị (thưa hay dày đặc), người học cần lựa chọn cách biểu diễn phù hợp để tối ưu hiệu năng tính toán và bộ nhớ.

4. BẬC CỦA ĐỈNH VÀ ĐỈNH LIÊN KÈ

4.2.1. Hai đỉnh liên kề (Adjacent Vertices)

Định nghĩa

Trong một đồ thị $G = (V, E)$, hai đỉnh u và v được gọi là **liên kề** (*adjacent*) nếu tồn tại một cạnh hoặc cung $e \in E$ sao cho:

- Với **đồ thị vô hướng**: $\{u, v\} \in E$
- Với **đồ thị có hướng**: $(u, v) \in E$ hoặc $(v, u) \in E$ (tùy theo ngữ cảnh)

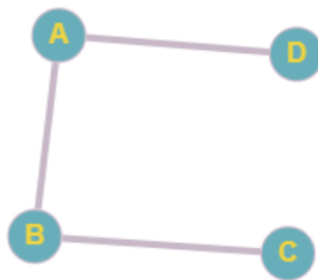
Cụ thể:

- Trong **đồ thị vô hướng**, tính liên kề là **hai chiều**: nếu u liên kề với v thì v cũng liên kề với u .
- Trong **đồ thị có hướng**, thường chỉ xét liên kề theo hướng đi của cung, nghĩa là u liên kề với v nếu tồn tại cung $(u, v) \in E$.

Ví dụ

Xét đồ thị vô hướng $G = (V, E)$ với:

- $V = \{A, B, C, D\}$
- $E = \{\{A, B\}, \{A, D\}, \{B, C\}\}$

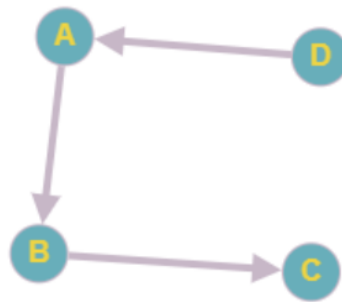


Ta có:

- A liên kề với B và D
- B liên kề với A và C
- C liên kề với B
- D liên kề với A

Nếu G là đồ thị có hướng với:

- $E = \{(A,B), (B,C), (D,A)\}$



Thì:

- A liên kề với B (vì có cung từ $A \rightarrow B$)
- B liên kề với C
- D liên kề với A
- Nhưng **B không liên kề với A**, vì không có cung từ $B \rightarrow A$

Biểu diễn

Trong danh sách kề của một đồ thị, các đỉnh liên kề với một đỉnh u được liệt kê trong danh sách liên kết (hoặc mảng) tương ứng với u.

Ứng dụng

Việc xác định đỉnh liên kề là nền tảng cho nhiều thuật toán đồ thị:

- Duyệt đồ thị theo chiều rộng (BFS) hoặc chiều sâu (DFS)
- Tìm đường đi ngắn nhất
- Phân tích liên thông, chu trình

Ghi chú mở rộng:

Trong lý thuyết đồ thị hiện đại, khái niệm "liền kề" đôi khi cũng được mở rộng cho cả **cạnh liền kề** (tức là hai cạnh cùng chia sẻ một đỉnh chung), nhưng trong phạm vi chương này, chúng ta chỉ đề cập đến **đỉnh liền kề**.

4.2.2. Bậc của đỉnh (Degree of a Vertex)

Định nghĩa chung

Bậc của một đỉnh trong một đồ thị là số lượng cạnh (hoặc cung) có liên quan đến đỉnh đó.

Tùy theo loại đồ thị, khái niệm bậc được định nghĩa cụ thể như sau:

A. Đối với đồ thị vô hướng

Cho đồ thị $G = (V, E)$ vô hướng.

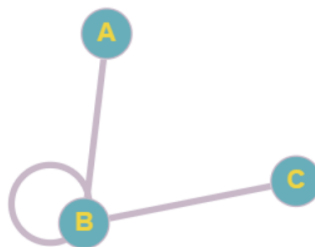
Bậc của một đỉnh $v \in V$, ký hiệu là $\deg(v)$, được định nghĩa là **số lượng cạnh nối tới** v .

- Mỗi cạnh nối hai đỉnh khác nhau đóng góp **1 đơn vị bậc** cho mỗi đỉnh.
- Mỗi **cạnh khuyết** $\{v, v\}$ (nếu có) đóng góp **2 đơn vị bậc** cho chính đỉnh v (vì nó kết nối v với chính nó ở cả hai đầu).

Ví dụ:

Với đồ thị vô hướng có:

- $V = \{A, B, C\}$
- $E = \{\{A, B\}, \{B, C\}, \{B, B\}\}$



Ta có:

- $\deg(A) = 1$
- $\deg(B) = 1(AB) + 1(BC) + 2(BB) = 4$
- $\deg(C) = 1$

B. Đối với đồ thị có hướng

Cho đồ thị có hướng $G = (V, E)$.

Với mỗi đỉnh $v \in V$, ta định nghĩa:

- **Bậc vào** (*in-degree*), ký hiệu $\deg^-(v)$
(v): là số cung có đỉnh đích là v .
- **Bậc ra** (*out-degree*), ký hiệu $\deg^+(v)$
(v): là số cung có đỉnh gốc là v .

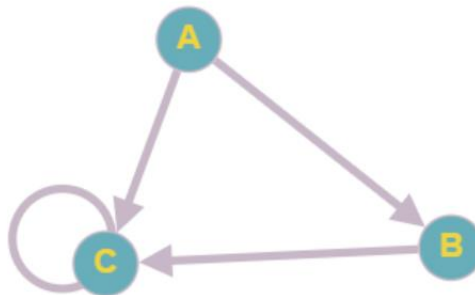
Tổng bậc của một đỉnh:

$$\deg(v) = \deg^-(v) + \deg^+(v)$$

Ví dụ:

Đồ thị có hướng G với:

- $V = \{A, B, C\}$
- $E = \{(A, B), (A, C), (B, C), (C, C)\}$



Ta tính:

- $\deg^+(A) = 2, \deg^-(A) = 0$
- $\deg^+(B) = 1, \deg^-(B) = 1$
- $\deg^+(C) = 1, \deg^-(C) = 3$ (bao gồm một cung từ B, một từ A, và một cung khuyết từ chính C)

Tổng bậc toàn đồ thị

- Với đồ thị vô hướng:

$$\sum_{v \in V} \deg(v) = 2|E|$$

- Với đồ thị có hướng:

$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$$

Đây là những hệ thức quan trọng để kiểm tra tính đúng đắn của một mô hình đồ thị.

Ứng dụng

- Phân tích luồng dữ liệu: số lượng thông tin đến và đi tại mỗi nút mạng.
- Tìm đỉnh đầu/cuối trong cây hoặc đồ thị có hướng.
- Hỗ trợ kiểm tra tính liên thông, chu trình, cân bằng của mạng.

Lưu ý:

- Đối với đồ thị có hướng, **không thể chỉ dùng** $\deg(v)$ mà phải xét riêng $\deg^-(v)$ và $\deg^+(v)$ vì chúng mang ý nghĩa khác nhau trong phân tích hệ thống.

4.2.3. Mệnh đề và hệ quả: Tổng bậc của đồ thị vô hướng

Mệnh đề

Trong một **đồ thị vô hướng**, tổng bậc của tất cả các đỉnh bằng **hai lần số cạnh** của đồ thị.

$$\sum_{v \in V} \deg(v) = 2 \cdot |E|$$

Chứng minh

Mỗi cạnh $e = \{u, v\} \in E$ nối hai đỉnh u và v , và do đó:

- Được tính một lần vào $\deg(u)$
- Và một lần vào $\deg(v)$

\Rightarrow Mỗi cạnh đóng góp **2 đơn vị** vào tổng bậc.

Vì có tổng cộng $|E|$ cạnh nên tổng bậc là $2|E|$.

Lưu ý: Điều này vẫn đúng ngay cả khi đồ thị chứa **cạnh khuyết**, vì mỗi cạnh khuyết $\{v, v\}$ được tính hai lần vào $\deg(v)$ (theo định nghĩa đã nêu ở phần 3.2.2).

Hệ quả 1: Số đỉnh có bậc lẻ là số chẵn

Từ mệnh đề:

- Tổng bậc là số **chẵn** (do bằng $2|E|$)

- Tổng của một tập hợp số nguyên là số chẵn \Rightarrow số lượng số **lẻ** trong tập hợp đó **phải chẵn**

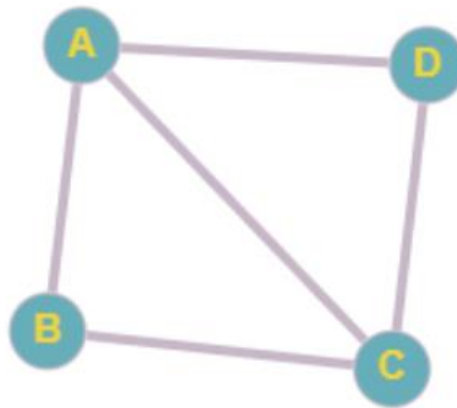
Do đó, trong bất kỳ đồ thị vô hướng nào, số lượng đỉnh có bậc **lẻ** luôn là **một số chẵn**.

Đây là một kết quả rất quan trọng, đặc biệt trong các bài toán liên quan đến đường đi Euler hoặc thiết kế mạng.

Ví dụ minh họa

Đồ thị GG với:

- $V = \{A, B, C, D\}$
- $E = \{\{A, B\}, \{B, C\}, \{C, D\}, \{D, A\}, \{A, C\}\}$



Tính bậc:

- $\deg(A) = 3$
- $\deg(B) = 2$
- $\deg(C) = 3$
- $\deg(D) = 2$

Tổng bậc: $3 + 2 + 3 + 2 = 10 = 2 \times 5 = 2|E|$ ☒

Số đỉnh có bậc lẻ = 2 (A và C) \Rightarrow là số chẵn ☒

Ứng dụng

- Kiểm tra tính hợp lệ của đồ thị được nhập vào chương trình.
- Hỗ trợ xác định khả năng tồn tại đường đi Euler (đường đi qua mỗi cạnh đúng một lần).

- Dùng trong thuật toán duyệt đồ thị và kiểm tra liên thông.

Lưu ý:

- Mệnh đề này **không áp dụng cho đồ thị có hướng**; trong trường hợp đó, tổng bậc vào = tổng bậc ra = số cung, nhưng tổng bậc không bằng $2|E|$.

5. TỔNG KẾT

Phần tổng kết nhằm giúp người học rà soát lại các khái niệm quan trọng đã học trong buổi học số 6, từ đó nắm vững kiến thức nền tảng về đồ thị để chuẩn bị tiếp cận các nội dung nâng cao trong các buổi học tiếp theo.

1. Phân biệt các loại đồ thị

Loại đồ thị	Ký hiệu cạnh	Cho phép đa cạnh	Cho phép cạnh khuyết	Cạnh có hướng
Đơn đồ thị	$\{u, v\}$	Không	Không	Không
Đa đồ thị	$\{u, v\}$	Có	Có (tùy chọn)	Không
Đồ thị có hướng	(u, v)	Không	Có (tùy chọn)	Có
Đa đồ thị có hướng	(u, v)	Có	Có	Có

Ghi nhớ:

- Ký hiệu $\{u, v\}$: không thứ tự \Rightarrow không có hướng.
- Ký hiệu (u, v) : có thứ tự \Rightarrow có hướng.

2. Cách biểu diễn đồ thị bằng ma trận

Ma trận	Kích thước	Đặc điểm chính	Phù hợp với
Ma trận kề	$n \times n$	Truy vấn nhanh, lưu toàn bộ liên kết	Đồ thị dày đặc
Ma trận liên thuộc	$n \times m$	Mỗi cột là một cạnh, phản ánh vai trò từng đỉnh	Đồ thị thưa, mô hình mạng

Ghi nhớ:

- Ma trận kề đơn giản hơn khi truy xuất dữ liệu giữa hai đỉnh.
- Ma trận liên thuộc thể hiện rõ cấu trúc đỉnh–cạnh, hỗ trợ phân tích luồng và cấu trúc tuyến tính.

3. Cách xác định bậc của đỉnh

- **Đồ thị vô hướng:**
 - Bậc của đỉnh = số cạnh nối đến đỉnh đó.
 - Tổng bậc của toàn bộ đỉnh = $2 \times$ số cạnh
- **Đồ thị có hướng:**
 - **Bậc vào:** số cung đi đến đỉnh.
 - **Bậc ra:** số cung đi ra từ đỉnh.
 - Tổng bậc vào = Tổng bậc ra = số cung.

Ghi nhớ:

- Số đỉnh có bậc lẻ trong đồ thị vô hướng luôn là số chẵn.
- Khi phân tích mạng có hướng, cần xét riêng bậc vào và bậc ra.

Gợi ý ôn tập

- Lấy ví dụ từ thực tế (mạng xã hội, hệ thống giao thông) và xác định:
 - Loại đồ thị phù hợp.
 - Cách biểu diễn bằng ma trận.
 - Bậc của từng đỉnh.
- Áp dụng lý thuyết đã học để kiểm tra tính hợp lệ của cấu trúc đồ thị: tổng bậc, sự liên kết, phân loại.

PHẦN 2: BÀI TẬP CÓ GIẢI

Bài 1. Phân loại đồ thị từ tập cạnh cho trước

Mục tiêu: Giúp sinh viên nhận biết loại đồ thị (đơn, đa, có hướng, đa có hướng) dựa trên danh sách cạnh.

Dữ liệu vào: Danh sách cạnh và tập đỉnh.

Yêu cầu: Phân loại đúng loại đồ thị.

Mở rộng: Viết chương trình tự động phân loại.

Mục tiêu

Giúp sinh viên:

- Nhận diện các loại đồ thị cơ bản (đơn đồ thị, đa đồ thị, đồ thị có hướng, đa đồ thị có hướng) từ danh sách các cạnh.

- Phân tích đặc điểm dữ liệu đầu vào để đưa ra phân loại chính xác.
- Hiểu rõ vai trò của thứ tự cặp đỉnh, số lần lặp lại cạnh, và cạnh khuyết.

Đề bài

Cho tập đỉnh V và danh sách các cạnh E của một đồ thị, hãy xác định loại đồ thị tương ứng là một trong bốn loại:

1. Đơn đồ thị (Simple Graph)
2. Đa đồ thị (Multigraph)
3. Đồ thị có hướng (Directed Graph)
4. Đa đồ thị có hướng (Directed Multigraph)

Dữ liệu ví dụ

Tập đỉnh:

$V = \{A, B, C, D\}$

Danh sách cạnh 1 (vô hướng): $E_1 = \{ \{A,B\}, \{B,C\}, \{C,A\}, \{B,A\} \}$

Danh sách cạnh 2 (có hướng): $E_2 = \{ (A,B), (B,C), (A,B), (C,C) \}$

Hướng dẫn phân tích

Để xác định loại đồ thị, cần kiểm tra:

1. **Cạnh có hướng hay không?**
→ Nếu là cặp có thứ tự (u, v) thì là đồ thị có hướng.
2. **Có cạnh lặp lại không?**
→ Nếu giữa cùng một cặp đỉnh có nhiều cạnh \Rightarrow là đa đồ thị.
3. **Có cạnh khuyết (loop) không?**
→ Nếu tồn tại $\{v, v\}$ hoặc (v, v) , cần xét kỹ loại đồ thị cho phép.

Lời giải chi tiết

Với danh sách cạnh E_1

- Các cạnh được biểu diễn bởi tập không thứ tự $\{u, v\} \Rightarrow$ đồ thị **vô hướng**.
- Có cả $\{A, B\}$ và $\{B, A\} \Rightarrow$ thực chất trùng nhau trong đồ thị vô hướng \Rightarrow **lặp cạnh**.
- \rightarrow Có **đa cạnh**.

✓ → Đây là **đa đồ thị (Multigraph)**

Với danh sách cạnh E_2

- Các cạnh được biểu diễn bởi cặp có thứ tự $(u, v) \Rightarrow$ đồ thị **có hướng**.
- Có hai lần cạnh $(A, B) \Rightarrow$ **đa cung có hướng**.
- Có cạnh $(C, C) \Rightarrow$ **cạnh khuyết có hướng**.

✓ → Đây là **đa đồ thị có hướng (Directed Multigraph)**

Phần mở rộng: Viết chương trình phân loại

Ý tưởng

- Phân tích định dạng cạnh: nếu là tuple $(u, v) \Rightarrow$ có hướng.
- Kiểm tra trùng lặp cạnh.
- Kiểm tra cạnh khuyết $u == v$.

Code minh họa (Python)

```
from collections import Counter

def classify_graph(edges, directed=False):
    edge_counter = Counter()
    has_loop = False
    is_multigraph = False

    for e in edges:
        u, v = e
        if u == v:
            has_loop = True
        key = (u, v) if directed else frozenset([u, v])
        edge_counter[key] += 1
        if edge_counter[key] > 1:
            is_multigraph = True
```

```

if directed:
    if is_multigraph:
        return "Đồ thị có hướng (Directed Multigraph)"
    else:
        return "Đồ thị có hướng (Directed Graph)"
else:
    if is_multigraph:
        return "Đồ thị (Multigraph)"
    else:
        return "Đơn đồ thị (Simple Graph)"

# Ví dụ sử dụng
edges1 = [ { "A", "B" }, { "B", "C" }, { "C", "A" }, { "B", "A" } ]
edges2 = [ ("A", "B"), ("B", "C"), ("A", "B"), ("C", "C") ]

# Chuyển tập hợp thành frozenset cho đồ thị vô hướng
edges1_processed = [frozenset(edge) for edge in edges1]

print("E1:", classify_graph(edges1_processed, directed=False)) # Multigraph
print("E2:", classify_graph(edges2, directed=True))           # Directed Multigraph

```

Kết luận

Bài tập này không chỉ giúp sinh viên nắm vững định nghĩa 4 loại đồ thị cơ bản, mà còn rèn luyện kỹ năng đọc và phân tích cấu trúc dữ liệu – một kỹ năng quan trọng trong lập trình thuật toán, xử lý dữ liệu mạng và xây dựng hệ thống.

Bài 2. Biểu diễn đồ thị bằng ma trận kề và ma trận liên thuộc

Mục tiêu: Rèn luyện kỹ năng xây dựng hai ma trận biểu diễn đồ thị.

Dữ liệu vào: Một đồ thị cụ thể (vẽ tay hoặc mô tả bằng danh sách cạnh).

Yêu cầu:

- Vẽ ma trận kề và ma trận liên thuộc.

- Phân tích ưu nhược điểm khi sử dụng mỗi loại.
Mở rộng: Viết chương trình nhập đồ thị và xuất ra hai ma trận.

Mục tiêu

- Rèn luyện khả năng xây dựng biểu diễn ma trận của đồ thị từ danh sách cạnh.
- So sánh và đánh giá hai phương pháp phổ biến: **ma trận kề** và **ma trận liên thuộc**.
- Chuẩn bị cho các bài toán xử lý đồ thị bằng thuật toán và lập trình.

Đề bài

Cho một đồ thị $G = (V, E)$, hãy:

1. Vẽ **ma trận kề (adjacency matrix)** và **ma trận liên thuộc (incidence matrix)**.
2. Phân tích ưu nhược điểm khi sử dụng mỗi loại.
3. (Mở rộng) Viết chương trình nhập danh sách cạnh và xuất ra hai ma trận trên.

Dữ liệu vào – Ví dụ cụ thể

Tập đỉnh:

$V = \{A, B, C, D\}$

Danh sách cạnh (đồ thị vô hướng):

$E = \{\{A,B\}, \{B,C\}, \{C,D\}, \{A,C\}\}$

Số đỉnh: 4 \rightarrow đánh số: A:0, B:1, C:2, D:3

Số cạnh: 4 \rightarrow đánh số: $e_1:AB, e_2:BC, e_3:CD, e_4:AC$

Lời giải

1. Ma trận kề (Adjacency Matrix)

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Hàng và cột lần lượt ứng với đỉnh A, B, C, D.

$A[i][j]=1$ nếu tồn tại cạnh giữa đỉnh i và j, 0 nếu không.

2. Ma trận liên thuộc (Incidence Matrix)

Hàng: các đỉnh $V = \{A, B, C, D\}$

Cột: các cạnh $E = \{e_1, e_2, e_3, e_4\}$

	e_1 (AB)	e_2 (BC)	e_3 (CD)	e_4 (AC)
A	1	0	0	1
B	1	1	0	0
C	0	1	1	1
D	0	0	1	0

Mỗi ô có giá trị 1 nếu đỉnh tham gia vào cạnh tương ứng, 0 nếu không.

3. Phân tích so sánh

Tiêu chí	Ma trận kề	Ma trận liên thuộc
Kích thước	$n \times n$	$n \times m$
Dễ truy xuất cạnh	Rất nhanh (truy cập $A[i][j]$)	Không trực tiếp
Lưu trữ đồ thị thưa	Không tối ưu	Tối ưu hơn
Dễ áp dụng thuật toán	Phù hợp với thuật toán DFS, BFS	Phù hợp cho bài toán luồng, tuyến tính
Thêm/xóa cạnh	Cập nhật đơn giản	Cần cập nhật lại toàn bộ ma trận

4. Mở rộng: Chương trình Python tạo 2 ma trận

```
def build_adjacency_matrix(vertices, edges):  
    n = len(vertices)  
    index_map = {v: i for i, v in enumerate(vertices)}  
    adj_matrix = [[0] * n for _ in range(n)]  
  
    for u, v in edges:  
        i, j = index_map[u], index_map[v]  
        adj_matrix[i][j] = 1  
        adj_matrix[j][i] = 1 # vì đồ thị vô hướng
```

```

    return adj_matrix

def build_incidence_matrix(vertices, edges):
    n = len(vertices)
    m = len(edges)
    index_map = {v: i for i, v in enumerate(vertices)}
    inc_matrix = [[0] * m for _ in range(n)]

    for idx, (u, v) in enumerate(edges):
        i, j = index_map[u], index_map[v]
        inc_matrix[i][idx] = 1
        inc_matrix[j][idx] = 1

    return inc_matrix

def print_matrix(matrix, row_labels, col_labels):
    print(f"{'>5}', *[f'{c:>5}' for c in col_labels])
    for i, row in enumerate(matrix):
        print(f"{row_labels[i]:>5}", *[f'{x:>5}' for x in row])

# Dữ liệu đầu vào
vertices = ['A', 'B', 'C', 'D']
edges = [('A', 'B'), ('B', 'C'), ('C', 'D'), ('A', 'C')]

# Xây dựng và in ma trận
adj = build_adjacency_matrix(vertices, edges)
inc = build_incidence_matrix(vertices, edges)

print("Ma trận kề:")
print_matrix(adj, vertices, vertices)

```

```
print("\nMa trận liên thuộc:")
print_matrix(inc, vertices, [f'e{i+1}' for i in range(len(edges))])
```

Kết luận

Bài tập này giúp sinh viên:

- Thành thạo hai cách biểu diễn đồ thị bằng ma trận.
- Nắm được ưu – nhược điểm để chọn cấu trúc phù hợp trong thực tiễn.
- Phát triển kỹ năng lập trình xử lý dữ liệu đồ thị – một kỹ năng quan trọng trong Khoa học máy tính.

Bài 3. Tính bậc của các đỉnh và kiểm tra hệ thức tổng bậc

Mục tiêu: Củng cố định nghĩa bậc của đỉnh, hệ thức tổng bậc.

Dữ liệu vào: Một đồ thị vô hướng và/hoặc có hướng.

Yêu cầu:

- Tính bậc vào, bậc ra (nếu có hướng) hoặc tổng bậc (nếu vô hướng).
- Kiểm tra mệnh đề tổng bậc $= 2 \times$ số cạnh.

Mở rộng: Viết chương trình kiểm tra tự động.

Mục tiêu

- Củng cố định nghĩa về **bậc của đỉnh, bậc vào, bậc ra**.
- Rèn luyện khả năng tính toán thủ công và kiểm chứng các mệnh đề toán học cơ bản trong đồ thị học.
- Ứng dụng các định lý lý thuyết trong việc kiểm tra tính đúng đắn của cấu trúc dữ liệu đồ thị.

Đề bài

Cho một đồ thị $G = (V, E)$, hãy thực hiện các yêu cầu sau:

1. Tính bậc của mỗi đỉnh:

- Nếu đồ thị **vô hướng**: tính tổng bậc của từng đỉnh.
- Nếu đồ thị **có hướng**: tính **bậc vào, bậc ra** và tổng bậc.

2. Kiểm tra mệnh đề:

- Đối với đồ thị vô hướng:

$$\sum_{v \in V} \deg(v) = 2|E|$$

- Đối với đồ thị có hướng:

$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$$

Ví dụ minh họa

A. Đồ thị vô hướng

Tập đỉnh: $V = \{A, B, C, D\}$


Cạnh: $E = \{\{A, B\}, \{A, C\}, \{B, C\}, \{C, D\}\}$

Tính bậc:

- $\deg(A) = 2$ (nối B, C)
- $\deg(B) = 2$ (nối A, C)
- $\deg(C) = 3$ (nối A, B, D)
- $\deg(D) = 1$ (nối C)

Tổng bậc: $2 + 2 + 3 + 1 = 8$

Số cạnh: $|E| = 4$

 $8 = 2 \times 4 \rightarrow$ Hệ thức đúng

B. Đồ thị có hướng


Tập đỉnh: $V = \{A, B, C\}$

Cung: $E = \{(A, B), (A, C), (C, B)\}$

Tính bậc vào/ra:

- A: $\deg^+(A) = 2, \deg^-(A) = 0$
- B: $\deg^+(B) = 0, \deg^-(B) = 2$
- C: $\deg^+(C) = 1, \deg^-(C) = 1$

Tổng:

- $\sum \deg^+ = 3, \sum \deg^- = 3, |E| = 3$
-  Hệ thức đúng

Mở rộng: Chương trình Python kiểm tra tự động

```
from collections import defaultdict
```

```

def analyze_undirected(vertices, edges):
    degree = defaultdict(int)
    for u, v in edges:
        degree[u] += 1
        degree[v] += 1
    total_degree = sum(degree.values())
    expected = 2 * len(edges)
    return degree, total_degree, expected, total_degree == expected

def analyze_directed(vertices, edges):
    indeg = defaultdict(int)
    outdeg = defaultdict(int)
    for u, v in edges:
        outdeg[u] += 1
        indeg[v] += 1
    sum_in = sum(indeg.values())
    sum_out = sum(outdeg.values())
    return indeg, outdeg, sum_in, sum_out, len(edges), sum_in == sum_out == len(edges)

# Ví dụ: đồ thị vô hướng
vertices1 = ['A', 'B', 'C', 'D']
edges1 = [('A', 'B'), ('A', 'C'), ('B', 'C'), ('C', 'D')]

deg, total, expect, ok = analyze_undirected(vertices1, edges1)
print("Đồ thị vô hướng:")
print("Bậc:", dict(deg))
print("Tổng bậc =", total, "==", expect, "→", "ĐÚNG" if ok else "SAI")

# Ví dụ: đồ thị có hướng
vertices2 = ['A', 'B', 'C']
edges2 = [('A', 'B'), ('A', 'C'), ('C', 'B')]

```

```
ind, outd, sum_in, sum_out, edge_count, ok2 = analyze_directed(vertices2, edges2)
print("\nĐồ thị có hướng:")
print("Bậc vào:", dict(ind))
print("Bậc ra:", dict(outd))
print("Tổng bậc vào =", sum_in, ", Tổng bậc ra =", sum_out, ", Số cung =", edge_count,
      "→", "ĐÚNG" if ok2 else "SAI")
```

Kết luận

Bài tập này giúp sinh viên:

- Hiểu và áp dụng khái niệm bậc đỉnh trong thực tế.
- Rèn kỹ năng kiểm chứng mệnh đề đồ thị.
- Phát triển năng lực phân tích cấu trúc dữ liệu đồ thị bằng lập trình.

Bài 4. Kiểm tra tính hợp lệ của một ma trận kề

Mục tiêu: Kiểm tra xem ma trận kề cho trước có tương ứng với một đồ thị hợp lệ không.

Dữ liệu vào: Một ma trận kề.

Yêu cầu:

- Xác định xem đây có phải là ma trận kề của một đồ thị vô hướng/hay có hướng hợp lệ không.
- Nếu không hợp lệ, chỉ ra lý do.

Mở rộng: Cài đặt hàm kiểm tra hợp lệ trong ngôn ngữ lập trình.

Mục tiêu

- Giúp sinh viên nhận biết các **đặc trưng hợp lệ** của ma trận kề cho từng loại đồ thị.
- Hiểu và phát hiện được các lỗi phổ biến như: không đối xứng, chứa giá trị bất hợp lệ, sai kích thước...
- Rèn luyện kỹ năng lập trình phân tích cấu trúc đồ thị đầu vào.

Đề bài

Cho một ma trận nhị phân A kích thước $n \times n$, hãy:

1. Kiểm tra xem đây có thể là ma trận kề của một **đồ thị vô hướng hợp lệ** hay không.

2. Kiểm tra xem đây có thể là ma trận kề của một **đồ thị có hướng hợp lệ** hay không.
3. Nếu không hợp lệ, **chỉ ra lý do cụ thể**.

Các điều kiện hợp lệ

1. Đối với đồ thị vô hướng:

- Ma trận **phải đối xứng**: $A[i][j]=A[j][i] \forall i, j$
- Giá trị hợp lệ: chỉ chứa 0 hoặc 1 (nếu là đơn đồ thị), hoặc số nguyên ≥ 0 (nếu là đa đồ thị).
- Giá trị đường chéo $A[i][i] = 0$ nếu **không cho phép cạnh khuyết**.

2. Đối với đồ thị có hướng:

- Ma trận **không cần đối xứng**
- Cho phép $A[i][j] \neq A[j][i]$
- Cho phép tồn tại $A[i][i] \neq 0$ nếu có cạnh khuyết có hướng

Ví dụ minh họa

Ma trận A1:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

✓ Hợp lệ: Ma trận đối xứng, không có cạnh khuyết \Rightarrow **Đồ thị vô hướng hợp lệ**

Ma trận A2:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

✓ Hợp lệ: Không đối xứng \Rightarrow **Đồ thị có hướng hợp lệ**

Ma trận A3:

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}$$

✓ Hợp lệ nếu là **đa đồ thị vô hướng**, vì có số lớn hơn 1 và ma trận đối xứng.

Ma trận A4:

$$\begin{bmatrix} 0 & 1 & \\ 1 & 0 & 1 \end{bmatrix}$$

✗ Không hợp lệ: không phải ma trận vuông

Mở rộng: Hàm kiểm tra hợp lệ (Python)

```
def is_square_matrix(matrix):
    return all(len(row) == len(matrix) for row in matrix)

def is_symmetric(matrix):
    n = len(matrix)
    for i in range(n):
        for j in range(n):
            if matrix[i][j] != matrix[j][i]:
                return False
    return True

def is_valid_adjacency_matrix(matrix, directed=False, allow_loops=False):
    n = len(matrix)
    if not is_square_matrix(matrix):
        return False, "Ma trận không vuông"

    for i in range(n):
        for j in range(n):
            val = matrix[i][j]
            if not isinstance(val, int) or val < 0:
                return False, f"Giá trị không hợp lệ tại ({i},{j}): {val}"
            if not allow_loops and i == j and val != 0:
                return False, f"Cạnh khuyên không được phép tại ({i},{i})"
```

```

if not directed and not is_symmetric(matrix):
    return False, "Ma trận không đối xứng (đồ thị vô hướng yêu cầu đối xứng)"

return True, "Hợp lệ"

# Ví dụ sử dụng
A1 = [
    [0, 1, 0],
    [1, 0, 1],
    [0, 1, 0]
]

A2 = [
    [0, 1, 0],
    [0, 0, 1],
    [1, 0, 0]
]

A3 = [
    [0, 1, 2],
    [1, 0, 1],
    [2, 1, 0]
]

for name, mat, directed in [("A1", A1, False), ("A2", A2, True), ("A3", A3, False)]:
    valid, message = is_valid_adjacency_matrix(mat, directed=directed)
    print(f"{name}: {message}")

```

Kết luận

Bài tập này giúp sinh viên:

- Phân biệt được các đặc điểm cấu trúc của ma trận kề hợp lệ.

- Nhận biết được loại đồ thị dựa trên đặc tính toán học của ma trận.
- Lập trình hiệu quả để xác minh dữ liệu đầu vào trong các ứng dụng mạng, đồ thị, hệ thống phức hợp.

Bài 5. Xây dựng và phân tích đồ thị bạn bè

Bài 5. Xây dựng và phân tích đồ thị bạn bè

Mục tiêu: Liên hệ kiến thức với ứng dụng thực tế.

Mô tả:

- Giả sử bạn có danh sách sinh viên trong lớp và mối quan hệ bạn bè giữa họ.
- Xây dựng đồ thị tương ứng.

Yêu cầu:

- Xác định loại đồ thị.
- Vẽ sơ đồ hoặc biểu diễn bằng ma trận.
- Tính bậc của mỗi sinh viên (đỉnh).

Mở rộng: Viết chương trình nhập tên và quan hệ, xuất ra thông tin phân tích.

Mục tiêu

- Vận dụng kiến thức lý thuyết đồ thị (đỉnh, cạnh, bậc, ma trận) vào tình huống thực tế.
- Giúp sinh viên hiểu rõ bản chất các mô hình mạng xã hội (social graphs).
- Rèn luyện kỹ năng chuyển đổi dữ liệu thực sang biểu diễn đồ thị toán học.

Mô tả bài toán

Giả sử bạn có danh sách sinh viên trong lớp, và thông tin về mối quan hệ bạn bè giữa họ (có thể từ khảo sát hoặc mô phỏng).

1. Mỗi sinh viên được biểu diễn bằng một **đỉnh**.
2. Mỗi mối quan hệ bạn bè được biểu diễn bằng một **cạnh**.

Yêu cầu

1. **Xác định loại đồ thị:**
 - Đồ thị **vô hướng** hay **có hướng**?
 - Có lặp cạnh không? Có cạnh khuyết không?
2. **Vẽ sơ đồ hoặc biểu diễn bằng:**

- **Ma trận kề**
- (Tùy chọn) ma trận liên thuộc

3. Tính bậc của mỗi đỉnh:

- Số bạn của mỗi sinh viên

Ví dụ minh họa

Danh sách sinh viên:

$V = ['\text{An}', '\text{Bình}', '\text{Châu}', '\text{Dũng}']$

Quan hệ bạn bè (song phương):

$E = \{('An', 'Bình'), ('An', 'Châu'), ('Châu', 'Dũng')\}$

Xác định loại đồ thị:

- Quan hệ bạn bè là **hai chiều** \Rightarrow Đồ thị **vô hướng**
- Không có cạnh trùng lặp hay cạnh khuyết \Rightarrow Đây là **đơn đồ thị**

Ma trận kề:

	An	Bình	Châu	Dũng
An	0	1	1	0
Bình	1	0	0	0
Châu	1	0	0	1
Dũng	0	0	1	0

Bậc của mỗi đỉnh:

- An: 2
- Bình: 1
- Châu: 2
- Dũng: 1

Mở rộng: Chương trình Python nhập dữ liệu và phân tích

```
from collections import defaultdict

def build_adjacency_matrix(vertices, edges):
    index_map = {v: i for i, v in enumerate(vertices)}
```

```

n = len(vertices)
adj = [[0]*n for _ in range(n)]

for u, v in edges:
    i, j = index_map[u], index_map[v]
    adj[i][j] = 1
    adj[j][i] = 1 # vì là đồ thị vô hướng

return adj, index_map

def compute_degrees(adj, vertices):
    return {vertices[i]: sum(row) for i, row in enumerate(adj)}

def print_matrix(matrix, row_labels, col_labels):
    print(f"{'':>7}", *[f"{c:>7}" for c in col_labels])
    for i, row in enumerate(matrix):
        print(f"{row_labels[i]:>7}", *[f"{x:>7}" for x in row])

# Dữ liệu đầu vào
vertices = ['An', 'Bình', 'Châu', 'Dũng']
edges = [('An', 'Bình'), ('An', 'Châu'), ('Châu', 'Dũng')]

# Xử lý
adj_matrix, idx_map = build_adjacency_matrix(vertices, edges)
degrees = compute_degrees(adj_matrix, vertices)

# Xuất kết quả
print("Ma trận kề:")
print_matrix(adj_matrix, vertices, vertices)

print("\nBậc của từng sinh viên:")

```

```
for sv, deg in degrees.items():  
    print(f"{sv}: {deg}")
```

Kết luận

Bài tập này giúp sinh viên:

- Hiểu rõ cách **chuyển mô hình thực tế sang biểu diễn đồ thị**.
- Vận dụng khái niệm bậc đỉnh, loại đồ thị, và cấu trúc ma trận trong phân tích mạng xã hội.
- Luyện tập lập trình xử lý dữ liệu đồ thị – nền tảng cho các ứng dụng phân tích mạng (social network analysis), hệ thống đề xuất, mô hình cộng tác...

PHẦN 3: TRẮC NGHIỆM

I. Định nghĩa cơ bản về đồ thị (Câu 1 – 6)

Câu 1

Trong một đơn đồ thị, mỗi cạnh nối giữa:

- A. Một đỉnh và chính nó
- B. Một cặp đỉnh bất kỳ, có thể lặp lại
- C. Một cặp đỉnh khác nhau, không có cạnh lặp
- D. Bất kỳ hai đỉnh, có hoặc không có hướng

Câu 2

Trong một đa đồ thị, điều nào sau đây là đúng?

- A. Mỗi cặp đỉnh chỉ được nối bởi một cạnh
- B. Không có cạnh khuyết
- C. Có thể có nhiều cạnh giữa hai đỉnh
- D. Mỗi đỉnh đều có bậc bằng nhau

Câu 3

Ký hiệu (u, v) thường được dùng cho:

- A. Cạnh trong đồ thị vô hướng
- B. Cung trong đồ thị có hướng
- C. Cạnh khuyết
- D. Bất kỳ đồ thị nào

Câu 4

Trong đồ thị có hướng, cạnh (u, v) đi từ:

- A. v đến u
- B. u đến v
- C. Đỉnh có bậc thấp hơn đến bậc cao hơn
- D. Cả hai chiều

Câu 5

Cạnh (v, v) là ví dụ của:

- A. Đa cạnh
- B. Cạnh khuyết
- C. Cạnh vô hướng
- D. Cạnh không hợp lệ

Câu 6

Đồ thị có cạnh lặp và cạnh có hướng được gọi là:

- A. Đơn đồ thị
- B. Đồ thị có hướng
- C. Đa đồ thị
- D. Đa đồ thị có hướng

II. Biểu diễn đồ thị bằng ma trận (Câu 7 – 12)

Câu 7

Ma trận kề của đồ thị vô hướng luôn có tính chất gì?

- A. Chứa toàn số 0 và 1
- B. Đối xứng
- C. Có tổng các hàng bằng nhau
- D. Không có đường chéo

Câu 8

Trong ma trận liên thuộc, cột tương ứng với:

- A. Một đỉnh
- B. Một cạnh
- C. Một cạnh khuyết
- D. Một thành phần liên thông

Câu 9

Đối với đồ thị có hướng, phần tử $M[i][j]$ trong ma trận liên thuộc bằng -1 khi:

- A. Đỉnh i là đỉnh đích của cạnh j
- B. Đỉnh i là đỉnh gốc của cạnh j
- C. Đỉnh j là liên kề với đỉnh i
- D. Không có liên kết

Câu 10

Ma trận kề phù hợp nhất khi nào?

- A. Khi đồ thị rất lớn và thưa
- B. Khi đồ thị dày đặc
- C. Khi có nhiều cung ngược
- D. Khi các đỉnh có trọng số

Câu 11

Đồ thị có 5 đỉnh và 6 cạnh. Ma trận kề có kích thước:

- A. 5×6
- B. 6×5
- C. 5×5
- D. 6×6

Câu 12

Điểm khác biệt lớn nhất giữa ma trận kề và ma trận liên thuộc là:

- A. Ma trận kề dễ đọc hơn
- B. Ma trận liên thuộc biểu diễn đỉnh tốt hơn
- C. Kích thước: một cái là $n \times n$, cái kia là $n \times m$
- D. Ma trận liên thuộc chỉ dùng cho đồ thị có hướng

III. Bậc của đỉnh và liên hệ tổng bậc (Câu 13 – 17)**Câu 13**

Trong đồ thị vô hướng, tổng bậc của tất cả các đỉnh bằng:

- A. Số cạnh
- B. Số đỉnh
- C. $2 \times$ số cạnh
- D. Không xác định

Câu 14

Trong đồ thị có hướng, tổng bậc vào của tất cả đỉnh bằng:

- A. Tổng số cạnh
- B. Số đỉnh chia 2
- C. Tổng bậc ra trừ 1
- D. 0

Câu 15

Trong đồ thị vô hướng có 4 đỉnh với bậc lần lượt là 2, 2, 3, 3. Đồ thị có bao nhiêu cạnh?

- A. 10
- B. 5
- C. 8
- D. 6

Câu 16

Nếu một đồ thị vô hướng có tổng bậc là 13, nhận xét nào sau đây đúng?

- A. Đồ thị không hợp lệ
- B. Có 13 cạnh
- C. Có 13 đỉnh
- D. Là đồ thị phân đôi

Câu 17

Trong đồ thị có hướng, nếu $\deg^-(v) = 0$ thì:

- A. v là đỉnh khuyết
- B. Không có cung đi ra từ v
- C. Không có cung đi vào v
- D. v không liên quan đến đồ thị

IV. Vận dụng thực tế và phân tích nâng cao (Câu 18 – 20)**Câu 18**

Một mạng xã hội là ví dụ điển hình của loại đồ thị nào?

- A. Đa đồ thị có hướng
- B. Đồ thị phân đôi
- C. Đơn đồ thị vô hướng
- D. Cây nhị phân

Câu 19

Bạn được cung cấp một ma trận kề không đối xứng. Điều nào sau đây chắc chắn đúng?

- A. Đây là đồ thị vô hướng
- B. Đây là đồ thị dày đặc
- C. Đây là đồ thị có hướng
- D. Đây là đồ thị không hợp lệ

Câu 20

Để biểu diễn hệ thống giao thông một chiều giữa các nút giao thông, bạn nên chọn:

- A. Ma trận liên thuộc của đồ thị vô hướng
- B. Ma trận kề của đơn đồ thị
- C. Danh sách cạnh
- D. Ma trận kề của đồ thị có hướng

PHẦN 4: BÀI TẬP LUYỆN TẬP

Phần I – Định nghĩa và phân loại đồ thị

Bài 1. Phân loại đồ thị từ tập cạnh

Đề bài:

Cho tập đỉnh $V = \{A, B, C, D\}$ và các tập cạnh sau, hãy xác định loại đồ thị tương ứng.

1. $E_1 = \{\{A, B\}, \{B, C\}, \{C, D\}, \{D, A\}\}$

2. $E_2 = \{\{A, B\}, \{A, B\}, \{B, B\}, \{C, D\}\}$

3. $E_3 = \{(A, B), (B, C), (C, A), (A, B)\}$

Hướng dẫn:

Xác định loại đồ thị dựa vào: có hướng hay không, có lặp hay không, có cạnh khuyết hay không.

Bài 2. Nhận diện đồ thị hợp lệ từ mô tả thực tế

Đề bài:

Cho các mô tả sau, xác định loại đồ thị phù hợp (vô hướng, có hướng, đa đồ thị...).

- 1. Mạng xã hội Facebook (kết bạn 2 chiều).
- 2. Hệ thống email (người gửi \rightarrow người nhận).
- 3. Mạng đường sắt: hai trạm có thể có nhiều tuyến chạy song song.
- 4. Sơ đồ call center: tổng đài gọi lại chính mình.

Hướng dẫn:

Phân tích chiều quan hệ, số lần liên kết và sự lặp lại để xác định đặc điểm đồ thị.

Phần II – Biểu diễn đồ thị bằng ma trận**Bài 3. Tạo ma trận kề từ danh sách cạnh****Đề bài:**

Cho đồ thị G với:

- $V = \{A, B, C, D\}$
- $E = \{\{A, B\}, \{B, C\}, \{C, D\}, \{A, D\}\}$

Hãy vẽ ma trận kề tương ứng.

Hướng dẫn:

Tạo ma trận 4×4 , đánh số đỉnh, điền 1 tại các cặp có cạnh.

Bài 4. Tạo ma trận liên thuộc từ danh sách cạnh**Đề bài:**

Dùng lại đồ thị ở Bài 3.

Hãy vẽ **ma trận liên thuộc** tương ứng.

Hướng dẫn:

Ma trận 4×4 (4 đỉnh \times 4 cạnh), mỗi cột tương ứng với 1 cạnh.

Bài 5. So sánh hai biểu diễn ma trận**Đề bài:**

Dựa trên hai ma trận của Bài 3 và Bài 4, hãy:

1. So sánh số dòng, cột của mỗi ma trận.
2. Nhận xét khi nào nên dùng ma trận nào.

Hướng dẫn:

Tập trung vào kích thước, tốc độ truy xuất và độ thưa/dày của đồ thị.

Phần III – Bậc của đỉnh và hệ thức tổng bậc**Bài 6. Tính bậc trong đồ thị vô hướng****Đề bài:**

Cho đồ thị vô hướng với:

- $V = \{A, B, C, D\}$

- $E = \{\{A,B\}, \{A,C\}, \{B,C\}, \{C,D\}\}$

1. Tính bậc của mỗi đỉnh.
2. Kiểm tra tổng bậc có bằng $2 \times |E|$ không?

Hướng dẫn:

Mỗi cạnh đóng góp 1 vào bậc của 2 đỉnh. Tổng bậc = $2 \times$ số cạnh.

Bài 7. Tính bậc vào – bậc ra trong đồ thị có hướng

Đề bài:

Cho đồ thị có hướng với:

- $V = \{A, B, C, D\}$
- $E = \{(A,B), (B,C), (C,A), (C,D)\}$

1. Tính $\deg^+(v)$, $\deg^-(v)$ cho từng đỉnh.
2. Kiểm tra tổng bậc vào = tổng bậc ra = số cung?

Hướng dẫn:

Duyệt từng cạnh, tăng số đếm vào/ra tương ứng.

Bài 8. Phát hiện lỗi trong ma trận kề

Đề bài:

Cho ma trận kề:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

1. Xác định loại đồ thị.
2. Có gì bất thường không?

Hướng dẫn:

Kiểm tra đối xứng, tính liên thông, các dòng có toàn số 0?

Phần IV – Ứng dụng thực tế

Bài 9. Biểu diễn đồ thị bạn bè lớp học

Đề bài:

Cho danh sách sinh viên: ['An', 'Bình', 'Châu', 'Dũng']

Quan hệ bạn bè: ('An', 'Bình'), ('An', 'Châu'), ('Châu', 'Dũng')

1. Xác định loại đồ thị.

2. Vẽ ma trận kề.
3. Tính bậc của từng sinh viên.

Hướng dẫn:

Xem bạn bè là quan hệ 2 chiều, mỗi cặp là 1 cạnh trong đồ thị vô hướng.

Bài 10. Viết mô tả từ ma trận kề cho trước

Đề bài:

Cho ma trận kề:

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

1. Xác định danh sách cạnh.
2. Tính bậc từng đỉnh.
3. Loại đồ thị?

Hướng dẫn:

Xác định các cặp i, j có giá trị 1, tránh lặp nếu vô hướng.

PHẦN 5: BÀI TẬP DỰ ÁN



Dự án 1: Phân tích mạng bạn bè trong lớp học

Mục tiêu:

- Vận dụng kiến thức về đồ thị vô hướng.
- Biểu diễn và phân tích quan hệ xã hội.
- Xây dựng mô hình đơn giản mô phỏng mạng xã hội.

Đề bài:

Bạn được giao xây dựng và phân tích **mạng bạn bè trong một lớp học**.

Dữ liệu đầu vào:

- Danh sách tên sinh viên trong lớp.
- Danh sách các cặp bạn bè (mỗi quan hệ là 2 chiều).

Yêu cầu:

1. Biểu diễn mạng bạn bè dưới dạng **đồ thị vô hướng**.

2. Xây dựng:

- **Ma trận kề**
- **Danh sách kề**

3. Tính:

- **Bậc của mỗi sinh viên**
- **Sinh viên có nhiều bạn nhất**

4. Vẽ sơ đồ mạng bằng công cụ (tùy chọn: vẽ tay, draw.io, hay thư viện Python như networkx).

Hướng dẫn triển khai:

Bước 1: Nhập dữ liệu

- Nhập danh sách tên sinh viên (tập đỉnh).
- Nhập danh sách các cặp bạn bè (tập cạnh vô hướng).

Bước 2: Xây dựng biểu diễn

- Tạo **ma trận kề** $n \times n$: điền 1 nếu 2 sinh viên là bạn.
- Tạo **danh sách kề**: với mỗi sinh viên, liệt kê các bạn của họ.

Bước 3: Tính bậc

- Với ma trận kề: bậc = tổng các phần tử trong hàng.
- Với danh sách kề: bậc = độ dài danh sách.

Bước 4: Phân tích

- Tìm sinh viên có bậc cao nhất.
- Nhận xét về cấu trúc lớp học: ai là “trung tâm mạng bạn bè”?

(Tùy chọn) Bước 5: Hiển thị đồ thị

- Dùng thư viện **networkx** + **matplotlib** để vẽ:

```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_edges_from([('An', 'Bình'), ('An', 'Châu'), ('Châu', 'Dũng')])
nx.draw(G, with_labels=True)
```

Dự án 2: Kiểm tra hợp lệ ma trận kề và sinh tự động biểu diễn

Mục tiêu:

- Hiểu sâu về các tiêu chuẩn của ma trận kề hợp lệ.
- Rèn kỹ năng lập trình và kiểm tra dữ liệu đồ thị.
- Thực hành sinh và kiểm tra đồ thị qua nhập liệu.

Đề bài:

Viết một chương trình cho phép:

1. Nhập một **ma trận kề** bất kỳ.
2. Kiểm tra ma trận đó có phải:
 - Ma trận kề hợp lệ của đồ thị vô hướng hay không?
 - Ma trận kề hợp lệ của đồ thị có hướng hay không?
3. Nếu hợp lệ:
 - Xuất ra danh sách cạnh.
 - Tính bậc của từng đỉnh.

Hướng dẫn triển khai:

Bước 1: Nhập ma trận

- Cho phép nhập từ bàn phím hoặc đọc từ file.
- Kiểm tra số dòng = số cột (ma trận vuông).

Bước 2: Kiểm tra tính chất

- Kiểm tra **đối xứng** \Rightarrow nếu có \Rightarrow có thể là đồ thị vô hướng.
- Kiểm tra các giá trị (chỉ 0 hoặc số nguyên ≥ 0).
- Nếu ma trận không đối xứng \Rightarrow xét như đồ thị có hướng.

Bước 3: Tạo danh sách cạnh

- Với mỗi $A[i][j] > 0$:
 - Nếu đồ thị vô hướng \Rightarrow chỉ xét khi $i < j$
 - Nếu có hướng \Rightarrow xét mọi i, j

Bước 4: Tính bậc

- Vô hướng: bậc = tổng hàng (hoặc cột)
- Có hướng:
 - Bậc vào = tổng cột
 - Bậc ra = tổng hàng

(Tuỳ chọn) Bước 5: Giao diện

- Tạo menu CLI cho người dùng chọn: nhập, kiểm tra, hiển thị danh sách cạnh, tính bậc...

Gợi ý triển khai code Python đơn giản

```
def is_symmetric(matrix):  
    return all(matrix[i][j] == matrix[j][i] for i in range(len(matrix)) for j in  
range(len(matrix)))  
  
def degree_info(matrix, directed=False):  
    n = len(matrix)  
    deg = { }  
    if directed:  
        for i in range(n):  
            out_deg = sum(matrix[i])  
            in_deg = sum(matrix[j][i] for j in range(n))  
            deg[i] = (in_deg, out_deg)  
    else:  
        for i in range(n):  
            deg[i] = sum(matrix[i])  
    return deg
```