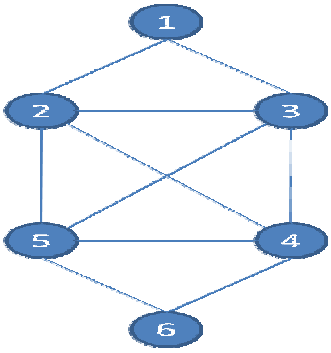
HƯỚNG DẪN CHU TRÌNH EULER, ĐƯỜNG ĐI EULER

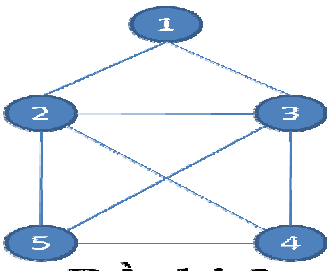
1. Lý thuyết

Chu trình Euler là chu trình đi qua tất cả các cạnh của đồ thị, mỗi cạnh đúng một lần trong đó đỉnh đầu và đỉnh cuối trùng nhau (đồ thị có thể có các đỉnh cô lập). Tương tự với đường đi Euler, ngoại trừ điểm đầu và điểm cuối không trùng nhau.

Ví dụ :



Đồ thị 1



Đồ thị 2

Đồ thị 1 có chu trình euler, chẳng hạn như 1 **à** 2 **à** 3 **à** 4 **à** 5 **à** 2 **à** 4 **à** 6 **à** 5 **à** 3 **à** 1.

Còn đồ thị 2 không có chu trình euler nhưng có đường đi euler, chẳng hạn như 4 **à** 5 **à** 2 **à** 3 **à** 4 **à** 2 **à** 1 **à** 3 **à** 5.

Điều kiện cần để tồn tại chu trình Euler là:

- Trong đồ thị vô hướng, bậc của tất cả các đỉnh phải là số chẵn.
- Trong đồ thị có hướng, bậc ngoài và bậc trong của mỗi đỉnh phải bằng nhau.

Trong một đồ thị, nếu không tìm ra chu trình Euler, vẫn có thể tồn tại đường đi Euler.

Điều kiện cần để tồn tại đường đi Euler trong trường hợp này là:

- Trong đồ thị vô hướng, tồn tại duy nhất hai đỉnh có bậc lẻ, tất cả các đỉnh còn lại là bậc chẵn.
- Trong đồ thị có hướng, bậc ngoài và bậc trong của mỗi đỉnh bằng nhau ngoại trừ một đỉnh tại đó bậc ngoài lớn hơn bậc trong 1 đơn vị (làm đỉnh bắt đầu) và một đỉnh tại đó bậc trong lớn hơn bậc ngoài 1 đơn vị (làm đỉnh kết thúc).

2. Thuật toán Fleury

Xuất phát từ 1 đỉnh nào đó của đồ thị G (G đã loại các đỉnh cô lập) ta đi theo các cạnh của nó một cách tuỳ ý chỉ cần tuân thủ 2 quy tắc sau:

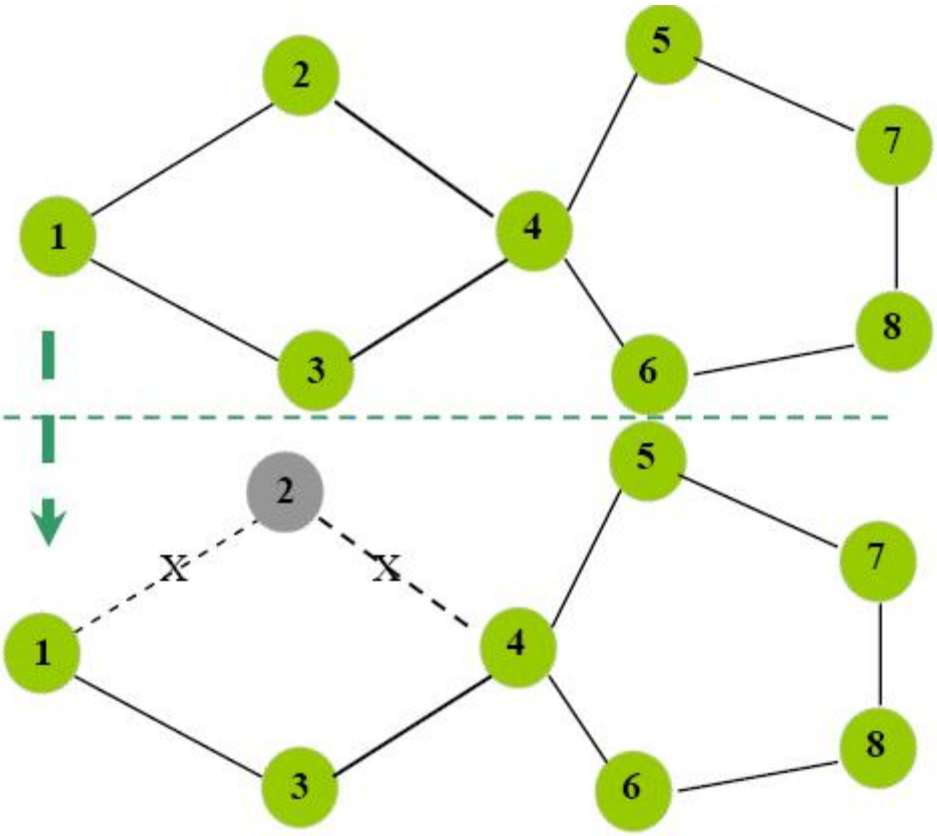
- Cạnh này không phải là "cầu" (việc bỏ cạnh này không làm cho đồ thị mất liên thông).
- Xóa bỏ cạnh đã đi qua và đồng thời xóa cả những đỉnh cô lập tạo thành.

Nếu không tìm thấy cạnh nào thỏa, thuật toán dừng. Ngược lại, từ đỉnh được nối đến, lặp lại quá trình trên. Một cách tự nhiên, chu trình Euler sẽ quay lại điểm bắt đầu, đường đi Euler sẽ đến điểm kết thúc.

Lưu ý: Khi xóa bỏ cạnh đi qua và đỉnh cô lập, sẽ hình thành một đồ thị mới. Việc xét "cầu" tiếp theo sẽ trên đồ thị mới này.

Nhận xét: Thuật toán này tuy đơn giản nhưng thường không được sử dụng bởi việc xác định "cầu" trong đồ thị không phải đơn giản.

Ví dụ: ta có đồ thị sau:



Nếu xuất phát từ đỉnh 1, có hai cách đi tiếp: hoặc sang 2 hoặc sang 3, giả sử ta sẽ sang 2 và xoá cạnh (1, 2) vừa đi qua. Từ 2 chỉ có cách duy nhất là sang 4, nên cho dù (2, 4) là cầu ta cũng phải đi sau đó xoá luôn cạnh (2, 4). Đến đây, các cạnh còn lại của đồ thị được vẽ bằng nét liền, các cạnh đã bị xoá được vẽ bằng nét đứt.

Bây giờ đang đứng ở đỉnh 4 thì ta có 3 cách đi tiếp: sang 3, sang 5 hoặc sang 6. Vì (4, 3) là "cầu" nên ta sẽ không đi theo cạnh (4, 3) mà sẽ đi (4, 5) hoặc (4, 6). Nếu đi theo (4, 5) và cứ tiếp tục đi như vậy, ta sẽ được chu trình Euler là (1, 2, 4, 5, 7, 8, 6, 4, 3, 1). Còn đi theo (4, 6) sẽ tìm được chu trình Euler là: (1, 2, 4, 6, 8, 7, 5, 4, 3, 1).

3. Thuật toán tìm chu trình/đường đi Euler

Từ nhận xét về thuật toán Fleury, chúng ta tìm thuật toán để thực thi một cách dễ hơn.

Cho G=(X, E) là một đồ thị gồm n đỉnh. Thuật toán tìm chu trình Euler xuất phát từ u với u là đỉnh có bậc khác 0 như sau:

‘tour’ là một stack;

Find_tour (u)

Với mỗi cạnh e=(u,v) trong E

Loại cạnh e khỏi tập E;

Find_tour (v);

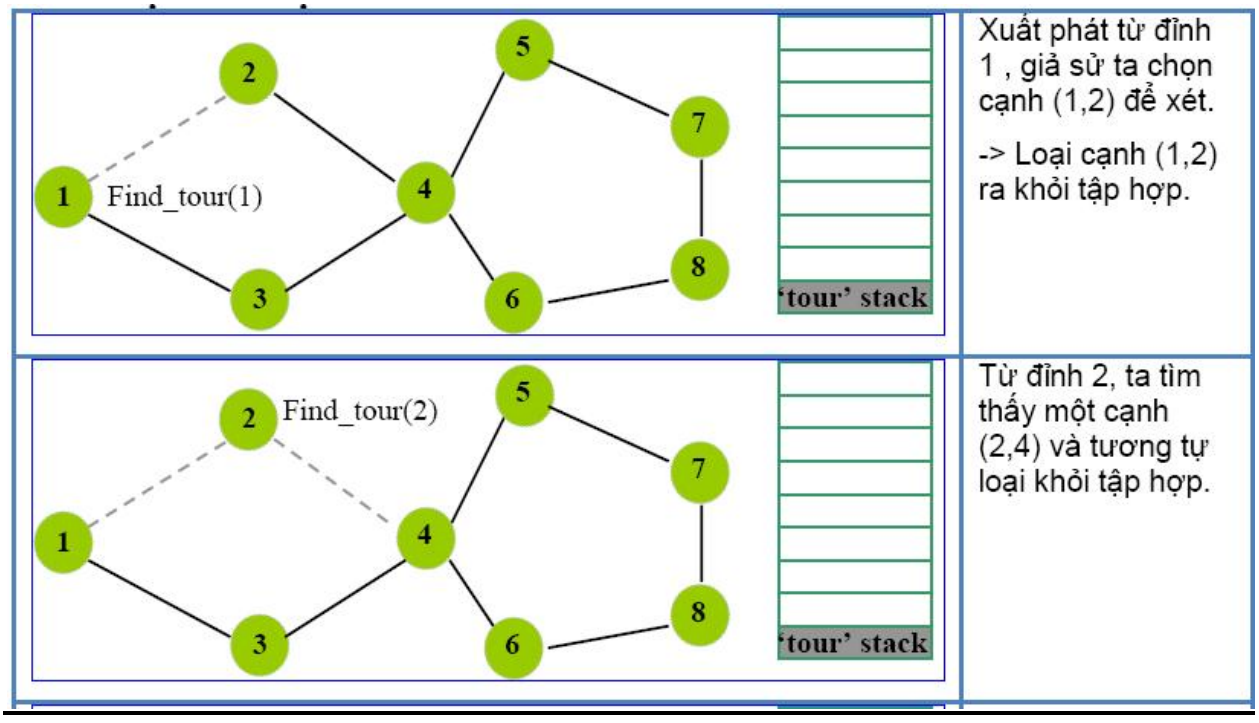
Cuối với mỗi

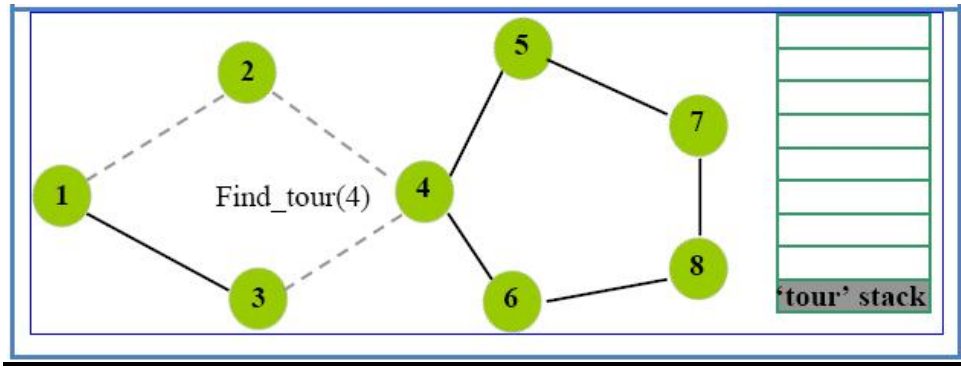
Thêm u vào stack ‘tour’;

Cuối hàm

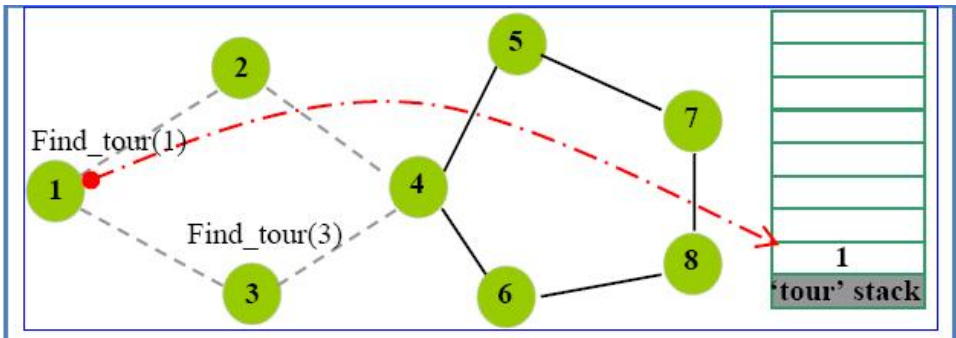
Lưu ý: thuật toán này cũng có thể được sử dụng để tìm đường đi Euler bằng cách tạo một cạnh “giả” (dummy edge) nối điểm đầu và điểm cuối với nhau.

4. Ví dụ minh họa



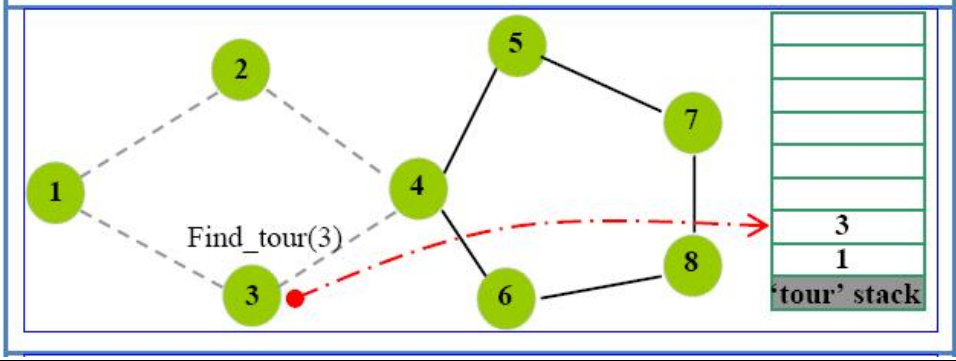


Tại đỉnh 4 có cạnh nối (4,3), (4,5), (4,6). Giả sử xét cạnh (4,3) trước.

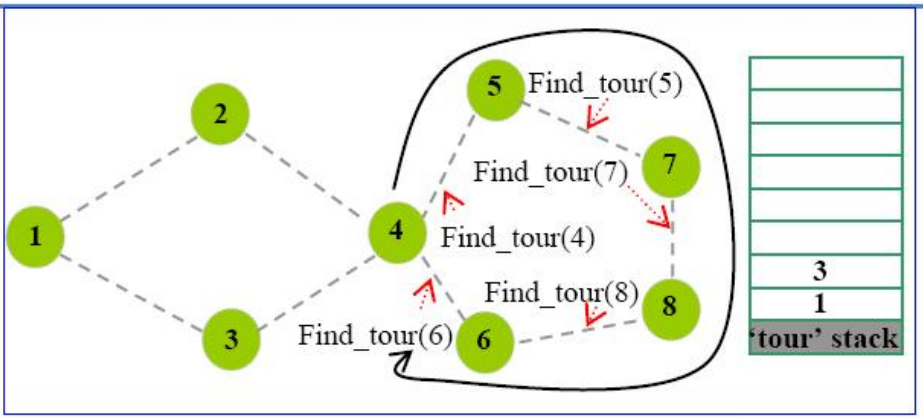


Tiếp đến xét đỉnh 3 và loại cạnh (3,1).

Lúc này từ 1 không còn cạnh nào chưa xét, vì vậy ta đưa đỉnh 1 vào stack **'tour'**



Quay lui lại find_tour(3). Từ 3 cũng không còn cạnh nào chưa xét, đưa đỉnh 3 vào stack 'tour'.



Tương tự, quay lui về đỉnh 4, xét tiếp cạnh (4,5).

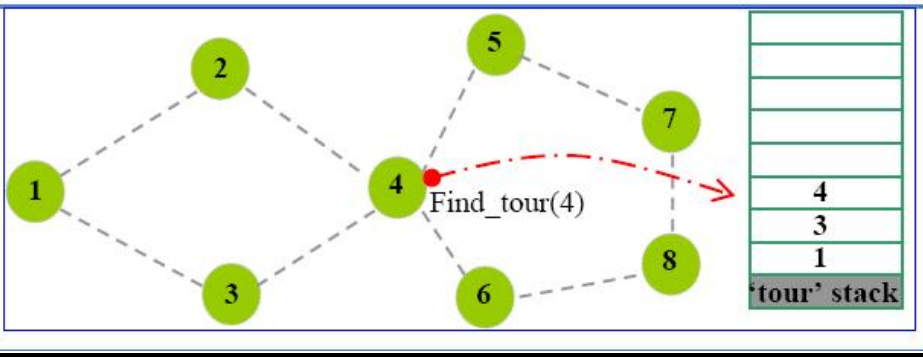
Ta tiếp tục xét các đỉnh kế tiếp.

5: (5,7)

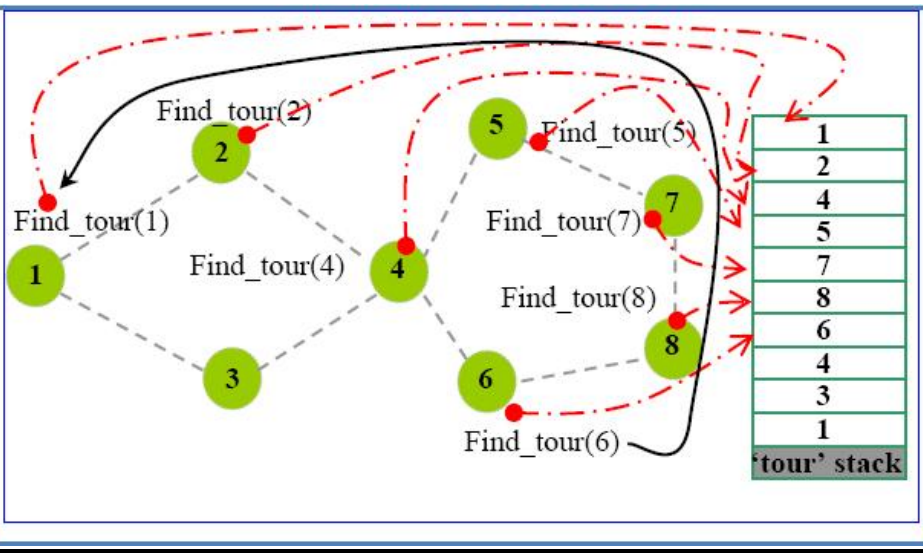
-> 7: (7,8)

-> 8: (8,6)

-> 6: (6,4)



Tại đỉnh 4, không còn tìm thấy cạnh nối nữa. Cho vào stack và quay lui.



Lần lượt quay lui và cho vào stack (vì các đỉnh không còn cạnh nối nào nữa).

Khi đó ta lấy theo nguyên tắc của stack là LIFO (Last In First Out) ta sẽ được một chu trình euler là 1 **à** 2 **à** 4 **à** 5 **à** 7 **à** 8 **à** 6 **à** 4 **à** 3 **à** 1.

----------HẾT----------