Handout Đường đi Euler và Đồ thị Euler

🎯 I. MỤC TIÊU BUỔI HỌC

Sau buổi học, sinh viên sẽ:

- Nắm vững định nghĩa đường đi Euler và đồ thị Euler.
- Hiểu và vận dụng các định lý liên quan đến đồ thị Euler.
- Giải được bài toán Người phát thư Trung Hoa (Chinese Postman Problem).
- Rèn luyện tư duy thuật toán trên đồ thị thông qua các bài tập ứng dụng thực tế.

PHẦN 1: LÝ THUYẾT

4.1.1 Định nghĩa chi tiết

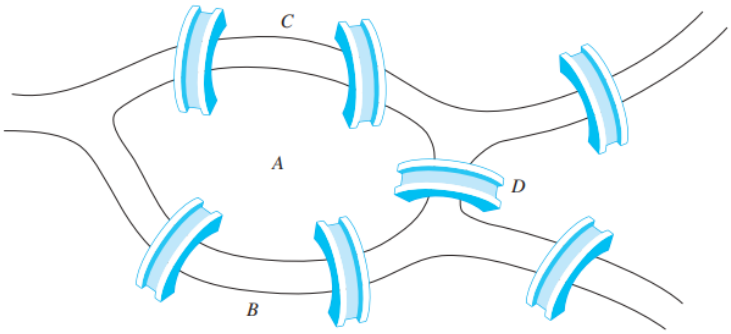


FIGURE 1 The seven bridges of Königsberg.

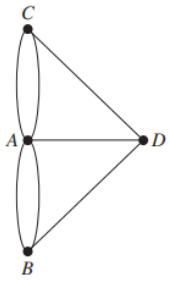


FIGURE 2 Multigraph model of the town of Königsberg.

Bài toán bảy cây cầu ở Königsberg xuất phát từ một thị trấn cổ ở Phổ (nay là Kaliningrad, Nga), nơi có bốn khu vực được ngăn cách bởi các nhánh sông Pregel và được nối với nhau bằng **bảy cây cầu**. Người dân từng thắc mắc liệu có thể **đi dạo qua tất cả các cầu một lần duy nhất** và **trở lại điểm xuất phát** hay không.

Nhà toán học Thụy Sĩ **Leonhard Euler** đã giải quyết bài toán này vào năm **1736**, đánh dấu sự khởi đầu của **lý thuyết đồ thị**. Ông mô hình hóa các khu vực thành **đỉnh** và các cây cầu thành **cạnh** trong một **đa đồ thị**, rồi chứng minh rằng **không tồn tại chu trình Euler** (tức là không thể đi qua mỗi cầu một lần mà quay lại điểm xuất phát).

Trong lĩnh vực lý thuyết đồ thị, một trong những chủ đề nền tảng và có nhiều ứng dụng thực tiễn chính là **đường đi Euler** và **đồ thị Euler**. Đây là kiến thức cốt lõi cho việc mô hình hóa các bài toán như: hành trình giao hàng, kiểm tra hệ thống mạng, hoặc lập lộ trình cho robot tự động. Trước khi đi sâu vào định lý hay bài toán, ta cần hiểu rõ các khái niệm cơ bản sau:

🔷 1. Đường đi Euler (Eulerian Trail / Path)

Một **đường đi Euler** trong đồ thị là một **dãy các cạnh liên tiếp nhau** sao cho:

- **Mỗi cạnh chỉ được đi đúng một lần.**
- **Các đỉnh có thể được đi qua nhiều lần.**

💡 *Lưu ý:* Đường đi Euler **không yêu cầu** phải quay lại điểm xuất phát.

📌 **Ký hiệu:** Gọi đồ thị là $G = (V,E)$. Một đường đi Euler là một dãy cạnh $e_1, e_2, \ldots, e_k \in E$ sao cho không có cạnh nào lặp lại.

📘 Ví dụ:

Giả sử ta có đồ thị đơn giản gồm 4 đỉnh A, B, C, D và các cạnh:

$$E = \{AB, BC, CD, DA, AC\}$$

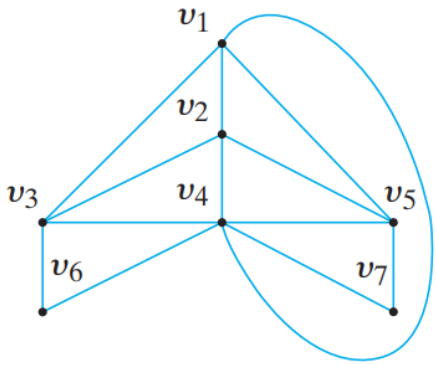
Đường đi: $A \to B \to C \to D \to A \to C$
→ Là đường đi Euler (đi qua mỗi cạnh đúng một lần).

🔷 2. Chu trình Euler (Eulerian Circuit / Cycle)

Một **chu trình Euler** là một đường đi Euler có **điểm đầu trùng với điểm cuối**.

Hay nói cách khác:

- Là một vòng khép kín đi qua **tất cả các cạnh** đúng một lần.
- **Bắt đầu và kết thúc tại cùng một đỉnh.**



Cho G là đồ thị trong trên. Cần xác minh rằng G có một chu trình Euler. Hãy tìm một chu trình Euler cho G.

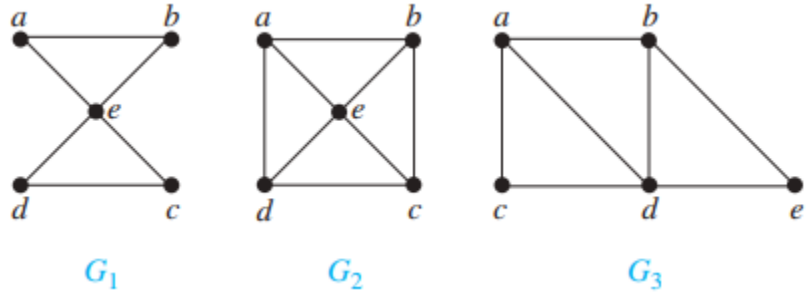
Với các đỉnh: $y_1, y_2, y_3, y_4, y_5, y_6, y_7$

LỜI GIẢI: Ta nhận thấy rằng đồ thị G là liên thông và có:

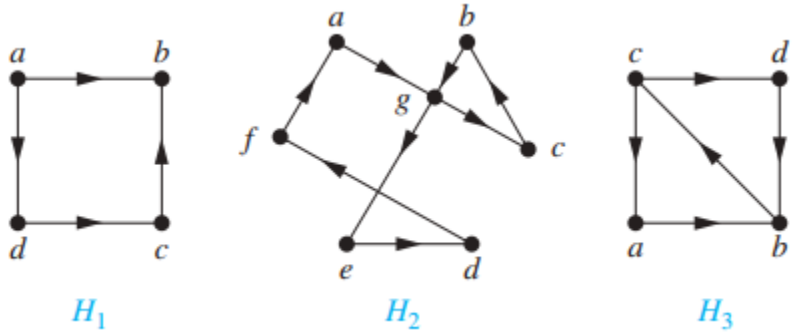
- $\delta(v_1) = \delta(v_2)$ =δ(v3)=δ(v5)=4,
- $\delta(v_4) = 6$
- $\delta(v_6) = \delta(v_7)$ =2.

Vì bậc của mọi đỉnh đều là số chẵn, theo Định lý 8.2.18, đồ thị G có một chu trình Euler.

Dựa trên quan sát, ta tìm được một chu trình Euler là:
$(v_6,v_4,v_7,v_5,v_1,v_3,v_4,v_1,v_2,v_5,v_4,v_2,v_3,v_6)$



Trong số các đồ thị vô hướng ở Hình 3, đồ thị nào có chu trình Euler? Trong số các đồ thị không có chu trình Euler, đồ thị nào có đường đi Euler?
Lời giải: Đồ thị **G1** có một chu trình Euler, ví dụ: a, e, c, d, e, b, a. Cả hai đồ thị **G2** và **G3** đều không có chu trình Euler. Tuy nhiên, **G3** có một đường đi Euler, cụ thể: a, c, d, e, b, d, a, b. **G2** không có đường đi Euler.



Trong số các đồ thị có hướng ở Hình 4, đồ thị nào có chu trình Euler? Trong số các đồ thị không có chu trình Euler, đồ thị nào có đường đi Euler?

Lời giải thêm: Đồ thị **H2** có một chu trình Euler, ví dụ: a, g, c, b, g, e, d, f, a. Cả hai đồ thị **H1** và **H3** đều không có chu trình Euler. Tuy nhiên, **H3** có một đường đi Euler, cụ thể: c, a, b, c, d, b. Còn **H1** thì không có đường đi Euler.

🔷 3. Đồ thị Euler (Eulerian Graph)

Một **đồ thị Euler** là một đồ thị **có chứa ít nhất một chu trình Euler.**

📌 Tức là: Nếu bạn có thể bắt đầu từ một đỉnh bất kỳ và **vòng lại chính nó sau khi đi qua mỗi cạnh một lần**, thì đồ thị đó là Euler.

📘 **Điều kiện cần và đủ** để một đồ thị **vô hướng liên thông** là đồ thị Euler:

- **Tất cả các đỉnh có bậc chẵn.**

💬 **Ghi chú chuyên môn:**
Điều kiện này rất dễ kiểm tra bằng lập trình: đếm bậc từng đỉnh và kiểm tra tính chẵn.

📎 Tóm tắt các khái niệm chính

Khái niệm	Đặc điểm chính	Yêu cầu điểm đầu – cuối
Đường đi Euler	Đi qua tất cả cạnh đúng một lần	Không yêu cầu trùng nhau
Chu trình Euler	Là đường đi Euler có đầu = cuối	Phải trùng nhau
Đồ thị Euler	Đồ thị có ít nhất một chu trình Euler	Mọi đỉnh đều bậc chẵn

📦 Ứng dụng thực tiễn

- **Lập lịch trình kiểm tra dây điện, cáp mạng, hay hệ thống ống dẫn.**
- **Tối ưu hóa hành trình giao hàng: đi qua mọi tuyến đường một lần duy nhất.**
- **Thiết kế hệ thống robot tuần tra không lặp đường.**

4.1.2. Một số định lý liên quan đến đường đi và đồ thị Euler

Đây là những định lý nền tảng, được chứng minh từ thế kỷ 18 bởi Leonhard Euler – người được xem là "cha đẻ của lý thuyết đồ thị". Những định lý này không chỉ mang tính lý thuyết mà còn là kim chỉ nam cho việc **kiểm tra và xác định tính Euler của đồ thị một cách nhanh chóng**.

🔷 Định lý 1:

Một đồ thị vô hướng liên thông có chu trình Euler ⇔ tất cả các đỉnh đều có bậc chẵn.

🧠 **Giải thích:**

- **Bậc của đỉnh** là số cạnh nối đến đỉnh đó.
- Khi bạn đi vào một đỉnh qua một cạnh, bạn phải rời khỏi đỉnh đó bằng một cạnh khác. Do đó, nếu số cạnh đến đỉnh là lẻ, bạn không thể thoát ra hết – không thể hình thành chu trình khép kín.

📌 **Điều kiện liên thông:** Đồ thị phải không bị chia cắt – tức là **mọi đỉnh đều có thể đi tới từ bất kỳ đỉnh nào khác**.

🧪 **Ví dụ minh họa:**
Đồ thị hình vuông với 4 đỉnh A, B, C, D và các cạnh AB, BC, CD, DA
→ Mỗi đỉnh có bậc 2 (chẵn) → có chu trình Euler.

🔷 **Định lý 2:**

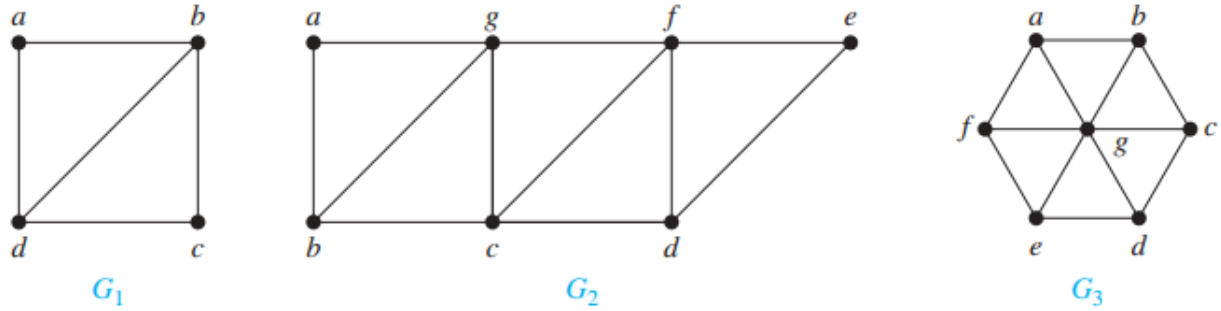
Một đồ thị vô hướng liên thông có đường đi Euler nhưng không có chu trình Euler ⇔ có đúng 2 đỉnh có bậc lẻ.

🧠 **Giải thích:**

- Hai đỉnh bậc lẻ sẽ là **điểm bắt đầu** và **kết thúc** của đường đi Euler.
- Tất cả các đỉnh còn lại phải có bậc chẵn để bảo đảm việc "đi vào – đi ra" như giải thích ở trên.

📘 **Lưu ý:** Nếu có **nhiều hơn hai đỉnh bậc lẻ**, đồ thị sẽ **không có đường đi Euler**.

🧪 **Ví dụ minh họa:**
Đồ thị với đỉnh A, B, C, D và các cạnh: AB, BC, CD, DA, AC
→ Bậc A = 3, bậc C = 3 → 2 đỉnh lẻ → có đường đi Euler, không có chu trình Euler.



Những đồ thị nào trong Hình trên có đường đi Euler?
Lời giải:
Đồ thị **G1** có đúng hai đỉnh bậc lẻ, đó là **b** và **d**. Do đó, nó có một đường đi Euler với điểm đầu và điểm cuối là **b** và **d**. Một ví dụ về đường đi Euler là: **d, a, b, c, d, b**.

Tương tự, đồ thị **G2** cũng có đúng hai đỉnh bậc lẻ, là **b** và **d**. Vì vậy, nó cũng có một đường đi Euler với điểm đầu và điểm cuối là **b** và **d**. Một ví dụ là: **b, a, g, f, e, d, c, g, b, c, f, d**.
Đồ thị **G3** không có đường đi Euler vì nó có đến **sáu đỉnh bậc lẻ**.

🔷 **Định lý 3 (đối với đồ thị có hướng):**

Đồ thị có hướng có chu trình Euler ⇔ với mỗi đỉnh, bậc vào = bậc ra.

🧠 **Giải thích:**

- Ở đồ thị có hướng, "đi vào" và "đi ra" là hai khái niệm khác biệt.
- Để hình thành chu trình khép kín đi qua mọi cung, bạn phải đi ra khỏi một đỉnh **mỗi lần bạn đi vào nó**, tức là:

$$\forall v \in V, \text{in-degree}(v) = \text{out-degree}(v)$$

🧪 **Ví dụ minh họa:**
Đồ thị có hướng với 3 đỉnh A, B, C và cung AB, BC, CA
→ Mỗi đỉnh có 1 cung vào và 1 cung ra → có chu trình Euler.

✅ **Tổng kết**

Đồ thị loại nào?	**Điều kiện để có đường/chu trình Euler**
Vô hướng – có chu trình	Liên thông & mọi đỉnh có **bậc chẵn**
Vô hướng – có đường đi	Liên thông & có đúng **2 đỉnh bậc lẻ**
Có hướng – chu trình Euler	Đồ thị liên thông & mỗi đỉnh: **bậc vào = bậc ra**

💡 **Mẹo thực hành:**

- **Bước 1:** Kiểm tra tính liên thông.
- **Bước 2:** Đếm bậc của các đỉnh (vô hướng) hoặc bậc vào/ra (có hướng).
- **Bước 3:** Áp dụng định lý để kết luận nhanh mà không cần vẽ đường đi.

4.1.3. Bổ đề và hệ quả

🔷 **Bổ đề về tính liên thông**

Bổ đề:
*Để một đồ thị có đường đi Euler hoặc chu trình Euler thì đồ thị đó **phải liên thông** (trừ các đỉnh cô lập).*

🧠 **Giải thích:**

- **Liên thông** nghĩa là **mọi cặp đỉnh trong đồ thị đều có đường đi nối với nhau** (trong đồ thị vô hướng).
- Với đồ thị có hướng, điều kiện tương ứng là **mạnh liên thông** hoặc ít nhất **liên thông yếu + điều kiện bậc phù hợp**.

📌 Nếu đồ thị không liên thông, sẽ tồn tại **một hoặc nhiều cụm** đỉnh riêng biệt, không có cách nào nối tất cả các cạnh bằng một đường đi duy nhất → **không thể có đường đi Euler**.

🧪 **Ví dụ minh họa:**

- Đồ thị gồm 2 tam giác tách biệt → không liên thông → không có đường đi Euler.

🔷 **Các hệ quả quan trọng từ định lý Euler**

Hệ quả 1:

Nếu một đồ thị vô hướng liên thông có **hơn hai đỉnh bậc lẻ**, thì **không tồn tại** đường đi Euler.

📌 Hệ quả này giúp ta **loại trừ nhanh** một đồ thị khi kiểm tra tính Euler.

Hệ quả 2:

Một đồ thị Euler (có chu trình Euler) thì **mọi đỉnh đều phải có bậc chẵn**
→ Điều này dẫn đến tính chất thú vị sau:

👉 **Tổng bậc của toàn bộ các đỉnh trong đồ thị Euler luôn là số chẵn.**

Vì mỗi cạnh đóng góp 2 đơn vị bậc (mỗi đầu một lần), nên tổng bậc luôn là số chẵn – **dù đồ thị có phải Euler hay không**.

🧠 Do đó, nếu bạn tính tổng bậc của các đỉnh mà ra số lẻ, chắc chắn bạn đã **tính sai đâu đó!**

Hệ quả 3:

Trong đồ thị có hướng, nếu tồn tại chu trình Euler thì:

$$\forall v \in V, \deg^{in}(v) = \deg^{out}(v)$$

📘 Hệ quả này thường được áp dụng trong mô hình **mạng máy tính**, nơi bạn cần lập trình một tiến trình đi qua tất cả liên kết đúng một lần và quay về gốc.

✍️ Tóm lược nhanh

Nội dung	**Ý nghĩa**
Bổ đề về liên thông	Đồ thị phải liên thông mới có đường đi/chu trình Euler
Hệ quả 1	>2 đỉnh bậc lẻ → không có đường đi Euler
Hệ quả 2	Tổng bậc các đỉnh luôn chẵn (áp dụng kiểm tra nhanh)
Hệ quả 3 (đồ thị có hướng)	Mỗi đỉnh: bậc vào = bậc ra → mới có chu trình Euler

🛠 Ứng dụng thực tiễn

- **Tối ưu hóa mạng giao thông đô thị:** Sử dụng điều kiện liên thông và bậc để thiết kế tuyến đường kiểm tra/tuần tra.
- **Phát hiện lỗi trong cấu trúc mạng:** Nếu mạng không liên thông, hệ thống định tuyến theo Euler sẽ thất bại.

Thuật toán Fleury và thuật toán Hierholzer

🚩 1. Thuật toán Fleury

Mục đích: Tìm chu trình Euler trong đồ thị vô hướng liên thông có chu trình Euler.
Nguyên tắc hoạt động:

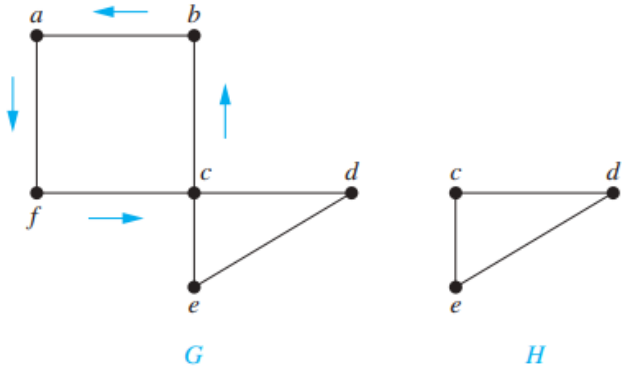
- Bắt đầu từ một đỉnh bất kỳ (đỉnh có bậc chẵn nếu là chu trình Euler).
- Ở mỗi bước, chọn cạnh **không phải là cầu (bridge)** nếu có thể.
- Xóa cạnh đã đi và tiếp tục đến khi không còn cạnh nào.

Các bước chính:

1. Chọn đỉnh xuất phát uuu.
2. Trong khi còn cạnh nối với uuu:
    - Chọn cạnh (u,v)(u, v)(u,v) sao cho:
        - Nếu chỉ có một cạnh → chọn luôn.
        - Nếu có nhiều cạnh → tránh chọn cầu.
    - In cạnh (u,v)(u, v)(u,v), xóa nó khỏi đồ thị.
    - Đặt u:=vu := vu:=v.
3. Lặp đến khi hết cạnh.

Ưu điểm: Dễ hiểu, dễ cài đặt.
Nhược điểm: Chậm với đồ thị lớn do phải kiểm tra cầu mỗi bước.

⚡ 2. Thuật toán Hierholzer



Một chu trình Euler được tạo ra nếu tất cả các cạnh đã được sử dụng. Nếu chưa, ta xét đồ thị con **H** tạo thành bằng cách xóa các cạnh đã dùng và các đỉnh không còn liên kết với cạnh nào. Vì **G** liên thông nên **H** có ít nhất một đỉnh chung với chu trình đã xóa (ví dụ, đỉnh **c**). Mọi đỉnh trong **H** đều có bậc chẵn. Từ một đỉnh như vậy, xây dựng một đường đi đơn trong **H** (kết thúc tại chính nó), rồi nối vào chu trình ban đầu tại điểm chung. Lặp lại quá trình này cho đến khi mọi cạnh được dùng. Kết quả là một chu trình Euler. Cách xây dựng này chứng minh rằng nếu một đa đồ thị liên thông có tất cả các đỉnh đều bậc chẵn thì nó có chu trình Euler.

Mục đích: Tìm chu trình Euler trong đồ thị (vô hướng hoặc có hướng).
Nguyên tắc hoạt động:

- Bắt đầu từ một đỉnh bất kỳ, đi theo các cạnh chưa sử dụng để tạo thành chu trình.
- Nếu trong chu trình còn đỉnh có cạnh chưa đi, từ đó tạo chu trình phụ và "nối" vào chu trình chính.
- Tiếp tục đến khi dùng hết các cạnh.

Các bước chính:

1. Tạo chu trình Euler ban đầu bắt đầu từ đỉnh uuu.
2. Trong khi còn đỉnh nào trong chu trình có cạnh chưa sử dụng:
    - Tạo chu trình phụ từ đỉnh đó.
    - Gắn chu trình phụ vào chu trình chính.
3. Lặp cho đến khi sử dụng hết các cạnh.
4. Kết quả là chu trình Euler hoàn chỉnh.

Ưu điểm: Hiệu quả cao, độ phức tạp **O(E)**.
Nhược điểm: Cần lưu cấu trúc dữ liệu tốt để quản lý việc "nối chu trình".

🔍 So sánh nhanh

Tiêu chí	Fleury	Hierholzer
Độ phức tạp	Chậm hơn ($O(E^2)$)	Nhanh (O(E))
Dễ cài đặt	Có	Cần khéo hơn

Kiểm tra cầu	Có (tốn thời gian)	Không cần
Thích hợp	Bài tập nhỏ	Dữ liệu lớn

4.1.4. Bài toán Người phát thư Trung Hoa (Chinese Postman Problem – CPP)

🔷 1. Mô tả bài toán

Bài toán:

> Một người phát thư cần đi qua tất cả các con đường trong một khu phố **ít nhất một lần**, sau đó quay về điểm xuất phát (tức là một **chu trình**).
> **Hỏi:** Làm thế nào để tìm được **lộ trình ngắn nhất** thỏa yêu cầu đó?

Mô hình hóa:

- Mỗi **ngã tư** là một **đỉnh**.
- Mỗi **con đường** là một **cạnh**.
- Bài toán trở thành: **Tìm chu trình ngắn nhất đi qua tất cả các cạnh trong một đồ thị có trọng số.**

💬 Đây là **bài toán tối ưu hóa trên đồ thị** – mở rộng của khái niệm chu trình Euler.

🔍 2. Ý nghĩa và ứng dụng thực tế

🗂️

Lĩnh vực thực tế	Ứng dụng bài toán CPP
📦 Giao hàng, chuyển phát	Tối ưu hóa tuyến đường giao hàng
🚓 Tuần tra, kiểm tra hệ thống	Robot/dịch vụ bảo trì đi qua toàn bộ mạng lưới
🔌 Mạng điện, viễn thông	Tối thiểu hóa chi phí bảo trì hệ thống dây cáp
🧹 Quét dọn đường phố, máy hút bụi	Lập lịch đường đi để bao phủ mọi khu vực

💡 **Khác với bài toán Người du lịch (TSP)** – đi qua **các đỉnh**, CPP đi qua **tất cả các cạnh**.

⚙️ 3. Thuật toán giải bài toán Người phát thư Trung Hoa

📌 Ý tưởng chính:

Nếu đồ thị đã là **đồ thị Euler** → giải xong (vì có chu trình Euler).

Nếu không, ta **thêm cạnh nhân tạo** (lặp lại đường) sao cho đồ thị trở thành Euler → giải chu trình Euler → suy ra đường đi ban đầu.

🧠 **Các bước thuật toán:**

Bước 1: Kiểm tra bậc các đỉnh

- Tìm tất cả các đỉnh có **bậc lẻ**.
- Gọi số đỉnh bậc lẻ là $2k$. (Luôn chẵn vì tổng bậc là chẵn.)

Bước 2: Ghép cặp các đỉnh bậc lẻ

- Tạo **tập các cặp ghép** giữa các đỉnh bậc lẻ sao cho:
  - Tổng chi phí (độ dài/khối lượng cạnh) được **thêm vào là nhỏ nhất**.
  - Dùng **thuật toán ghép cặp ngắn nhất** (Minimum Weight Perfect Matching).

Bước 3: Thêm các đoạn đường tương ứng với cặp ghép

- Mỗi cặp được nối bằng **đường đi ngắn nhất** giữa chúng (dùng Dijkstra hoặc Floyd-Warshall).
- Thêm các cạnh này vào đồ thị (có thể trùng cạnh đã tồn tại).

Bước 4: Tìm chu trình Euler trong đồ thị đã chỉnh sửa

- Lúc này, đồ thị có bậc chẵn tại mọi đỉnh → tồn tại chu trình Euler.
- Áp dụng **thuật toán Fleury hoặc Hierholzer** để tìm chu trình.

Bước 5: Chuyển chu trình đã tìm thành đường đi gốc

- Đường đi tìm được bao gồm cả các cạnh trùng – chính là lộ trình tối ưu cần tìm.

📘 Ví dụ minh họa:

Đồ thị có 4 đỉnh A, B, C, D và các cạnh:
AB (2), BC (2), CD (2), DA (2), AC (1)

- Bậc A = 3, C = 3 (lẻ) → cần thêm đường ngắn nhất giữa A và C (chính là AC, trọng số 1).
- Sau khi thêm → tất cả bậc chẵn → tìm chu trình Euler → suy ra hành trình phát thư tối ưu.

🔧 Công cụ tính toán hỗ trợ:

- Sử dụng **thuật toán Dijkstra** hoặc **Floyd-Warshall** để tìm đường ngắn nhất giữa các cặp đỉnh bậc lẻ.
- Dùng **thuật toán Hierholzer** để tìm chu trình Euler hiệu quả hơn Fleury.

✅ Tổng kết:

Bài toán phát thư	**Kết nối lý thuyết Euler**
Đi qua mọi cạnh	Mở rộng khái niệm chu trình Euler
Thêm cạnh lặp	Để biến đồ thị ban đầu thành đồ thị Euler
Tối ưu hóa lộ trình	Giảm tổng chiều dài/khoảng cách của hành trình

PHẦN 2: BÀI TẬP CÓ GIẢI

🧠 BÀI 1: Kiểm tra một đồ thị có phải là đồ thị Euler không?

📜 Đề bài:

Cho đồ thị vô hướng G có các đỉnh:

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

và các cạnh:

$$E = \{AB,BC,CD,DA,AC\}$$

Hãy kiểm tra xem G có phải là **đồ thị Euler** hay không.

🔍 **Phân tích:**

- G là **đồ thị vô hướng**.
- Một đồ thị vô hướng **có chu trình Euler** nếu:
  - G **liên thông**.
  - **Mọi đỉnh có bậc chẵn**.

📋 **Tính bậc các đỉnh:**

Đỉnh	Cạnh liên quan	Bậc
A	AB, DA, AC	3
B	AB, BC	2
C	BC, CD, AC	3
D	CD, DA	2

→ Có 2 đỉnh (A và C) có **bậc lẻ** ⇒ Không phải là đồ thị Euler.

✅ **Kết luận:**

G không phải là đồ thị Euler, vì có đỉnh bậc lẻ.

💻 **Code Python kiểm tra:**

```
from collections import defaultdict

def is_eulerian(graph):
    degrees = defaultdict(int)

    for u, v in graph:
        degrees[u] += 1
        degrees[v] += 1

    odd_vertices = [v for v in degrees if degrees[v] % 2 == 1]

    if len(odd_vertices) == 0:
        return "Đồ thị Euler (có chu trình Euler)"
    elif len(odd_vertices) == 2:
        return "Không phải đồ thị Euler, nhưng có đường đi Euler"
    else:
        return "Không có đường đi/chu trình Euler"
```

```
# Danh sách cạnh
edges = [('A', 'B'), ('B', 'C'), ('C', 'D'), ('D', 'A'), ('A', 'C')]
print(is_eulerian(edges))
```

🧠 BÀI 2: Tìm đường đi Euler trong một đồ thị đơn giản

📜 Đề bài:

Cho đồ thị:

$$V=\{1,2,3,4\},\ E=\{(1,2),(2,3),(3,4),(4,1),(1,3)\}$$

Tìm một **đường đi Euler** nếu tồn tại.

🔍 Phân tích:

Tính bậc các đỉnh:

Đỉnh	Bậc
1	3
2	2
3	3
4	2

→ Có đúng **2 đỉnh bậc lẻ** → **Có đường đi Euler**, bắt đầu tại 1 và kết thúc tại 3 (hoặc ngược lại).

✅ Kết luận:

Tồn tại **đường đi Euler**, ví dụ:
1 → 2 → 3 → 4 → 1 → 3

💻 Code Python: Tìm đường đi Euler (thuật toán Fleury đơn giản)

```
from collections import defaultdict

def build_graph(edges):
    graph = defaultdict(list)
    for u, v in edges:
        graph[u].append(v)
        graph[v].append(u)
    return graph
```

```
def remove_edge(graph, u, v):
    graph[u].remove(v)
    graph[v].remove(u)

def dfs_count(graph, v, visited):
    visited[v] = True
    count = 1
    for neighbor in graph[v]:
        if not visited.get(neighbor, False):
            count += dfs_count(graph, neighbor, visited)
    return count

def is_valid_next_edge(graph, u, v):
    if len(graph[u]) == 1:
        return True
    visited = {}
    count1 = dfs_count(graph, u, visited)
    remove_edge(graph, u, v)
    visited = {}
    count2 = dfs_count(graph, u, visited)
    graph[u].append(v)
    graph[v].append(u)
    return count1 == count2

def fleury(graph, u):
    path = [u]
    while True:
        for v in graph[u]:
            if is_valid_next_edge(graph, u, v):
                path.append(v)
                remove_edge(graph, u, v)
                u = v
                break
        else:
            break
    return path

edges = [(1,2), (2,3), (3,4), (4,1), (1,3)]
graph = build_graph(edges)

start = 1  # Một trong hai đỉnh bậc lẻ
euler_path = fleury(graph, start)
```

```
print("Đường đi Euler:", euler_path)
```

🧠 BÀI 3: Tìm chu trình Euler bằng thuật toán Fleury

📜 Đề bài:

Cho đồ thị:

$$V=\{1,2,3,4\}, E=\{(1,2),(2,3),(3,4),(4,1)\}$$

Tìm **chu trình Euler** bắt đầu từ đỉnh 1.

🔍 Phân tích:

- Tính bậc các đỉnh: mỗi đỉnh có bậc 2 → tất cả chẵn.
- Đồ thị liên thông. ⇒ **Là đồ thị Euler** → tồn tại **chu trình Euler**.

✅ Lời giải:

Áp dụng **thuật toán Fleury**:

- Bắt đầu tại 1.
- Lần lượt chọn các cạnh **không phải cầu** nếu còn lựa chọn.

→ Một chu trình Euler:
1 → 2 → 3 → 4 → 1

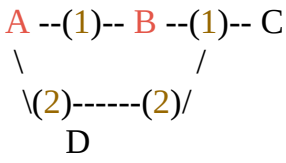
💻 Code Python:

```
# Dùng lại code Fleury ở Bài 2

edges = [(1,2), (2,3), (3,4), (4,1)]
graph = build_graph(edges)

start = 1
euler_cycle = fleury(graph, start)
print("Chu trình Euler:", euler_cycle)
```

🧠 BÀI 4: Giải bài toán Người phát thư Trung Hoa (CPP) trên đồ thị nhỏ

📜 Đề bài:

Đồ thị có trọng số:



Cạnh: AB=1, BC=1, CD=2, DA=2
→ Có 4 đỉnh: A, B, C, D

Hãy tìm hành trình phát thư ngắn nhất quay về điểm xuất phát (CPP).

🔍 Phân tích:

- Tính bậc:
  - A = 2, B = 2, C = 2, D = 2 → tất cả bậc chẵn → đã là **đồ thị Euler**.

✅ Kết luận:

- Vì là đồ thị Euler → không cần thêm cạnh → dùng chu trình Euler là hành trình phát thư.
- Chu trình ví dụ: A → B → C → D → A

💻 Code Python:

```
edges = [('A','B',1), ('B','C',1), ('C','D',2), ('D','A',2)]

# Xây dựng graph có trọng số
from collections import defaultdict

def build_weighted_graph(edges):
    graph = defaultdict(list)
    for u, v, w in edges:
        graph[u].append((v, w))
        graph[v].append((u, w))
    return graph

graph = build_weighted_graph(edges)
# Vì là Euler nên dùng thuật toán Fleury như bài trước với bỏ trọng số
graph_simple = [(u,v) for u,v,w in edges]
graph_unweighted = build_graph(graph_simple)
path = fleury(graph_unweighted, 'A')
```

print("Lộ trình phát thư (chu trình Euler):", path)

🧠 BÀI 5: Chứng minh định lý Euler cho đồ thị có hướng

📜 Đề bài:

Phát biểu:
Một đồ thị có hướng liên thông **có chu trình Euler** ⇔ với mọi đỉnh v:

$$\deg^{in}(v) = \deg^{out}(v)$$

Yêu cầu: Chứng minh định lý.

🔍 Phân tích:

- Giả thiết: Đồ thị có hướng, liên thông.
- Chu trình Euler: đi qua mọi cung đúng 1 lần và quay về điểm đầu.

✅ Lời giải:

(⇒) Nếu G có chu trình Euler ⇒ mọi đỉnh có in-degree = out-degree

- Mỗi lần đi qua đỉnh, nếu đi vào qua 1 cung thì cũng phải đi ra qua 1 cung.
- Chu trình Euler bắt đầu và kết thúc cùng đỉnh, nên:
    - Số lần vào = số lần ra ở mọi đỉnh.

(⇐) Nếu mọi đỉnh có in-degree = out-degree và đồ thị liên thông ⇒ tồn tại chu trình Euler

- Với điều kiện bậc in = bậc out tại mọi đỉnh:
    - Có thể xây dựng từng đoạn đi qua mỗi cung mà không "bế tắc".
- Dùng thuật toán Hierholzer để tìm chu trình.
- Đảm bảo quay về điểm xuất phát sau khi đi qua tất cả các cung.

⇒ Tồn tại chu trình Euler.

💡 Kết luận:

Đã chứng minh định lý Euler cho đồ thị có hướng.

💻 **Code Python kiểm tra điều kiện in-degree = out-degree:**

```
def check_directed_euler(graph):
    in_deg = defaultdict(int)
    out_deg = defaultdict(int)

    for u, v in graph:
        out_deg[u] += 1
        in_deg[v] += 1

    for v in set(in_deg.keys()).union(out_deg.keys()):
        if in_deg[v] != out_deg[v]:
            return False
    return True

# Ví dụ đồ thị có hướng: A→B, B→C, C→A
edges = [('A','B'), ('B','C'), ('C','A')]
print("Chu trình Euler tồn tại:" if check_directed_euler(edges) else "Không có chu trình Euler")
```

PHẦN 3: TRẮC NGHIỆM

📘 Phần 1: Câu hỏi định nghĩa (Câu 1–5)

Câu 1: Đường đi Euler là gì?
A. Đường đi qua tất cả các đỉnh đúng một lần
B. Đường đi qua tất cả các cạnh đúng một lần
C. Đường đi quay lại đỉnh ban đầu
D. Đường đi ngắn nhất trong đồ thị

Câu 2: Chu trình Euler là gì?
A. Đường đi qua các đỉnh không lặp
B. Đường đi qua các cạnh bất kỳ
C. Đường đi Euler bắt đầu và kết thúc tại cùng một đỉnh
D. Một cây khung có trọng số nhỏ nhất

Câu 3: Đồ thị Euler là gì?
A. Đồ thị liên thông có ít nhất một chu trình Hamilton
B. Đồ thị có chu trình Euler
C. Đồ thị vô hướng không có chu trình
D. Đồ thị có số đỉnh lẻ lớn hơn 2

Câu 4: Định lý Euler áp dụng cho loại đồ thị nào?
A. Chỉ đồ thị có hướng
B. Chỉ đồ thị vô hướng
C. Đồ thị liên thông có hoặc không hướng
D. Chỉ đồ thị có trọng số

Câu 5: Tên nhà toán học gắn liền với khái niệm chu trình Euler là:
A. Gauss
B. Newton
C. Euler
D. Dijkstra

📘 Phần 2: Điều kiện tồn tại đường đi/chu trình Euler (Câu 6–10)

Câu 6: Một đồ thị vô hướng liên thông có chu trình Euler nếu:
A. Tất cả đỉnh có bậc chẵn
B. Có đúng 2 đỉnh bậc lẻ
C. Mỗi cạnh có trọng số giống nhau
D. Có ít nhất một chu trình

Câu 7: Một đồ thị vô hướng liên thông có đường đi Euler (không phải chu trình) nếu:
A. Có nhiều hơn 2 đỉnh bậc lẻ
B. Tất cả đỉnh có bậc lẻ
C. Có đúng 2 đỉnh bậc lẻ
D. Tất cả đỉnh có bậc chẵn

Câu 8: Với đồ thị có hướng, tồn tại chu trình Euler khi:
A. Tất cả cung có hướng ngược
B. Mỗi đỉnh có bậc vào = bậc ra
C. Có đúng 2 đỉnh bậc vào khác bậc ra
D. Có chu trình đơn

Câu 9: Đồ thị có 4 đỉnh, bậc các đỉnh là (2, 4, 2, 4). Kết luận?
A. Không có đường đi Euler
B. Có chu trình Euler
C. Có đúng 2 đỉnh bậc lẻ
D. Có chu trình Hamilton

Câu 10: Số đỉnh bậc lẻ trong một đồ thị Euler là:
A. Luôn bằng 2

B. Có thể là số lẻ bất kỳ
C. Luôn là số chẵn
D. Không xác định được

📘 Phần 3: Câu hỏi về bậc đỉnh (Câu 11–14)

Câu 11: Bậc của một đỉnh trong đồ thị vô hướng là:
A. Tổng các cạnh đi ra từ đỉnh
B. Số đỉnh kề với nó
C. Số cạnh nối với đỉnh đó
D. Số chu trình đi qua đỉnh

Câu 12: Một đồ thị có 5 đỉnh, bậc các đỉnh là (3,3,2,2,2). Đồ thị có đường đi Euler không?
A. Có
B. Không
C. Chưa đủ thông tin
D. Chỉ có trong đồ thị có hướng

Câu 13: Tổng bậc các đỉnh trong đồ thị vô hướng luôn:
A. Là số nguyên tố
B. Chẵn
C. Bằng số đỉnh
D. Bằng số chu trình

Câu 14: Đỉnh có bậc lẻ ảnh hưởng gì đến đường đi Euler?
A. Không ảnh hưởng
B. Gây lỗi thuật toán
C. Là điểm bắt đầu/kết thúc của đường đi Euler
D. Là đỉnh không thể đi qua

📘 Phần 4: Bài toán Chinese Postman (Câu 15–17)

Câu 15: Mục tiêu của bài toán người phát thư Trung Hoa là:
A. Đi qua mọi đỉnh
B. Tìm chu trình Hamilton
C. Tối ưu hóa hành trình đi qua mọi cạnh
D. Tối ưu hóa số bước đi qua các cung

Câu 16: Khi giải bài toán CPP, ta cần:
A. Loại bỏ các cạnh lặp
B. Chuyển đồ thị thành Euler bằng cách thêm cạnh
C. Tìm chu trình Hamilton
D. Giảm số đỉnh

Câu 17: Nếu một đồ thị không Euler, bước đầu tiên khi giải CPP là gì?
A. Đếm số chu trình
B. Tạo đồ thị con
C. Xác định các đỉnh bậc lẻ và ghép cặp
D. Vẽ lại đồ thị

📘 Phần 5: Nhận dạng đồ thị Euler (Câu 18–20)

Câu 18: Nhìn vào đồ thị với tất cả đỉnh bậc 2. Kết luận?
A. Không thể có chu trình Euler
B. Có thể có chu trình Euler
C. Không đủ thông tin
D. Phải có chu trình Hamilton

Câu 19: Nếu đồ thị có hình chuông khép kín với 6 cạnh nối tiếp, tất cả đỉnh bậc 2, thì:
A. Không có đường đi Euler
B. Có chu trình Euler
C. Có 2 đỉnh bậc lẻ
D. Đồ thị không liên thông

Câu 20: Nếu hình vẽ đồ thị có 3 đỉnh bậc lẻ, thì:
A. Không có đường đi Euler
B. Có đường đi Euler
C. Có chu trình Euler
D. Có đồ thị con Euler

PHẦN 4: BÀI TẬP LUYỆN TẬP
Bài 1. Vẽ một đồ thị Euler với 6 đỉnh

📝 **Yêu cầu:**
Vẽ một đồ thị vô hướng có 6 đỉnh sao cho:

- Đồ thị liên thông.
- Tất cả các đỉnh có bậc chẵn.

🧭 **Gợi ý cách làm:**

- Chọn bậc của các đỉnh là 2 hoặc 4 (tùy ý).
- Có thể dùng hình lục giác khép kín:

$$V = \{A,B,C,D,E,F\}, E = \{AB,BC,CD,DE,EF,FA\}$$

→ Mỗi đỉnh bậc 2 → đồ thị Euler.

Bài 2. Cho đồ thị G, kiểm tra có phải đồ thị Euler không

📜 **Cho:**

$$V=\{1,2,3,4\},E=\{(1,2),(2,3),(3,4),(4,1),(1,3)\}$$

🧭 **Hướng dẫn:**

- Tính bậc các đỉnh.
- Kiểm tra điều kiện: tất cả bậc chẵn ⇒ Euler; đúng 2 bậc lẻ ⇒ đường đi Euler.

Bài 3. Viết thuật toán tìm đường đi Euler

📜 **Yêu cầu:**
Viết thuật toán (bằng lời hoặc giả mã) tìm đường đi Euler trên đồ thị vô hướng liên thông.

🧭 **Gợi ý cách làm:**

- Dùng thuật toán Fleury hoặc Hierholzer.
- Mô tả bước:
    1. Bắt đầu từ đỉnh bậc lẻ (nếu có) hoặc bất kỳ.
    2. Tại mỗi bước, chọn cạnh không phải cầu nếu có.
    3. Xóa cạnh, đi tiếp đến đỉnh kề.
    4. Lặp lại đến hết.

Bài 4. Cho đồ thị có 2 đỉnh bậc lẻ → tìm cách thêm cạnh tạo đồ thị Euler

📜 **Cho:**
Đồ thị có các đỉnh A, B, C, D. Bậc: A=3, B=2, C=3, D=2

🧭 **Hướng dẫn:**

- A và C là đỉnh bậc lẻ.
- Thêm cạnh AC để tăng bậc A và C thêm 1 → thành chẵn. → G trở thành đồ thị Euler.

Bài 5. Ứng dụng thuật toán Fleury vào đồ thị có 4 đỉnh, 5 cạnh

📜 **Cho:**
Đồ thị:

$$V=\{1,2,3,4\}, E=\{(1,2),(2,3),(3,4),(4,1),(1,3)\}$$

🧭 **Gợi ý cách làm:**

- Áp dụng thuật toán Fleury từng bước.
- Theo dõi từng cạnh được chọn và lý do (ưu tiên không phải cầu).

Bài 6. Giải bài toán Chinese Postman bằng thuật toán cơ bản

📜 **Cho:**
Đồ thị có trọng số:

A --(2)-- B
| |
(3) (4)
| |
D --(1)-- C

🧭 **Hướng dẫn:**

- Kiểm tra bậc các đỉnh: xác định đỉnh bậc lẻ.
- Tìm cặp ghép ngắn nhất giữa các đỉnh bậc lẻ.
- Thêm các cạnh để tạo Euler.
- Tìm chu trình Euler là hành trình phát thư tối ưu.

Bài 7. Phân tích thời gian chạy của thuật toán Euler

📜 **Yêu cầu:**
Phân tích độ phức tạp của thuật toán Fleury và Hierholzer.

🧭 **Gợi ý:**

- Fleury: $O(E^2)$ vì mỗi bước kiểm tra cầu tốn thời gian.

- Hierholzer: O(E) → tối ưu hơn, thường dùng trong thực tế.

Bài 8. Phân biệt đồ thị Hamilton và Euler qua ví dụ

📜 **Yêu cầu:**
Cho 2 đồ thị:

1. Đi qua mọi cạnh đúng một lần.
2. Đi qua mọi đỉnh đúng một lần.

Hãy chỉ ra cái nào là Euler, cái nào là Hamilton.

🧭 **Gợi ý:**

- Euler → đi qua **cạnh**, có thể lặp đỉnh.
- Hamilton → đi qua **đỉnh**, không lặp.

Bài 9. Áp dụng Euler vào mạng lưới điện thành phố

📜 **Tình huống thực tế:**
Một kỹ sư cần kiểm tra hệ thống cáp điện ngầm, đi qua tất cả các đoạn cáp một lần. Mỗi nút giao là một đỉnh.

🧭 **Hướng dẫn:**

- Mô hình hóa hệ thống cáp thành đồ thị.
- Nếu không Euler → dùng thuật toán CPP để lập lộ trình kiểm tra tối ưu.

Bài 10. Chứng minh đồ thị hoàn chỉnh chẵn đỉnh là đồ thị Euler

📜 **Yêu cầu:**
Chứng minh: Đồ thị hoàn chỉnh K_n, với nn là số chẵn và $n \geq 2n \geq 2$, là đồ thị Euler.

🧭 **Gợi ý:**

Trong K_n, mỗi đỉnh có bậc $n - 1$.
Nếu n chẵn ⇒ $n - 1$ lẻ ⇒ bậc lẻ ⇒ không phải Euler. 🛑 → Nhận ra: **phát biểu sai!**
→ Phát biểu đúng: K_n
với **n lẻ** ⇒ bậc chẵn ⇒ có thể là Euler nếu liên thông.

🧠 Bài này dùng để kiểm tra kỹ năng phản biện và nhận dạng điều kiện đúng/sai.

PHẦN 5: BÀI TẬP DỰ ÁN

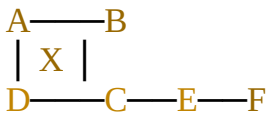
🎯 Dự án 1: Mô hình hóa và giải bài toán người đưa thư cho khu dân cư ảo

📜 Mục tiêu:

Mô phỏng **bài toán người phát thư Trung Hoa (Chinese Postman Problem)** trong một **khu dân cư ảo**, sử dụng **Python + Graphviz** để trực quan hóa hành trình tối ưu.

🗺 Mô tả bài toán:

Giả sử có một khu dân cư với 6 giao lộ (đỉnh) và 8 đoạn đường (cạnh):



Cạnh (trọng số):
AB=2, AC=3, BC=1, BD=4, CD=2, CE=3, EF=2, CF=4

Người đưa thư cần đi **qua tất cả các đoạn đường ít nhất một lần** và **quay lại điểm xuất phát**, sao cho tổng quãng đường **ngắn nhất**.

🧰 Công cụ đề xuất:

- **Python** (sử dụng NetworkX)
- **Graphviz** (để trực quan hóa đồ thị và hành trình)
- **Google Colab / Jupyter Notebook**

🧭 Hướng dẫn thực hiện:

1. **Mô hình hóa đồ thị bằng NetworkX**:
    - Định nghĩa các đỉnh và cạnh có trọng số.
2. **Kiểm tra bậc đỉnh**:
    - Xác định các đỉnh có bậc lẻ.
3. **Ghép cặp tối ưu các đỉnh lẻ**:
    - Tìm đường ngắn nhất giữa chúng (Dijkstra).
4. **Thêm cạnh ảo để biến đồ thị thành Euler**.
5. **Tìm chu trình Euler** (Hierholzer hoặc thư viện).

6. **Hiển thị đồ thị và đường đi bằng Graphviz**.

💡 **Gợi ý mở rộng:**

- Cho phép người dùng chọn điểm bắt đầu.
- Gắn màu khác nhau cho cạnh lặp lại.
- Thêm giao diện nhập dữ liệu từ file.

🎯 **Dự án 2: Phân tích đường đi hiệu quả cho robot kiểm tra đường ống trong nhà máy**

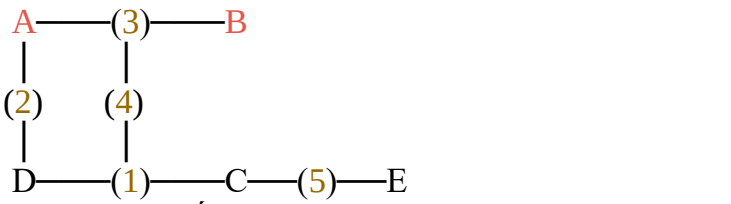
📄 **Mục tiêu:**

Xây dựng thuật toán cho robot tự hành để **kiểm tra toàn bộ hệ thống ống dẫn** trong nhà máy, sao cho:

- Đi qua tất cả các đoạn ống **ít nhất một lần**.
- Tổng chiều dài di chuyển là **nhỏ nhất**.

🗺️ **Mô tả bài toán:**

Sơ đồ hệ thống ống dưới dạng đồ thị có trọng số:



Cạnh (trọng số): AB=3, AD=2, BC=4, CD=1, CE=5

📐 **Yêu cầu:**

- **Gắn nhãn cạnh** (mã ống, độ dài).
- Tính **tổng chiều dài** cần đi qua.
- Trình bày **thuật toán tối ưu hóa đường đi** (dựa vào Euler hoặc CPP).

🧰 **Công cụ đề xuất:**

- Python (NetworkX + matplotlib hoặc Plotly)
- Có thể xuất file hành trình cho robot điều khiển thực tế (CSV)

🧭 **Hướng dẫn thực hiện:**

1. **Mô hình hóa đồ thị với trọng số**.
2. **Kiểm tra số đỉnh bậc lẻ**, ghép cặp để chuyển sang Euler.
3. **Thêm cạnh ảo để giải bài toán phát thư (CPP)**.
4. **Tìm đường đi Euler**.
5. **Tính tổng chiều dài cần đi**.
6. **Hiển thị sơ đồ và hành trình**.

💡 **Gợi ý mở rộng:**

- Cho phép nhiều robot chia tuyến.
- Giới hạn chiều dài tối đa mỗi lượt đi (nếu pin robot giới hạn).
- Thêm lựa chọn loại bỏ điểm giao không cần kiểm tra (nút phụ).