

LÝ THUYẾT ĐỒ THỊ

CHU TRÌNH HAMILTON

Phiên bản: 0.3 – tháng 10 năm 2014

Tác giả: Nguyễn Trung Thành (abcxyz tcIT)

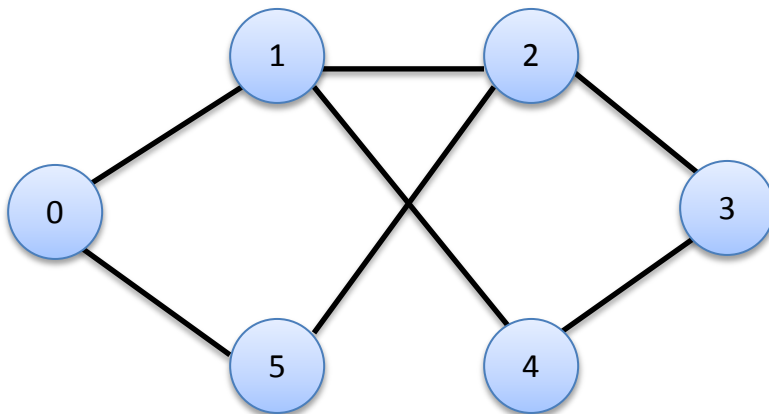
Sinh viên năm 2 khoa CNTT – Đại học Khoa học Tự nhiên TP HCM

<https://www.facebook.com/abcxyztcit>

1. Chu trình Hamilton là gì

Mấy cái này chắc là các bạn xem tài liệu :D Tài liệu này hướng về thực hành và hiểu hơn là lý thuyết. Đơn giản nó là **một đường đi qua tất cả các đỉnh** trong đồ thị, **mỗi đỉnh đi qua đúng 1 lần**.

Ví dụ với đồ thị vô hướng ở dưới



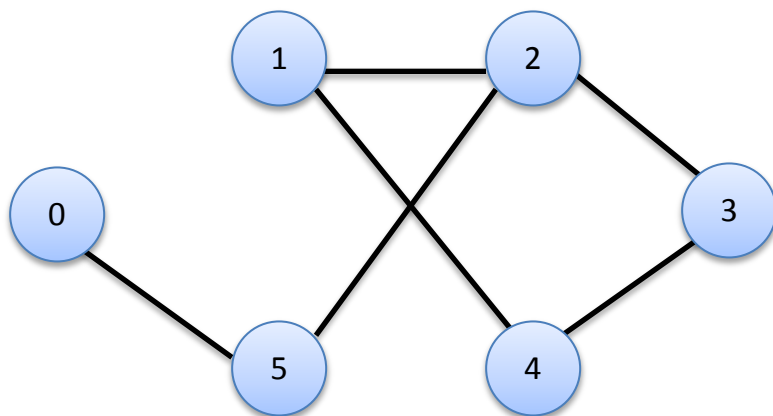
Ta có chu trình đi là $0 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1 \rightarrow 0$.

Điểm xuất phát ban đầu cũng là điểm kết thúc.

Dĩ nhiên, ta có thể xuất phát từ các đỉnh khác nhưng chu trình vẫn y chang, ví dụ như :D

$5 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1 \rightarrow 0 \rightarrow 5$

Tuy nhiên, nếu đồ thị của ta như hình dưới thì không có chu trình Hamilton mà chỉ có đường đi Hamilton



Đường đi của chúng ta là $0 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$.

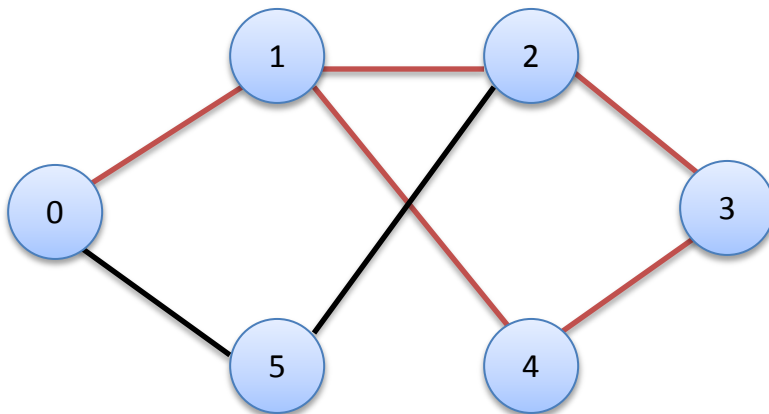
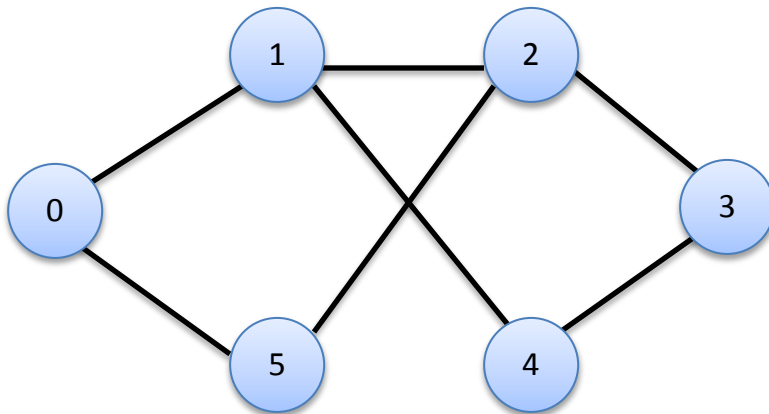
Vẫn đi qua các đỉnh đúng 1 lần nhưng điểm khởi đầu khác điểm kết thúc, chỉ vậy thôi :D

2. Thuật toán

Theo kiến thức hạn hẹp của mình thì cho đến ngày nay chưa có một thuật toán nào thật sự hiệu quả để tìm chu trình Hamilton cả. Hiện tại chỉ có một thuật toán đơn giản là thuật toán Fleury. Tuy nhiên trong tài liệu này mình sẽ trình bày hướng giải quyết một cách đơn giản nhất.

Xuất phát từ một đỉnh, ta cứ "đi đại" tùy ý, đi đã đời mà thấy nó quay về đỉnh xuất phát thì coi như OK ta đã tìm được chu trình Hamilton. Suy nghĩ quá đơn giản nhỉ !

Xét lại đồ thị như hình dưới



Nếu ta "đi đại" thì theo thứ tự từ điển, đường đi đầu tiên ta đi được là..

$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1 \implies \text{oops !!!}$

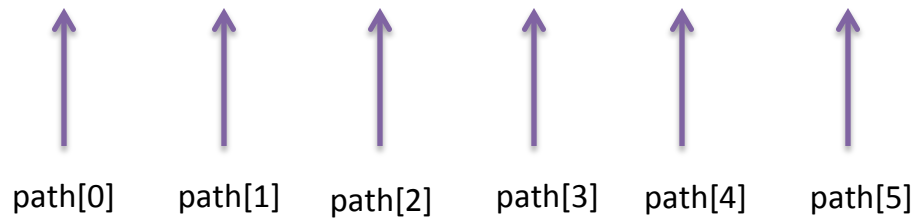


Vấn đề phát sinh: dùng 1 mảng bool OK để đánh dấu, đỉnh i đi qua rồi thì $OK[i] = \text{false}$. Ban đầu mảng OK này khởi tạo là true hết.

Vấn đề tiếp theo: ta cần một mảng $\text{path}[n]$ để lưu lại đường đi.

Ví dụ ta có chu trình Hamilton là

0 → 1 → 4 → 3 → 2 → 5 → 0



Ban đầu, ta chọn đại 1 đỉnh để đi, ví dụ đỉnh 0 → $\text{path}[0] = 0$.

Ở bước thứ 1, ta xác định $\text{path}[1]$

Ở bước thứ 2, ta xác định $\text{path}[2]$

...

Ở bước thứ k, ta xác định $\text{path}[k]$

Cuối cùng là ở bước thứ $k = n-1$, ta xác định $\text{path}[n-1]$ và in ra kết quả :D

Quan trọng: ta để ý

+ Đỉnh 1 có đường đi tới đỉnh 4, tức là $\text{path}[1]$ có đường đi tới $\text{path}[2]$.

+ Đỉnh 4 có đường đi tới đỉnh 3, tức là $\text{path}[2]$ $\text{path}[3]$.

+ Đỉnh 3 có đường đi tới đỉnh 2, tức là $\text{path}[3]$ $\text{path}[4]$.

Ta có công thức

Đỉnh $i = \text{path}[k-1]$

Đỉnh $j = \text{path}[k]$

Đỉnh i đi được đến đỉnh j tức là $\text{path}[k-1]$ đến được $\text{path}[k]$.

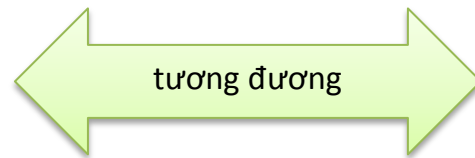
Nói nôm na, ta đang ở bước đệ quy thứ k thì ta phải tìm được đỉnh j phù hợp theo công thức ở trên. Sau đó ta đến bước $k+1$, đến bước $k+2$, bước $k+3$, v.v



(bước này hơi khó hiểu, chịu khó ngâm nghĩ tí nhen)

Đang ở bước thứ k, tìm path[k]

$i = \text{path}[k-1] \rightarrow j = \text{path}[k]$

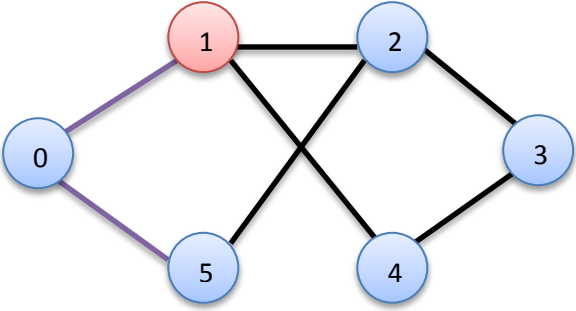
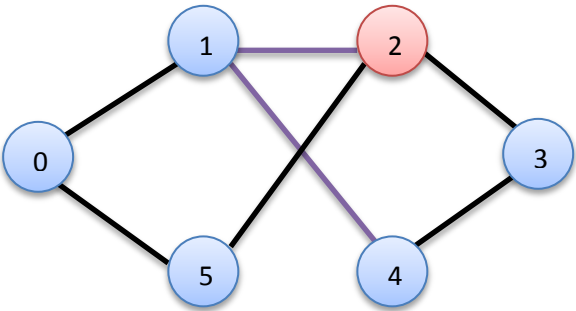
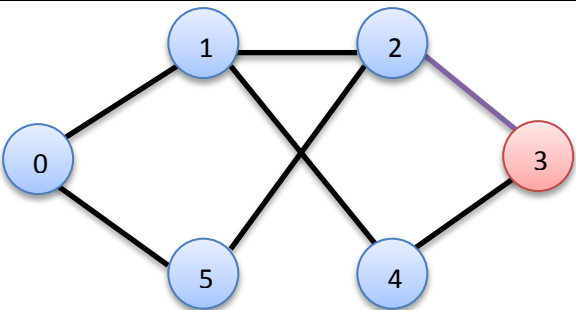


Tìm tất cả đỉnh j sao cho i đến được j ($a[i][j]$ khác 0)

```
int i = path[k-1];  
for (int j = 0; j < n; j++)  
    if (a[i][j])  
    {  
        path[k] = j;  
    }
```

Lấy lại cái đồ thị ở trên.

Bước 0: giả sử ta bắt đầu từ đỉnh 0, $\text{path}[0] = 0$.

	<p>$\text{path} = \{0\}$</p> <p>Bước 1: $i = \text{path}[1-1] = 0$.</p> <p>Đỉnh $i = 0$ tới được đỉnh $j = 1$ và $j = 5$, chọn tùy ý luôn, chọn $j = 1$ thử.</p> <p>Như vậy $\text{path}[1] = 1$</p>
	<p>$\text{path} = \{0, 1\}$</p> <p>Bước 2: $i = \text{path}[2-1] = 1$.</p> <p>Đỉnh $i = 1$ tới được đỉnh $j = 2$ và $j = 4$, vẫn chọn tùy ý, chọn $j = 2$ trước.</p> <p>Như vậy $\text{path}[2] = 2$.</p>
	<p>$\text{path} = \{0, 1, 2\}$</p> <p>Bước 3: $i = \text{path}[3-1] = 2$.</p> <p>Đỉnh $i = 2$ tới được đỉnh $j = 3$ duy nhất.</p> <p>Như vậy $\text{path}[3] = 3$</p>
<p>Bước k: tương tự ☺. Quá trình "mò đường" của ta sẽ kết thúc khi mò hết các đỉnh (kiếm được đường đi Hamilton rồi, vui quá !) hoặc thất bại không tìm được đường đi khác.</p>	

Để minh họa cụ thể hơn, mình ghi đại code ví dụ, áp dụng với ma trận kề, đồ thị vô hướng.



Dĩ nhiên đây chỉ là ví dụ minh họa nên mình chơi luôn mảng toàn cục :3, Ma trận 2 chiều với kích thước $n \times n$.

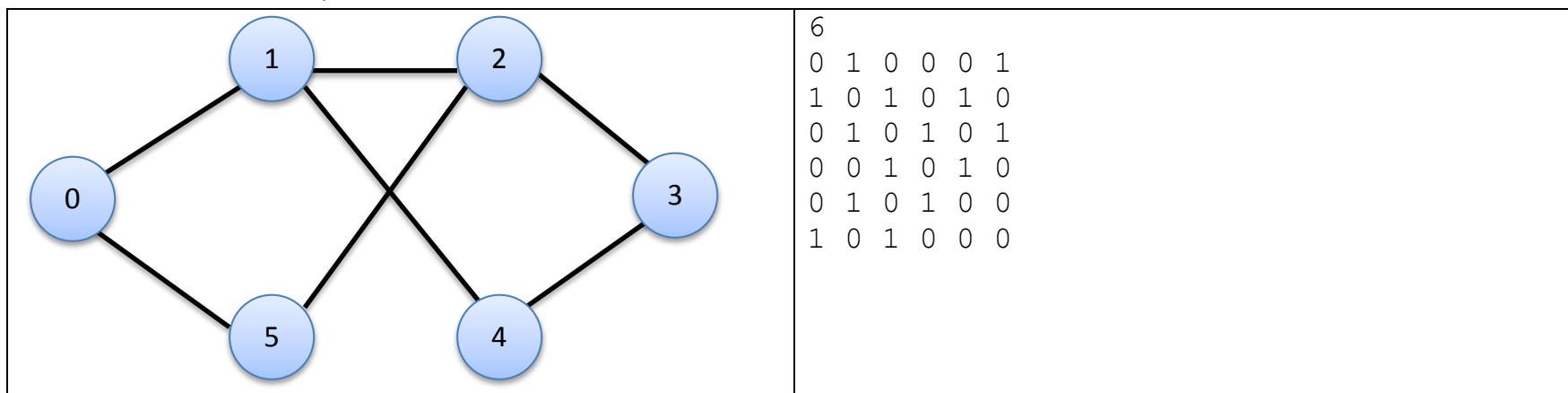
	Code
1	<code>#define MAX_N 20</code>
2	<code>int path [MAX_N] = {0};</code>
3	<code>bool OK [MAX_N] = {true}; // nhớ khởi tạo ban đầu OK[i] = true hết</code>
4	
5	<code>void TRY (int k, int a[MAX_N][MAX_N], int n) // Ở bước k, tìm path[k]</code>
6	<code>{</code>
7	<code> if (k >= n)</code>
8	<code> return; // đi lỗi rồi, dừng lại :3</code>
9	
10	<code> int i = path[k-1]; // lấy node cuối cùng của đường đi</code>
11	
12	<code> for (int j = 0; j < n; j++)</code>
13	<code> if (OK[j] && a[i][j]) // Đỉnh j chưa bị vô hiệu hóa</code>
14	<code> // đồng thời i đến được j thì...</code>
15	<code> {</code>
16	<code> - Lưu đường đi, path[k] = j</code>
17	<code> - Vô hiệu hóa đỉnh j → OK[j] = false</code>
18	<code> - Đi đến bước k+1 để tìm path[k+1].</code>
19	<code> - Trả lại sự tự cho cho đỉnh j → OK[j] = true.</code>
20	<code> }</code>
21	<code>}</code>

Nếu bạn tình ý thì đoạn code này đã mang chút dáng vẻ của "đệ quy quay lui".

Việc khử đệ quy hoặc áp dụng danh sách kề hay kiểu khác có thể code dài hơn và khó hiểu hơn.

OK. Đó là sườn chính. Nói chung code cũng khá là đơn giản, **mình đã code sẵn một demo nho nhỏ**, bạn có thể mở xem nhé :D
Nhắc lại là tất cả chỉ là code minh họa, ngắn, dễ hiểu, cho nên mình sẽ loại bỏ mọi bước trung gian rườm rà. Dĩ nhiên khi code thật sự để nộp bài thì các bạn cần sửa lại ít nhiều rồi.

Ôn lại kỉ niệm xưa, lại lấy cái đồ thị cũ rích kia



Chạy thử demo chương trình, ta sẽ thấy có 3 đường đi Hamilton và 2 chu trình Hamilton :D

3. Kết luận

Nhìn chung, tìm một đường đi / chu trình Hamilton không quá khó nếu ta xài kiểu "mì ăn liền". Hi vọng với đoạn code ngắn ngủn này bạn sẽ dễ hình dung dễ hiểu hơn :D

Mình viết ra tài liệu này trong thời gian khá nhanh, gấp gáp (bận quá), khó tránh khỏi sai sót, không còn kĩ lưỡng như hồi mình học năm 1 nữa. Mong các bạn bỏ qua ☺.