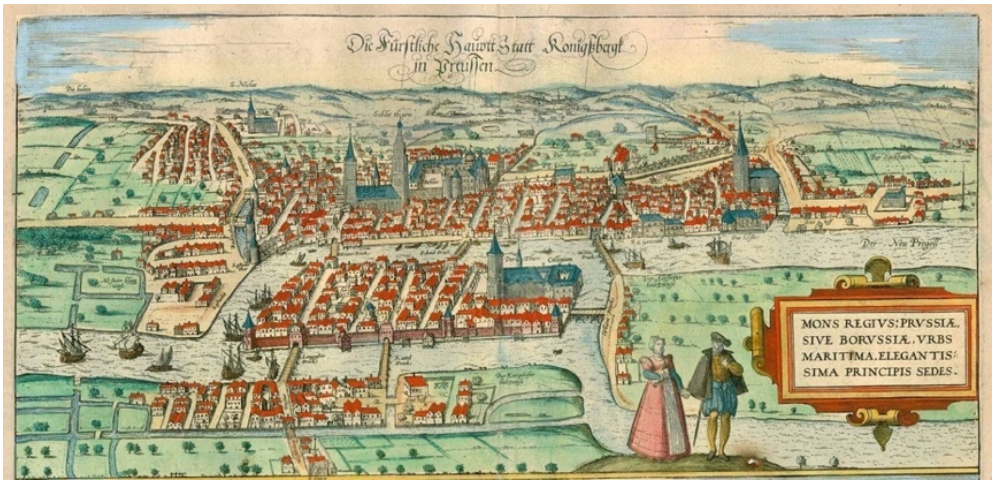
BÀI 6
ĐƯỜNG ĐI VÀ CHU TRÌNH EULER

Mục tiêu

- Hiểu các khái niệm đường đi Euler, chu trình Euler, đồ thị Euler và đồ thị nửa Euler;
- Hiểu ý tưởng và cài đặt được thuật toán **Hierholzer** để tìm đường đi/chu trình Euler;
- Vận dụng giải các bài toán liên quan.

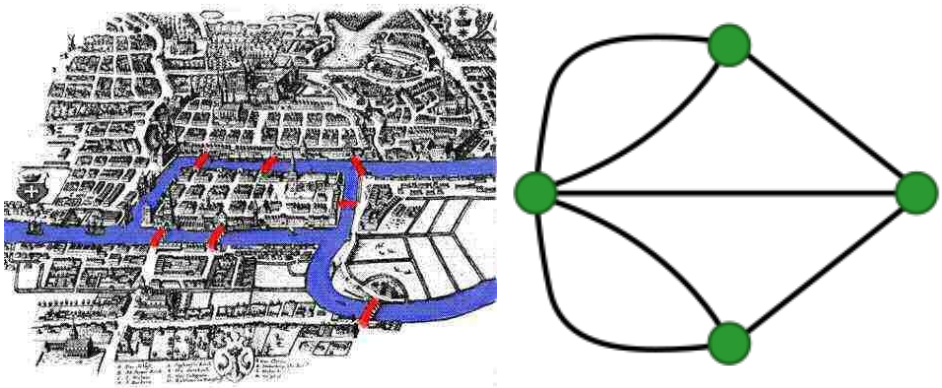
1. Một số khái niệm



Hình 1. Thành phố Königsberg (Nga) có 4 vùng đất được nối với nhau bởi 7 cây cầu

Bài toán: Người dân trong vùng thách đố nhau là thử tìm cách xuất phát từ một vùng đi dạo qua mỗi chiếc cầu đúng một lần và trở về nơi xuất phát.

Video giới thiệu bài toán: https://youtu.be/nZwSo4vfw6c?si=DILGp8DIcmYC2lov



Hình 2. Mô hình đồ thị bài toán 7 cây cầu ở thành phố Königsberg

Đường đi Euler *(Eulerian path):* Một đường đi trong đồ thị $G = (V, E)$ được gọi là đường đi Euler nếu **đi qua tất cả các cạnh** của đồ thị, **mỗi cạnh đúng một lần**.

Chu trình Euler *(Eulerian cycle):* là đường đi Euler có đỉnh đầu và đỉnh cuối trùng nhau.

Đồ thị Euler *(Eulerian graph):* là đồ thị **chứa chu trình Euler**.

Đồ thị nửa Euler *(semi-Eulerian graph):* là đồ thị **chứa đường đi Euler**.

Định lý Euler 1: *(Điều kiện cần và đủ để đồ thị vô hướng có chu trình Euler)*

Đồ thị **vô hướng** $G = (V, E)$ là **đồ thị Euler** khi và chỉ khi *tất cả các đỉnh có bậc lớn hơn 0*:

(1) Thuộc cùng một thành phần liên thông và;

(2) Có bậc chẵn.

Định lý Euler 2: *(Điều kiện cần và đủ để đồ thị vô hướng có đường đi Euler)*

Đồ thị **vô hướng** $G = (V, E)$ là **đồ thị nửa Euler** khi và chỉ khi *tất cả các đỉnh có bậc lớn hơn 0*:

(1) Thuộc cùng một thành phần liên thông và;

*(2) Có **đúng 2 đỉnh có bậc lẻ, các đỉnh còn lại có bậc chẵn**.*

Định lý Euler 3: *(Điều kiện cần và đủ để đồ thị có hướng có chu trình Euler)*

Đồ thị **có hướng** $G = (V, E)$ là **đồ thị Euler** khi và chỉ khi *tất cả các đỉnh có bậc lớn hơn 0*:

(1) Thuộc cùng một thành phần liên thông và;

*(2) Có **bán bậc vào bằng bán bậc ra**.*

Định lý Euler 4: *(Điều kiện cần và đủ để đồ thị có hướng có đường đi Euler)*

Đồ thị **có hướng** $G = (V, E)$ là **đồ thị nửa Euler** khi và chỉ khi *tất cả các đỉnh có bậc lớn hơn 0*:

(1) Thuộc cùng một thành phần liên thông và;

*(2) **Tồn tại đúng 2 đỉnh** $u, v \in V$ sao cho:*

$deg^+(u) = deg^-(u) + 1$ *(đỉnh bắt đầu đường đi)*

$deg^-(v) = deg^+(v) + 1$ *(đỉnh kết thúc đường đi)*

*(3) Các **đỉnh còn lại của G có bán bậc vào bằng bán bậc ra**.*

2. Thuật toán Hierholzer tìm chu trình Euler

2.1. Kiểm tra đồ thị có chứa chu trình Euler hay không?

Input: đồ thị vô hướng $G(V, E)$

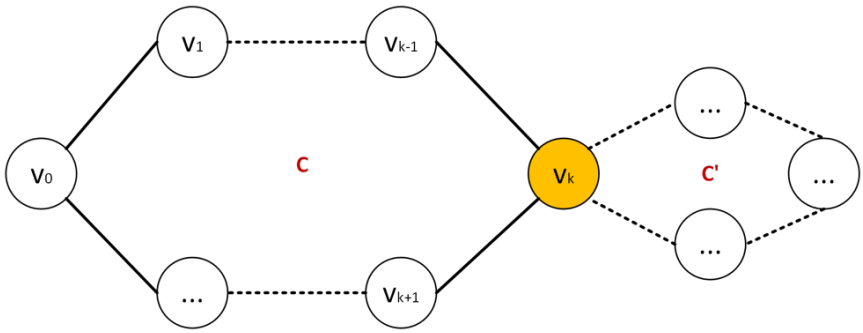
Output: *true*/*false*: đồ thị *có*/*không có* chu trình Euler

```
HasEulerianCycle()
    //Kiểm tra các đỉnh có bậc lớn hơn 0
    //thuộc cùng một thành phần liên thông? (dùng BFS/DFS)
    IF(!IsConnected())
        RETURN false

    //Kiểm tra các đỉnh có bậc lớn hơn 0 đều có bậc chẵn
    FOREACH v IN V
        IF(deg[v] > 0 AND deg[v] mod 2 ≠ 0)
            RETURN false
    RETURN true
```

2.2. Xác định chu trình Euler bằng thuật toán Hierholzer

Ý tưởng:

- Xuất phát từ một đỉnh bất kỳ (*có bậc lớn hơn 0*), tìm một chu trình C đi qua một số cạnh của đồ thị *(mỗi cạnh đi đúng một lần và không nhất thiết phải đi qua tất cả các cạnh);*
  Tại sao chắc chắn sẽ tìm được một chu trình?
- Nếu trong chu trình C có đỉnh v_k liên thuộc với cạnh chưa đi qua thì tìm chu trình C' bắt đầu tại đỉnh v_k. Sau đó ghép nối C' vào chu trình C để mở rộng chu trình hiện có;

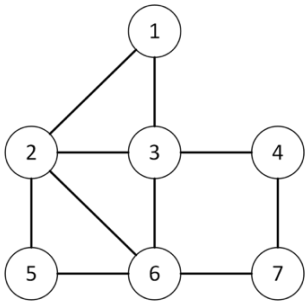


- Lặp lại cho đến khi trong chu trình C không còn bất kỳ đỉnh nào liên thuộc với cạnh chưa đi qua, tức là chu trình C đã đi qua tất cả các cạnh của đồ thi → tìm được chu trình Euler, thuật toán kết thúc.

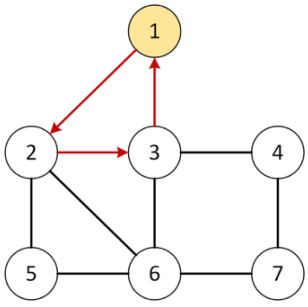
Cài đặt:

- **Cách 1:** Nối chu trình C' vào cuối chu trình C: dùng **thuật toán đối xứng gương** để đảo chu trình tại vị trí v_k.
  - Giả sử có chu trình C: $\{v_0, v_1, \dots, v_{k-1}, v_k, v_{k+1}, \dots, v_0\}$
  - Đảo chu trình tại vị trí v_k:
    - Đảo đoạn đầu $\{v_0, v_1, \dots, v_{k-1}\}$: $\{v_{k-1}, \dots, v_1, v_0, v_k, v_{k+1}, \dots, v_0\}$
    - Đảo đoạn cuối $\{v_k, v_{k+1}, \dots, v_0\}$: $\{v_{k-1}, \dots, v_1, v_0, v_0, \dots, v_{k+1}, v_k\}$
    - Đảo toàn bộ chu trình: $\{v_k, v_{k+1}, \dots, v_0, v_0, v_1, \dots, v_{k-1}\}$
  - Lưu ý: ban đầu chu trình bắt đầu tại v_0 nên v_0 xuất hiện 2 lần *(đỉnh đầu cũng là đỉnh cuối)*, nhưng khi đảo chu trình tại vị trí k thì v_0 chỉ xuất hiện 1 lần ở giữa chu trình và v_k sẽ xuất hiện 2 lần *(đỉnh đầu cũng là đỉnh cuối)*. Thuật toán đối xứng gương hiệu chỉnh:
    - Đảo đoạn đầu $\{v_1, \dots, v_{k-1}\}$: $\{v_0, v_{k-1}, \dots, v_1, v_k, v_{k+1}, \dots, v_0\}$
    - Đảo đoạn cuối $\{v_k, v_{k+1}, \dots, v_0\}$: $\{v_0, v_{k-1}, \dots, v_1, v_0, \dots, v_{k+1}, v_k\}$
    - Đảo toàn bộ chu trình: $\{v_k, v_{k+1}, \dots, v_0, v_1, \dots, v_{k-1}, v_0\}$
    - Thay thế phần tử cuối trong chu trình thành v_k.

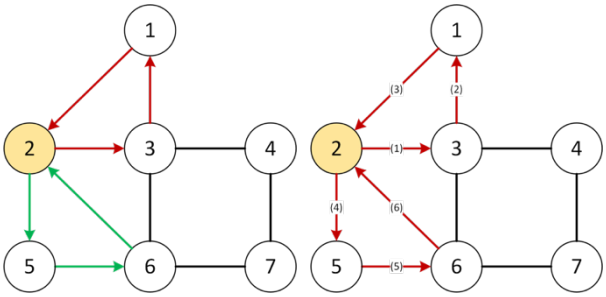
Ví dụ: Tìm chu trình Euler trong đồ thị dưới đây bằng cách đảo chu trình:



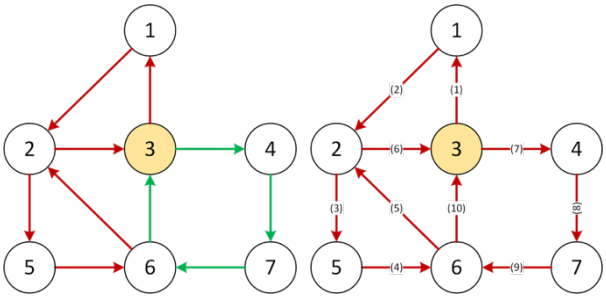
- **Kiểm tra sự tồn tại chu trình Euler**: Đồ thị liên thông và tất các các đỉnh đều có bậc chẵn → Đồ thị có chu trình Euler.
- Giả sử bắt đầu từ **đỉnh 1**, tìm được chu trình C: 1 → 2 → 3 → 1. Chu trình này có đỉnh 2 và 3 liên thuộc với các cạnh chưa đi qua.



- Xuất phát từ **đỉnh 2**, tìm được chu trình C': 2 → 5 → 6 → 2. Ghép C' vào C để tạo thành chu trình mới (*đảo C tại đỉnh 2*): 2 → 3 → 1 → 2 → 5 → 6 → 2. Chu trình mới có đỉnh 3 và 6 liên thuộc với các cạnh chưa đi qua.



- Xuất phát từ **đỉnh 3**, tìm được chu trình C': 3 → 4 → 7 → 6 → 3. Ghép C' vào C để tạo thành chu trình mới (*đảo C tại đỉnh 3*): 3 → 1 → 2 → 5 → 6 → 2 → 3 → 4 → 7 → 6 → 3. Chu trình mới không có đỉnh nào liên thuộc với cạnh chưa đi qua → tìm được chu trình Euler, thuật toán kết thúc.



Input: đồ thị vô hướng $G(V, E)$

Output: một chu trình Euler (nếu có)

```
FindEulerianCycle()
    //Kiểm tra đồ thị có chứa chu trình Euler hay không?
    IF(!HasEulerianCycle())
        RETURN null
    //Chọn đỉnh xuất phát x0: đỉnh bất kỳ có bậc lớn hơn 0
    u ← x0
    eulerCycle ← ∅
    //Lặp cho đến khi chu trình đi qua hết các cạnh
    WHILE(true)
        //Tìm chu trình mới và nối vào cuối chu trình hiện tại
        eulerCycle ← eulerCycle + FindCycleFrom(u)
        //Tìm đỉnh k liên thuộc với cạnh chưa đi qua
        k ← FindVertex(eulerCycle)
        //Không có đỉnh k thì thoát vòng lặp
        IF(k == -1) break
        //Đảo chu trình tại vị trí k
        ReverseCycle(eulerCycle, k)
        u ← k
    RETURN eulerCycle
```

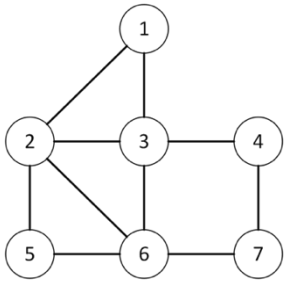
- **Cách 2:** Chèn chu trình C' vào vị trí v_k của chu trình C: **dùng Stack**

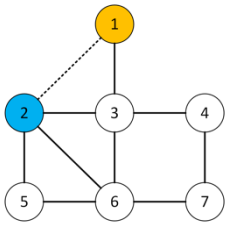
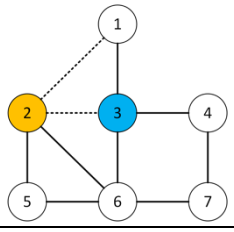
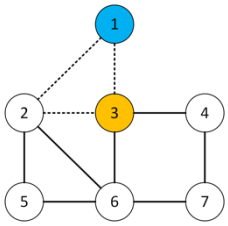
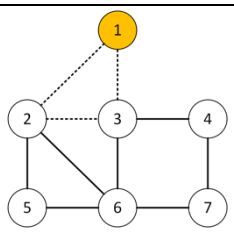
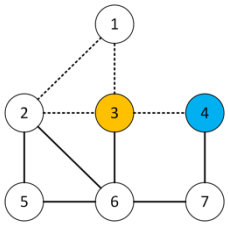
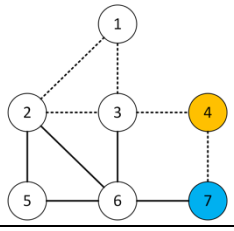
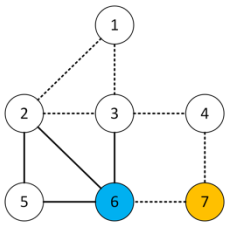
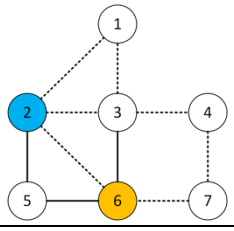
Input: đồ thị $G(V, E)$

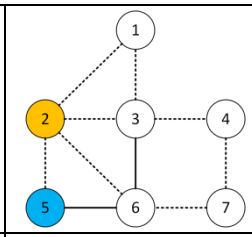
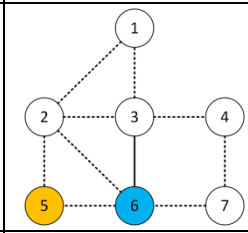
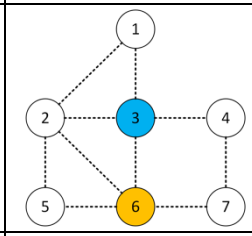
Output: một chu trình Euler (nếu có)

```
FindEulerianCycle()
    //Kiểm tra đồ thị có chứa chu trình Euler hay không?
    IF(!HasEulerianCycle())
        RETURN null
    //Chọn đỉnh xuất phát x0: đỉnh bất kỳ có bậc lớn hơn 0
    s.Push(x0)
    eulerStack ← ∅
    visitedEdges ← ∅
    WHILE(s ≠ ∅)
        u ← s.Top() //lấy giá trị đỉnh stack, không xóa
        hasUnvisitedEdge ← false
        //Có cạnh (u,v) chưa đi qua
        FOREACH v IN adjList[u]
            IF((u, v) ∉ visitedEdges)
                s.Push(v)
                visitedEdges.Add((u,v))
                visitedEdges.Add((v,u))
                hasUnvisitedEdge ← true
                break
        //Không còn cạnh (u,v) chưa đi qua
        IF(!hasUnvisitedEdge)
            eulerStack.Push(u)
            s.Pop()
    eulerCycle ← ∅
    WHILE(eulerStack ≠ ∅)
        eulerCycle ← eulerCycle + eulerStack.Top()
        eulerStack.Pop()
    RETURN eulerCycle
```

Ví dụ: Tìm chu trình Euler trong đồ thị dưới đây *(sử dụng Stack)*:



Stack	u	v	eulerStack	Ghi chú
1	1	2	∅	
1, 2	2	3	∅	
1, 2, 3	3	1	∅	
1, 2, 3, 1	1	-	1	
1, 2, 3	3	4	1	
1, 2, 3, 4	4	7	1	
1, 2, 3, 4, 7	7	6	1	
1, 2, 3, 4, 7, 6	6	2	1	

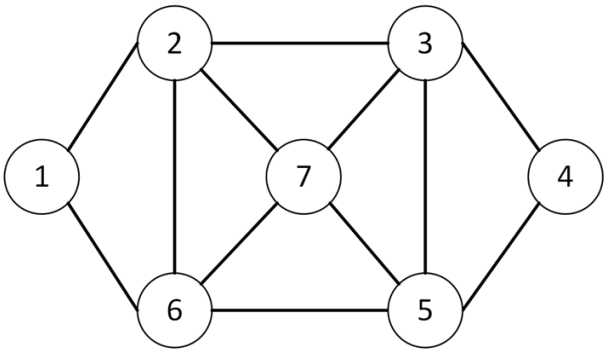
1, 2, 3, 4, 7, 6, 2	2	5	1	
1, 2, 3, 4, 7, 6, 2, 5	5	6	1	
1, 2, 3, 4, 7, 6, 2, 5, 6	6	3	1	
1, 2, 3, 4, 7, 6, 2, 5, 6, 3	3	-	1, 3	
1, 2, 3, 4, 7, 6, 2, 5, 6	6	-	1, 3, 6	
1, 2, 3, 4, 7, 6, 2, 5	5	-	1, 3, 6, 5	
1, 2, 3, 4, 7, 6, 2	2	-	1, 3, 6, 5, 2	
1, 2, 3, 4, 7, 6	6	-	1, 3, 6, 5, 2, 6	
1, 2, 3, 4, 7	7	-	1, 3, 6, 5, 2, 6, 7	
1, 2, 3, 4	4	-	1, 3, 6, 5, 2, 6, 7, 4	
1, 2, 3	3	-	1, 3, 6, 5, 2, 6, 7, 4, 3	
1, 2	2	-	1, 3, 6, 5, 2, 6, 7, 4, 3, 2	
1	1	-	1, 3, 6, 5, 2, 6, 7, 4, 3, 2, 1	
∅				

➔ Chu trình Euler tìm được là *(lấy lần lượt từng phần tử từ đỉnh eulerStack)*:

{1, 2, 3, 4, 7, 6, 2, 5, 6, 3, 1}

Độ phức tạp: **O**(|V| + |E|)

Bài tập: Hãy cho biết đồ thị dưới đây có chứa chu trình Euler hay không? Nếu có hãy dùng thuật toán **Hierholzer** để xác định chu trình Euler *(sử dụng Stack)*.



3. Thuật toán Hierholzer tìm đường đi Euler

Đường đi Euler bắt đầu tại một trong hai đỉnh có bậc lẻ và kết thúc tại đỉnh còn lại. Do đó, có thể **sử dụng thuật toán Hierholzer tương tự như chu trình Euler tương nhưng đỉnh bắt đầu phải chọn một trong hai đỉnh có bậc lẻ**.

Input: đồ thị $G(V, E)$

Output: một chu trình Euler (nếu có)

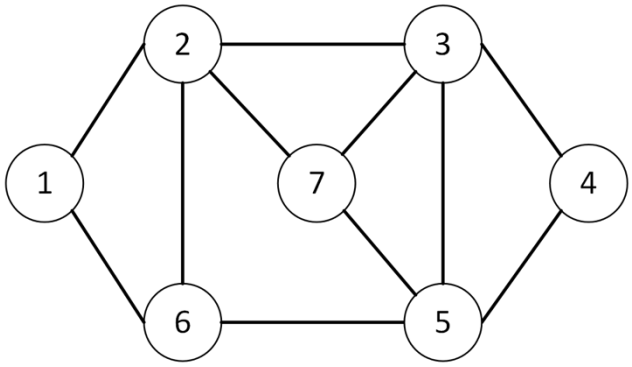
```
FindEulerianPath()
    //Kiểm tra đồ thị có chứa chu trình Euler hay không?
    IF(!HasEulerianPath())
        RETURN null
    //Chọn đỉnh xuất phát x0: một trong hai đỉnh có bậc lẻ
    //Trường hợp đồ thị chứa chu trình Euler
    //thì có thể chọn đỉnh bất kỳ có bậc lớn hơn 0
    s.Push(x0)
    eulerStack ← ∅
    visitedEdges ← ∅

    WHILE(s ≠ ∅)
        u ← s.Top() //lấy giá trị đỉnh stack, không xóa
        hasUnvisitedEdge ← false
        //Có cạnh (u,v) chưa đi qua
        FOREACH v IN adjList[u]
            IF((u, v) ∉ visitedEdges)
                s.Push(v)
                visitedEdges.Add((u,v))
                visitedEdges.Add((v,u))
                hasUnvisitedEdge ← true
                break
        //Không còn cạnh (u,v) chưa đi qua
        IF(!hasUnvisitedEdge)
            eulerStack.Push(u)
            s.Pop()
    eulerPath ← ∅
    WHILE(eulerStack ≠ ∅)
        eulerPath ← eulerPath + eulerStack.Top()
        eulerStack.Pop()
    RETURN eulerPath
```

Input: đồ thị vô hướng $G(V, E)$

Output: *true*/*false*: đồ thị *có*/*không có* đường đi Euler

```
HasEulerianPath()
    //Kiểm tra các đỉnh có bậc lớn hơn 0
    //thuộc cùng một thành phần liên thông? (dùng BFS/DFS)
    IF(!IsConnected())
        RETURN false
    //Kiểm tra các đỉnh bậc lớn hơn 0 đều có bậc chẵn
    //hoặc có đúng 2 đỉnh bậc lẻ
    oddDegreeCount ← 0
    FOREACH v IN V
        IF(deg[v] mod 2 ≠ 0)
            oddDegreeCount ← oddDegreeCount + 1
    IF(oddDegreeCount == 0 OR oddDegreeCount == 2)
        RETURN true
    RETURN false
```

Bài tập: Hãy cho biết đồ thị dưới đây có chứa đường đi Euler hay không? Nếu có hãy dùng thuật toán **Hierholzer** để xác định đường đi Euler *(sử dụng Stack)*.



--- HẾT ---