04. CÁC BÀI TOÁN ĐƯỜNG ĐI

Bài giảng Lý thuyết đồ thị

Định nghĩa

- Đường đi (path) (độ dài n) từ u tới v trong một đồ thị vô hướng là một dãy các cạnh e_1, e_2 … e_n của đồ thị sao cho $f(e_1) = \{x_0, x_1\}$, $f(e_2) = \{x_1, x_2\}$… $f(e_n) = \{x_{n-1}, x_n\}$, với $x_0 = u$ và $x_n = v$.
- Khi đồ thị là đơn ta ký hiệu đường đi này bằng dãy các đỉnh x_0, x_1, … x_n.

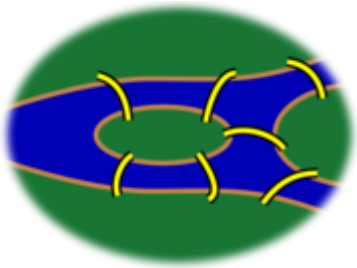
ĐỒ THỊ EULER

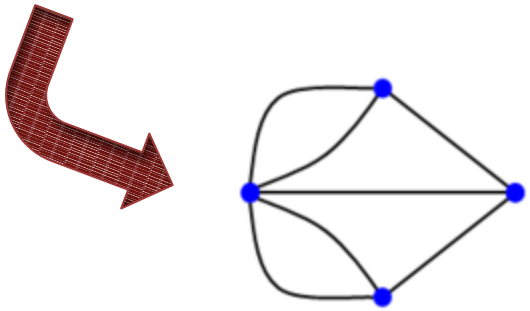
Lịch sử

- Thành phố Königsberg thuộc Phổ (bây giờ gọi là Kaliningrad thuộc Nga), được chia thành 4 vùng bằng các nhánh sông Pregel. Các vùng này gồm hai vùng bên bờ sông, đảo Kneiphof và một miền nằm giữa hai nhánh của sông Pregel. Vào thế kỷ XVIII người ta đã xây 7 chiếc cầu nối các vùng này với nhau.
- Vào chủ nhật, người dân ở đây thường đi bộ dọc theo các phố. Họ tự hỏi không biết có thể xuất phát tại một địa điểm nào đó trong thành phố, đi qua tất cả các cầu, mỗi chiếc cầu không đi qua nhiều hơn một lần, rồi lại trở về điểm xuất phát được không.

Lịch sử

- Leonhard Euler (1707 – 1783)





Euler's Konigsberg's Bridges Problem, 1736

Phát biểu bài toán

- Bài toán tìm đường đi qua tất cả các cầu, mỗi cầu không quá một lần có thể được phát biểu lại bằng mô hình như sau:

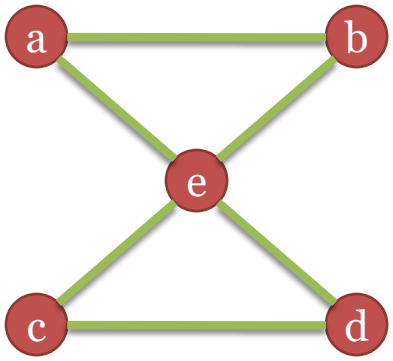
Có tồn tại chu trình đơn
trong đa đồ thị chứa tất cả các cạnh?

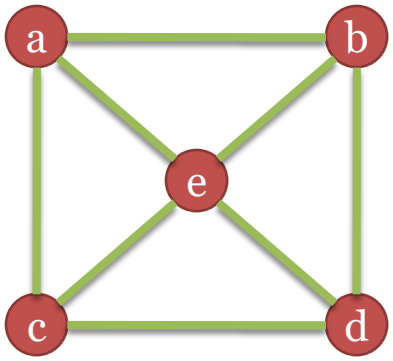
Định nghĩa

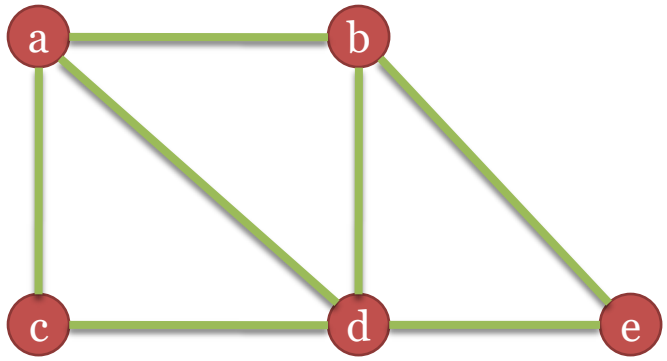
- Chu trình đơn chứa tất cả các cạnh của đồ thị G được gọi là chu trình Euler (Eulerian circuit – Euler circuit).
- Đường đi Euler (Euler path) trong G là đương đi đơn chứa mọi cạnh của G.
- Đồ thị có chu trình Euler được gọi là đồ thị Euler (Euler graph).
- Đồ thị có đường đi Euler được gọi là đồ thị nửa Euler (Semi-euler graph).

Ví dụ

- Đồ thị nào trong 3 hình dưới có chu trình Euler? Nếu không, liệu nó có đường đi Euler hay không?



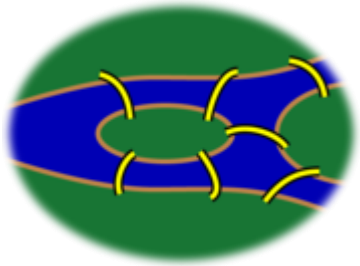


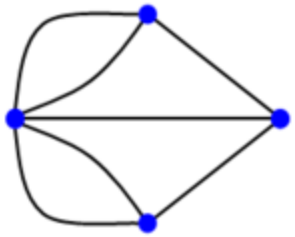


Định lý

- Một đa đồ thị liên thông có chu trình Euler khi và chỉ khi mỗi đỉnh của nó đều có bậc chẵn.

- *Như vậy bài toán 7 cây cầu đã được giải: Do đồ thị có 4 đỉnh bậc lẻ nên không thể có chu trình Euler. Vì vậy không thể bắt đầu từ một địa điểm bất kỳ, đi qua mỗi cây cầu một lần sao cho quay về lại địa điểm xuất phát.*





Hệ quả

- Đồ thị vô hướng liên thông G là nửa Euler khi và chỉ khi nó có không quá 2 đỉnh bậc lẻ.

- Chứng minh:
  - Nếu G không có đỉnh bậc lẻ: theo định lý, hệ quả đúng.
  - Nếu G có 2 đỉnh bậc lẻ u và v. Thêm cạnh {u, v}, G trở nên không có bậc lẻ. Do vậy khi đó G có chu trình Euler. Suy ra trước khi thêm cạnh {u, v}, G phải có đường đi Euler từ u đến v.

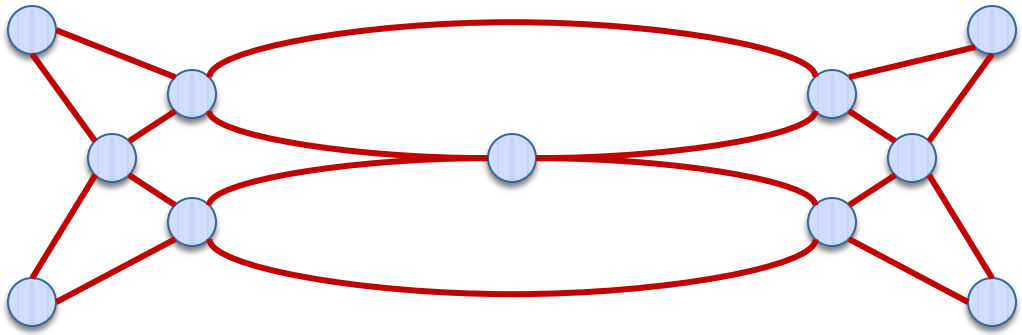
Thuật toán tìm chu trình Euler

- Bước khởi tạo:
  - Tìm chu trình C bất kỳ trong G = (V, E).
  - Loại bỏ khỏi G các cạnh trong chu trình C.
- Bước lặp: Trong khi $E \neq \emptyset$ thực hiện các bước sau:
  - Bước 1. Tìm chu trình C’ trong G sao cho tồn tại 1 đỉnh trong C’ thuộc C.
  - Bước 2. Loại bỏ khỏi G các cạnh trong chu trình C’.
  - Bước 3. Chèn vào C’ chu trình C ở vị trí thích hợp (vị trí đỉnh chung).

Kết thúc thuật toán ta có C chính là chu trình Euler cần tìm

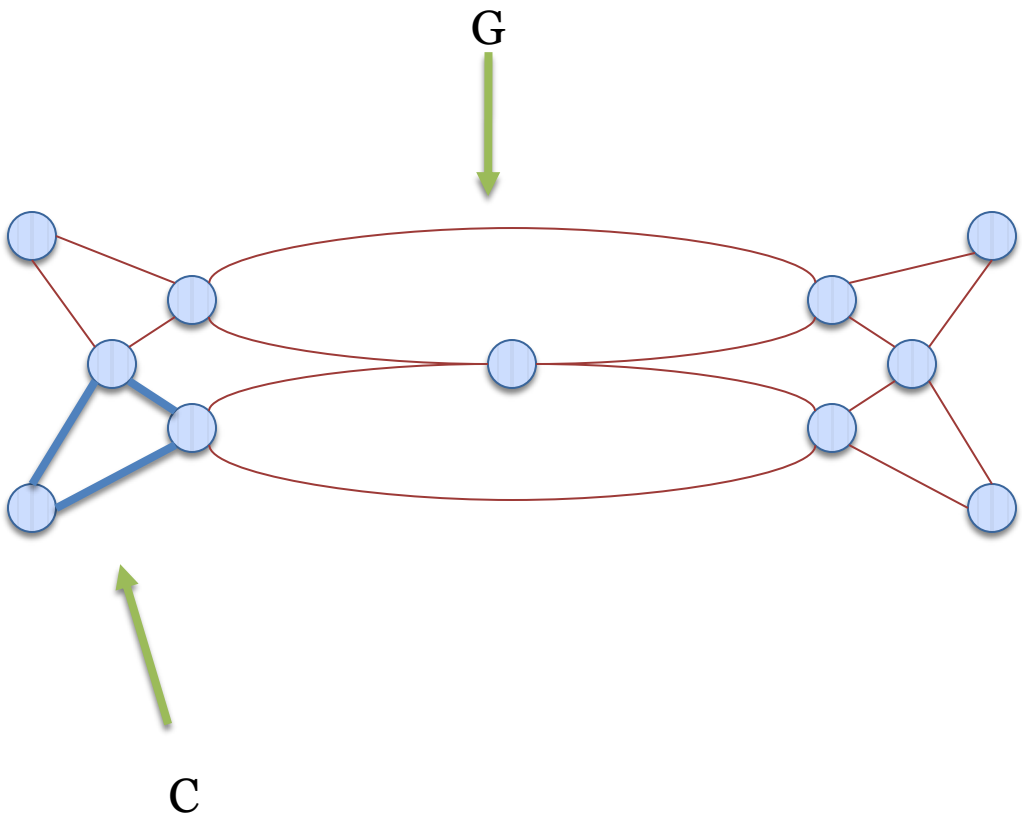
Ví dụ

- Có thể vẽ *thanh mã tấu của Mohammed* xuất phát và kết thúc tại một điểm bằng cách vẽ liên tục và không nhất bút lên được không? Vẽ như thế nào?



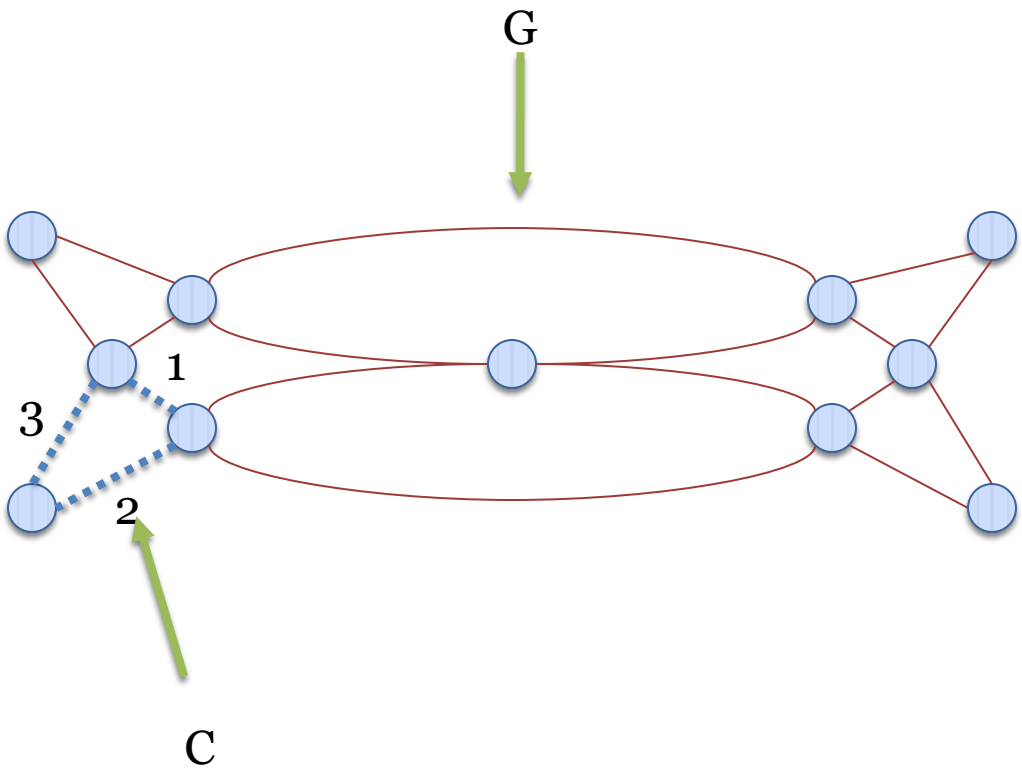
Ví dụ

- Tìm một chu trình C bất kỳ.



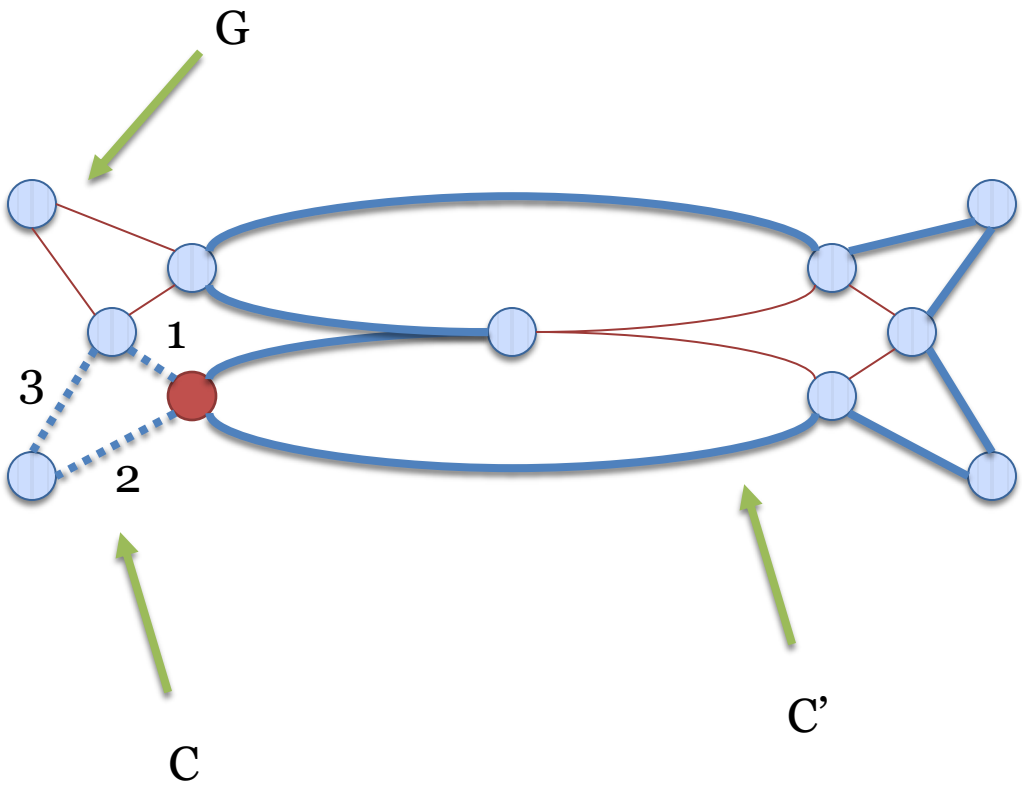
Ví dụ

- Xóa chu trình C đã tìm được ra khỏi G.



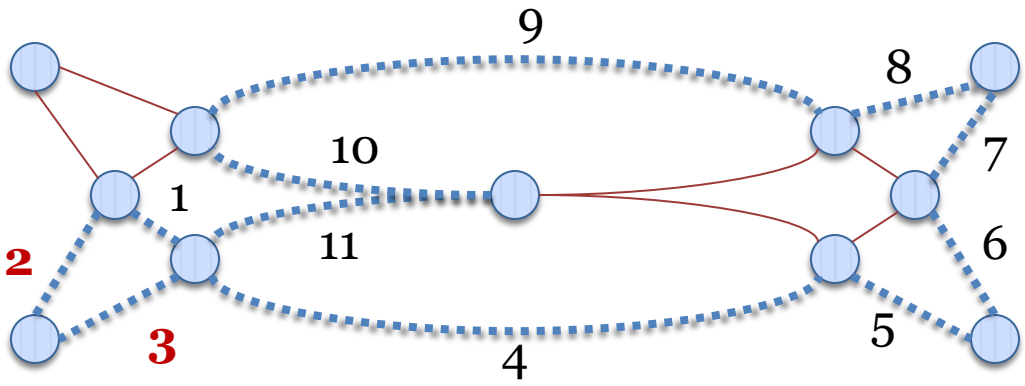
Ví dụ

- Tìm chu trình C’ trên G có đỉnh trên chu trình C đã tìm được.



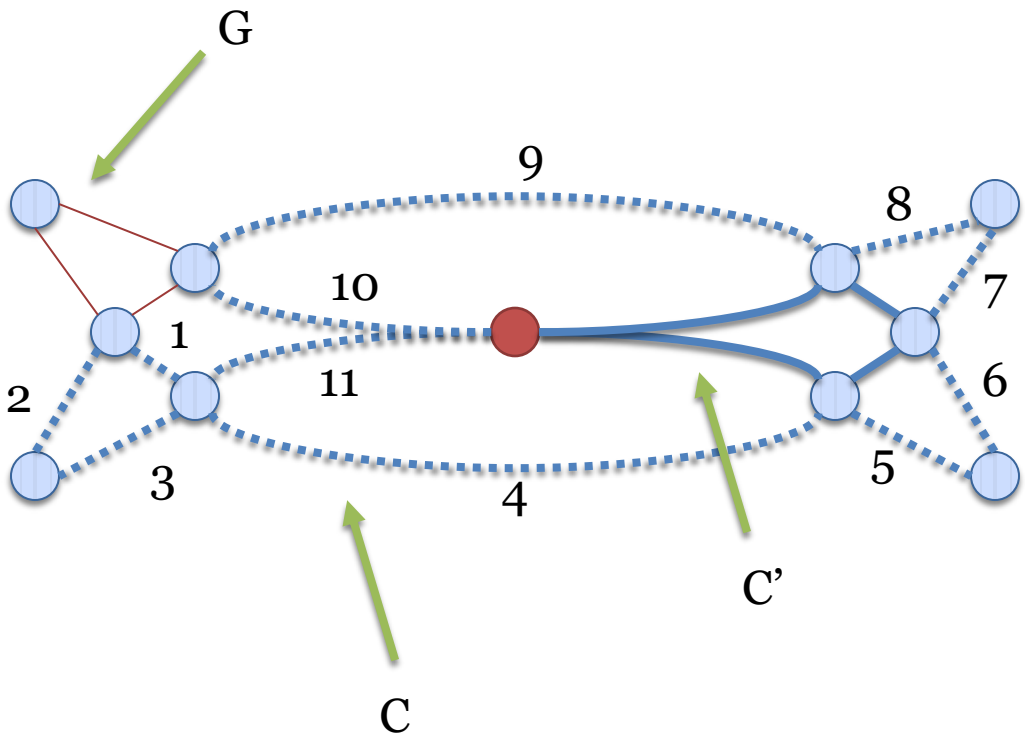
Ví dụ

- Thêm chu trình C’ vừa tìm được vào C. Xóa C’ ra khỏi G.



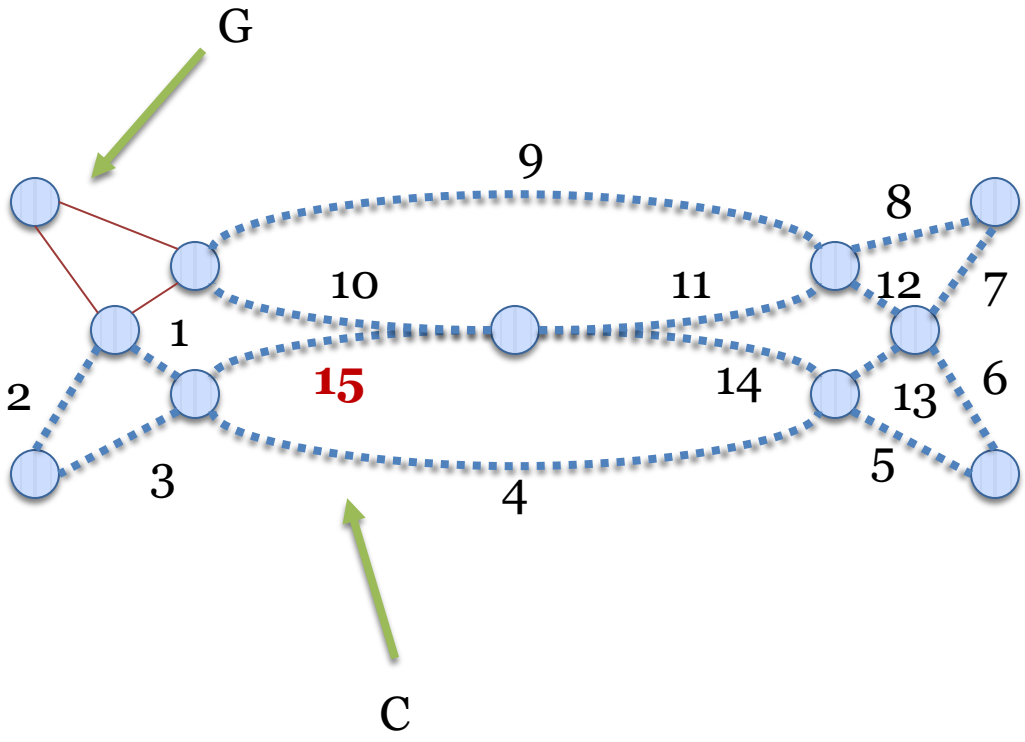
Ví dụ

- Tìm chu trình C’ trên G có đỉnh trên chu trình C đã tìm được.



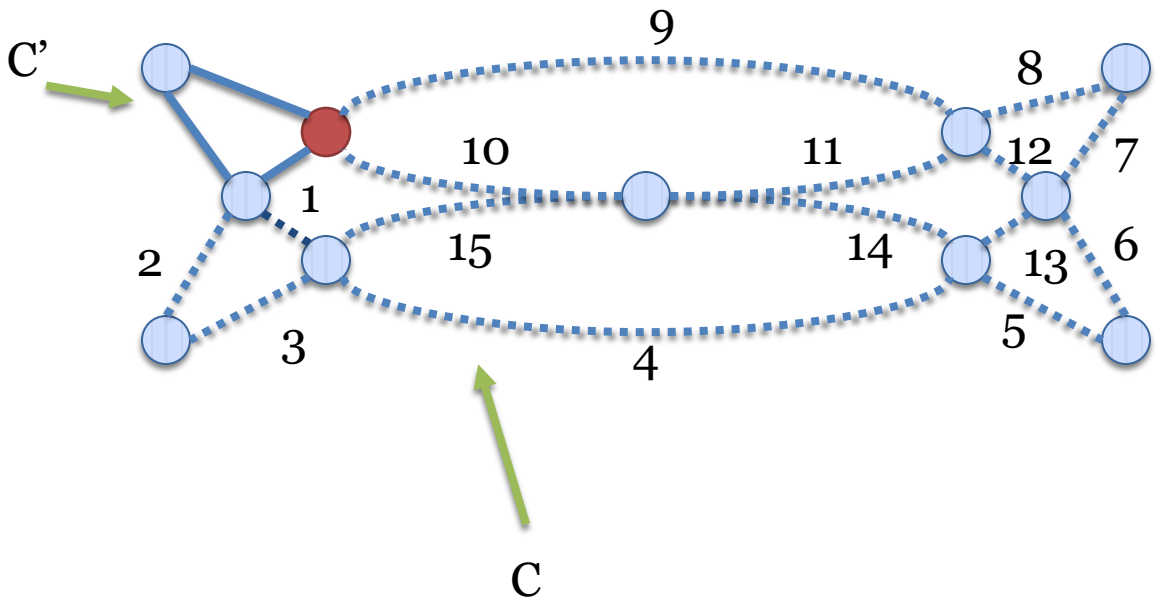
Ví dụ

- Thêm chu trình C’ vừa tìm được vào C. Xóa C’ ra khỏi G.



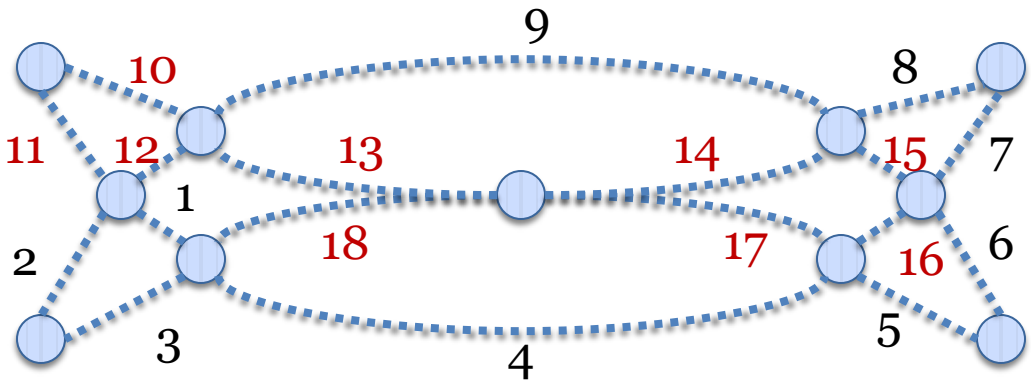
Ví dụ

- Tìm chu trình C’ trên G có đỉnh trên chu trình C đã tìm được.



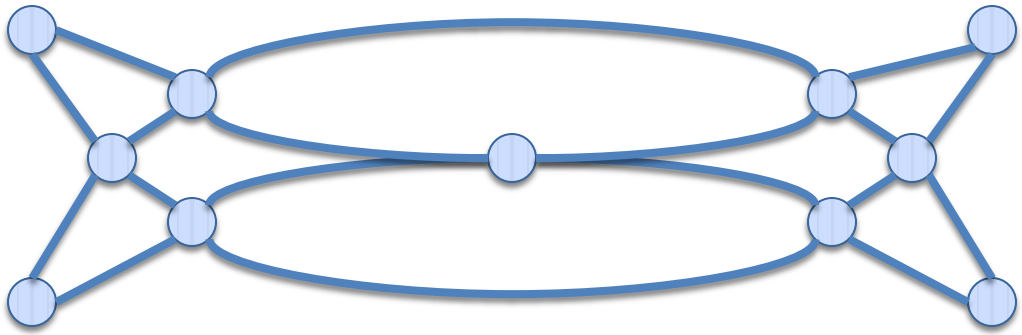
Ví dụ

- Thêm chu trình C’ vừa tìm được vào C. Xóa C’ ra khỏi G.



Ví dụ

- Vậy ta có thể vẽ thanh mã tấu Mohammed liên tục bằng một nét và xuất phát/kết thúc cùng một điểm.

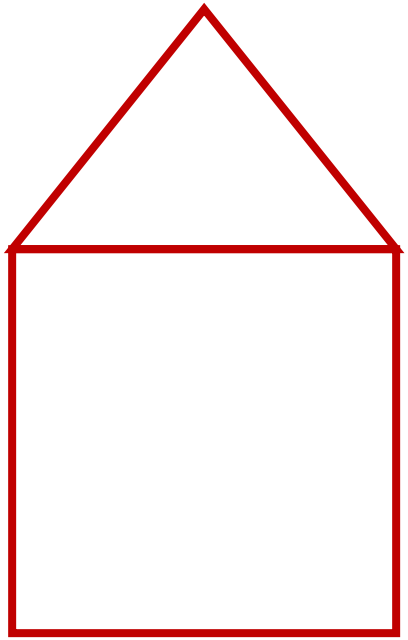


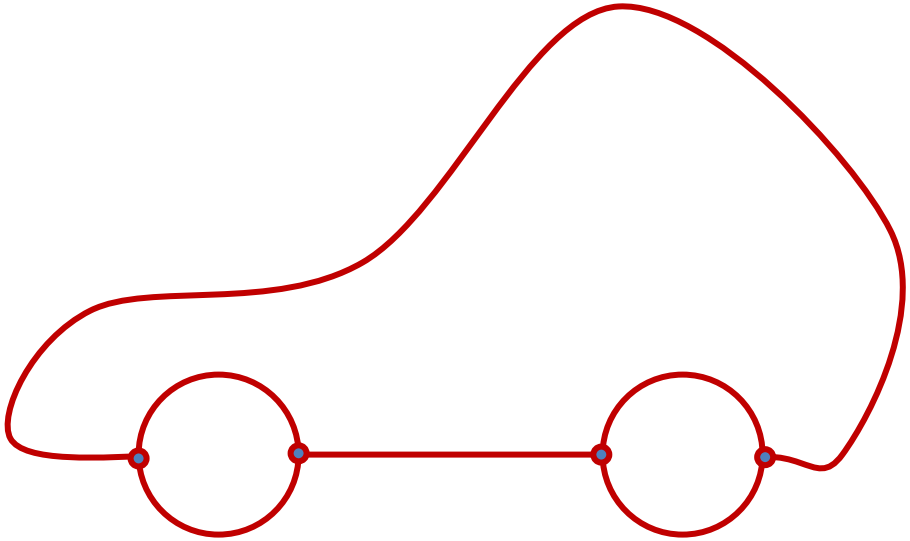
Định lý

- Đa đồ thị liên thông có đường đi Euler nhưng không có chu trình Euler nếu và chỉ nếu nó có đúng 2 đỉnh bậc lẻ.

Ví dụ

- Có thể vẽ các hình sau bằng một nét được không?

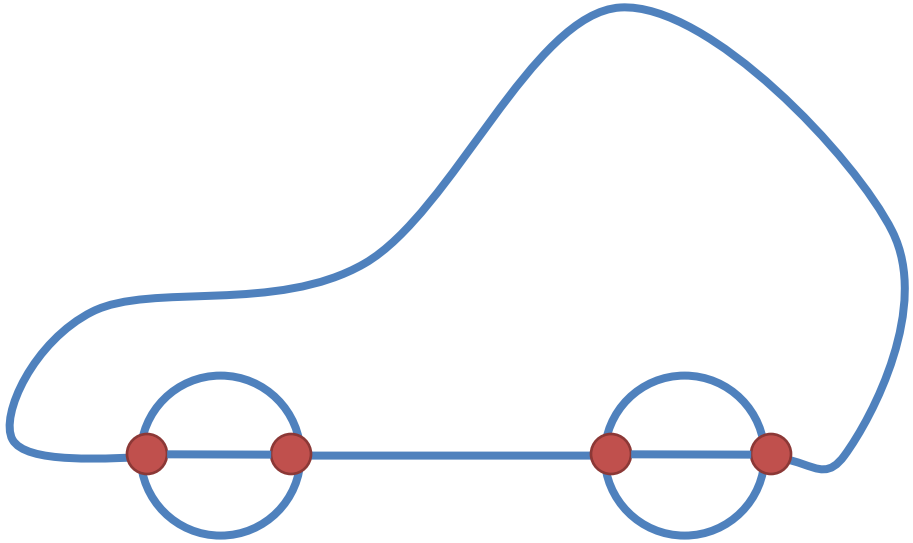




Thuật toán Fleury tìm chu trình Euler

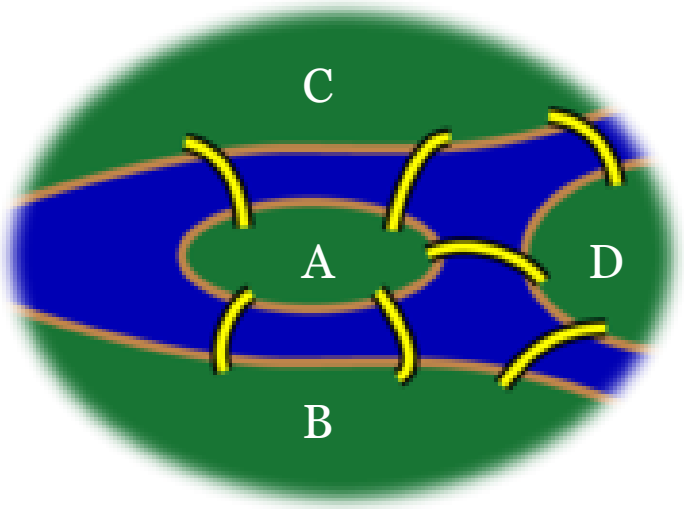
- Bước 1. Chọn một đỉnh u tùy ý để bắt đầu.
- Bước 2. Chọn một cạnh để đi tiếp, chỉ chọn cạnh cầu khi nào không còn lựa chọn nào khác.
- Bước 3. Đánh dấu cạnh đã đi qua cho biết ta không thể quay lại cạnh đó.
- Bước 4. Đi theo cạnh đó đến đỉnh tiếp theo.
- Bước 5. Lặp lại bước 2 cho đến khi nào mọi cạnh đều đã được duyệt.

Ví dụ



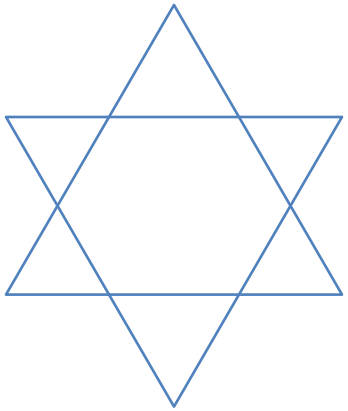
Bài tập

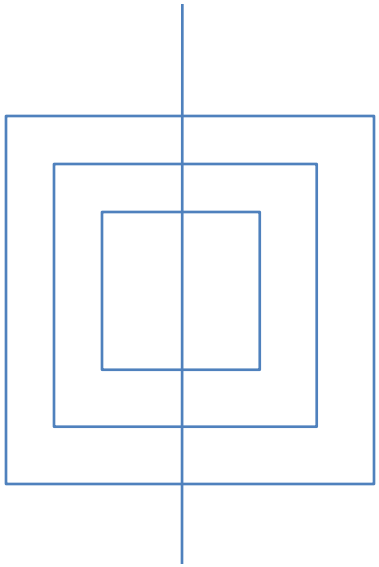
1. Ngoài 7 chiếc cầu đã được xây dựng từ thế kỷ XVIII ở Kaliningrad, người ta xây thêm 2 chiếc nữa nối khu B với khu C và khu B với khu D. Một người nào đó có thể đi qua 9 chiếc cầu, mỗi chiếc đi qua đúng một lần và trở về nơi xuất phát được không?

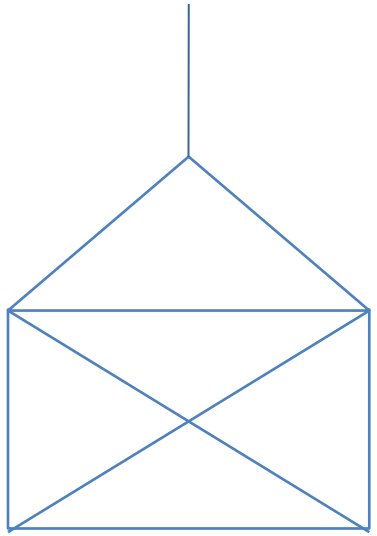


Bài tập

2. Xác định xem các hình sau đây có thể vẽ bằng một nét hay không.







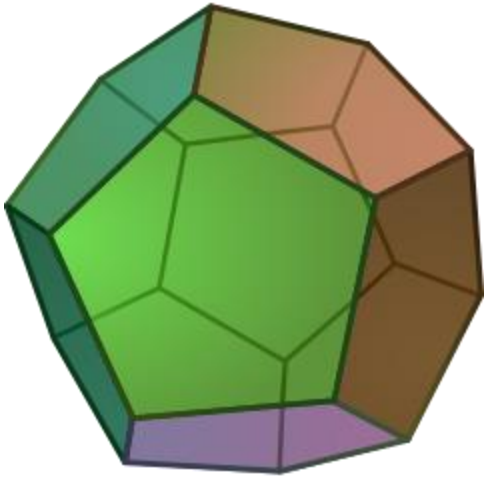
Bài tập

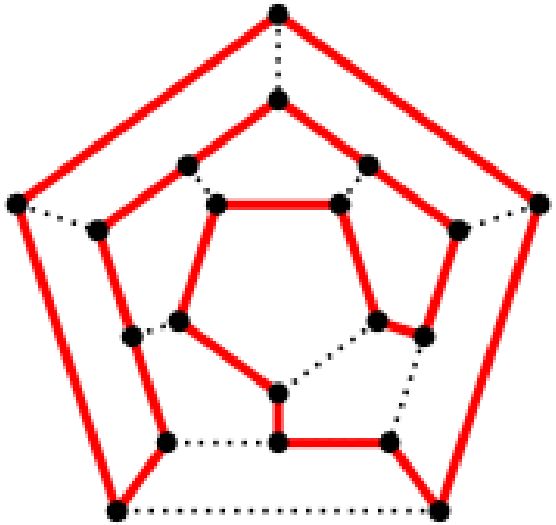
3. Với giá trị nào của m, n các đồ thị sau đây có chu trình Euler? Đường đi Euler?
   a) K_n
   b) C_n
   c) $K_{m, n}$

ĐỒ THỊ HAMILTON

Lịch sử

- Thuật ngữ này xuất phát từ trò đố vui do William Rowan Hamilton, nhà toán học người Ailen đưa ra vào năm 1857.
- Giả sử có một khối thập nhị diện (dodecahedron), mỗi mặt là một ngũ giác đều. Mỗi đỉnh trong 20 đỉnh khối này được đặt tên một thành phố. Hãy tìm một đường xuất phát từ một thành phố, đi dọc theo cạnh của khối, ghé thăm mỗi một trong 19 thành phố còn lại đúng một lần, cuối cùng trở về thành phố ban đầu.





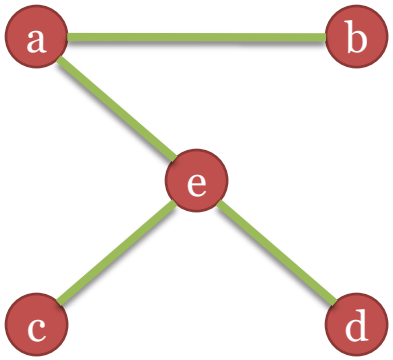
W. R. Hamilton
(1805 – 1865)

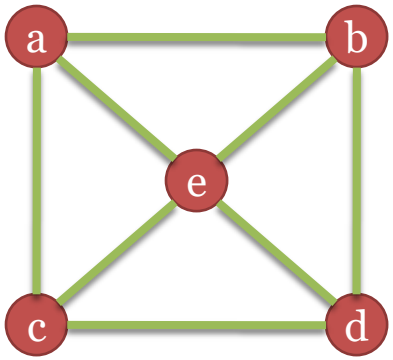
Định nghĩa

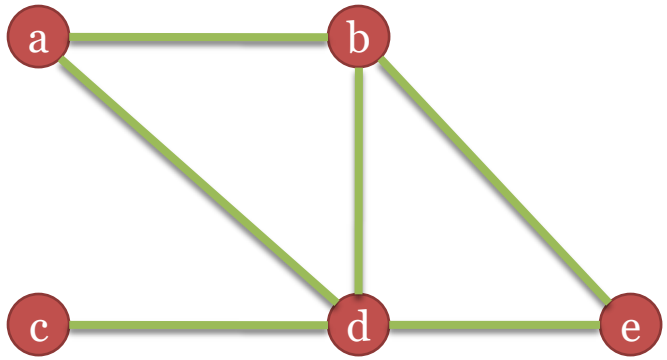
- Đường đi $x_0, x_1 \ldots x_{n-1}, x_n$ trong đồ thị G = (V, E) được gọi là đường đi Hamilton (Hamiltonia path, Hamilton path) nếu $V = \{x_0, x_1, \ldots x_{n-1}, x_n\}$ và $x_i \neq x_j$, với $0 \leq i < j \leq n$.
- Chu trình $x_0, x_1, \ldots x_{n-1}, x_n, x_0$ $(n > 1)$ trong đồ thị G = (V, E) được gọi là chu trình Hamilton (Hamilton circuit) nếu $x_0, x_1, \ldots x_{n-1}, x_n$ là đường đi Hamilton.
- Đồ thị được gọi là đồ thị Hamilton (Hamilton graph) nếu nó chứa chu trình Hamilton và gọi là đồ thị nửa Hamilton (Semi-hamilton graph) nếu nó chứa đường đi Hamilton.

Ví dụ

- Đồ thị nào trong số các đồ thị sau đây là đồ thị Hamilton? Là nửa Hamilton?







Nhận biết đồ thị Hamilton

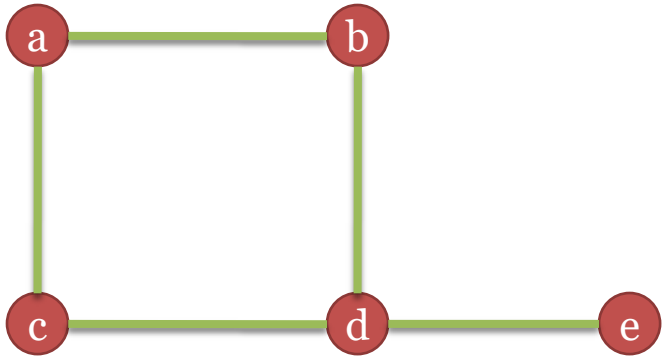
- Cho đến nay, người ta vẫn chưa tìm ra được điều kiện cần và đủ để tồn tại chu trình Hamilton.
- Các kết quả thu được phần lớn là các điều kiện đủ để một đồ thị là đồ thị Hamilton. Phần lớn chúng đều có dạng *"nếu G có số cạnh đủ lớn thì G là Hamilton"*

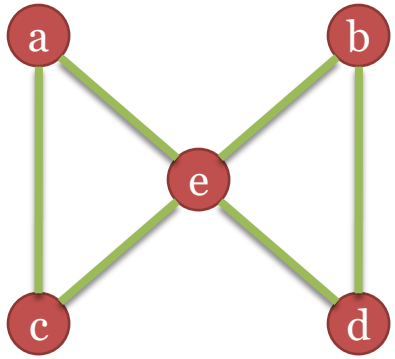
Nhận biết đồ thị không là Hamilton

- Một số tính chất sau đây có thể dùng để nhận biết một đồ thị không là Hamilton:
  - Tính chất 1: Đồ thị có đỉnh treo không thể có chu trình Hamilton. *(thật vậy, vì trong chu trình Hamilton mỗi đỉnh đều gắn với 2 cạnh trong chu trình).*
  - Tính chất 2: Nếu một đỉnh có bậc 2 thì cả 2 cạnh liên thuộc với đỉnh này phải là một phần của chu trình Hamilton.
- Lưu ý: Chu trình Hamilton không thể chứa chu trình nhỏ hơn trong nó.

Ví dụ

- Hãy chỉ ra rằng các đồ thị sau không là đồ thị Hamilton





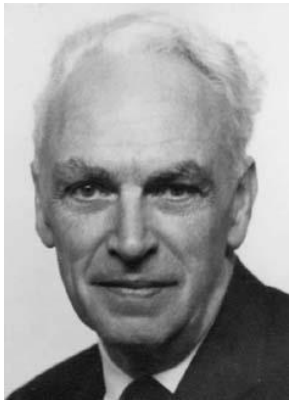
Một số định lý

- Định lý Dirac (1952):
  Đơn đồ thị vô hướng G = (V, E) với n đỉnh (n ≥ 3) có chu trình Hamilton nếu

$$\deg(v) \geq n/2, \forall v \in V$$

- Định lý Ore (1960):
  Đồ thị vô hướng G = (V, E) với |V| ≥ 3 có chu trình Hamilton khi

$$\forall (u, v) \notin E \rightarrow \deg(u) + \deg(v) \geq n$$



Ghi chú:
•Øystein Ore (1899 – 1968) là nhà toán học người Nauy.
•Gabriel Andrew Dirac (1925 – 1985) là nhà toán học người Thụy Sĩ, ông là con trước của nhà toán học Paul Dirac (1902 – 1984).

Lưu ý

- Cả hai định lý Ore và Dirac đều chỉ là điều kiện đủ để một đơn đồ thị liên thông có tồn tại chu trình Hamilton. Tuy nhiên, chúng không phải là điều kiện cần.
- Ví dụ: đồ thị C_5 có chu trình Hamilton nhưng không thỏa mãn các giả định của định lý Ore hay định lý Dirac.

Chứng minh định lý Dirac

- Chứng minh này dựa trên định lý Ore.

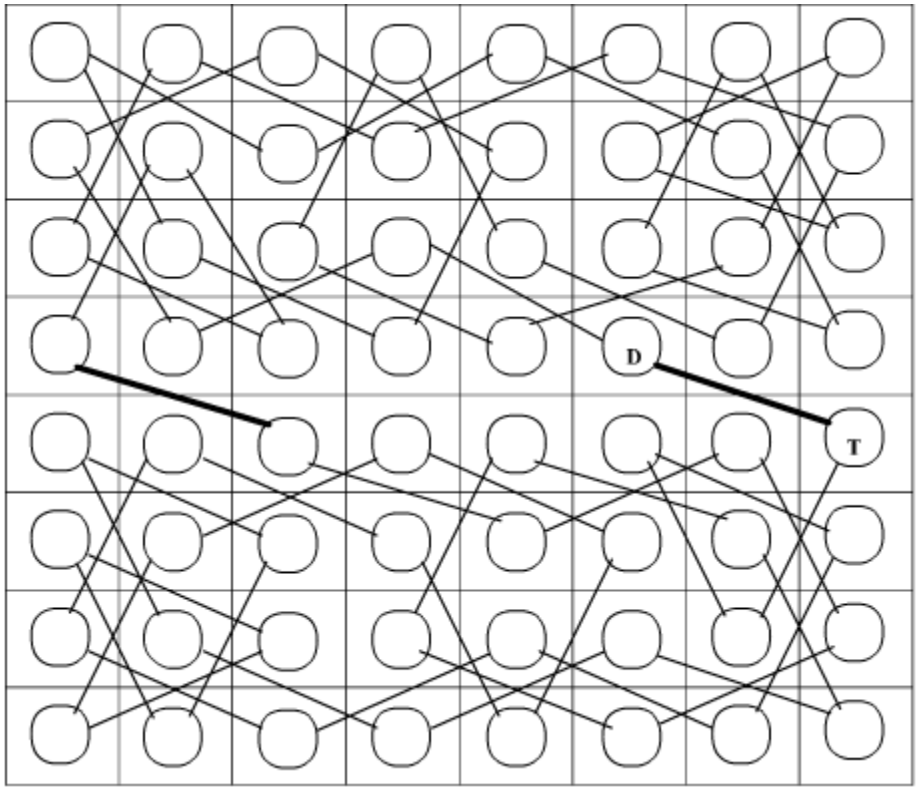
- *Chứng minh:*
  - Với mọi cặp đỉnh u, v $\in$ V, không quan tâm có tồn tại cạnh (u, v) $\in$ E hay không nếu ta đều có:

    $$\deg(u) + \deg(v) \geq n/2 + n/2 = n$$

  - Do đó, theo định lý Ore, đồ thị có chứa chu trình Hamilton [đpcm].

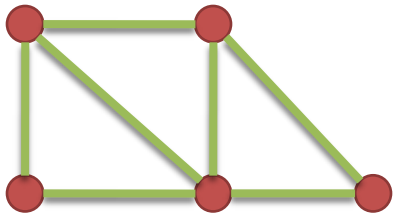
Ứng dụng chu trình Hamilton

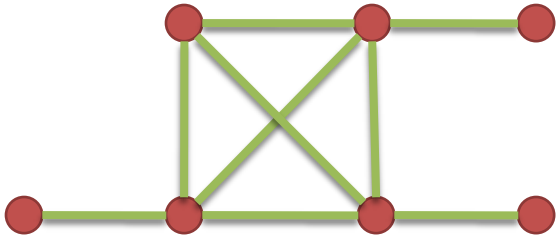
- Bài toán Mã đi tuần (knight tour): Cho bàn cờ vua 8x8. Đặt một con mã ở vị trí bất kỳ (x, y). Hãy tìm lộ trình sao cho con mã đi qua hết các ô của bàn cờ, mỗi ô đúng một lần.

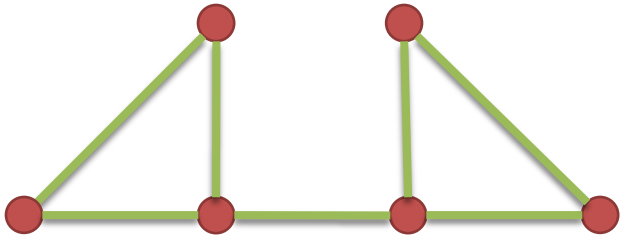


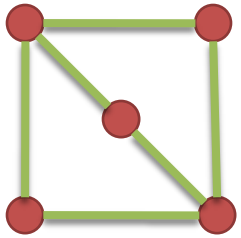
Bài tập

1. Xác định xem các đồ thị sau đây có là đồ thị Hamilton hay không? Nếu có, hãy chỉ ra, nếu không hãy giải thích. Câu hỏi tương tự với đồ thị là nửa Hamilton.









Bài tập

3. Với các giá trị nào của n, đồ thị sau đây có đường đi Hamilton
   a) K_n
   b) C_n

Bài tập

4. Với giá trị nào của m và n đồ thị phân đôi đầy đủ $K_{m,n}$ có chu trình Hamilton.

Bài tập

5. Đồ thị có một đỉnh kề với ba đỉnh bậc 2 thì có chu trình Hamilton hay không? Giải thích và cho ví dụ minh họa, nếu có.

BÀI TOÁN
ĐƯỜNG ĐI NGẮN NHẤT

Phát biểu bài toán

- ***Định nghĩa đường đi có trọng số:***

  Cho đồ thị G = (V, E) là đồ thị có trọng số và trọng số mỗi cạnh e là w(e). Với G' là một đồ thị con của G thì trọng số của G' được định nghĩa là:

$$w(G') = \sum_{e \in G'} w(e)$$

  - Nếu G' là đường đi hay chu trình thì w(G') gọi là độ dài của G'.
  - Nếu G' là một mạch (chu trình có các đỉnh không lặp lại) và w(G') < 0 thì ta goị G' là *mạch âm*.

Phát biểu bài toán

- Cho đồ thị G = (V, E) là một đồ thị có trọng số và s, t $\in$ V. Gọi P là tập hợp tất cả các đường đi từ s tới t. Xét bài toán:

  Tìm $p_0 \in P$ sao cho $p_0 = \min\{w(p): p \in P\}$

- Bài toán này gọi là bài toán đường đi ngắn nhất (shortest path problem) và p_0 gọi là đường đi ngắn nhất (shortest path) từ s đến t.

Các nhận xét

1. Mặc dù bài toán được phát biểu cho đồ thị có hướng có trọng, nhưng các thuật toán sẽ trình bày đều có thể áp dụng cho các đồ thị vô hướng có trọng bằng cách xem mỗi cạnh của đồ thị vô hướng như hai cạnh có cùng trọng lượng nối cùng một cặp đỉnh nhưng có chiều ngược nhau.
2. Khi làm bài toán tìm đường đi ngắn nhất thì chúng ta có thể bỏ bớt đi các cạnh song song và chỉ chừa lại một cạnh có trọng lượng nhỏ nhất trong số các cạnh song song.

Các nhận xét

3. Đối với các khuyên có trọng lượng không âm thì cũng có thể bỏ đi mà không làm ảnh hưởng đến kết quả của bài toán. Đối với các khuyên có trọng lượng âm thì có thể đưa đến bài toán đường đi ngắn nhất không có lời giải.
4. Nếu G có mạch âm q trên đường đi từ u đến v thì đường đi ngắn nhất từ u đến v không tồn tại. Nhận xét số 4 này thực chất là mở rộng của nhận xét 3.

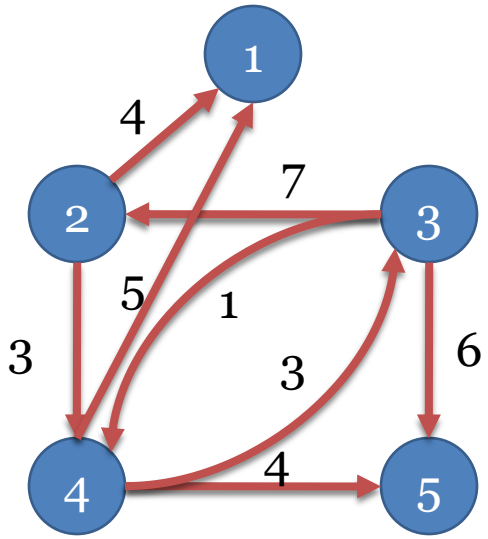
Ma trận khoảng cách

- Từ các nhận xét vừa nêu, có thể xem dữ liệu nhập của bài toán đường đi ngắn nhất là ma trận khoảng cách (distance matrix) D được định nghĩa như sau:
  - Ma trận khoảng cách của G là ma trận D = (D_{ij}) với

$$D_{ij} = \begin{cases} 0 & i = j \\ w(i,j) & (i,j) \in E \\ +\infty & (i,j) \notin E \end{cases}$$

  - Ma trận khoảng cách còn được gọi là ma trận trọng lượng.

Ví dụ về ma trận khoảng cách



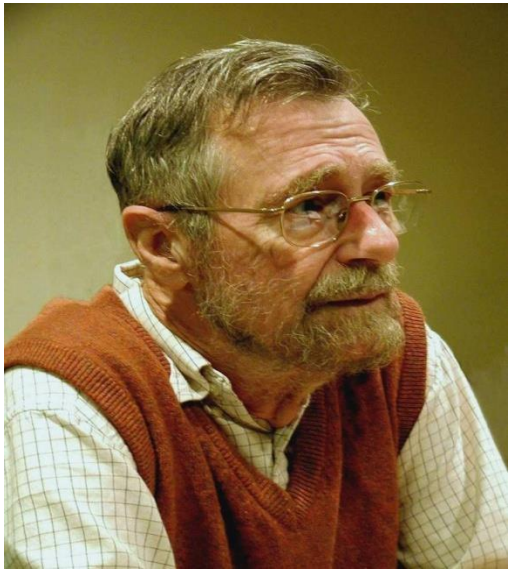
	1	2	3	4	5
1	0	+∞	+∞	+∞	+∞
2	4	0	+∞	3	+∞
3	+∞	7	0	1	6
4	5	+∞	3	0	4
5	+∞	+∞	+∞	+∞	0

BÀI TOÁN ĐƯỜNG ĐI NGẮN NHẤT

THUẬT TOÁN DIJKSTRA

Thuật toán Dijkstra

- **Edsger Wybe Dijkstra** /dɛɪkstra/ là một nhà toán học người Hà Lan. Ông đưa ra thuật toán tìm đường đi ngắn nhất – thuật toán mang tên ông – vào năm 1959.



Dijkstra (1930 – 2002)

Thuật toán Dijkstra – Dữ liệu vào/ra

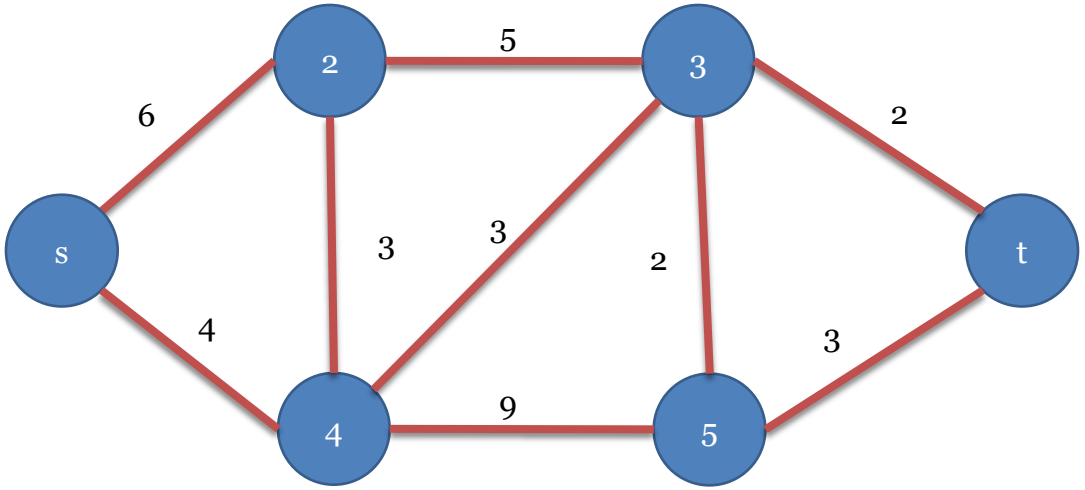
- Xét đồ thị G = (V, E) có trọng giả sử không âm.
  - Dữ liệu nhập cho thuật toán là ma trận trọng lượng D và hai đỉnh u, v cho trước.
  - Dữ liệu xuất là đường đi ngắn nhất từ u đến v.

Thuật toán Dijkstra

- **Bước 1.** Gán T = V và gán các nhãn:
  L[u] = 0; L[k] = $+\infty$, $\forall k \in V \backslash \{u\}$;
  Prev[k] = -1, $\forall k \in V$.
- **Bước 2.** Nếu v $\notin$ T thì dừng và giá trị L[v] chính là độ dài đường đi ngắn nhất từ u đến v và Prev[v] là đỉnh nằm ngay trước v trên đường đi đó.
- **Bước 3.** Chọn đỉnh i $\in$ T sao cho L[i] nhỏ nhất và gán T = T\{i}.
- **Bước 4.**
  - Với $\forall k \in T$ và từ đỉnh i (ở bước 3) đến đỉnh k có cạnh nối:
    nếu $L[k] > L[i] + D_{ik}$ thì
    Gán $L[k] = L[i] + D_{ik}$ và $Prev[k] = i$
  - Trở về bước 2.

Ví dụ Dijkstra

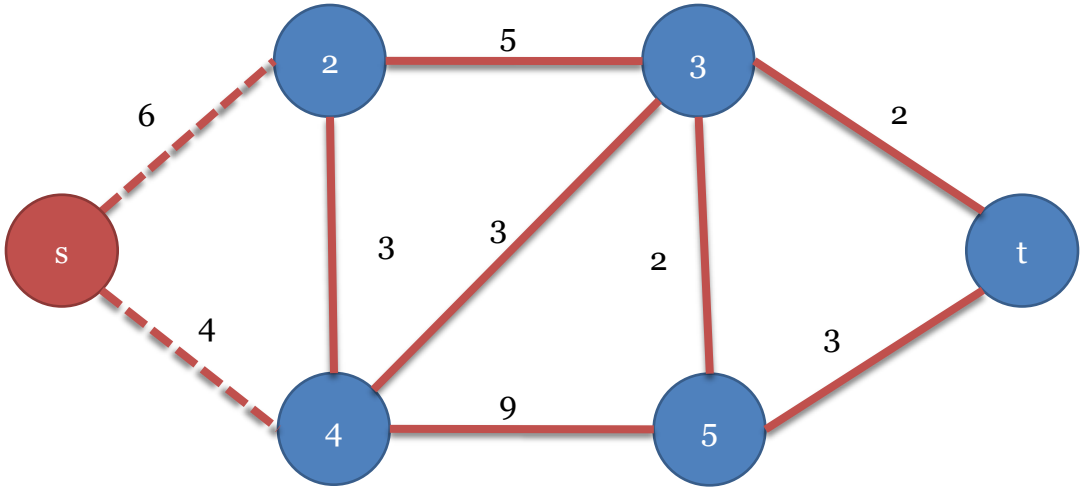
- Ví dụ 1



	s	2	3	4	5	t
T	s	2	3	4	5	t
L	0	+∞	+∞	+∞	+∞	+∞
Prev	-1	-1	-1	-1	-1	-1

Ví dụ Dijkstra

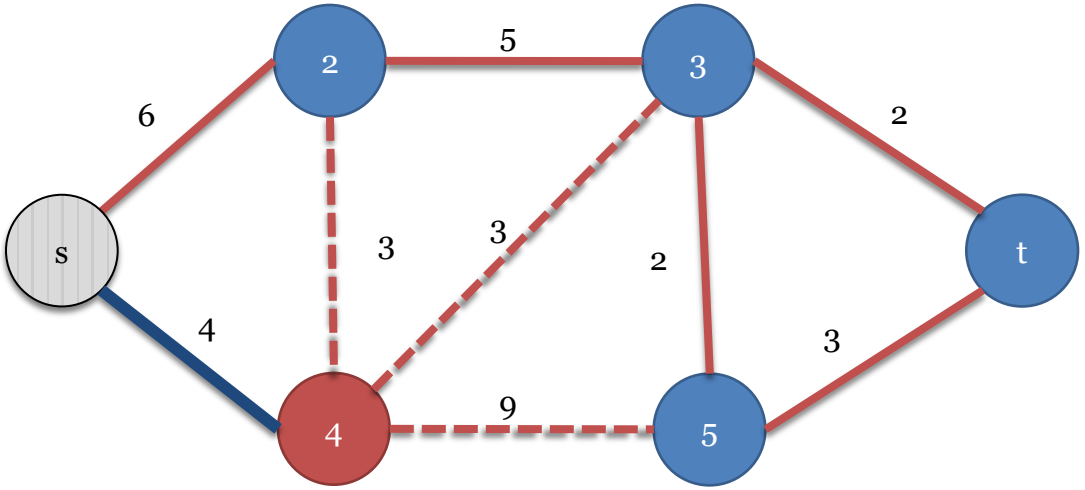
- Ví dụ 1



	s	2	3	4	5	t
T		2	3	4	5	t
L	0	6	+∞	4	+∞	+∞
Prev	-1	s	-1	s	-1	-1

Ví dụ Dijkstra

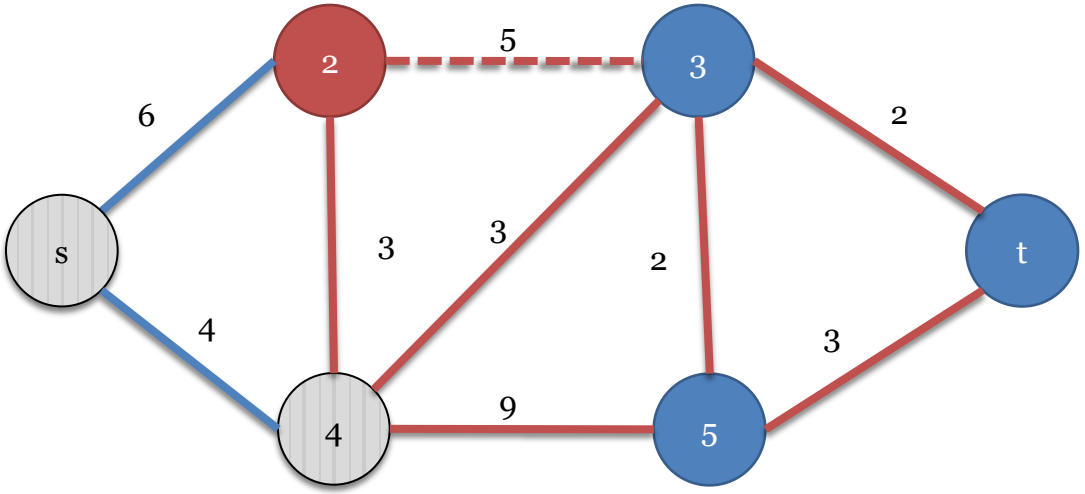
- Ví dụ 1



	s	2	3	4	5	t
T		2	3		5	t
L	0	6	7	4	13	+∞
Prev	-1	s	4	s	4	-1

Ví dụ Dijkstra

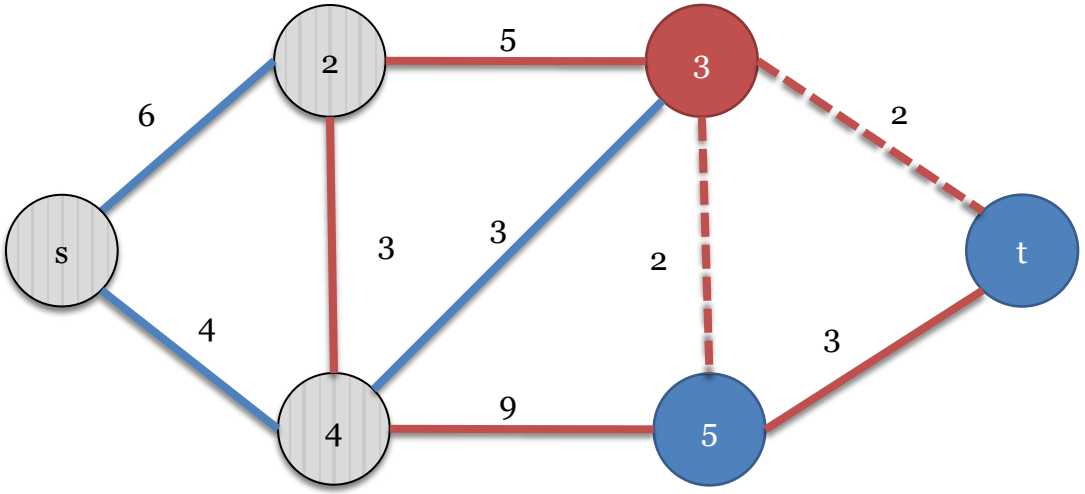
- Ví dụ 1



	s	2	3	4	5	t
T			3		5	t
L	0	6	7	4	13	+∞
Prev	-1	s	4	s	4	-1

Ví dụ Dijkstra

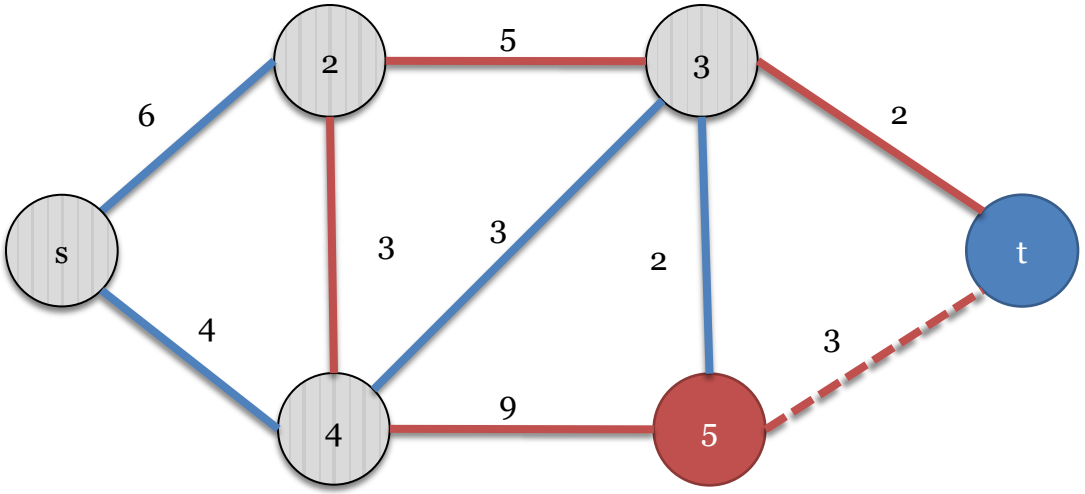
- Ví dụ 1



	s	2	3	4	5	t
T					5	t
L	0	6	7	4	9	9
Prev	-1	s	4	s	3	3

Ví dụ Dijkstra

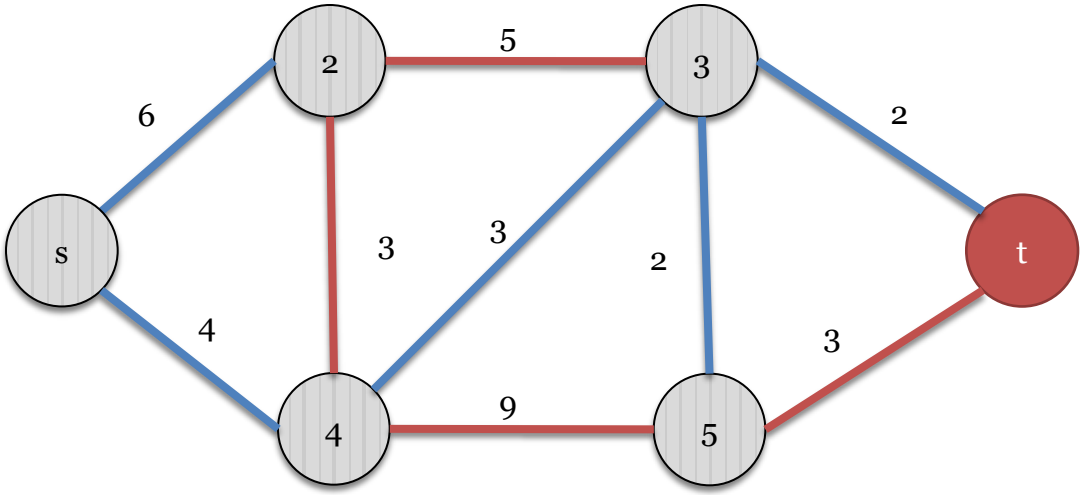
- Ví dụ 1



	s	2	3	4	5	t
T						t
L	0	6	7	4	9	9
Prev	-1	s	4	s	3	3

Ví dụ Dijkstra

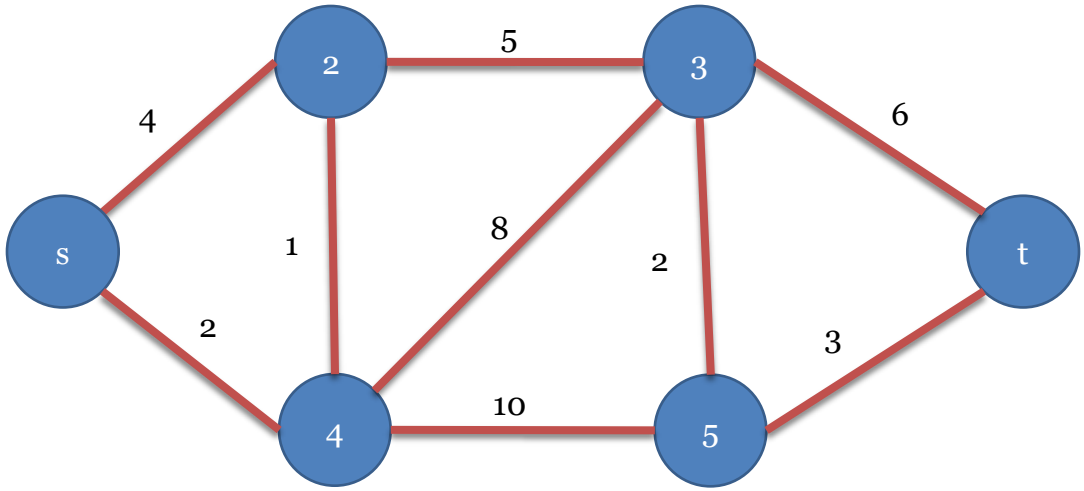
- Ví dụ 1



	s	2	3	4	5	t
T						
L	0	6	7	4	9	**9**
Prev	-1	s	4	s	3	3

Bài tập

- Tìm đường đi ngắn nhất bằng thuật toán Dijkstra từ đỉnh a đến đỉnh z cho đồ thị sau:



BÀI TOÁN ĐƯỜNG ĐI NGẮN NHẤT

THUẬT TOÁN BELLMAN

Lịch sử

- Thuật toán Bellman hay còn gọi là thuật toán Bellman – Ford do hai tác giả Bellman và Ford (Lester Randolph Ford, (1886 –1967)) đưa ra.

Thuật toán Bellman – Dữ liệu vào/ra

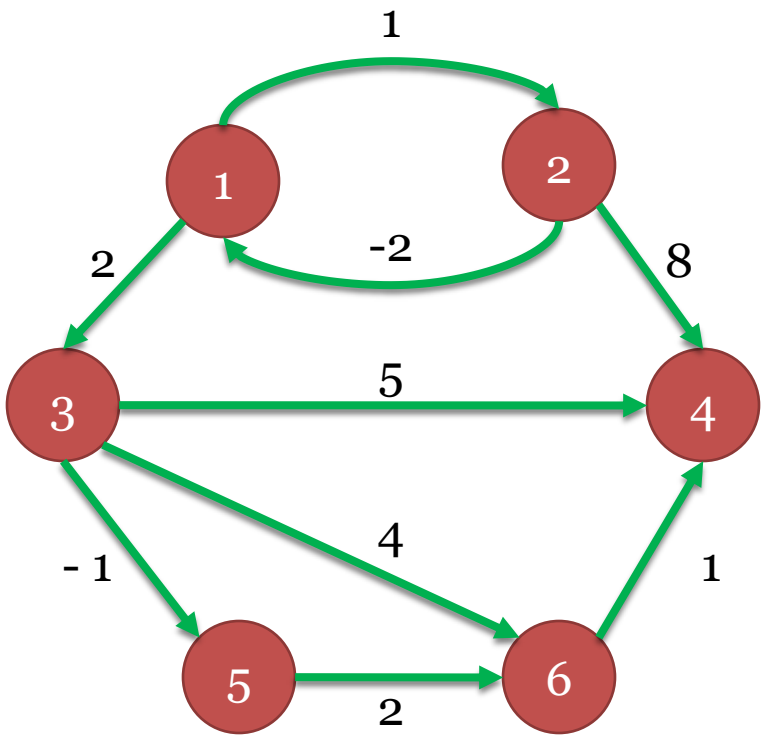
- Thuật toán Bellman có thể tìm được đường đi ngắn nhất trên đồ thị có trọng số âm.
  - Dữ liệu nhập cho thuật toán là ma trận trọng lượng D và đỉnh bắt đầu x cho trước.
- Thuật toán này tìm đường đi ngắn nhất từ một đỉnh cho trước của đồ thị đến mỗi đỉnh khác *nếu đường đi không đến được mạch âm*.
  - Nếu phát hiện đồ thị có mạch âm thì thuật toán dừng.

Thuật toán Bellman

- Cho trước đỉnh $x \in V$.
- Bước 1. Khởi tạo:
  - $\pi(0, x) = 0$; $\pi(0, i) = +\infty$, $\forall i \neq x$; $k = 1$.
  - $Prev(i) = i$, $\forall i \in V$
- Bước 2. Với mỗi $i \in V$ ta cập nhật:
  - $\pi(k, i) = min(\{\pi(k-1, i)\} \cup \{\pi(k-1, j) + D_{ji}\})$
  - *Nếu* $\pi(k, i) = \pi(k-1, j) + D_{ji}$ *:* $Prev(i) = j$
- Bước 3.
  - Nếu $\pi(k, i) = \pi(k-1, i)$ với $\forall i \in V$ thì $\pi(k, i)$ chính là độ dài đường đi ngắn nhất từ x đến i.
  - Ngược lại
    - Nếu $k < n$ thì tăng $k = k+1$ và trở lại bước 2;
    - Ngược lại thì dừng vì từ x đi tới được một mạch âm.

Ví dụ Bellman – Ford

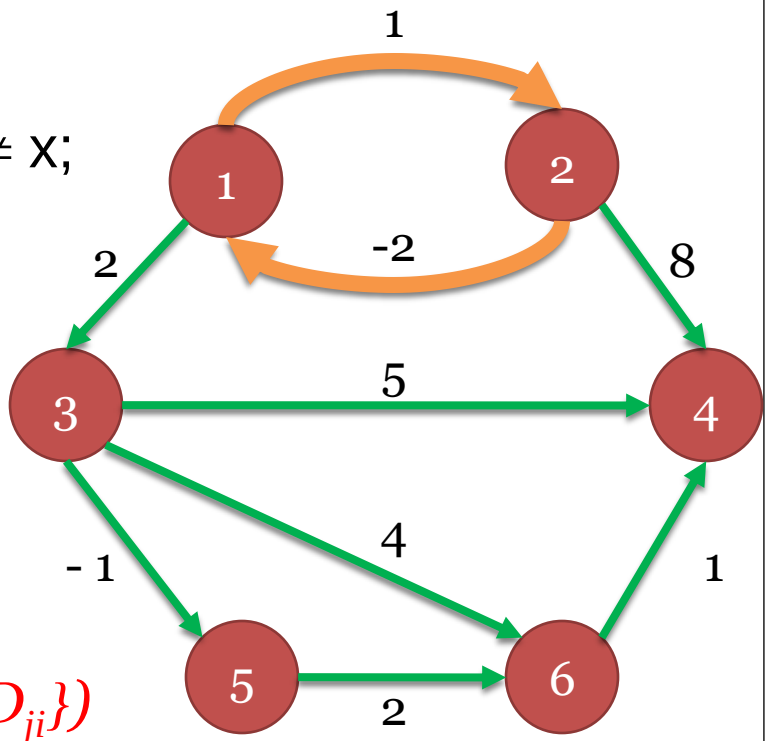
- Xem đồ thị trong hình vẽ, chúng ta sẽ tính toán cho 2 trường hợp:
  - Các đường đi khởi đầu từ đỉnh 1, và
  - Các đường đi khởi đầu từ đỉnh 3.



Ví dụ Bellman – Ford

- Bắt đầu từ đỉnh 1
  - Khởi tạo: $\pi(0, 1) = 0$; $\pi(0, i) = +\infty$, $\forall i \neq x$;
  - k = 1

π và k	1	2	3	4	5	6
k=0 và π =	0	+∞	+∞	+∞	+∞	+∞



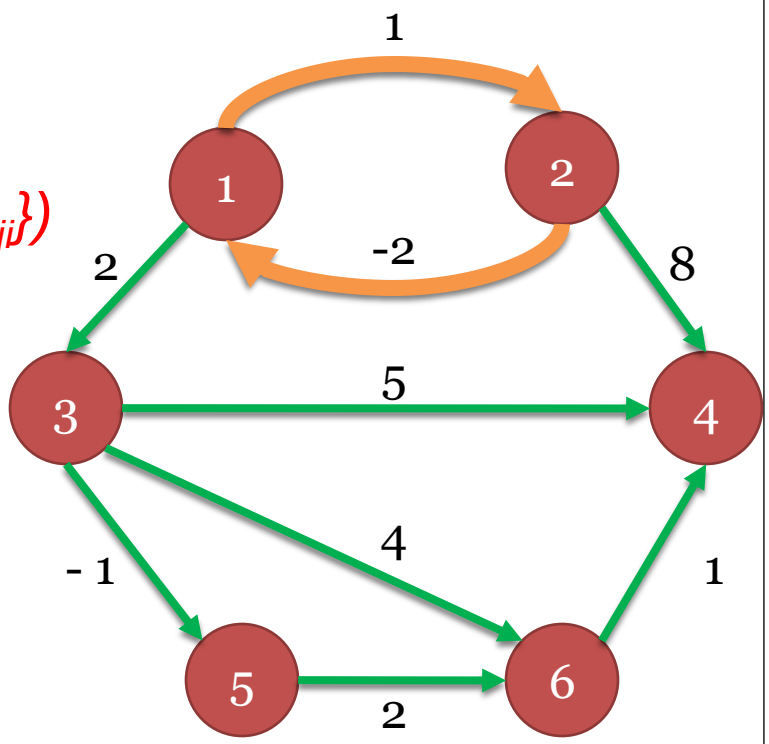
- Với k = 1, cập nhật $\pi(1, i)$ $\forall i \in V$ theo công thức:
  - $\pi(k, i) = min(\{\pi(k-1, i)\} \cup \{\pi(k-1, j) + D_{ji}\})$

π và k	1	2	3	4	5	6
k=0 và π =	0	+∞	+∞	+∞	+∞	+∞
k=1 và π =	0	1	2	+∞	+∞	+∞

Ví dụ Bellman – Ford

- Với k = 2, cập nhật $\pi(2, i)\ \forall i \in V$ theo công thức:
  - $\pi(k, i) = min(\{\pi(k-1, i)\} \cup \{\pi(k-1, j) + D_{ji}\})$

π và k	1	2	3	4	5	6
k=0 và π =	0	+∞	+∞	+∞	+∞	+∞
k=1 và π =	0	1	2	+∞	+∞	+∞
k=2 và π =	-1	1	2	7	1	6



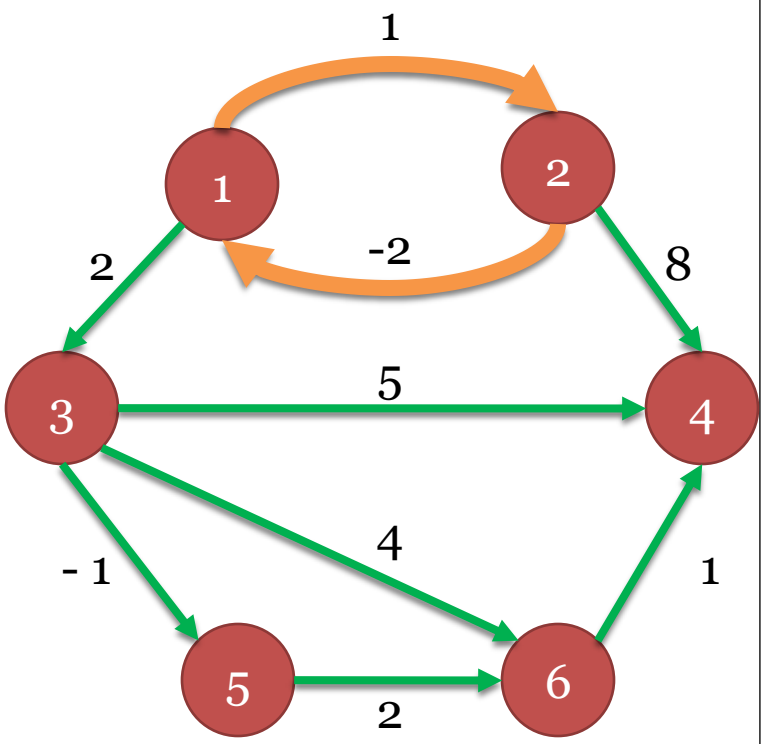
- Với k = 3

π và k	1	2	3	4	5	6
k=0 và π =	0	+∞	+∞	+∞	+∞	+∞
k=1 và π =	0	1	2	+∞	+∞	+∞
k=2 và π =	-1	1	2	7	1	6
k=3 và π =	-1	0	1	7	1	3

Ví dụ Bellman – Ford

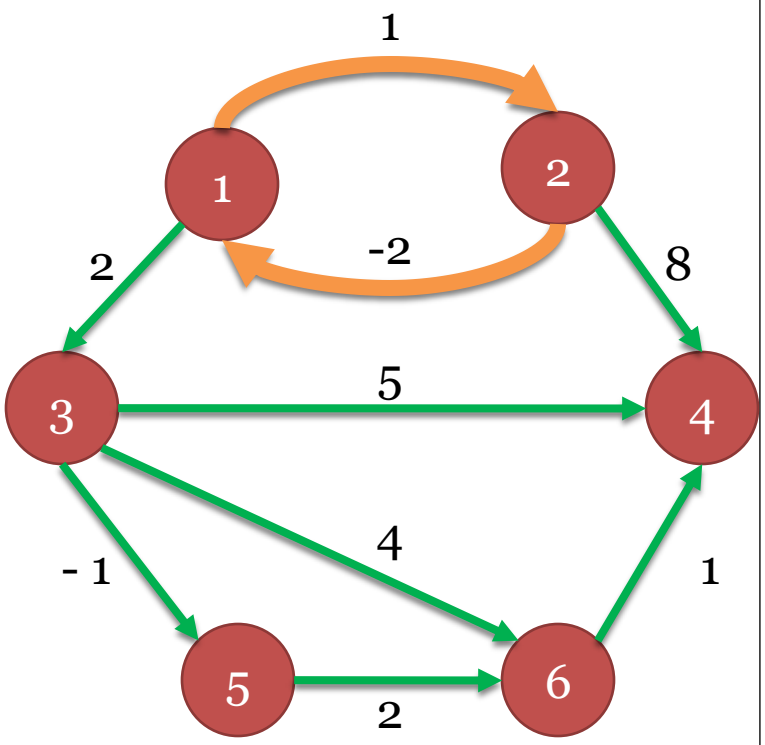
- Với k = 4
- Với k = 5
- Với k = 6

π và k	1	2	3	4	5	6
k=0 và π =	0	+∞	+∞	+∞	+∞	+∞
k=1 và π =	0	1	2	+∞	+∞	+∞
k=2 và π =	-1	1	2	7	1	6
k=3 và π =	-1	0	1	7	1	3
k=4 và π =	-2	0	1	4	0	3
k=5 và π =	-2	-1	0	4	0	2
k=6 và π =	-3	-1	0	3	-1	2



Ví dụ Bellman – Ford

- Trường hợp đường đi khởi đầu từ đỉnh 3, thuật toán dừng và cho biết có đường đi ngắn nhất từ đỉnh 3 đến mỗi đỉnh còn lại hay không. Các số trong ngoặc là các giá trị của đỉnh trước.



π và k	3	1	2	4	5	6
k=0 và π =	0	∞	∞	∞	∞	∞
k=1 và π =	0	∞	∞	5(3)	-1(3)	4(3)
k=2 và π =	0	∞	∞	5(3)	-1(3)	1(5)
k=3 và π =	0	∞	∞	2(6)	-1(3)	1(5)
k=4 và π =	0	∞	∞	2(6)	-1(3)	1(5)

Ví dụ Bellman – Ford

- Dựa vào bảng trên có thể suy ra:
  - Đường đi từ 3 đến 1 hay 2: không có;
  - Đường đi ngắn nhất từ 3 đến 4 (độ dài 2): 4← 6← 5← 3;
  - Đường đi ngắn nhất từ 3 đến 5 (độ dài -1):5← 3;
  - Đường đi ngắn nhất từ 3 đến 6 (độ dài 1): 6← 5← 3.

π và k	3	1	2	4	5	6
k=0 và π =	0	∞	∞	∞	∞	∞
k=1 và π =	0	∞	∞	5(3)	-1(3)	4(3)
k=2 và π =	0	∞	∞	5(3)	-1(3)	1(5)
k=3 và π =	0	∞	∞	2(6)	-1(3)	1(5)
k=4 và π =	0	∞	∞	2(6)	-1(3)	1(5)

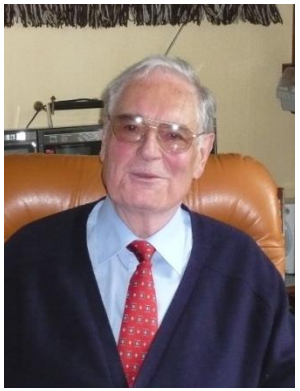
BÀI TOÁN ĐƯỜNG ĐI NGẮN NHẤT

THUẬT TOÁN FLOYD – WARSHALL

Lịch sử

- Thuật toán được nhà toán học người Mỹ, Robert Floyd (1936 – 2001) đưa ra vào năm 1962.
- Bernard Roy (1934 -) cũng đưa ra thuật toán tương tự vào năm 1959. Do vậy, thuật toán còn có tên gọi là Roy – Floyd.
- Stephen Warshall (1935 – 2006) cũng công bố một thuật toán tương tự vào năm 1962.

Robert Floyd



Bernard Roy

Stephen Warshall

Thuật toán Floyd

- Thuật toán Floyd được dùng để tìm ra đường đi ngắn nhất giữa tất cả cặp đỉnh bất kỳ của một đồ thị G với các cạnh có trọng lượng dương.
- Dữ liệu nhập cho thuật toán là ma trận trọng lượng D.

Thuật toán Floyd

- Khởi đầu với ma trận trọng số D.
- Thực hiện n lần lặp trên D. Sau bước lặp thứ k, D[i,j] chứa độ dài đường đi ngắn nhất từ đỉnh i đến đỉnh j mà chỉ đi qua các đỉnh có chỉ số không vượt quá k.
- Vậy trong bước lặp thứ k ta thực hiện theo công thức sau đây:

$$D^{(k)}[i,j] = \min (D^{(k-1)}[i,j] , D^{(k-1)}[i,k] + D^{(k-1)}[k,j])$$

với k = 1, 2, ... , n.

Cài đặt thuật toán Floyd

```
void Floyd()
{
  for (i = 0; i < n; i++)
       for (j = 0; j < n; j++) {
             d[i][j] = a[i][j];
             p[i][j] = i;
       }
  for (k = 0; k < n; k++)
       for (i = 0; i < n; i++)
             for (j = 0; j<n; j++)
                   if (d[i][j] > d[i][k] + d[k][j]) {
                         d[i][j] = d[i][k] + d[k][j];
                         p[i][j] = p[k][j];
                   }
}
```

Hỏi đáp