Phần 2

LÝ THUYẾT ĐỒ THỊ

Graph Theory

Nội dung

Chương 1. Các khái niệm cơ bản

- Đồ thị vô hướng và có hướng
- Các thuật ngữ cơ bản
- Một số dạng đồ thị vô hướng đặc biệt

Chương 2. Biểu diễn đồ thị

- Ma trận kề, ma trận trọng số, Ma trận liên thuộc đỉnh cạnh
- Danh sách cạnh, Danh sách kề

Chương 3. Duyệt đồ thị

- Tìm kiếm theo chiều sâu; Tìm kiếm theo chiều rộng
- Tìm đường đi và kiểm tra tính liên thông

Nội dung

Chương 4. Cây và cây khung của đồ thị

- Cây và các tính chất của cây
- Cây khung của đồ thị
- Bài toán cây khung nhỏ nhất

Chương 5. Bài toán đường đi ngắn nhất

- Phát biểu bài toán
- Đường đi ngắn nhất xuất phát từ một đỉnh (Thuật toán Dijkstra, Ford-Bellman)
- Đường đi ngắn nhất trên đồ thị không có chu trình
- Đường đi ngắn nhất giữa mọi cặp đỉnh (Thuật toán Floyd)

Chương 6. Bài toán luồng cực đại trong mạng

- Mạng, luồng và bài toán luồng cực đại
- Định lý Ford-Fulkerson
- Thuật toán Ford-Fulkerson
- Một số ứng dụng

Chương 1

CÁC KHÁI NIỆM CƠ BẢN

Chương 1
CÁC KHÁI NIỆM CƠ BẢN

1.1. Đồ thị trong thực tế

1.2. Các loại đồ thị

1.3. Bậc của đỉnh

1.4. Đồ thị con

1.5. Đồ thị đẳng cấu

1.6. Đường đi và chu trình

1.7. Tính liên thông

1.8. Một số loại đồ thị đặc biệt

1.9. Tô màu đồ thị

Đồ thị là gì?

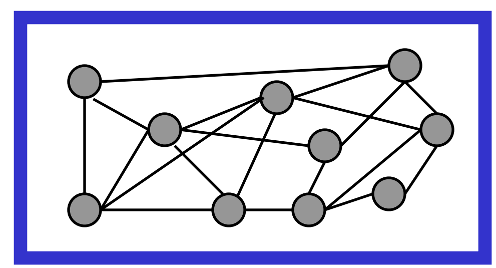
- Trong toán học đời thường hiểu là:

  Bản vẽ hay Sơ đồ biểu diễn dữ liệu nhờ sử dụng hệ thống toạ độ.

Không phải cái ta muốn đề cập

- Trong toán rời rạc:

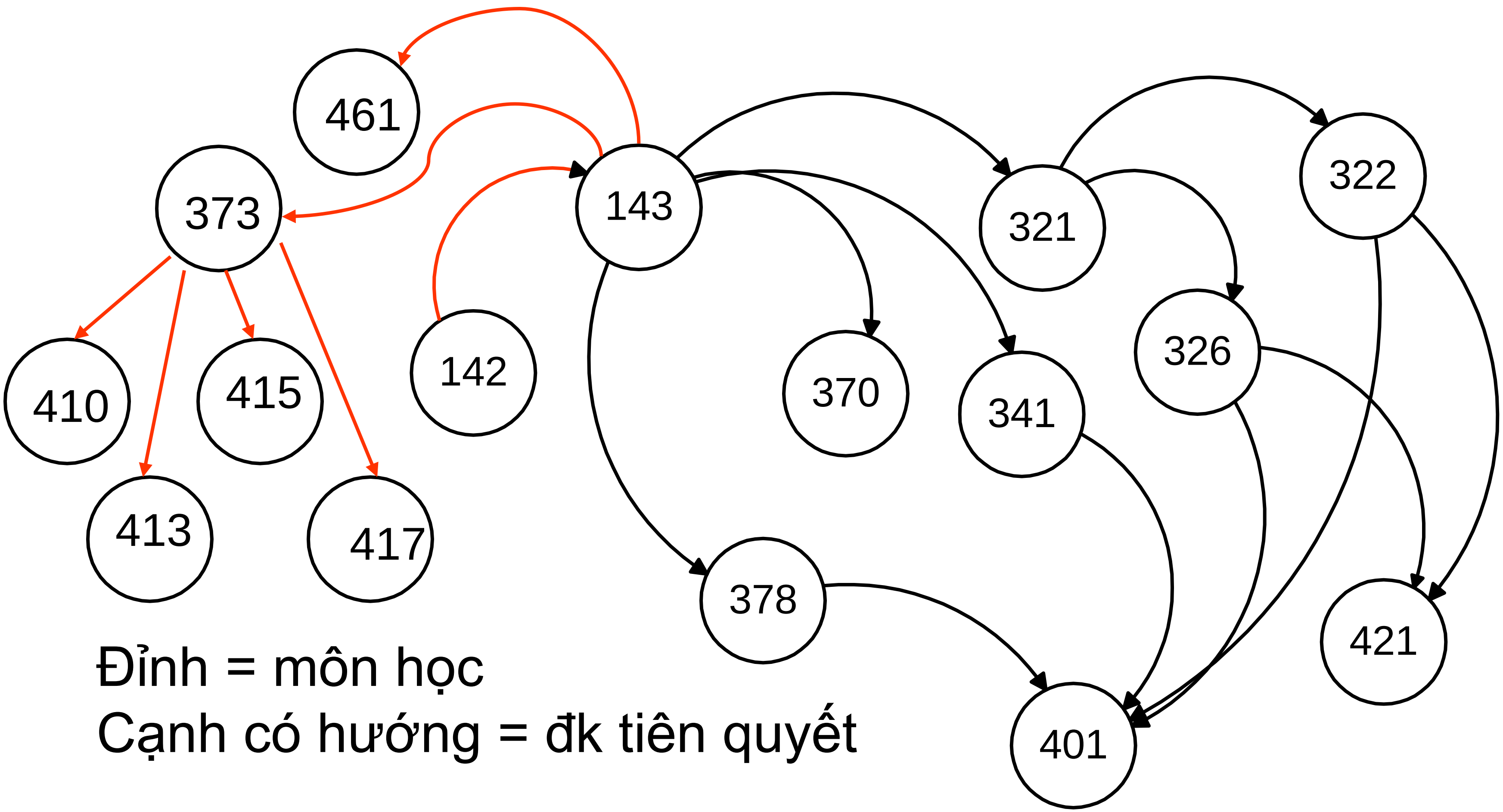
  Đây là cấu trúc rời rạc có tính trực quan cao, rất tiện ích để biểu diễn các quan hệ.



Các ứng dụng thực tế của đồ thị

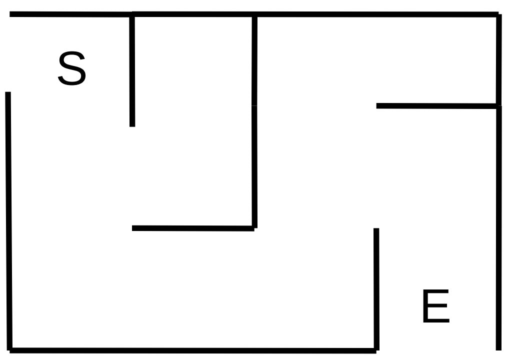
- Có tiềm năng ứng dụng trong nhiều lĩnh vực (Đồ thị có thể dùng để biểu diễn các quan hệ. Nghiên cứu quan hệ giữa các đối tượng là mục tiêu của nhiều lĩnh vực khác nhau).
- Ứng dụng trong mạng máy tính, mạng giao thông, mạng cung cấp nước, mạng điện,…) lập lịch, tối ưu hoá luồng, thiết kế mạch, quy hoạch phát triển...
- Các ứng dụng khác: Phân tích gen, trò chơi máy tính, chương trình dịch, thiết kế hướng đối tượng, …

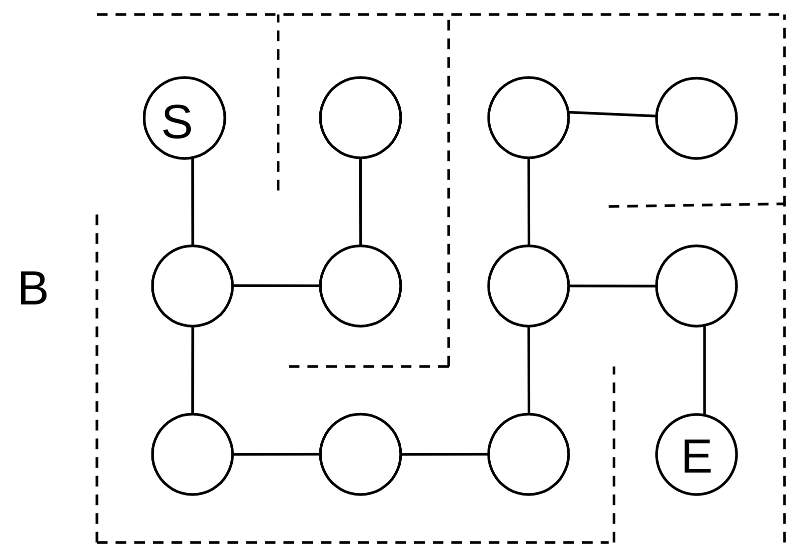
Mối liên hệ giữa các môn học



Đỉnh = môn học
Cạnh có hướng = đk tiên quyết

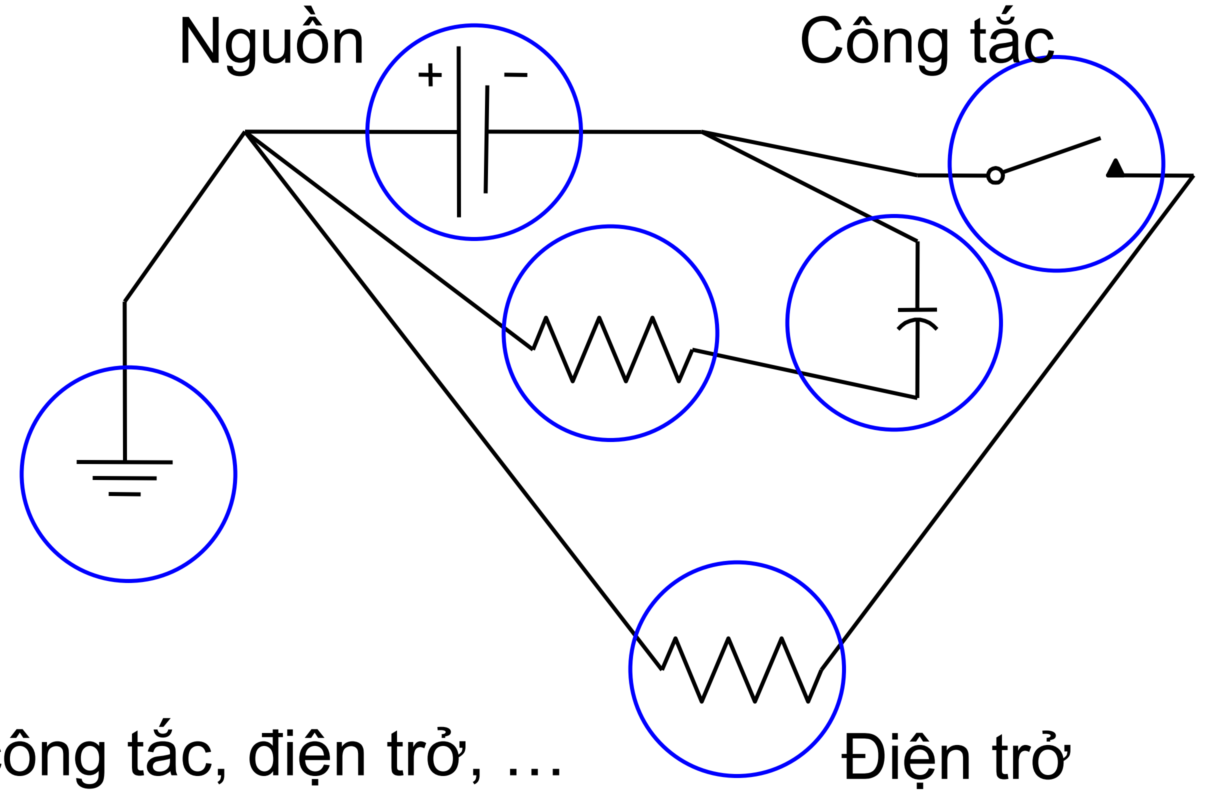
Biểu diễn mê cung





Đỉnh = phòng
Cạnh = cửa thông phòng hoặc hành lang

Biểu diễn mạch điện
(Electrical Circuits)



Đỉnh = nguồn, công tắc, điện trở, …

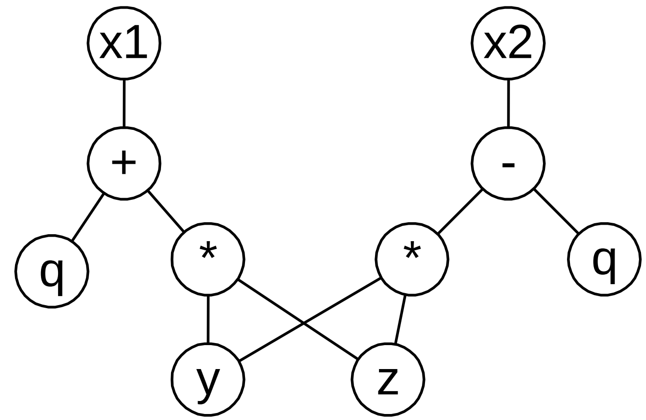
Cạnh = đoạn dây nối

Các câu lệnh của chương trình
Program statements

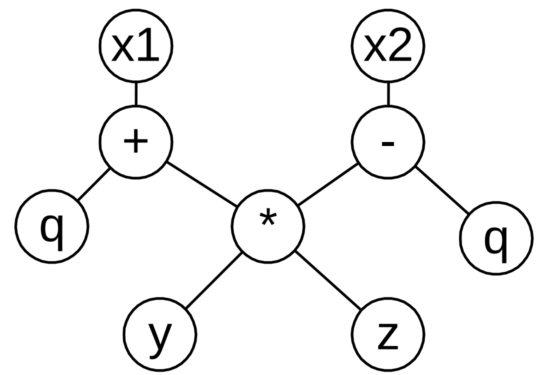
```
x1=q+y*z
x2=y*z-q
```

Thoạt nghĩ:

y*z tính hai lần



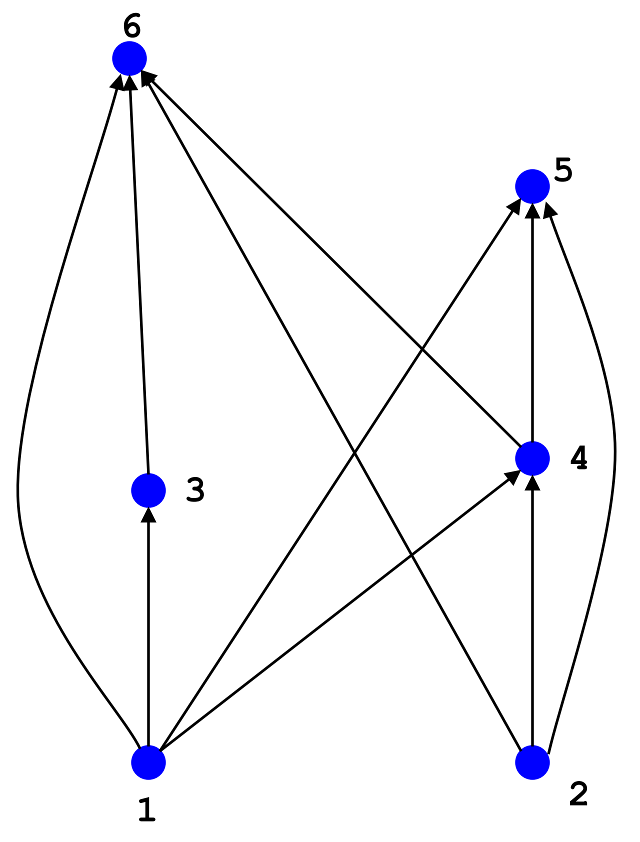
Loại
Biểu thức con
chung:



Đỉnh = ký hiệu/phép toán
Cạnh = mối quan hệ

Yêu cầu trình tự (Precedence)

```
S1        a=0;
S2        b=1;
S3        c=a+1
S4        d=b+a;
S5        e=d+1;
S6        e=c+d;
```



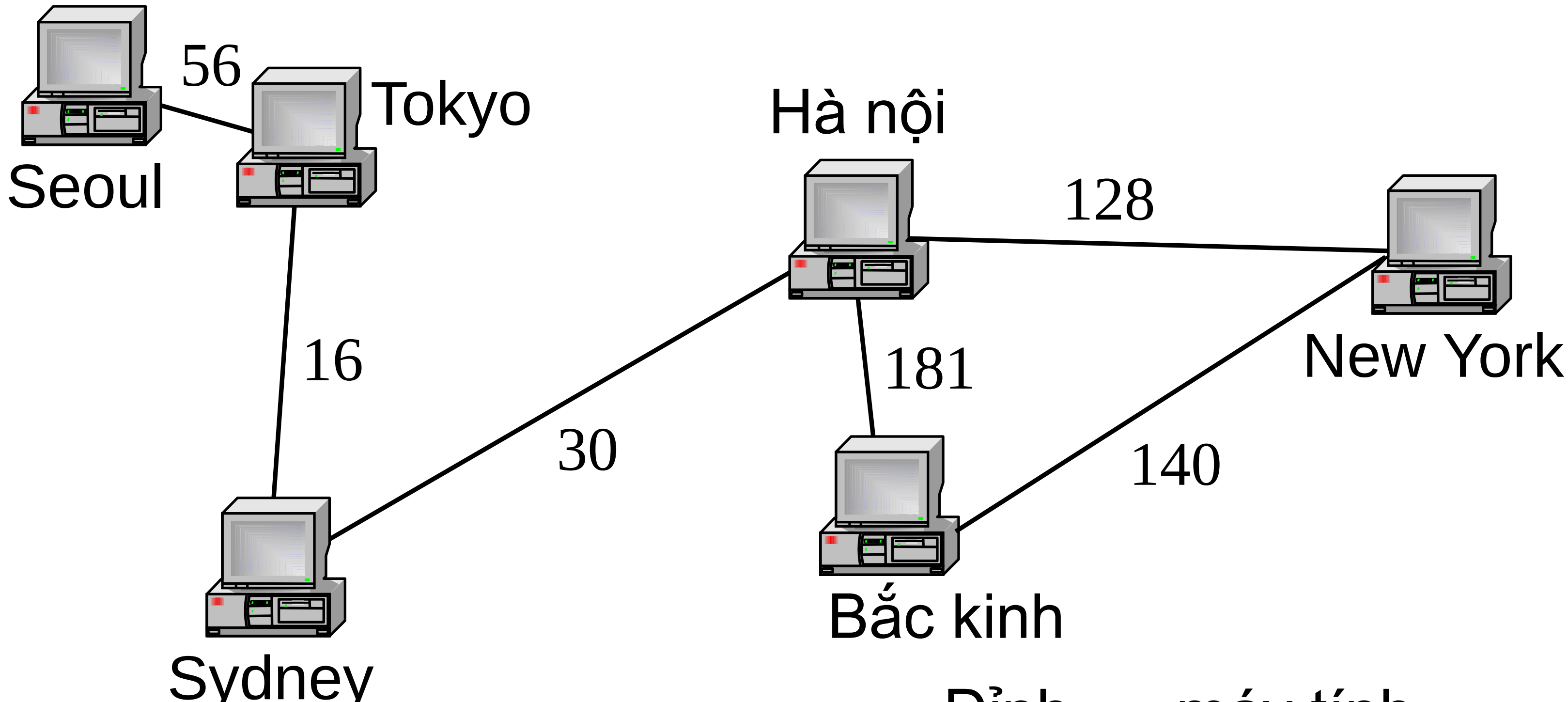
Các câu lệnh nào phải thực hiện trước S_6?

S_1, S_2, S_3, S_4

Đỉnh = câu lệnh

Cạnh = yêu cầu trình tự

Truyền thông trong mạng máy tính
(Information Transmission in a Computer Network)

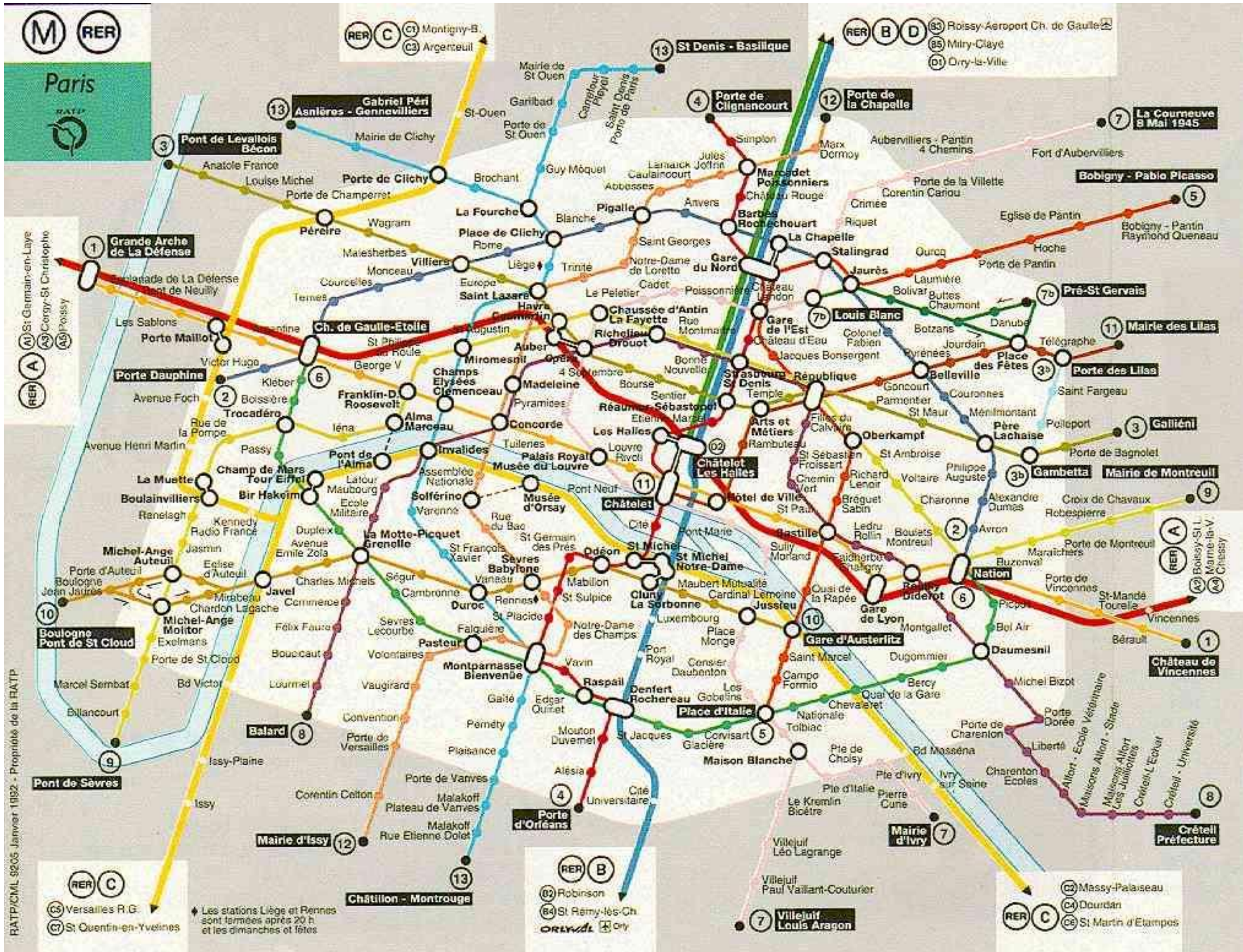


Đỉnh = máy tính
Cạnh = tốc độ truyền thông

Luồng giao thông trên xa lộ
(Traffic Flow on Highways)

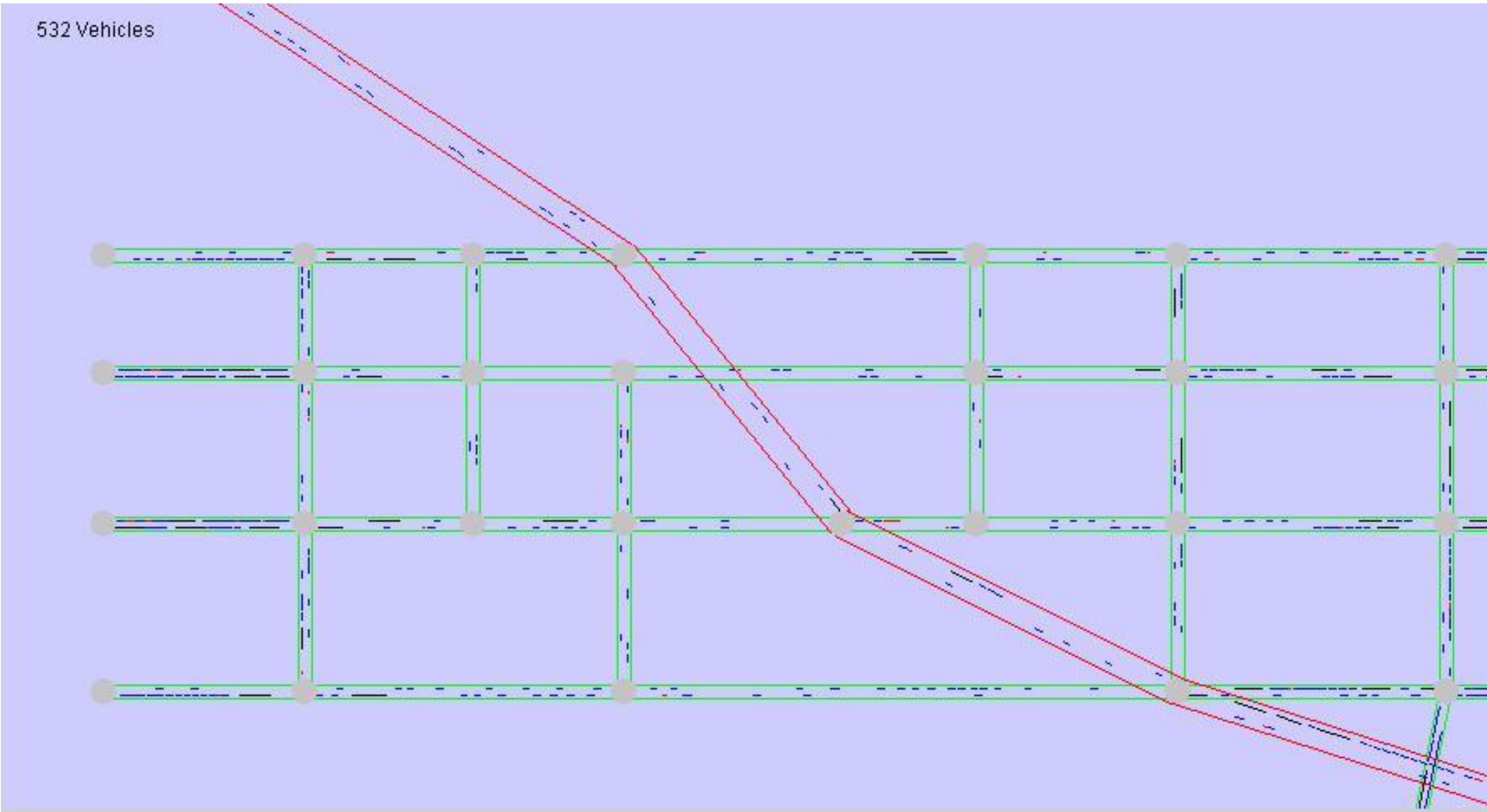
Đỉnh = thành phố
Cạnh = lượng xe cộ trên tuyến đường cao tốc kết nối giữa các thành phố

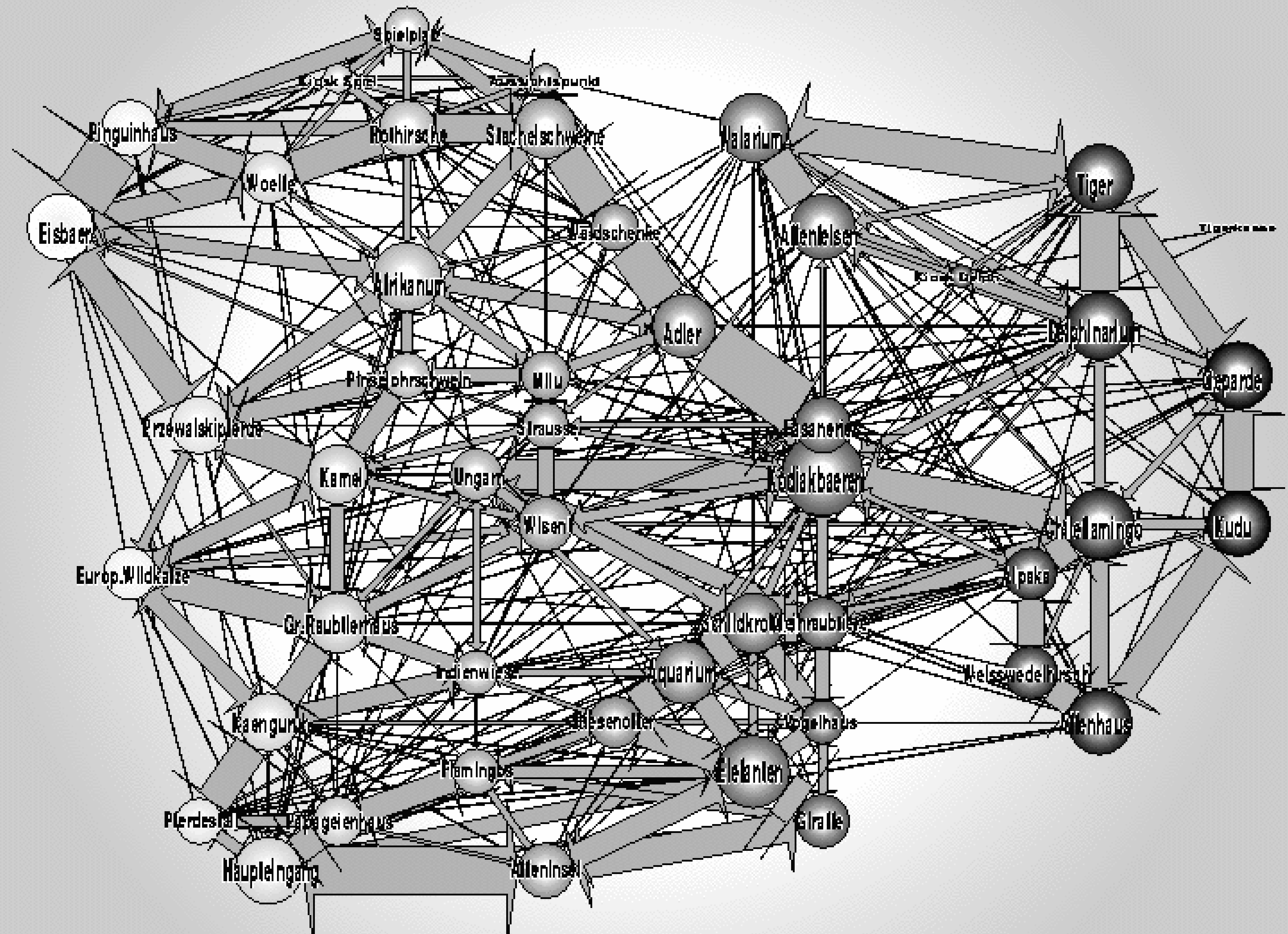
Mạng xe buýt



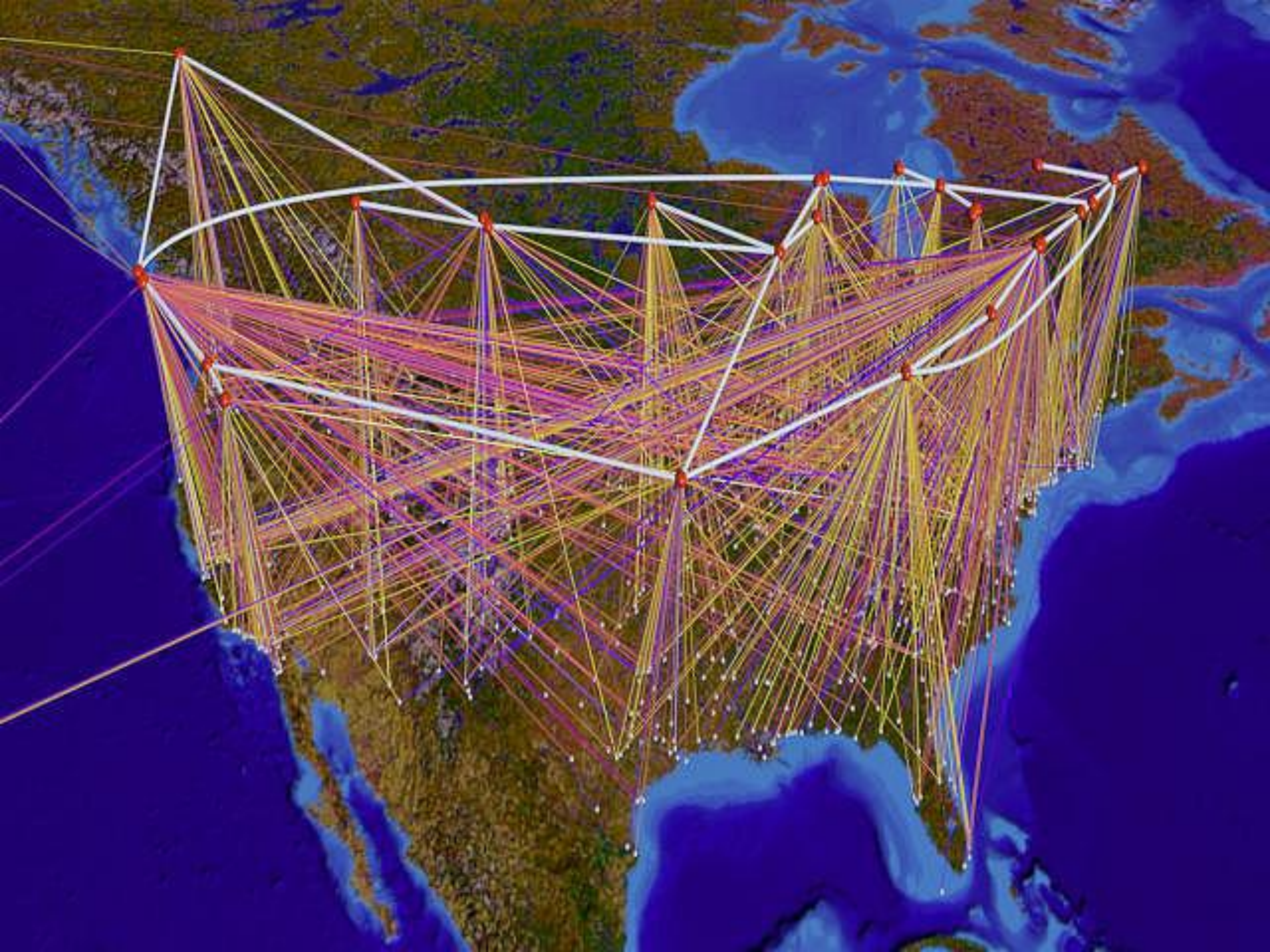
Mạng tàu điện ngầm

Sơ đồ đường phố





Spielplatz
Kiosk Spiel
Aussichtspunkt
Pinguinhaus
Rothirsche
Stachelschweine
Walarium
Tiger
Woelfe
Eisbaer
Waldschenke
Affenfelsen
Afrikanum
Adler
Delphinarium
Pinselohrschwein
Milu
Przewalskipferde
Strausse
Fasanerie
Kamel
Ungarn
Kodiakbaeren
Wisent
Chileflamingo
Kudu
Europ.Wildkatze
Alpaka
Gr.Raubtierhaus
Schildkroete
Kleinraubtiere
Indienwiese
Aquarium
Weisswedelhirsch
Kaenguruh
Riesenotter
Vogelhaus
Affenhaus
Flamingos
Elefanten
Pferdestall
Papageienhaus
Giraffe
Haupteingang
Affeninsel



Chương 1

CÁC KHÁI NIỆM CƠ BẢN

1.1. Đồ thị trong thực tế

1.2. Các loại đồ thị

1.3. Bậc của đỉnh

1.4. Đồ thị con

1.5. Đồ thị đẳng cấu

1.6. Đường đi và chu trình

1.7. Tính liên thông

1.8. Một số loại đồ thị đặc biệt

1.9. Tô màu đồ thị

Đồ thị vô hướng
(Undirected Graphs)

Định nghĩa. Đơn (đa) đồ thị vô hướng $G = (V,E)$ là cặp gồm:

- Tập đỉnh V là tập hữu hạn phần tử, các phần tử gọi là các ***đỉnh***
- Tập cạnh E là tập (họ) các bộ không có thứ tự dạng

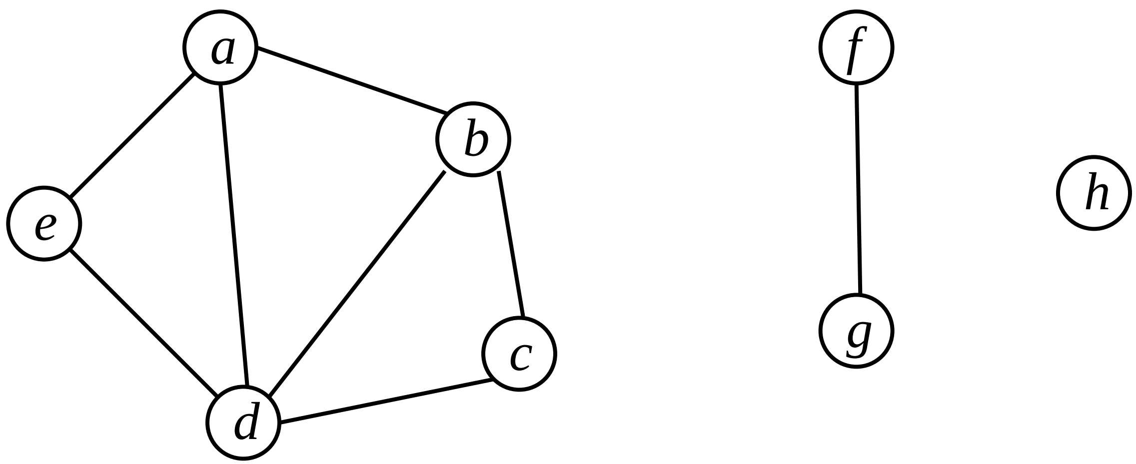
$$(u, v), u, v \in V, u \neq v$$

Đơn đồ thị vô hướng
(Simple Graph)

- **Ví dụ:** Đơn đồ thị $G_1 = (V_1, E_1)$, trong đó

  $V_1=\{a, b, c, d, e, f, g, h\}$,

  $E_1=\{(a,b), (b,c), (c,d), (a,d), (d,e), (a,e), (d,b), (f,g)\}$.



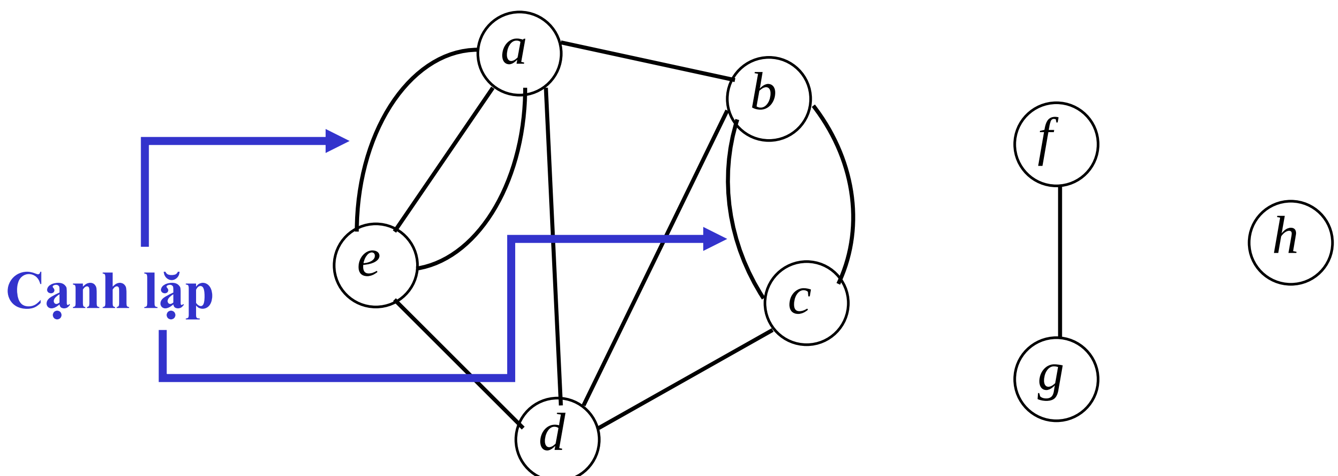
Đồ thị G_1

Đa đồ thị vô hướng
(Multi Graphs)

- **Ví dụ:** Đa đồ thị $G_2 = (V_2, E_2)$, trong đó

  $V_2=\{a, b, c, d, e, f, g, h\}$,

  $E_2=\{(a,b), (b,c), (b,c), (c,d), (a,d), (d,e), (a,e), (a,e), (a, e), (d,b), (f,g)\}$.



Đồ thị G_2

Đồ thị có hướng
(Directed Graph)

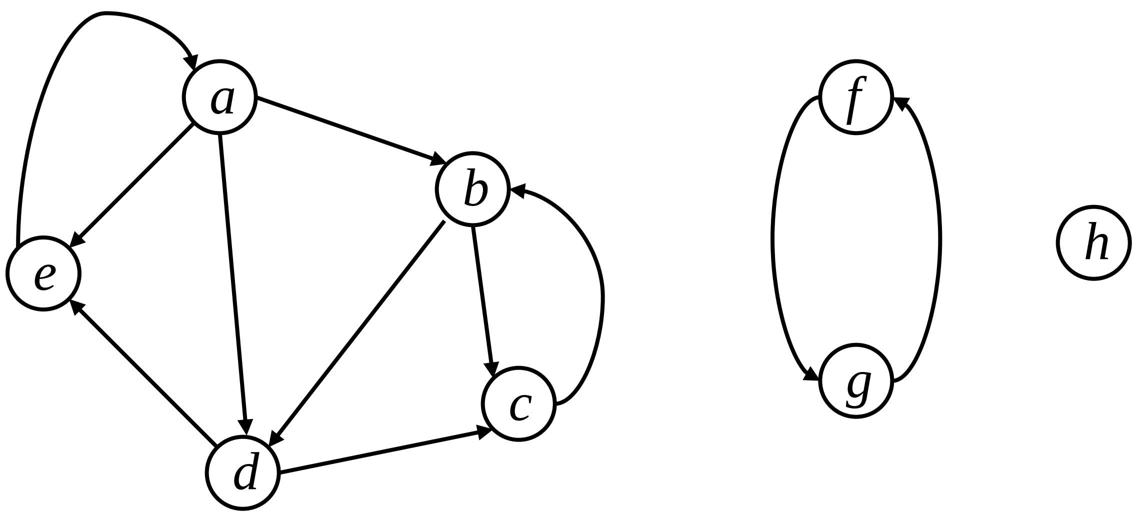
Định nghĩa. Đơn (đa) đồ thị có hướng $G = (V,E)$ là cặp gồm:

- Tập đỉnh V là tập hữu hạn phần tử, các phần tử gọi là các ***đỉnh***
- Tập cung E là tập (họ) các bộ có thứ tự dạng

$$(u, v),\ u, v \in V,\ u \neq v$$

Đơn đồ thị có hướng
(Simple digraph)

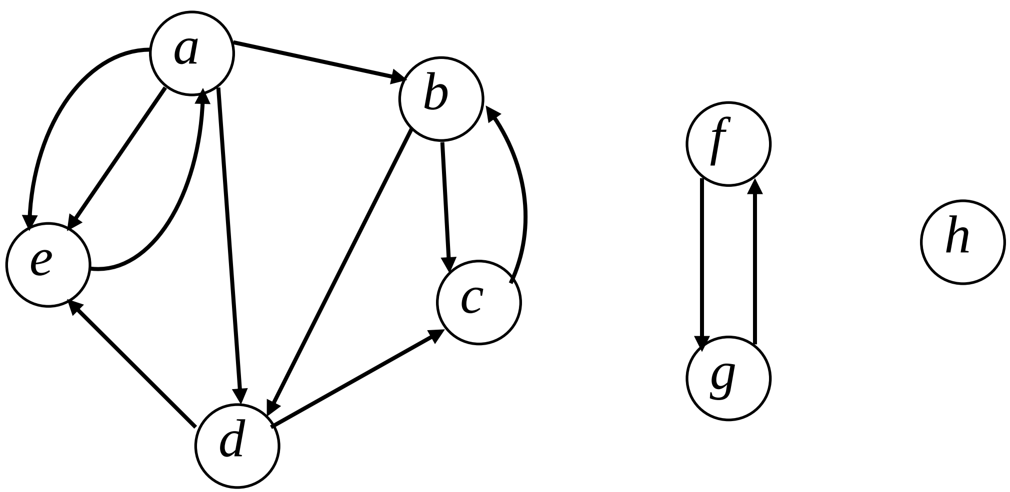
- **Ví dụ:** Đơn đồ thị có hướng $G_3 = (V_3, E_3)$, trong đó
  $V_3 = \{a, b, c, d, e, f, g, h\}$,
  $E_3 = \{(a,b), (b,c), (c,b), (d,c), (a,d), (b, d), (a,e), (d,e),$
  $(e,a), (f,g), (g,f)\}$



Đồ thị G_3

Đa đồ thị có hướng
(Multi Graphs)

- **Ví dụ:** Đa đồ thị có hướng $G_4 = (V_4, E_4)$, trong đó
  $V_4 = \{a, b, c, d, e, f, g, h\}$,
  $E_4 = \{(a,b), (b,c), (c,b), (d,c), (a,d), (b, d), (a,e), (a,e), (d,e), (e,a), (f,g), (g,f)\}$



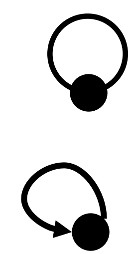
Đồ thị G_4

Các loại đồ thị: Tóm tắt

Loại	Kiểu cạnh	Có cạnh lặp?
Đơn đồ thị vô hướng	Vô hướng	Không
Đa đồ thị vô hướng	Vô hướng	Có
Đơn đồ thị có hướng	Có hướng	Không
Đa đồ thị có hướng	Có hướng	Có

- Chú ý:
  - Một dạng đồ thị ít sử dụng hơn, đó là giả đồ thị. **Giả đồ thị** là đa đồ thị mà trong đó có các **khuyên** (cạnh nối 1 đỉnh với chính nó).
  - Cách phân loại đồ thị dùng ở đây chưa chắc đã được chấp nhận trong các tài liệu khác...

Khuyên (loop)

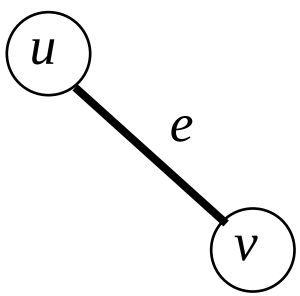


Các thuật ngữ

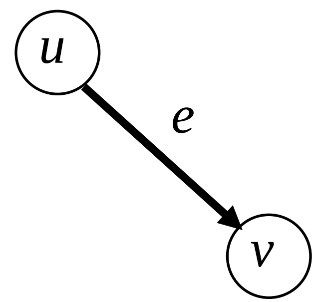
Graph Terminology

Chúng ta cần các thuật ngữ liên quan đến mối quan hệ giữa các đỉnh và các cạnh của đồ thị sau:

- *Kề nhau, nối, đầu mút, bậc, bắt đầu, kết thúc, bán bậc vào, bán bậc ra,…*

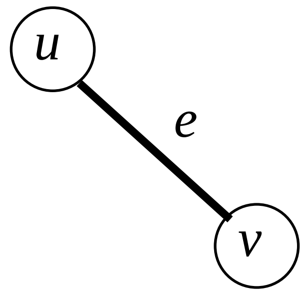


Cạnh vô hướng $e=(u,v)$



Cạnh có hướng (cung) $e=(u,v)$

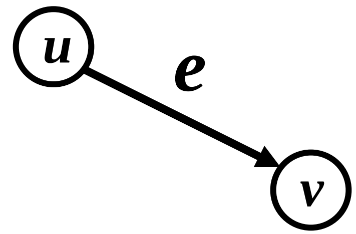
Kề (Adjacency)



Cho *G* là đồ thị vô hướng với tập cạnh *E*. Giả sử $e \in E$ là cặp (*u*,*v*). Khi đó ta nói:

- *u*, *v* là *kề nhau/lân cận/nối với nhau* (*adjacent* / *neighbors* / *connected*).
- Cạnh *e* là *liên thuộc* với hai đỉnh *u* và *v*.
- Cạnh *e nối* (*connect*) *u* và *v*.
- Các đỉnh *u* và *v* là các *đầu mút* (*endpoints*) của cạnh *e*.

Tính kề trong đồ thị có hướng



- Cho G là đồ thị có hướng (có thể là đơn hoặc đa) và giả sử $e = (u,v)$ là cạnh của G. Ta nói:
  - – *u* và *v* là *kề nhau*, *u* là kề tới *v*, *v* là kề từ *u*
  - – *e đi ra khỏi u, e đi vào v.*
  - – *e nối u với v, e đi từ u tới v*
  - – Đỉnh đầu (*initial vertex) của e* là *u*
  - – Đỉnh cuối (*terminal vertex*) của *e* là *v*

Chương 1
CÁC KHÁI NIỆM CƠ BẢN

1.1. Đồ thị trong thực tế

1.2. Các loại đồ thị

1.3. Bậc của đỉnh

1.4. Đồ thị con

1.5. Đồ thị đẳng cấu

1.6. Đường đi và chu trình

1.7. Tính liên thông

1.8. Một số loại đồ thị đặc biệt

Bậc của đỉnh (Degree of a Vertex)

- Giả sử G là đồ thị vô hướng, $v \in V$ là một đỉnh nào đó.
- *Bậc* của đỉnh v, deg(v), là số cạnh kề với nó.
- Đỉnh bậc 0 được gọi là *đỉnh cô lập* (*isolated*).
- Đỉnh bậc 1 được gọi là *đỉnh treo* (*pendant*).
- Các ký hiệu thường dùng:

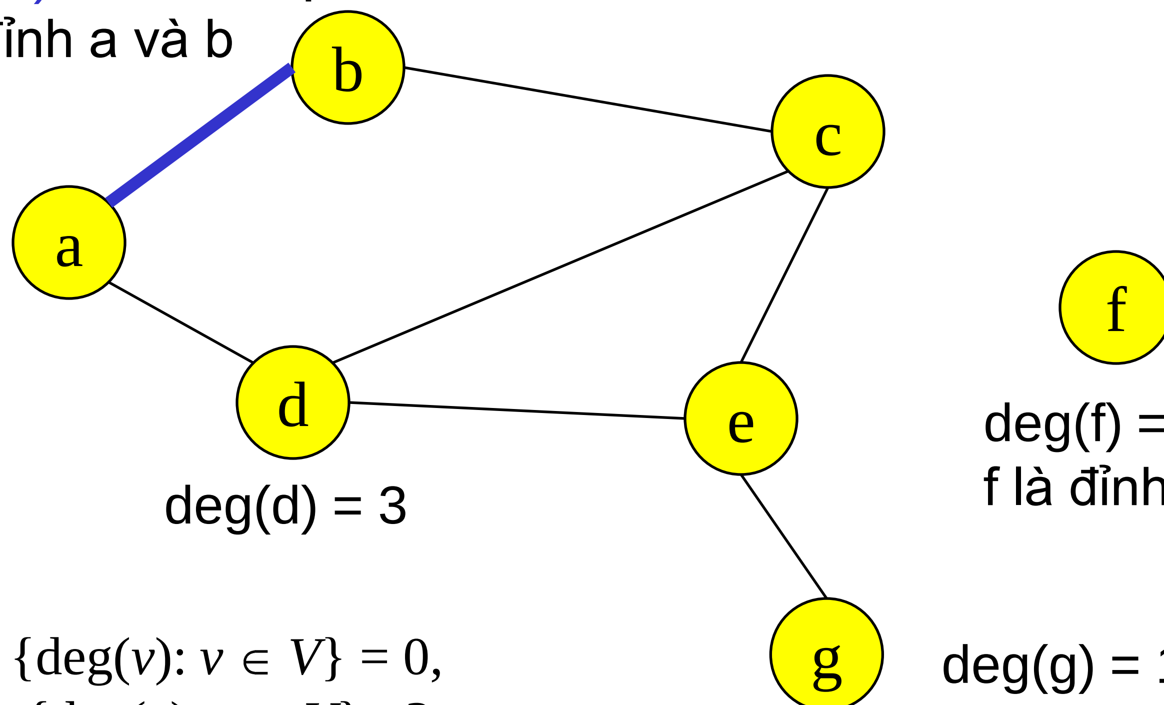
$$\delta(G) = \min \{\deg(v) : v \in V\},$$

$$\Delta(G) = \max \{\deg(v) : v \in V\}.$$

Ví dụ

Cạnh (a,b) là liên thuộc
với hai đỉnh a và b

b là kề với c và c là kề với b



deg(d) = 3

deg(f) = 0
f là đỉnh cô lập

deg(g) = 1
g là đỉnh treo

$\delta(G) = \min \{\deg(v): v \in V\} = 0,$
$\Delta(G) = \max \{\deg(v): v \in V\} = 3.$

Định lý về các cái bắt tay
(Handshaking Theorem)

- **Định lý.** Giả sử *G* là đồ thị vô hướng (đơn hoặc đa) với tập đỉnh *V* và tập cạnh *E*. Khi đó

$$\sum_{v \in V} \deg(v) = 2|E|$$

CM: Trong tổng ở vế trái mỗi cạnh $e=(u,v)\in E$ được tính hai lần: trong deg(u) và deg(v).

- **Hệ quả:** Trong một đồ thị vô hướng bất kỳ, số lượng đỉnh bậc lẻ (đỉnh có bậc là số lẻ) bao giờ cũng là số chẵn.

Ví dụ.

Biết rằng mỗi đỉnh của đồ thị vô hướng $G=(V,E)$ với 14 đỉnh và 25 cạnh đều có bậc là 3 hoặc 5.

Hỏi G có bao nhiêu đỉnh bậc 3?

Giải. Giả sử G có x đỉnh bậc 3.

Khi đó có $14-x$ đỉnh bậc 5.

Do $|E| = 25$, nên tổng tất cả các bậc là 50.

Từ đó, $3x + 5(14-x) = 50$

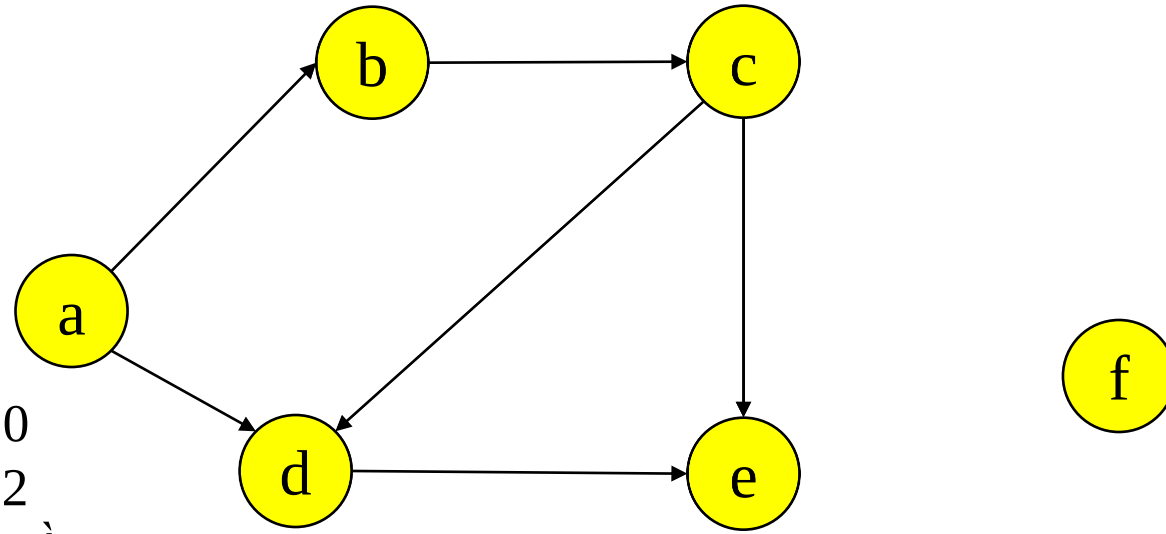
Suy ra $x = 10$.

Bậc của đỉnh của đồ thị có hướng

- Cho *G* là đồ thị có hướng, *v* là đỉnh của *G*.
  - *Bán bậc vào* (*in-degree*) của *v*, $\deg^-(v)$, là số cạnh đi vào *v*.
  - *Bán bậc ra* (*out-degree*) của *v*, $\deg^+(v)$, là số cạnh đi ra khỏi *v*.
  - *Bậc* của *v*, $\deg(v) :\equiv \deg^-(v) + \deg^+(v)$, là tổng của bán bậc vào và bán bậc ra của *v*.

Ví dụ

b kề tới c và c kề từ b



deg-(a) = 0
deg+(a)= 2
a- đỉnh nguồn

$deg^{-}(d) = 2$
$deg^{+}(d) = 1$

deg-(e) = 2
deg+(e)= 0
e – đỉnh đích (target)

deg-(f) = 0
deg+(f)= 0

Định lý về các cái bắt tay có hướng

Directed Handshaking Theorem

- **Định lý.** Giả sử G là đồ thị có hướng (có thể là đơn hoặc đa) với tập đỉnh V và tập cạnh E. Khi đó:

$$\sum_{v \in V} \deg^{-}(v) = \sum_{v \in V} \deg^{+}(v) = \frac{1}{2} \sum_{v \in V} \deg(v) = |E|$$

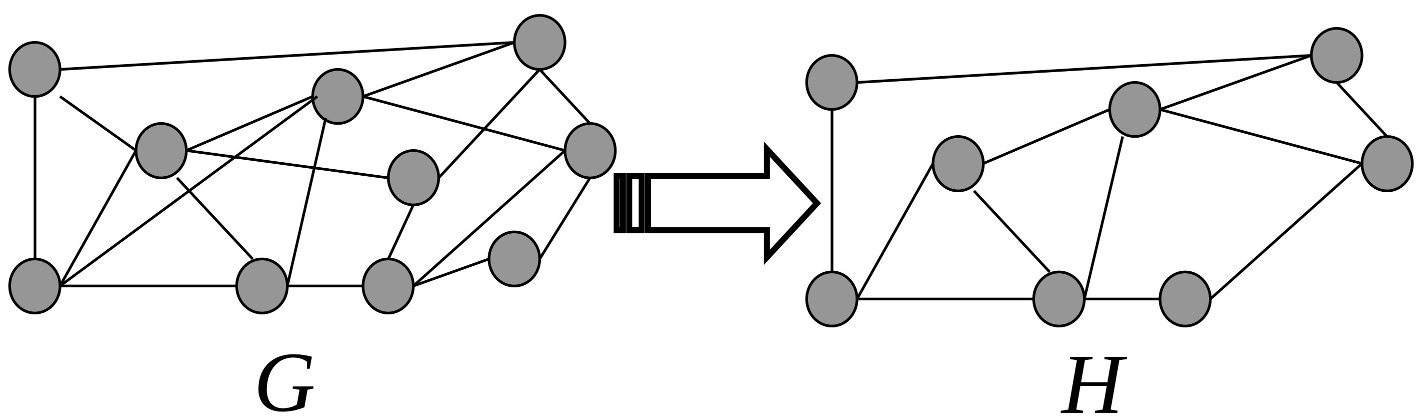
- Chú ý là khái niệm bậc của đỉnh là không thay đổi cho dù ta xét đồ thị vô hướng hay có hướng.

Chương 1

CÁC KHÁI NIỆM CƠ BẢN

Đồ thị con (Subgraphs)

- **Định nghĩa.** Đồ thị $H=(W,F)$ được gọi là đồ thị con của đồ thị $G=(V,E)$ nếu $W \subseteq V$ và $F \subseteq E$.
- Ký hiệu: $H \subseteq G$.

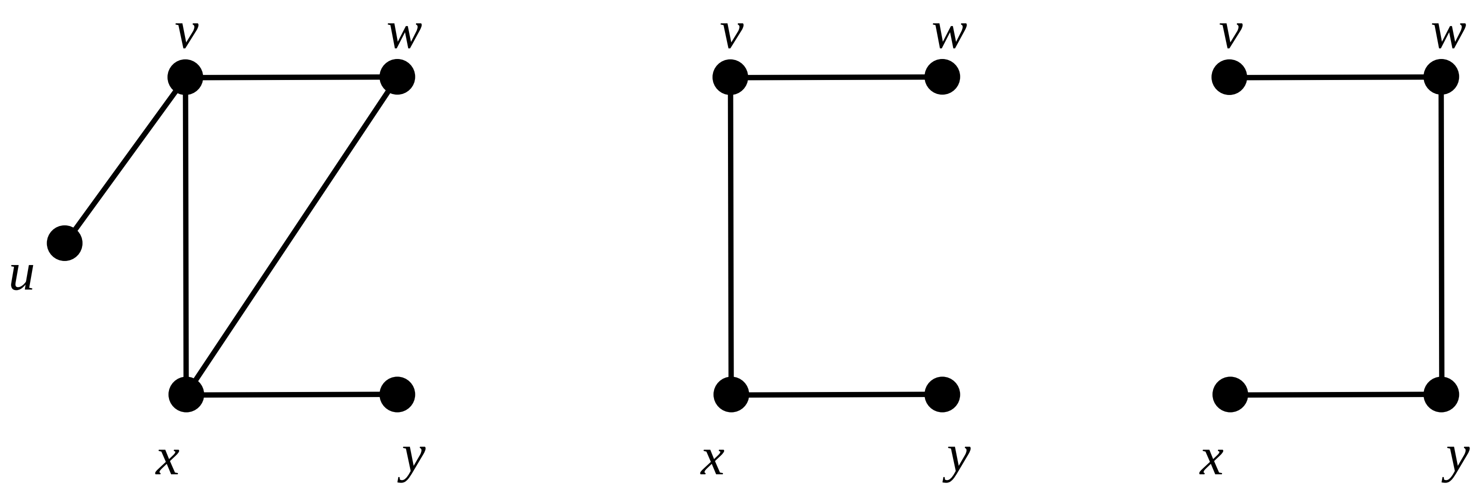


Ví dụ

Definition.

A graph H is a subgraph of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$ (denote $H \subseteq G$).

Ví dụ



G $\qquad$ $H \subseteq G$ $\qquad$ $F \not\subseteq G$

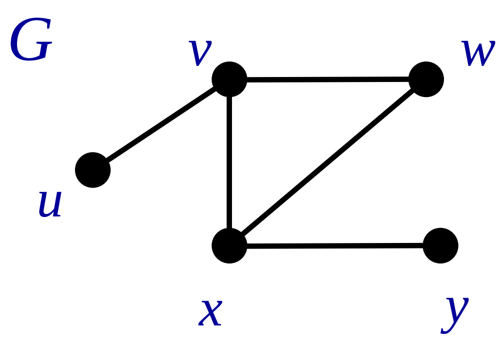
Đồ thị con cảm sinh
Induced Subgraph

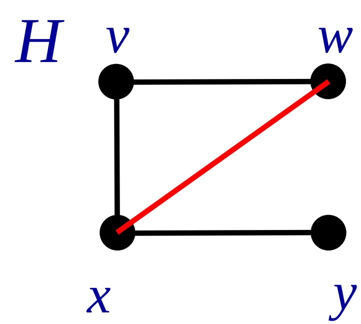
Định nghĩa. Cho $G = (V, E)$ là đồ thị vô hướng.

Giả sử $S \subseteq V$, $S \neq \varnothing$. Đồ thị con cảm sinh bởi S là đồ thị con cực đại của G với tập đỉnh là S. (thường ký hiệu là $<S>$)

Đồ thị con H của đồ thị G được gọi là đồ thị con cảm sinh đỉnh (vertex-induced subgraph) của G nếu tìm được $S \subseteq V$ sao cho $H=<S>$.

Ví dụ:





H không là đồ thị con cảm sinh của G.

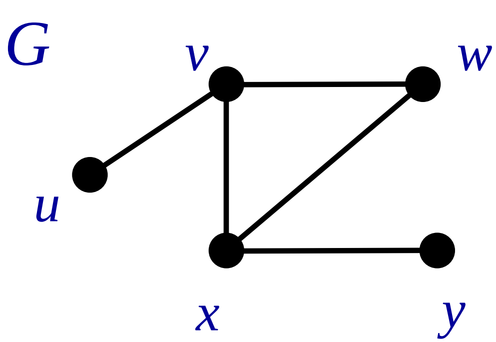
$H \cup \{(x,w)\}$ đúng

Loại bỏ đỉnh

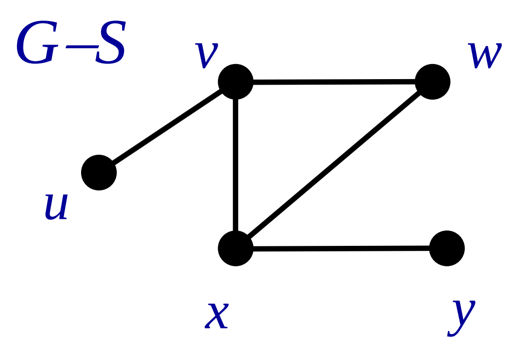
The deletion of vertices

Định nghĩa. Cho $G = (V, E)$ là đồ thị vô hướng. Giả sử $S \subseteq V$. Ta gọi việc loại bỏ tập đỉnh S khỏi đồ thị là việc loại bỏ tất cả các đỉnh trong S cùng các cạnh kề với chúng.

- Như vậy nếu ký hiệu đồ thị thu được là $G–S$, ta có $G–S = \langle V–S \rangle$. Nếu $S=\{v\}$, thì để đơn giản ta viết $G–v$.



Giả sử $S=\{x,u\} \Rightarrow$



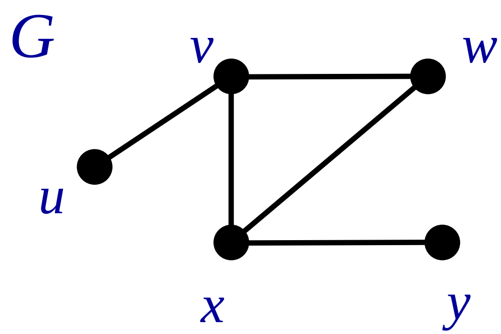
Đồ thị con cảm sinh cạnh
Edge Induced Subgraph

Định nghĩa. Cho $G = (V, E)$ là đồ thị vô hướng.

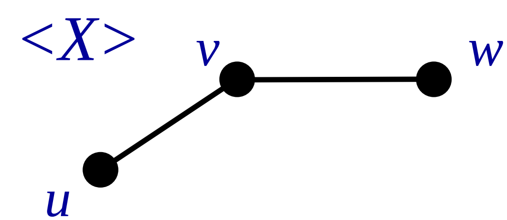
Giả sử $X \subseteq E$, $X \neq \emptyset$. Đồ thị con cảm sinh bởi X là đồ thị con ***nhỏ nhất*** của G với tập cạnh là X. (ký hiệu bởi $<X>$)

Đồ thị con H của G được gọi là đồ thị con cảm sinh cạnh (edge-induced subgraph) nếu $H=<X>$ đối với một tập con nào đó $X \subseteq E$.

Ví dụ:



$Cho\ X=\{(u,v),(v,w)\} \Rightarrow$

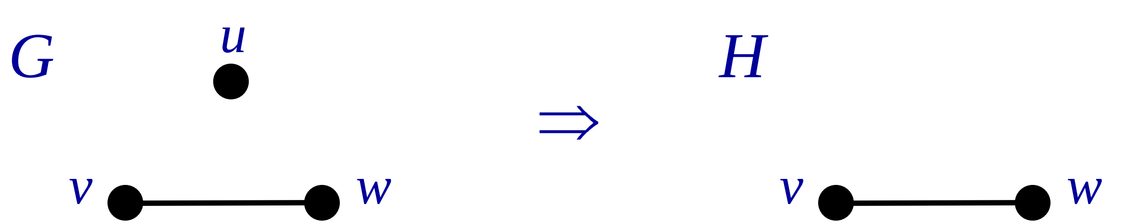


Đồ thị con cảm sinh cạnh và cảm sinh đỉnh

Ví dụ. Cho $G=(V,E)$ là đồ thị vô hướng.

Nếu $H=<E(G)>$, thì có thể suy ra $H=<V(G)>$ được không?

No



Dễ thấy, $G = <V(G)>$.

Ví dụ trên cho thấy $<E(G)>$ không nhất thiết trùng với G

Đồ thị con bao trùm

Spanning Subgraph

Định nghĩa.

Đồ thị con $H \subseteq G$ được gọi là đồ thị con bao trùm của G nếu tập đỉnh của H là tập đỉnh của G: $V(H) = V(G)$.

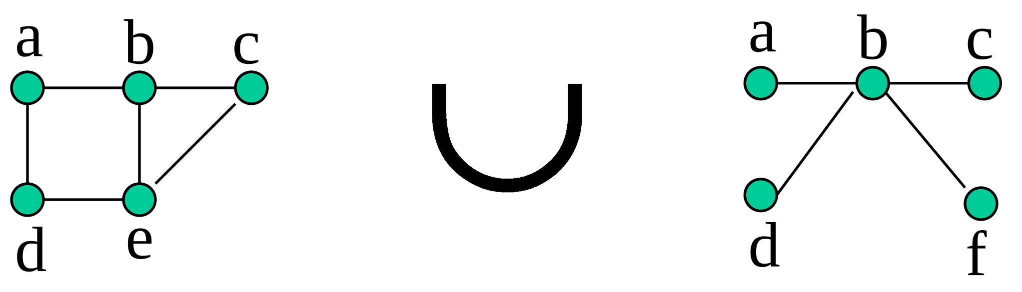
Định nghĩa.

Ta viết $H = G + \{(u,v), (u,w)\}$ hiểu là

$E(H) = E(G) \cup \{(u,v), (u,w)\}$, trong đó $(u,v), (u,w) \notin E(G)$.

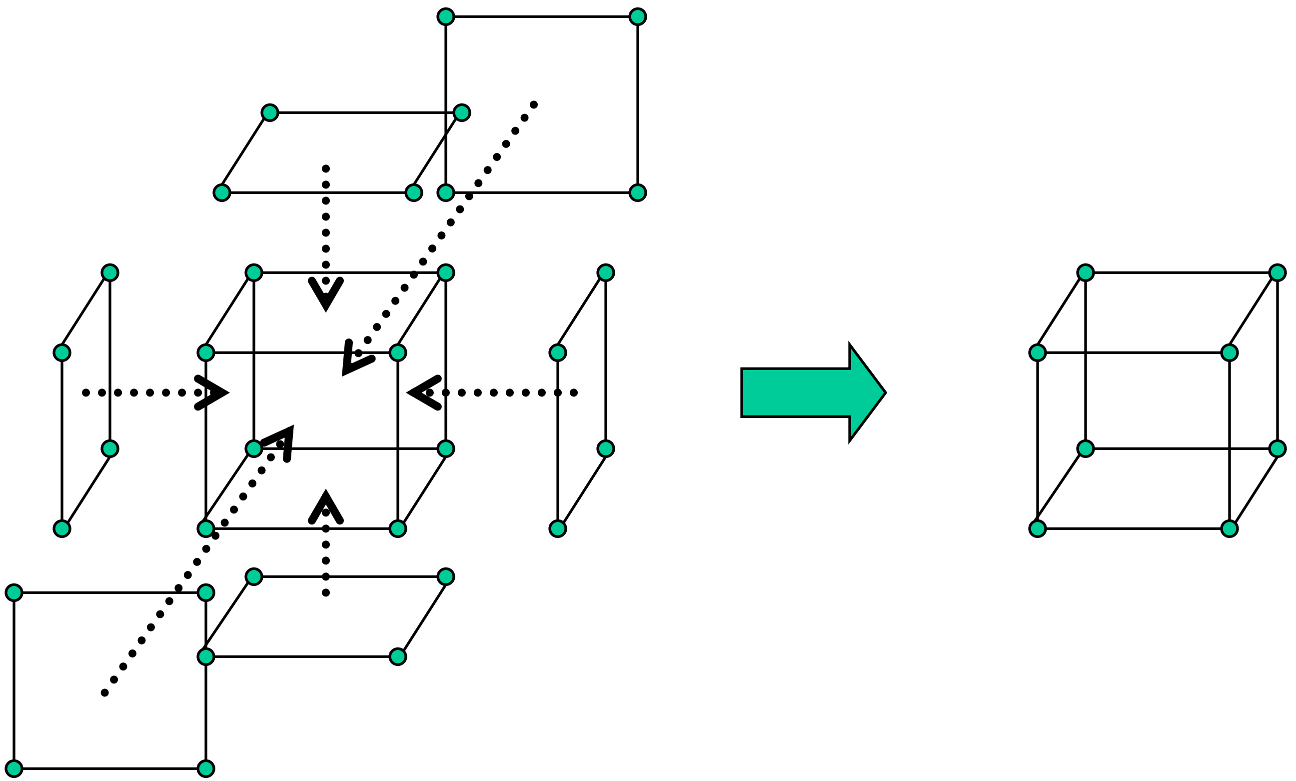
Hợp của hai đồ thị

- Hợp $G_1 \cup G_2$ của hai đơn đồ thị $G_1=(V_1, E_1)$ và $G_2=(V_2,E_2)$ là đơn đồ thị $(V_1 \cup V_2, E_1 \cup E_2)$.



Hợp của các đồ thị

Nếu S_1, S_2, S_3, S_4, S_5, S_6 là các hình vuông, khi đó Q_3 là hợp của các diện của nó: $Q_3 = S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5 \cup S_6$



Chương 1
CÁC KHÁI NIỆM CƠ BẢN

1.1. Đồ thị trong thực tế

1.2. Các loại đồ thị

1.3. Bậc của đỉnh

1.4. Đồ thị con

1.5. Đồ thị đẳng cấu

1.6. Đường đi và chu trình

1.7. Tính liên thông

1.8. Một số loại đồ thị đặc biệt

1.9. Tô màu đồ thị

Đồ thị đẳng cấu
Graph Isomorphism

- **Định nghĩa:**

  Hai đơn đồ thị vô hướng $G_1=(V_1, E_1)$ và $G_2=(V_2, E_2)$ là *đẳng cấu* (*isomorphic*) iff $\exists$ song ánh $f : V_1 \rightarrow V_2$ sao cho $\forall\ a, b \in V_1$, a và b là kề nhau trên G_1 iff $f(a)$ và $f(b)$ là kề nhau trên G_2.
- f là hàm đặt tên lại các đỉnh để cho hai đồ thị là đồng nhất.
- Có thể tổng quát định nghĩa này cho các loại đồ thị còn lại.

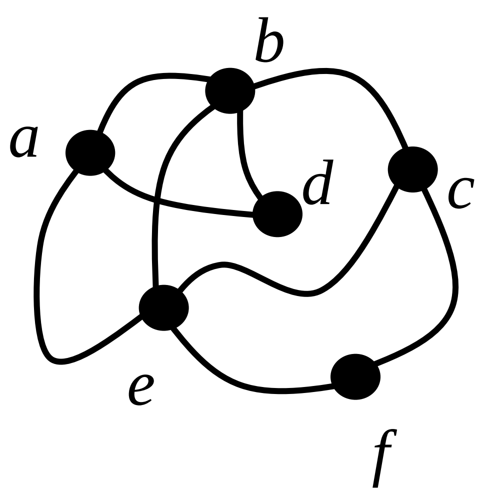
Bất biến đối với đẳng cấu

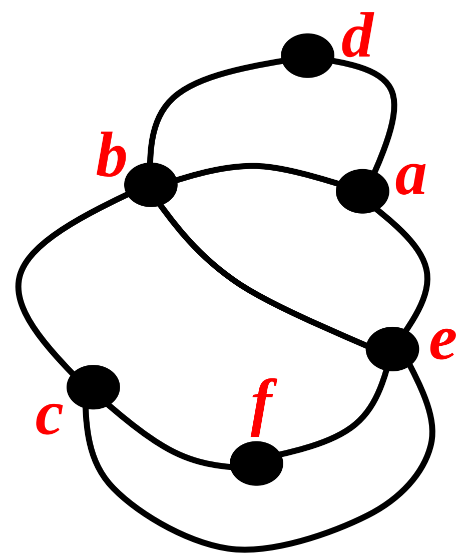
Điều kiện cần nhưng không phải là *đủ* để $G_1=(V_1, E_1)$ là đẳng cấu với $G_2=(V_2, E_2)$:

- Ta phải có $|V_1|=|V_2|$, và $|E_1|=|E_2|$.
- Số lượng đỉnh bậc *k* ở hai đồ thị là như nhau.

Ví dụ đẳng cấu

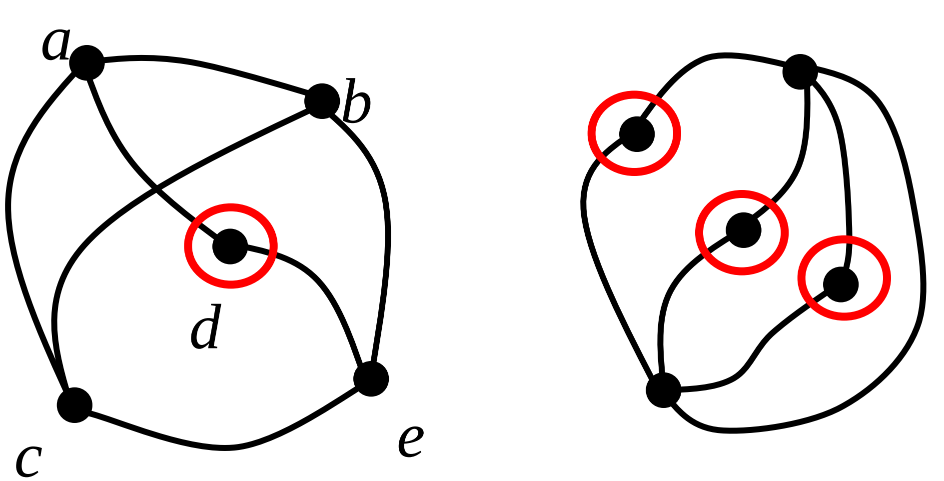
- Nếu là đẳng cấu thì hãy gán tên cho đồ thị thứ hai để thấy rõ sự đẳng cấu, trái lại hãy nêu rõ sự khác biệt.





Có đẳng cấu không?

- Nếu là đẳng cấu thì hãy gán tên cho đồ thị thứ hai để thấy rõ sự đẳng cấu, trái lại hãy nêu rõ sự khác biệt.



- *Cùng số lượng đỉnh*
- *Cùng số lượng cạnh*
- *Khác số lượng đỉnh bậc 2 (1 < >3)*

Chương 1
CÁC KHÁI NIỆM CƠ BẢN

1.1. Đồ thị trong thực tế
1.2. Các loại đồ thị
1.3. Bậc của đỉnh
1.4. Đồ thị con
1.5. Đồ thị đẳng cấu
1.6. Đường đi và chu trình
1.7. Tính liên thông
1.8. Một số loại đồ thị đặc biệt
1.9. Tô màu đồ thị

Đường đi, Chu trình

- **§Þnh nghÜa.** ***§-êng ®i P ®é dµi n*** *tõ ®Ønh u ®Õn ®Ønh v, trong ®ã n lµ sè nguyªn d-¬ng, trªn ®å thÞ G=(V,E) lµ d·y*

  $$P: \quad x_0, x_1, \ldots, x_{n-1}, x_n$$

  trong ®ã $u = x_0$, $v = x_n$, $(x_i, x_{i+1}) \in E$, $i = 0, 1, 2, \ldots, n-1$.

  §-êng ®i nãi trªn cßn cã thÓ biÓu diÔn d-íi d¹ng d·y c¸c c¹nh:

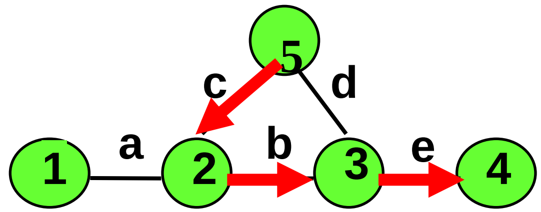
  $$(x_0, x_1), \; (x_1, x_2), \ldots, (x_{n-1}, x_n).$$

  *§Ønh u gäi lµ **®Ønh ®Çu**, cßn ®Ønh v gäi lµ **®Ønh cuèi** cña ®-êng ®i.*

Đường đi, Chu trình

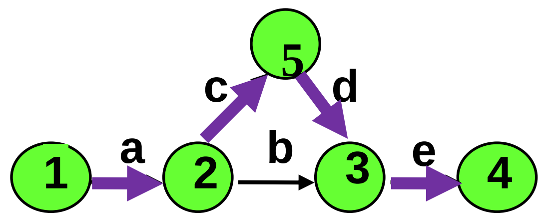
- Đường đi gọi là **đường đi đơn** nếu không có đỉnh nào bị lặp lại trên nó.
- Đường đi gọi là **đường đi cơ bản** nếu không có cạnh nào bị lặp lại trên nó.
- Nếu có đường đi từ u đến v thì ta nói đỉnh v **đạt đến được** từ đỉnh u. Ta quan niệm rằng một đỉnh v luôn đạt đến được từ chính nó.

Đường đi (Path)



Ví dụ: 5, 2, 3, 4 hoặc 5, c, 2, b, 3, e, 4.

Không có đỉnh lặp nên là đường đi đơn

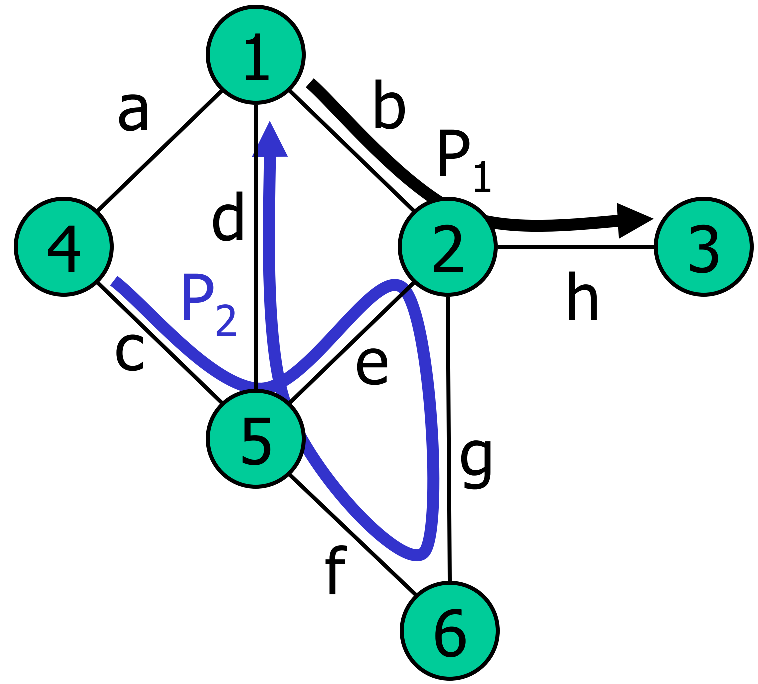


Ví dụ: 1, 2, 5, 3, 4 hoặc 1, a, 2, c, 5, d, 3, e, 4

- Là đường đi đơn

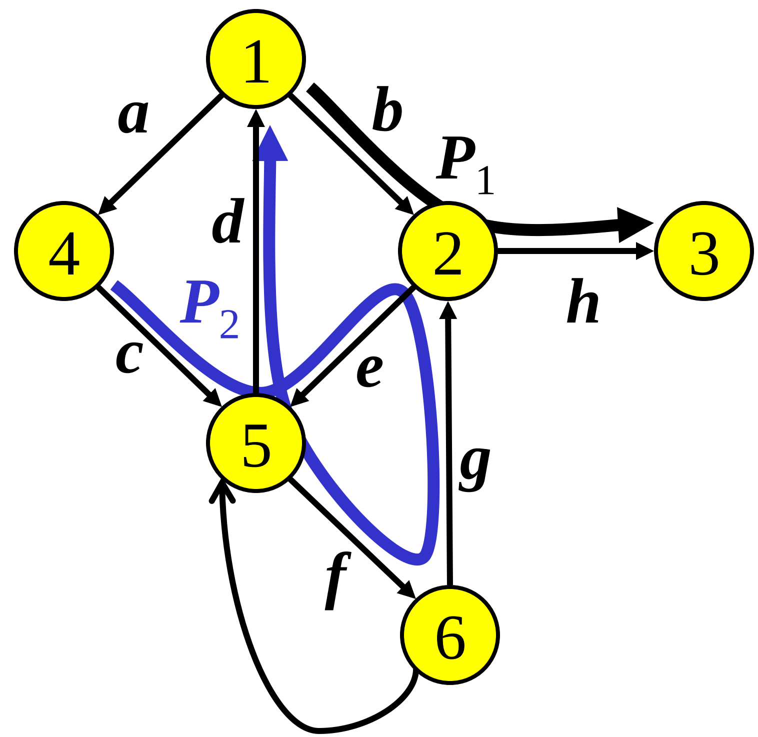
Ví dụ (cont.)

- P_1=(1,b,2,h,3) là đường đi đơn
- P_2=(4,c,5,e,2,g,6,f,5,d,1) là đường đi nhưng không là đường đi đơn



Ví dụ (cont.)

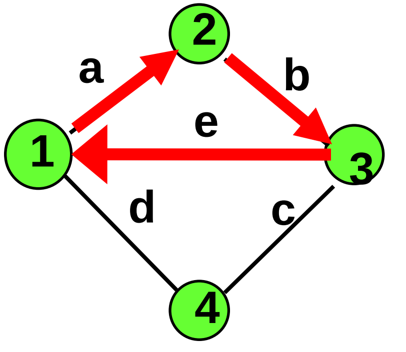
- $P_1=(1, b, 2, h, 3)$ là đường đi đơn
- $P_2=(4,c,\mathbf{5},e,2,g,6,f,\mathbf{5},d,1)$ là đường đi nhưng không là đường đi đơn



Chu trình

- Đường đi cơ bản có đỉnh đầu trùng với đỉnh cuối (tức là $u = v$) được gọi là **chu trình.**
- Chu trình được gọi là **đơn** nếu như ngoại trừ đỉnh đầu trùng với đỉnh cuối, không có đỉnh nào bị lặp lại.

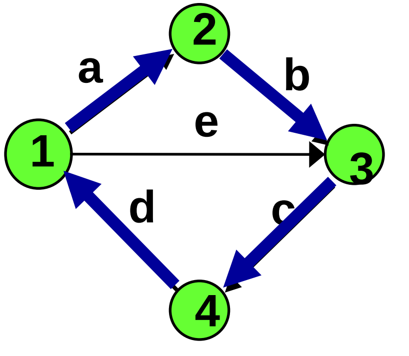
Chu trình (Cycle)



Chu trình

1, 2, 3, 1. (hay 1, a, 2, b, 3, e)

- **Chu trình đơn**



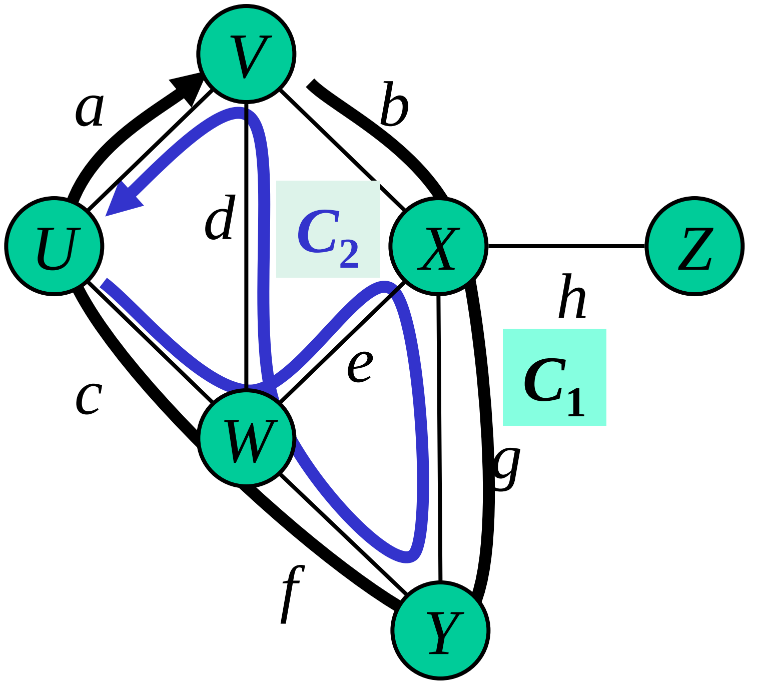
Chu trình: (1, 2, 3, 4, 1) hay

1, a, 2, b, 3, c, 4, d, 1

- **Chu trình đơn**

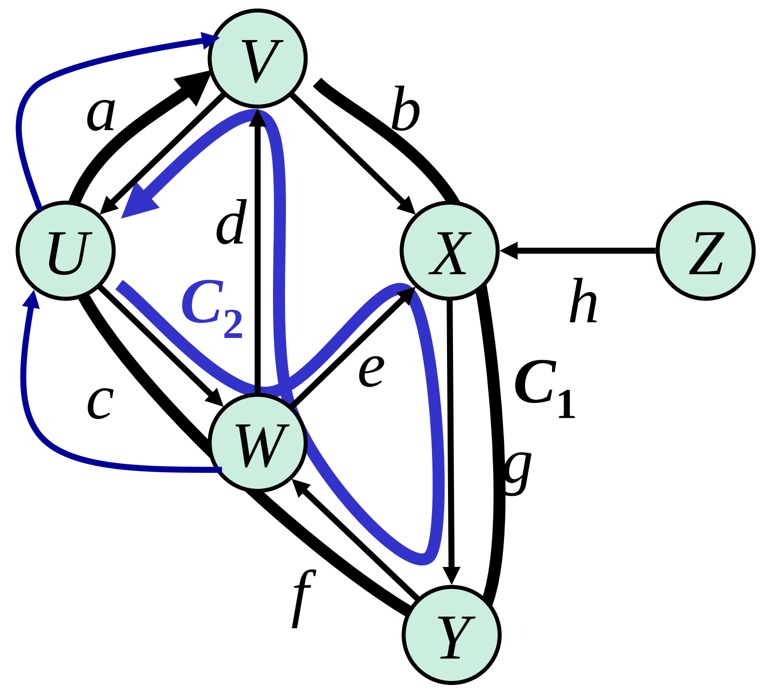
Ví dụ: Chu trình trên đồ thị vô hướng

- $C_1=(V,b,X,g,Y,f,W,c,U,a,V)$ là chu trình đơn
- $C_2=(U,c,W,e,X,g,Y,f,W,d,V,a,U)$ là chu trình nhưng không là chu trình đơn



Ví dụ: Chu trình trên đồ thị có hướng

- $C_1=(V,b,X,g,Y,f,W,c,U,a,V)$ là chu trình đơn
- $C_2=(U,c,W,e,X,g,Y,f,W,d,V,a,U)$ là chu trình nhưng không là chu trình đơn



Chương 1

CÁC KHÁI NIỆM CƠ BẢN

1.1. Đồ thị trong thực tế

1.2. Các loại đồ thị

1.3. Bậc của đỉnh

1.4. Đồ thị con

1.5. Đồ thị đẳng cấu

1.6. Đường đi và chu trình

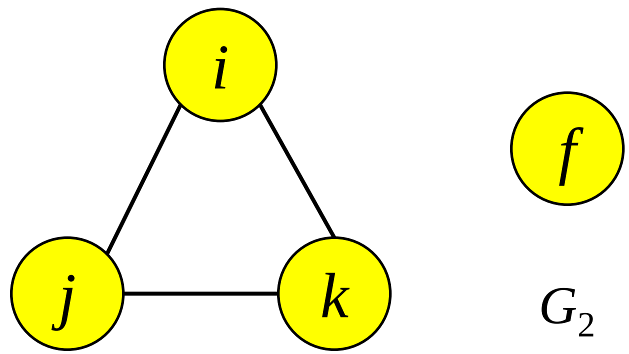
1.7. Tính liên thông

1.8. Một số loại đồ thị đặc biệt

1.9. Tô màu đồ thị

Tính liên thông (Connectedness)

- Đồ thị vô hướng được gọi là *liên thông* nếu luôn tìm được đường đi nối hai đỉnh bất kỳ của nó.
- **Ví dụ**



G_1

G_2

- G_1 và G_2 là các đồ thị liên thông
- Đồ thị G bao gồm G_1 và G_2 không là đồ thị liên thông

Tính liên thông (Connectedness)

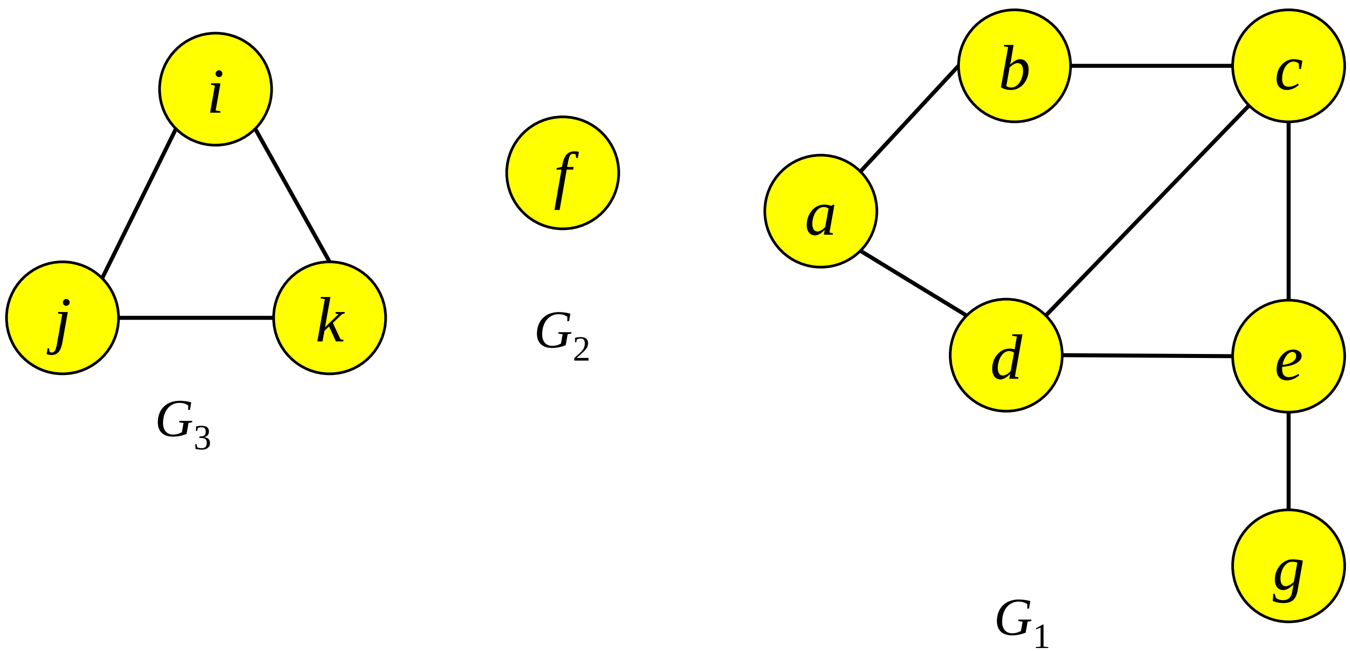
- **Mệnh đề:** Luôn tìm được đường đi đơn nối hai đỉnh bất kỳ của đồ thị vô hướng liên thông.

- **Chứng minh.**

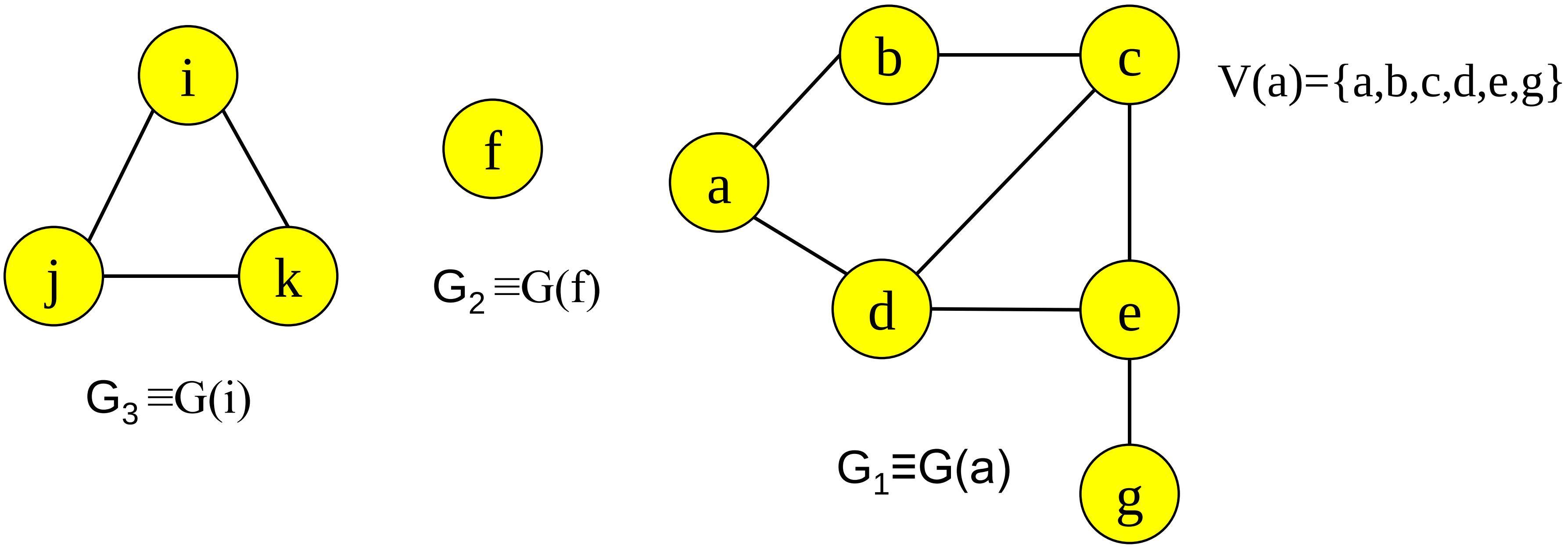
Theo định nghĩa, luôn tìm được đường đi nối hai đỉnh bất kỳ của đồ thị liên thông. Gọi *P* là đường đi ngắn nhất nối hai đỉnh *u* và *v*. Rõ ràng *P* phải là đường đi đơn.

Tính liên thông (Connectedness)

- *Thành phần liên thông* (*Connected component*): Đồ thị con liên thông cực đại của đồ thị vô hướng *G* được gọi là thành phần liên thông của nó.
- **Ví dụ:** Đồ thị *G* có 3 thành phần liên thông G_1, G_2, G_3



Thành phần liên thông



Gỉa sử $v \in V$. Gọi

- $V(v)$ – tập các đỉnh của đồ thị đạt đến được từ v,
- $E(v)$ – tập các cạnh có ít nhất một đầu mút trong $V(v)$.

Khi đó $G(v) = (V(v), E(v))$ là đồ thị liên thông và được gọi là thành phần liên thông sinh bởi đỉnh v. Dễ thấy $G(v)$ là thành phần liên thông sinh bởi mọi đỉnh $u \in V(v)$.

Ví dụ

Ví dụ: Cho G là đồ thị vô hướng $n \geq 2$ đỉnh. Biết rằng

$$\delta(G) = \min\{\deg(v) : v \in V\} \geq (n-1)/2.$$

Chứng minh rằng G liên thông.

Chứng minh.

Phản chứng. Giả sử G không liên thông, khi đó do

$$\delta(G) \geq (n-1)/2,$$

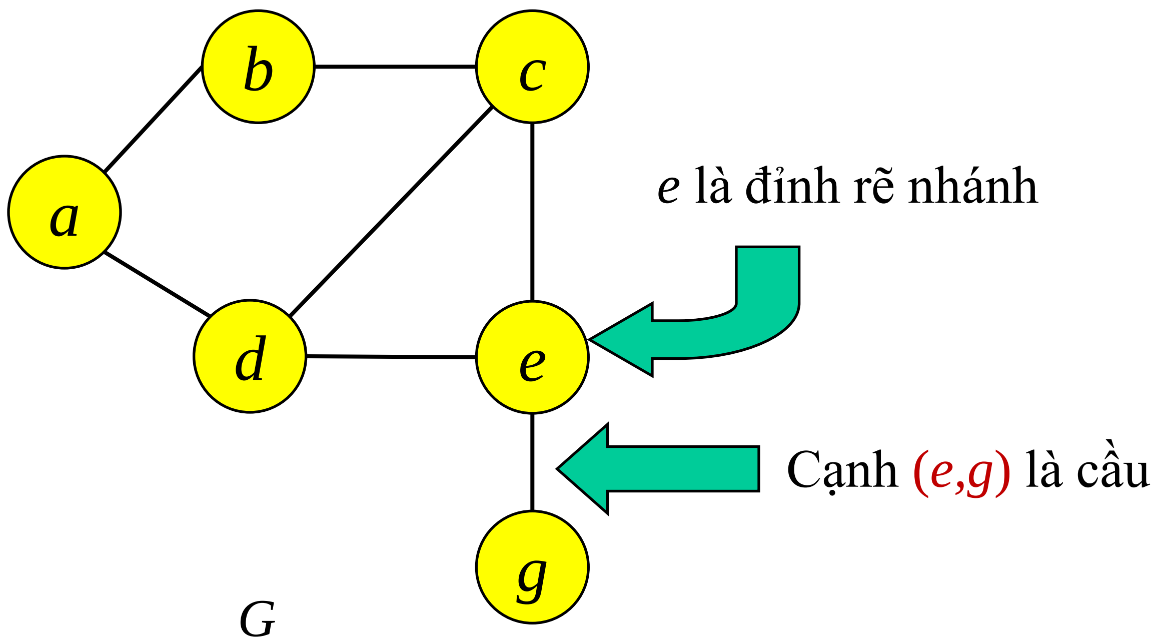
nên mỗi thành phần liên thông phải chứa ít ra

$$(n-1)/2+1 = (n+1)/2 \text{ đỉnh.}$$

Suy ra đồ thị có ít ra $(n+1)$ đỉnh?!

Đỉnh rẽ nhánh và cầu (Connectedness)

- *Đỉnh rẽ nhánh* (*cut vertex*): là đỉnh mà việc loại bỏ nó làm tăng số thành phần liên thông của đồ thị
- *Cầu* (*bridge*): Cạnh mà việc loại bỏ nó làm tăng số thành phần liên thông của đồ thị .
- **Ví dụ:**



Ví dụ

Mệnh đề. Cạnh e của đồ thị liên thông G là cầu iff e không thuộc bất cứ chu trình nào trên G.

Chứng minh

($\Rightarrow$) Cho $\boldsymbol{e}$ là cầu của $\boldsymbol{G}$.

Giả sử $\boldsymbol{e} = (\boldsymbol{u},\boldsymbol{v})$, và giả sử ngược lại là $\boldsymbol{e}$ nằm trên chu trình

$$\boldsymbol{C} : \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}, \ldots, \boldsymbol{x}, \boldsymbol{u}.$$

Khi đó

$$\boldsymbol{C} - \boldsymbol{e} : \boldsymbol{v}, \boldsymbol{w}, \ldots, \boldsymbol{x}, \boldsymbol{u}$$

là đường đi từ u đến v trên đồ thị $\boldsymbol{G} - \boldsymbol{e}$.

Ta sẽ chứng minh: $\boldsymbol{G} - \boldsymbol{e}$ là là liên thông.

(Điều đó sẽ mâu thuẫn với giả thiết e là cầu)

Chứng minh mệnh đề (cont)

Thực vậy, giả sử $\boldsymbol{u_1}, \boldsymbol{v_1} \in \boldsymbol{V}(\boldsymbol{G}-\boldsymbol{e})=\boldsymbol{V}(\boldsymbol{G})$

Do $\boldsymbol{G}$ là liên thông, nên $\exists$ đường đi $\boldsymbol{P}$: $\boldsymbol{u_1}\rightarrow\boldsymbol{v_1}$ trên $\boldsymbol{G}$.

Nếu $\boldsymbol{e} \notin \boldsymbol{P}$, thì $\boldsymbol{P}$ cũng là đường đi trên $\boldsymbol{G}-\boldsymbol{e}$

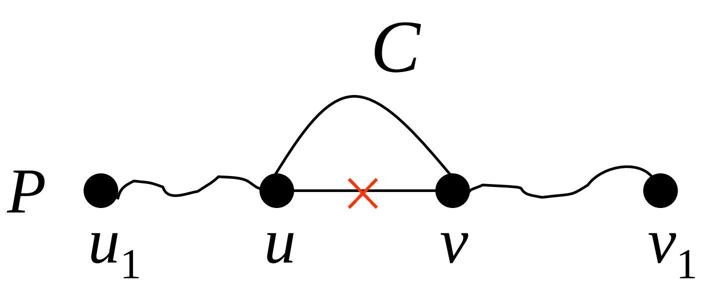
$\Rightarrow \exists$ đường đi $\boldsymbol{u_1}\rightarrow\boldsymbol{v_1}$ trên $\boldsymbol{G}-\boldsymbol{e}$

Nếu $\boldsymbol{e} \in \boldsymbol{P}$, thì

$(\boldsymbol{P} \cup \boldsymbol{C})-\boldsymbol{e}$ là đường đi $\boldsymbol{u_1}\rightarrow\boldsymbol{v_1}$ trên $\boldsymbol{G}-\boldsymbol{e}$ (*xem hình*)

Vậy luôn tìm được đường đi $\boldsymbol{u_1}\rightarrow\boldsymbol{v_1}$ trên $\boldsymbol{G}-\boldsymbol{e}$

Vậy $\boldsymbol{G}-\boldsymbol{e}$ là liên thông.



Chứng minh mệnh đề (cont)

($\Leftarrow$) Giả sử $\boldsymbol{e=(u,v)}$ là cạnh không nằm trên bất cứ chu trình nào của $\boldsymbol{G}$. Khi đó $\boldsymbol{G-e}$ không chứa đường đi $\boldsymbol{u \to v}$.

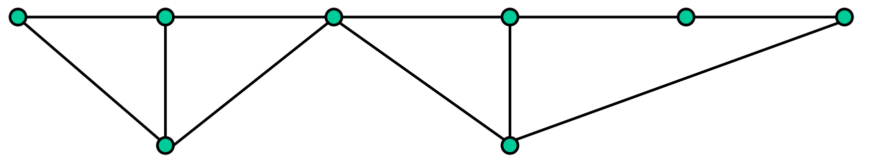
Trái lại, nếu $\boldsymbol{P}$ là đường đi $\boldsymbol{u \to v}$ trên $\boldsymbol{G-e}$, thì $\boldsymbol{P \cup \{(u,v)\}}$ là chu trình trên G chứa $\boldsymbol{e}$ **?!**

k-liên thông (*k*-Connectivity)

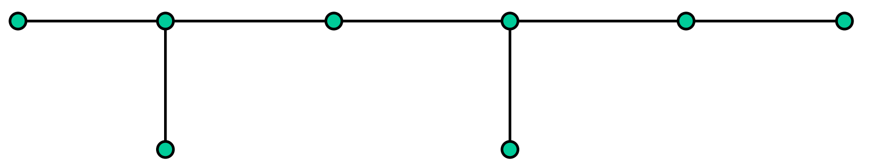
Không phải tất cả các đồ thị liên thông là đồng giá trị!

Q: Hãy đánh giá xem đồ thị nào dưới đây là sơ đồ nối mạng máy tính có giá trị hơn:

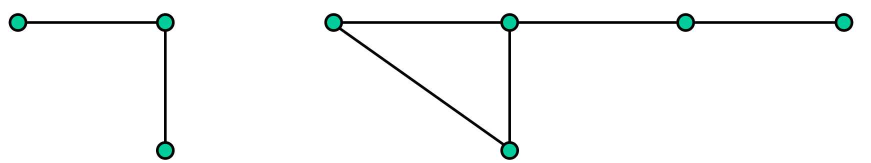
1) G_1



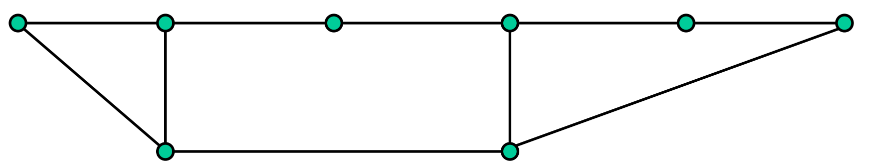
2) G_2



3) G_3



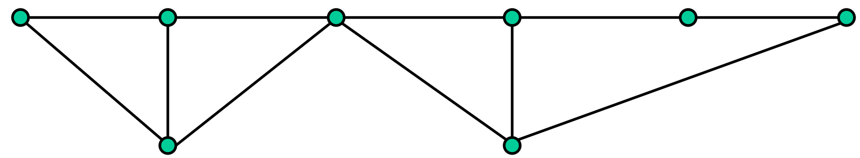
4) G_4



k-Connectivity

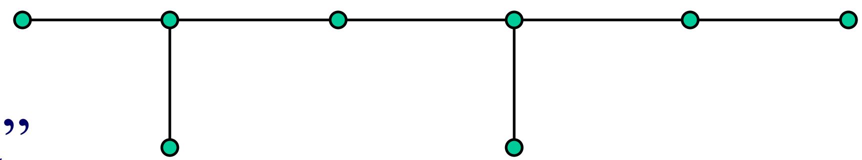
A: Ta muốn mạng máy tính vẫn là thông suốt ngay cả khi có một máy bị hỏng:

1) 2nd best. Vẫn có một điểm yếu— "cut vertex"



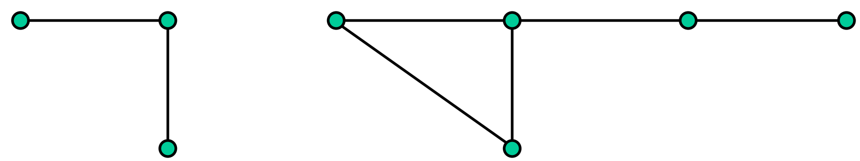
---

2) 3rd best. Thông suốt nhưng mỗi máy đều là điểm "yếu"



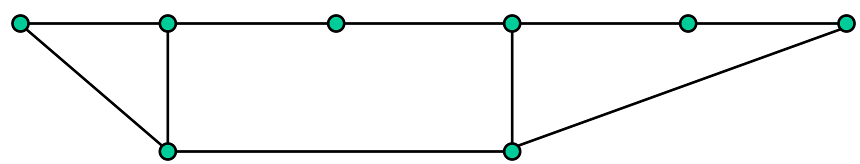
---

3) Tồi nhất! Không thông suốt



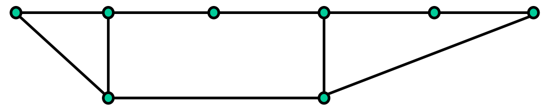
---

4) Tốt nhất! Mạng chỉ không thông suốt nếu hỏng 2 máy



k-Connectivity

Mạng



là tốt nhất bởi vì nó mất tính liên thông chỉ khi có 2 đỉnh bị loại bỏ. Nói cách khác mạng là 2-liên thông (song liên thông).

Định nghĩa. Đơn đồ thị vô hướng liên thông với $n \geq 3$ đỉnh được gọi là song liên thông nếu nó vẫn là liên thông sau khi loại bỏ một đỉnh bất kỳ.

Q: Tại sao lại có điều kiện với số đỉnh?

A: Tránh trường hợp đồ thị chỉ có 1 cạnh.

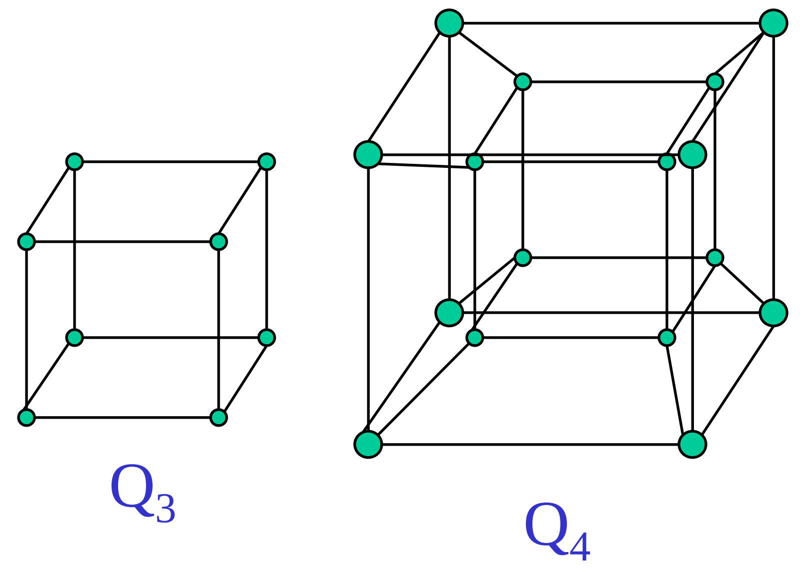
k-liên thông

Tổng quát:

Định nghĩa. Đơn đồ thị vô hướng được gọi là *k*-liên thông nếu như muốn phá vỡ tính liên thông của nó ta phải loại bỏ ít nhất *k* đỉnh.

Ví dụ:

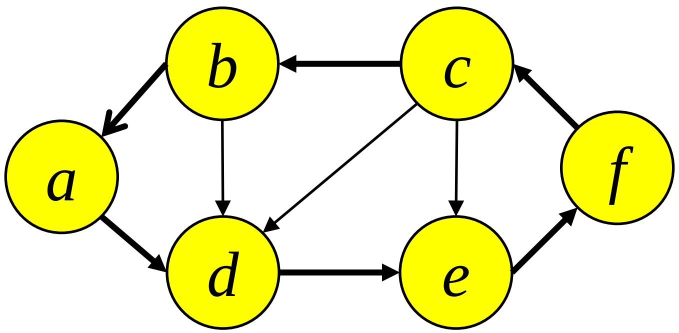
- Q_3 là 3-liên thông
- Q_4 là ?-liên thông

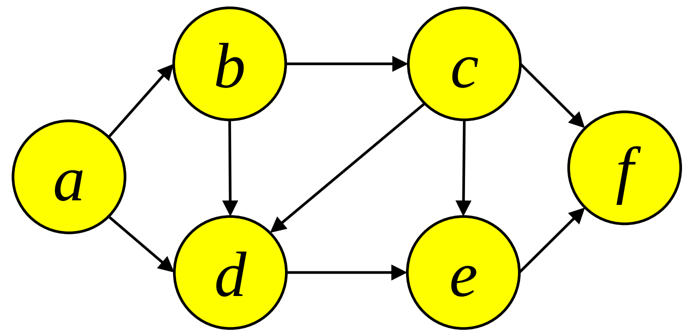


Tính liên thông của Đồ thị có hướng

- Đồ thị có hướng được gọi là *liên thông mạnh* (*strongly connected*) nếu như luôn tìm được đường đi nối hai đỉnh bất kỳ của nó.
- Đồ thị có hướng được gọi là *liên thông yếu* (*weakly connected*) nếu như đồ thị vô hướng thu được từ nó bởi việc bỏ qua hướng của tất cả các cạnh của nó là đồ thị vô hướng liên thông.
- Dễ thấy là nếu *G* là liên thông mạnh thì nó cũng là liên thông yếu, nhưng điều ngược lại không luôn đúng.

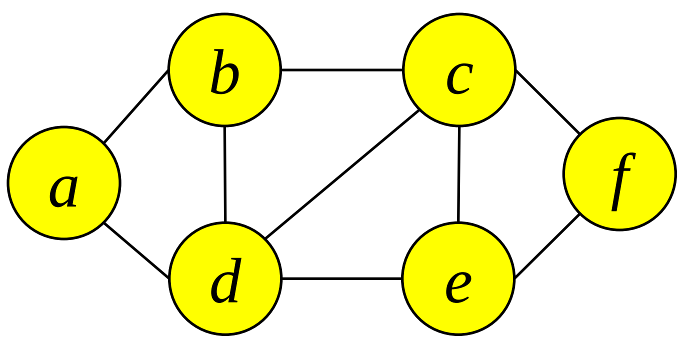
Ví dụ





- Đồ thị liên thông mạnh

Đồ thị liên thông yếu



Chương 1
CÁC KHÁI NIỆM CƠ BẢN

1.1. Đồ thị trong thực tế

1.2. Các loại đồ thị

1.3. Bậc của đỉnh

1.4. Đồ thị con

1.5. Đồ thị đẳng cấu

1.6. Đường đi và chu trình

1.7. Tính liên thông

1.8. Một số loại đồ thị đặc biệt

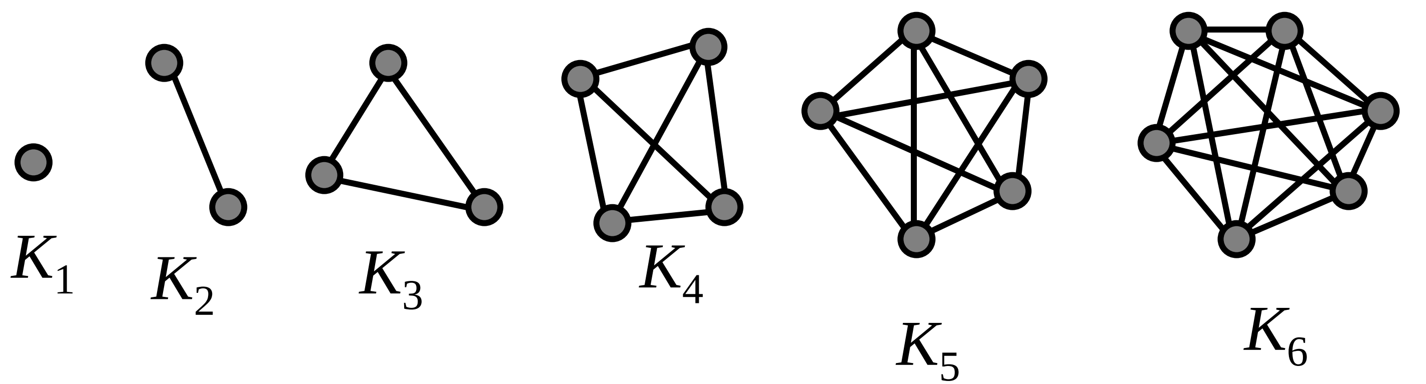
1.9. Tô màu đồ thị

Một số dạng đơn đồ thị vô hướng đặc biệt

- Đồ thị đầy đủ (Complete graphs) K_n
- Chu trình (Cycles) C_n
- Bánh xe (Wheels) W_n
- n-Cubes Q_n
- Đồ thị hai phía (Bipartite graphs)
- Đồ thị hai phía đầy đủ (Complete bipartite graphs) $K_{m,n}$
- Đồ thị chính qui
- Cây và rừng
- Đồ thị phẳng

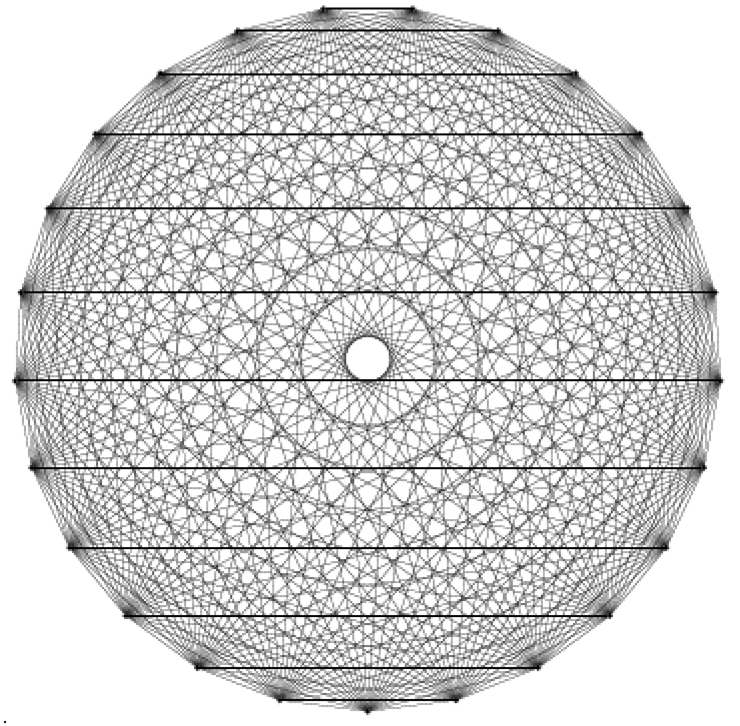
Đồ thị đầy đủ
Complete Graphs

- Với $n \in \mathbf{N}$, đồ thị đầy đủ n đỉnh, K_n, là đơn đồ thị vô hướng với n đỉnh trong đó giữa hai đỉnh bất kỳ luôn có cạnh nối: $\forall u,v \in V$: $u \neq v \leftrightarrow (u,v) \in E$.



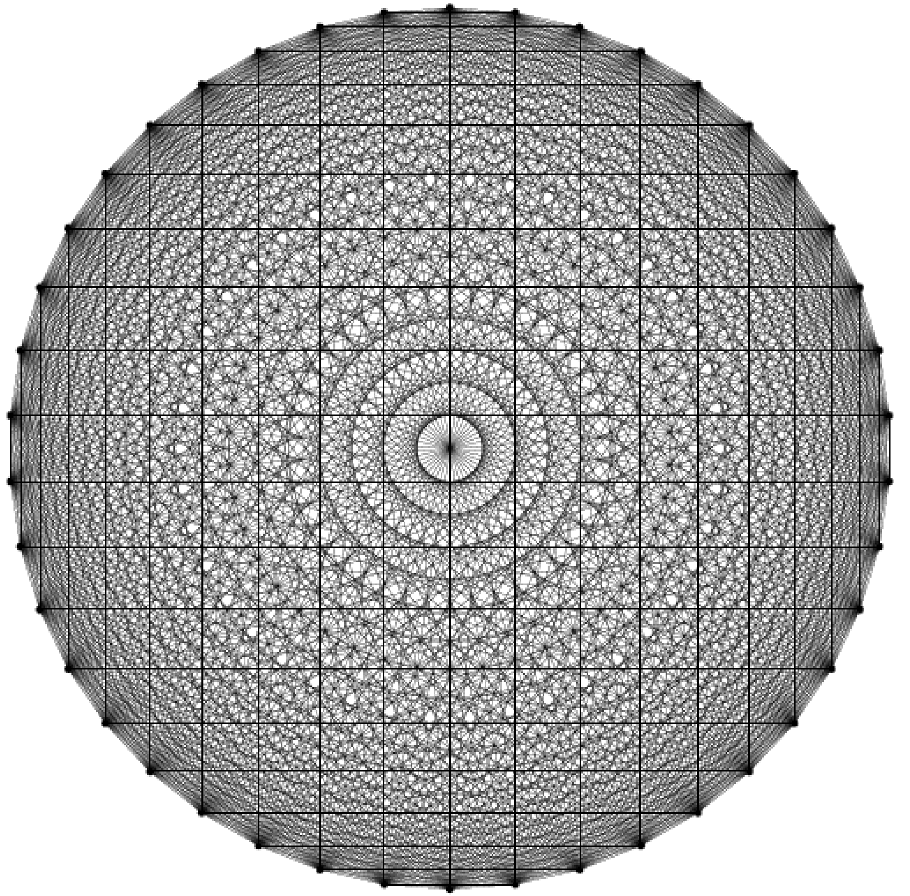
Để ý là K_n *có* $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$ *cạnh.*

Đồ thị đầy đủ
Complete Graphs



K_{25}

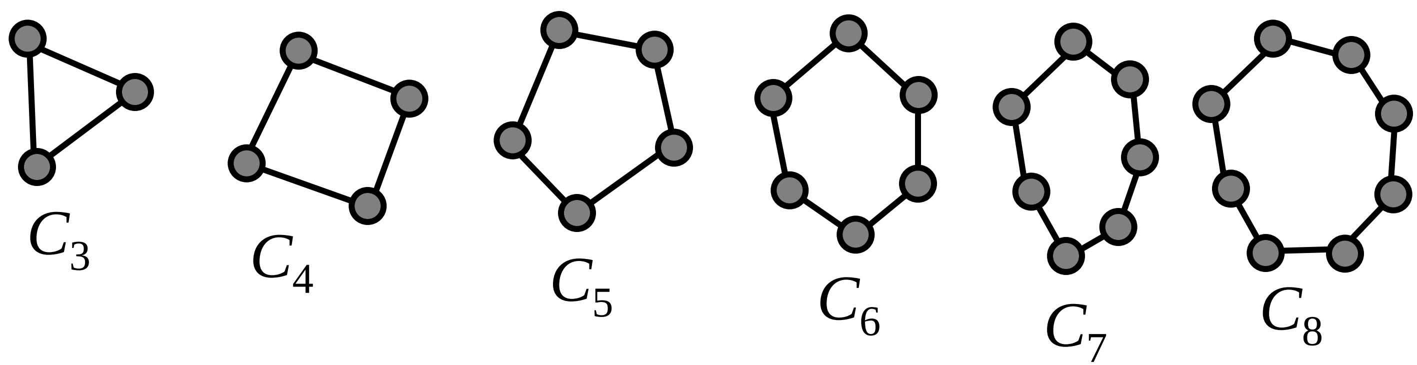
Đồ thị đầy đủ
Complete Graphs



K_{42}

Chu trình (Cycles)

- Giả sử $n \geq 3$. Chu trình *n* đỉnh, C_n, là đơn đồ thị vô hướng với $V=\{v_1,v_2,\ldots,v_n\}$ và $E=\{(v_1,v_2),(v_2,v_3),\ldots,(v_{n-1},v_n),(v_n,v_1)\}$.

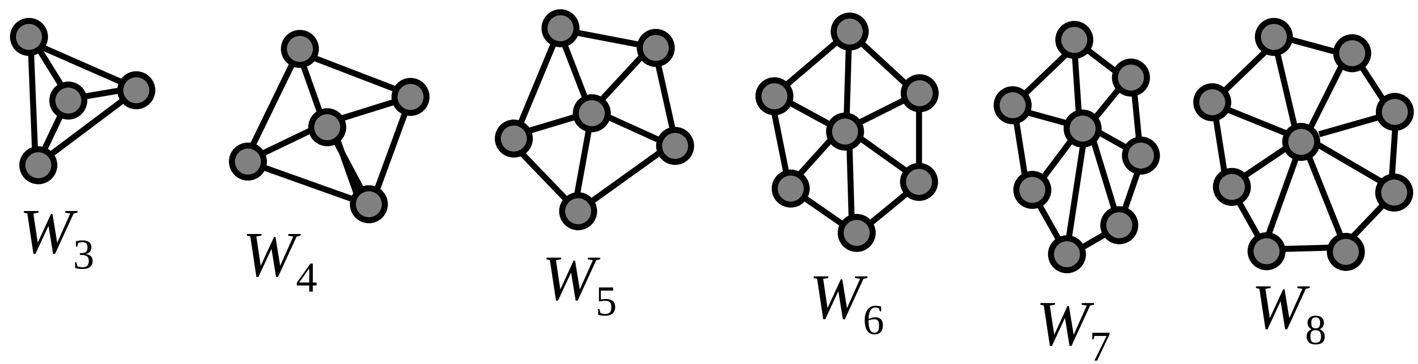


Có bao nhiêu cạnh trong C_n?

Bánh xe (Wheels)

- Với $n \geq 3$, *bánh xe* W_n, là đơn đồ thị vô hướng thu được bằng cách bổ sung vào chu trình C_n một đỉnh v_{hub} và n cạnh nối

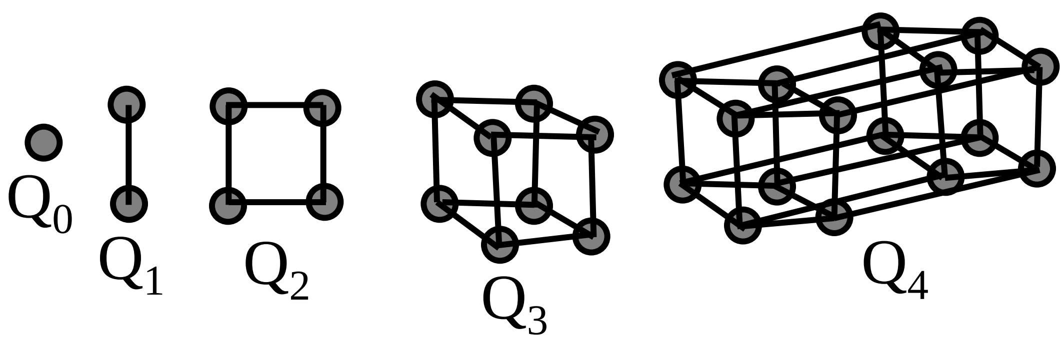
  $\{(v_{\text{hub}}, v_1), (v_{\text{hub}}, v_2), \ldots, (v_{\text{hub}}, v_n)\}$.



Có bao nhiêu cạnh trong W_n?

Siêu cúp
(n-cubes /hypercubes)

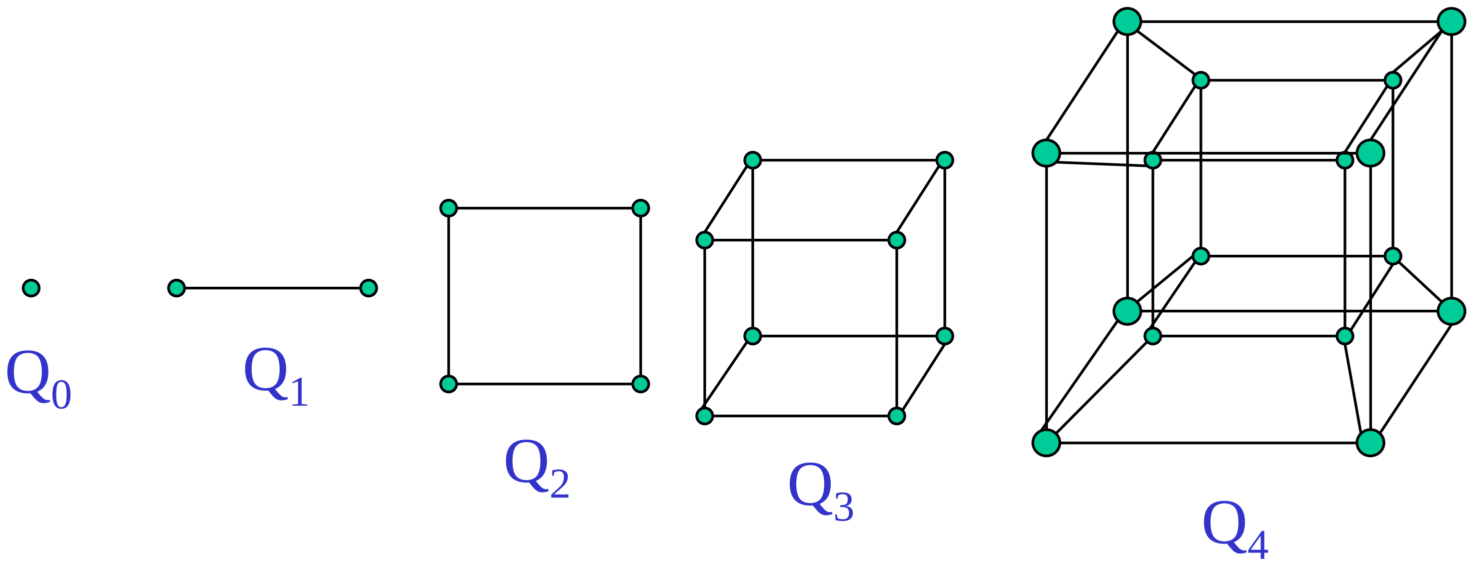
- Với $n \in \mathbf{N}$, siêu cúp Q_n là đơn đồ thị vô hướng gồm hai bản sao của Q_{n-1} trong đó các đỉnh tương ứng được nối với nhau. Q_0 gồm duy nhất 1 đỉnh.



Số đỉnh: 2^n. *Số cạnh: ?*

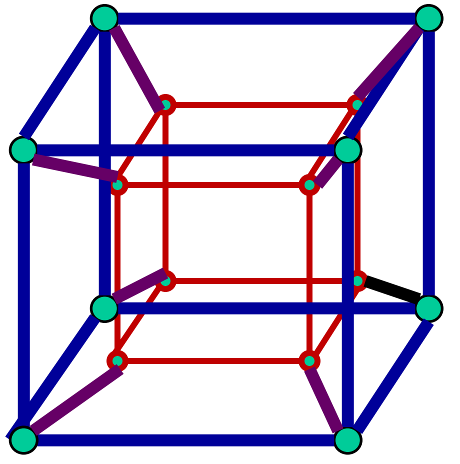
Siêu cúp
(*n*-cubes /hypercubes)

- Với $n \in \mathbf{N}$, siêu cúp Q_n là đơn đồ thị vô hướng gồm hai bản sao của Q_{n-1} trong đó các đỉnh tương ứng được nối với nhau. Q_0 gồm duy nhất 1 đỉnh.



Số đỉnh: 2^n. *Số cạnh: ?*

Siêu cúp Q_4

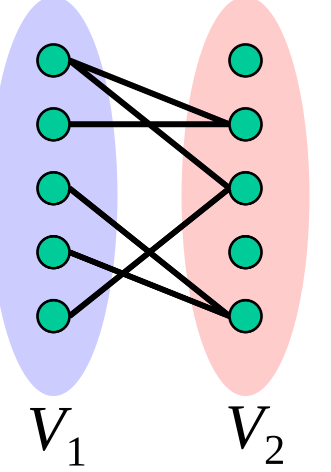


Siêu cúp
(n-cubes /hypercubes)

- Với $n \in \mathbf{N}$, siêu cúp Q_n có thể định nghĩa đệ qui như sau:
  - $Q_0=\{\{v_0\},\varnothing\}$ (một đỉnh và không có cạnh)
  - Với mọi $n \in \mathbf{N}$, nếu $Q_n=(V,E)$, trong đó $V=\{v_1,\ldots,v_a\}$ và $E=\{e_1,\ldots,e_b\}$, thì $Q_{n+1}=(V\cup\{v_1{'},\ldots,v_a{'}\}, E\cup\{e_1{'},\ldots,e_b{'}\}\cup\{(v_1,v_1{'}),(v_2,v_2{'}),\ldots,(v_a,v_a{'})\})$
- Nghĩa là siêu cúp Q_{n+1} thu được từ hai siêu cúp Q_n và Q'_n bằng việc nối các cặp đỉnh tương ứng.

Đồ thị hai phía (Bipartite Graphs)

- **Định nghĩa.** Đồ thị $G=(V,E)$ là hai phía nếu và chỉ nếu $V = V_1 \cup V_2$ với $V_1 \cap V_2 = \varnothing$ và $\forall e \in E: \exists v_1 \in V_1, v_2 \in V_2: e=(v_1,v_2)$.
- **Bằng lời:** Có thể phân hoạch tập đỉnh thành hai tập sao cho mỗi cạnh nối hai đỉnh thuộc hai tập khác nhau.

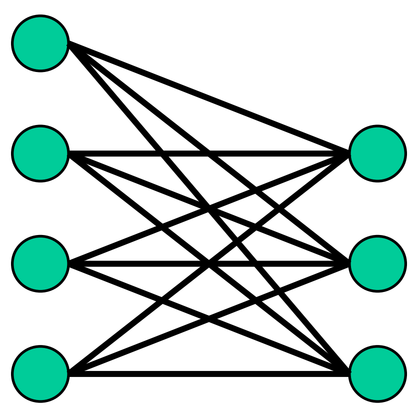


Định nghĩa này là chung cho cả đơn lẫn đa đồ thị vô hướng, có hướng.

Đồ thị hai phía đầy đủ

(Complete Bipartite Graphs)

- Với $m, n \in \mathbf{N}$, đồ thị hai phía đầy đủ $K_{m,n}$ là đồ thị hai phía trong đó $|V_1| = m$, $|V_2| = n$, và
  $E = \{(v_1,v_2)|v_1 \in V_1 \text{ và } v_2 \in V_2\}$.
- $K_{m,n}$ có m đỉnh ở tập bên trái, n đỉnh ở tập bên phải, và mỗi đỉnh ở phần bên trái được nối với mỗi đỉnh ở phần bên phải.



$K_{4,3}$

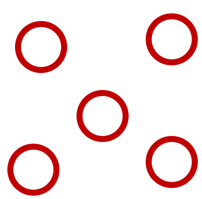
$K_{m,n}$ có ______ đỉnh và ______ cạnh.

Đồ thị chính qui

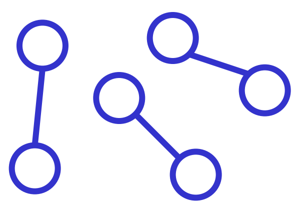
r-regular graph

- **Định nghĩa.** Đồ thị *G* được gọi là đồ thị chính qui bậc *r* nếu tất cả các đỉnh của nó có bậc bằng *r*.

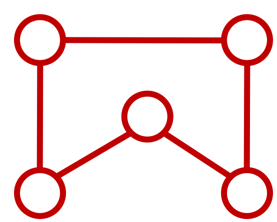
- **Ví dụ:**



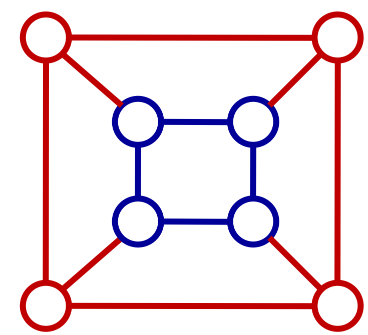
Đồ thị chính qui bậc 0



Đồ thị chính qui bậc 1



Đồ thị chính qui bậc 2

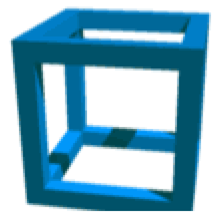


Đồ thị chính qui bậc 3

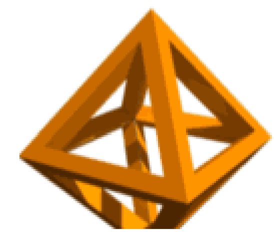
Đồ thị Platonic

- Xét các khối đa diện Platonic trong không gian 3-chiều

Tetrahedron
Tứ diện

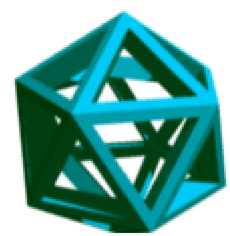


Hexahedron (cube)
Lục diện



Octahedron
Bát diện

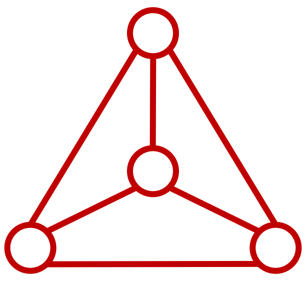
Dodecahedron
Thập nhị diện



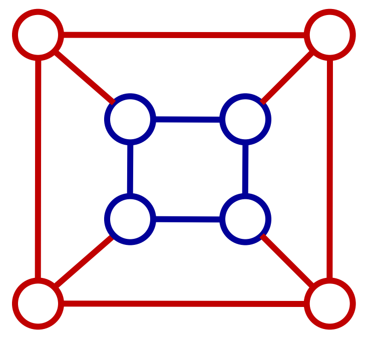
Icosahedron
Thập bát diện

Đồ thị Platonic

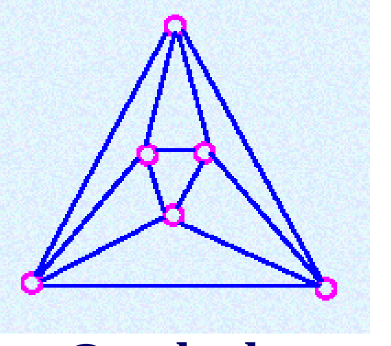
- Đồ thị platonic thu được bằng việc chiếu các khối đa diện tương ứng xuống mặt phẳng



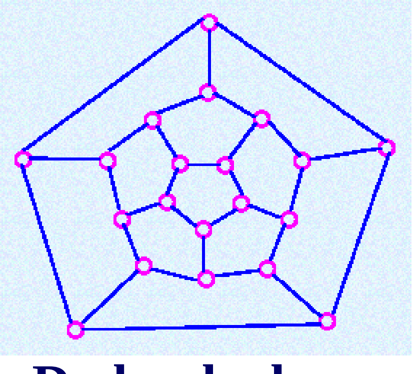
Tetrahedron



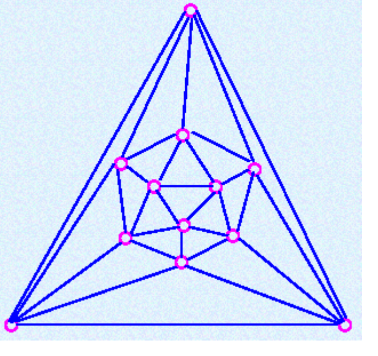
Hexahedron (cube)



Octahedron



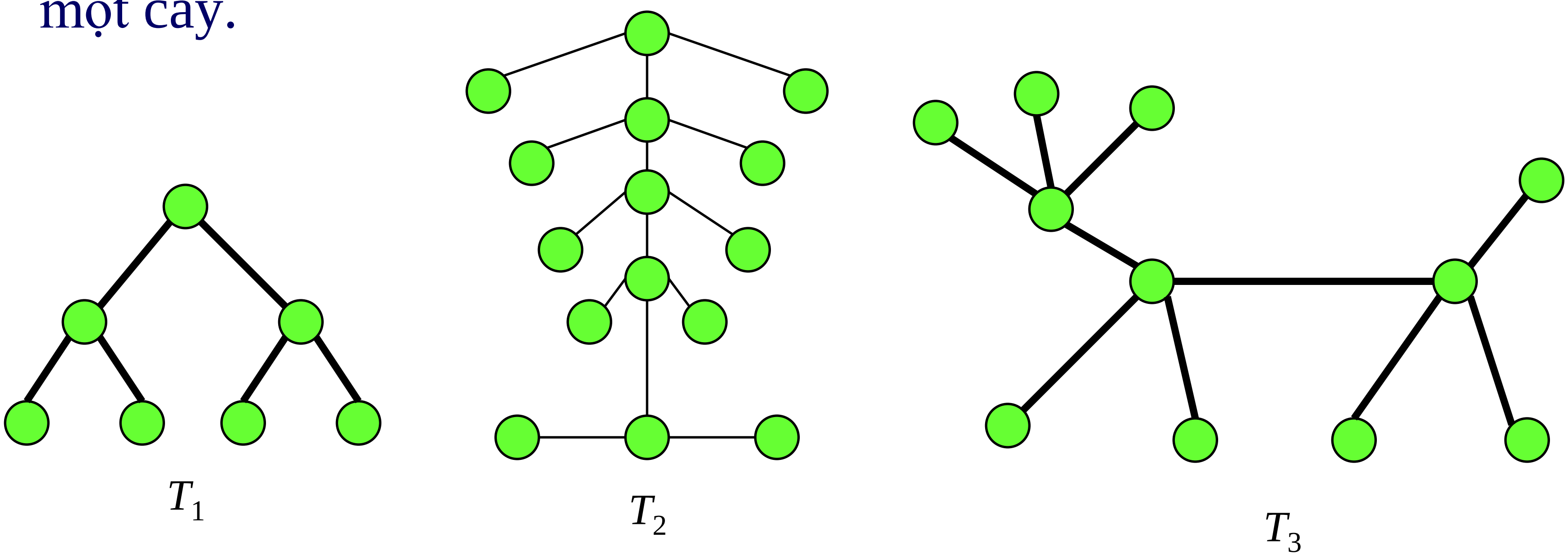
Dodecahedron



Icosahedron

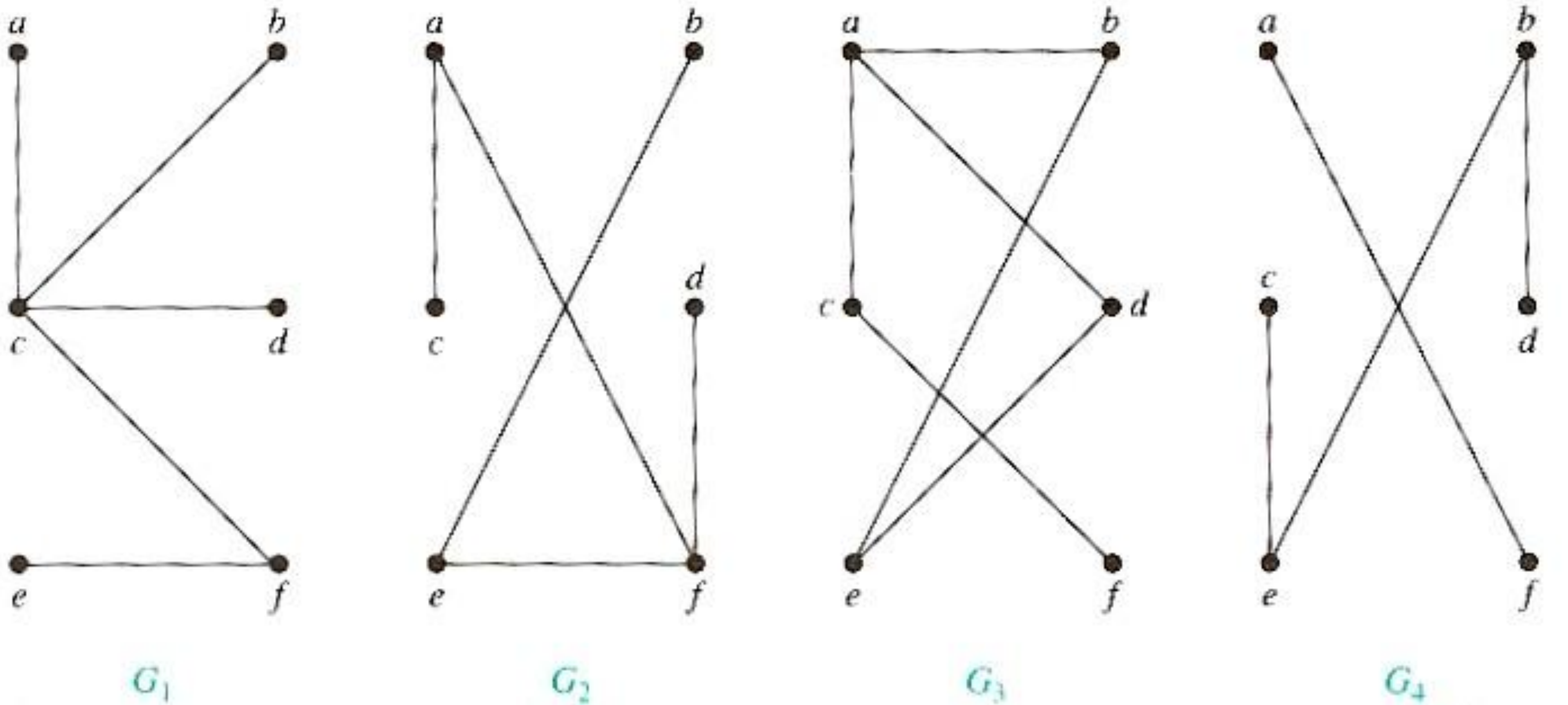
Cây và rừng (Tree and Forest)

- **Định nghĩa.** *Ta gọi cây là đồ thị vô hướng liên thông không có chu trình. Đồ thị không có chu trình được gọi là rừng.*
- Như vậy, rừng là đồ thị mà mỗi thành phần liên thông của nó là một cây.



Rừng F gồm 3 cây T_1, $T_{2,}$, T_3

VÍ DỤ



G_1, G_2 là cây

G_3, G_4 không là cây

Các tính chất cơ bản của cây

- **Định lý.** *Giả sử T=(V,E) là đồ thị vô hướng n đỉnh. Khi đó các mệnh đề sau đây là tương đương:*

  (1) *T là cây;*

  (2) *T không chứa chu trình và có n-1 cạnh;*

  (3) *T liên thông và có n-1 cạnh;*

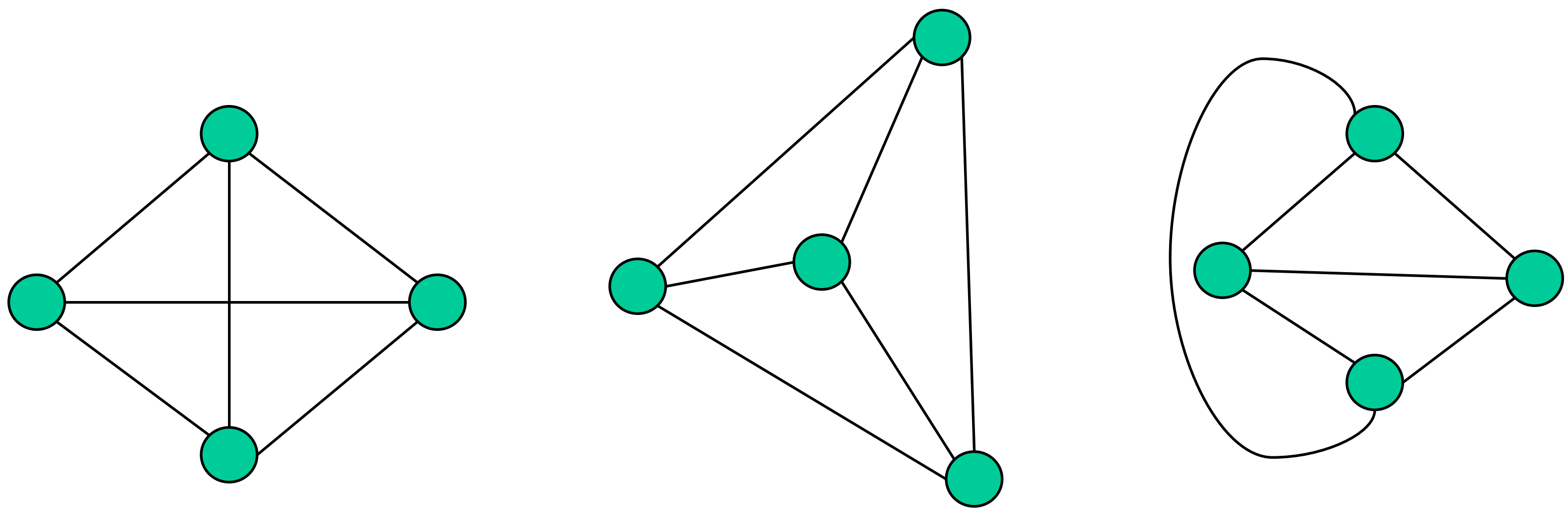
  (4) *T liên thông và mỗi cạnh của nó đều là cầu;*

  (5) *Hai đỉnh bất kỳ của T được nối với nhau bởi đúng một đường đi đơn;*

  (6) *T không chứa chu trình nhưng hễ cứ thêm vào nó một cạnh ta thu được đúng một chu trình.*

Đồ thị phẳng
(Planar Graphs)

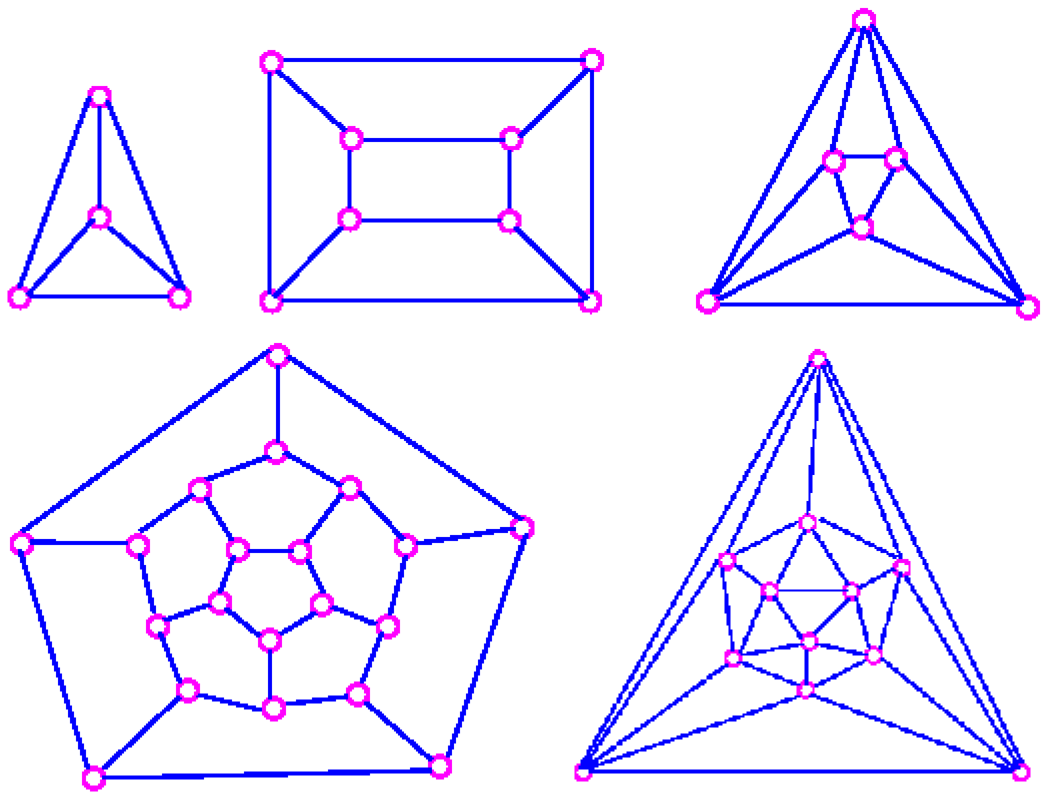
- **Định nghĩa.** Đồ thị vô hướng G được gọi là đồ thị phẳng nếu như có thể vẽ nó trên mặt phẳng sao cho không có hai cạnh nào cắt nhau ngoài ở đỉnh.
- **Ví dụ:** K_4 là đồ thị phẳng?



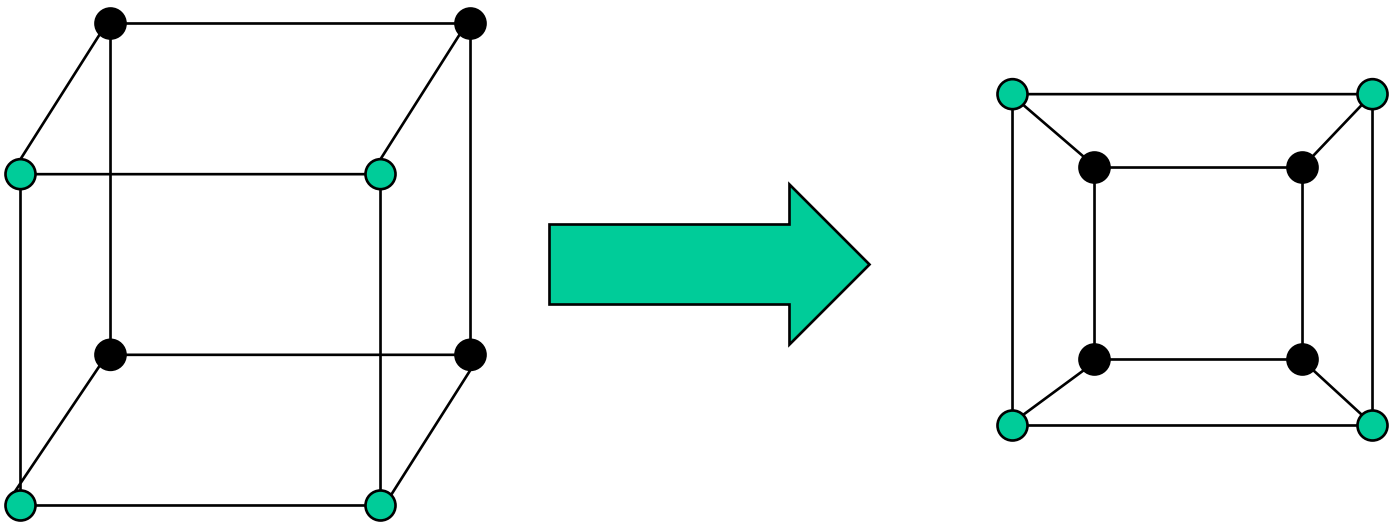
K_4 là đồ thị phẳng!

Các đồ thị Platonic đều phẳng

- Tất cả 5 đồ thị Platonic đều là đồ thị phẳng

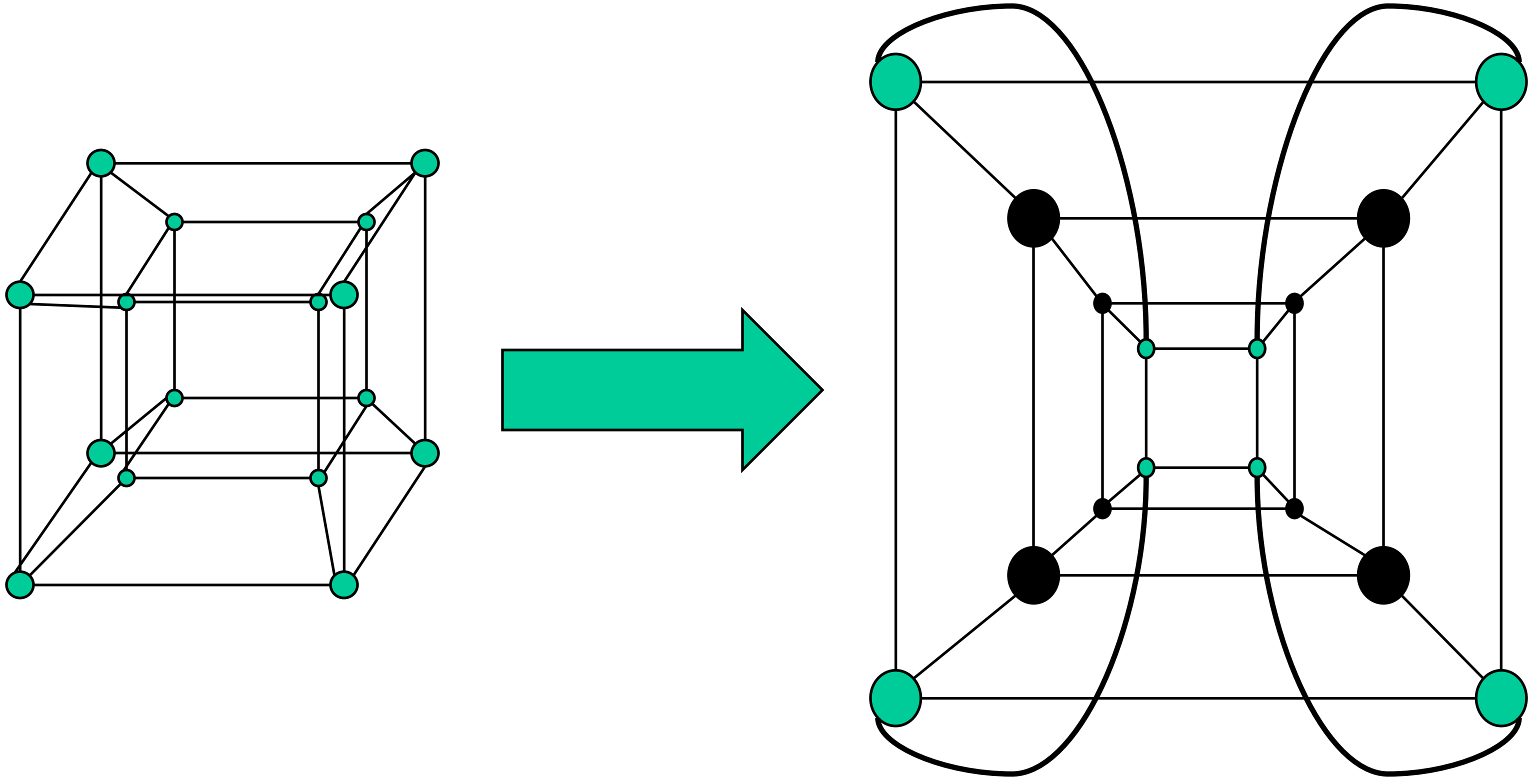


3-Cube là đồ thị phẳng



L25

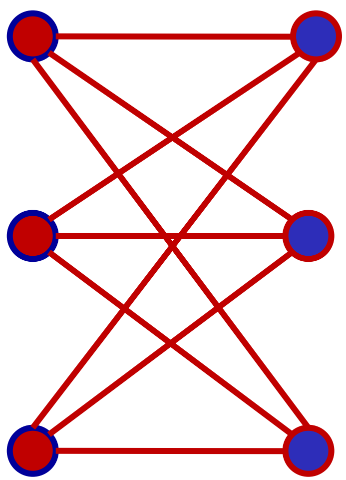
4-Cube có là đồ thị phẳng không?

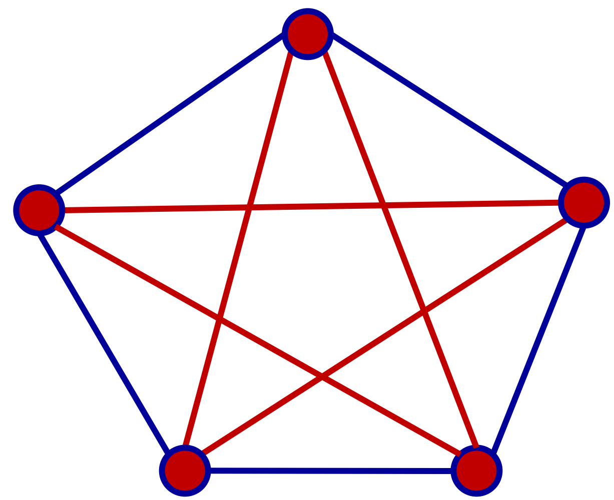


Có vẻ phẳng, nhưng chứng minh bằng cách nào?

$K_{3,3}$ và K_5 không là đồ thị phẳng

- Đồ thị $K_{3,3}$ và K_5 không là đồ thị phẳng





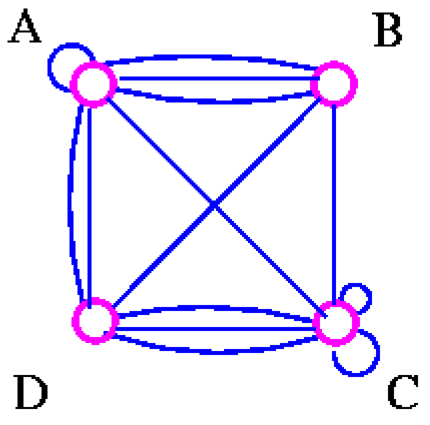
- Mọi cách vẽ $K_{3,3}$ đều phải có ít nhất một giao điểm ngoài đỉnh (gọi là vết cắt).

Khảo sát đồ thị phẳng

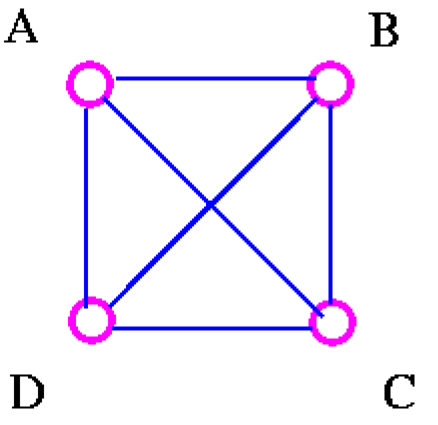
- Để khảo sát đồ thị phẳng ta có thể chỉ hạn chế ở đơn đồ thị. Bởi vì:
- Nếu đồ thị phẳng có cạnh lặp hay là khuyên (loop)
  - Chập các cạnh lặp lại thành một cạnh đơn.
  - Loại bỏ tất cả các khuyên.
- Vẽ đơn đồ thị thu được sao cho không có vết cắt.
- Sau đó chèn vào các khuyên và cạnh lặp.

Khảo sát đồ thị phẳng

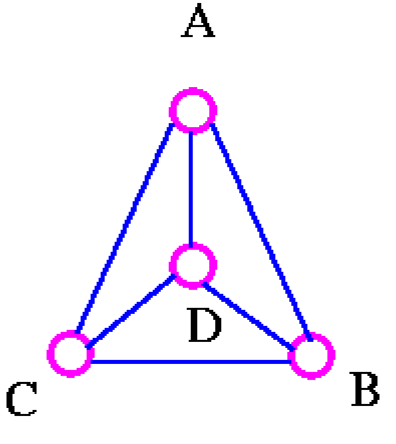
- **Ví dụ:** Xét đồ thị phẳng



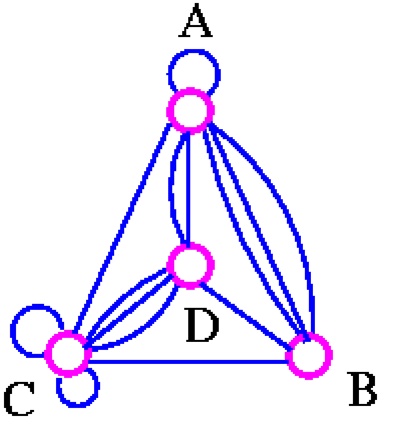
- Loại bỏ khuyên và cạnh lặp:



- Vẽ đơn đồ thị thu được:



- Bổ sung khuyên và cạnh lặp:



Công thức Euler

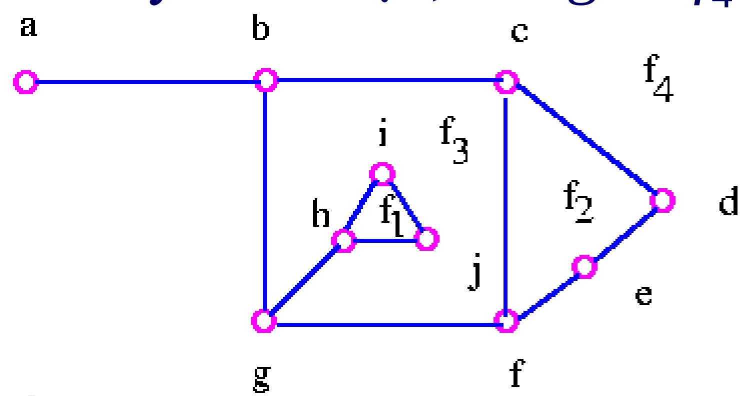
Euler's Formula

- Nếu *G* là đồ thị phẳng, thì mọi cách vẽ phẳng *G* đều chia mặt phẳng ra thành các vùng mà ta sẽ gọi là các **diện** (faces).
- Một trong các diện này là không bị chặn và nó được gọi là **diện vô hạn.**
- Giả sử *f* là một diện nào đó, ta gọi **bậc** của *f* , ký hiệu bởi deg(*f*), là số cạnh trên đường đi vòng quanh biên của diện *f*.
- Nếu tất cả các diện đều có cùng bậc (chẳng hạn, *g*), thì *G* được gọi là **diện chính quy bậc *g*.**

Công thức Euler

Euler's Formula

- **Ví dụ:** Đồ thị *G* sau đây có 4 diện, trong đó f_4 là diện vô hạn.



- Dễ thấy là trong đồ thị trên:

$$\deg(f_1)=3,\ \deg(f_2)=4,\ \deg(f_3)=9,\ \deg(f_4)=8.$$

- Nhận thấy là ***tổng bậc của các diện là bằng 2 lần số cạnh*** của đồ thị, bởi vì mỗi cạnh là biên chung của hai diện (ví dụ, bg, cd, và cf) hoặc xuất hiện hai lần khi đi vòng quanh một diện (ví dụ, các cạnh ab và gh).

Công thức Euler

- Công thức Euler cho biết mối liên hệ giữa số đỉnh, số cạnh và số diện của đồ thị phẳng. Nếu n, m, và f theo thứ tự là số đỉnh, cạnh và diện của đồ thị phẳng liên thông thì ta có $n – m+f = 2$.
- Công thức Euler khẳng định rằng mọi cách vẽ phẳng của đồ thị phẳng liên thông đều cho cùng một số diện như nhau là $2 – n + m$.
- **Theorem (Euler's Formula)** *Let G be a connected planar graph, and let* ***n***, ***m*** *and* ***f*** *denote, respectively, the numbers of vertices, edges, and faces in a plane drawing of G. Then* $\boldsymbol{n – m + f = 2}$.

Chứng minh công thức Euler

- **Chứng minh.** Qui nạp theo số cạnh m.
- ***Cơ sở qui nạp:*** Khi m=0, ta có n=1 và f=1. Do đó $n–m+f = 2$.
- ***Bước qui nạp:*** Giả sử khẳng định đúng cho mọi đồ thị phẳng liên thông có ít hơn m cạnh, trong đó $m \geq 1$, và giả sử rằng G có m cạnh. Nếu G là cây, thì $n=m+1$ và f=1 và do đó công thức là đúng. Mặt khác, nếu G không là cây thì gọi e là một ***cạnh trên chu trình*** của G và xét $G\backslash e$. Đồ thị phẳng liên thông $G\backslash e$ có n đỉnh, m-1 cạnh, và f – 1 diện, do đó theo giả thiết qui nạp

$$\boldsymbol{n - (m - 1) + (f - 1) = 2}$$

từ đó suy ra

$$\boldsymbol{n - m + f = 2.}$$

- Ta sẽ phát biểu một loạt hệ quả thú vị suy ra từ công thức Euler.

Hệ quả

- **Hệ quả 1.** Giả sử *G là đơn đồ thị phẳng liên thông với n đỉnh, trong đó* $n \geq 3$ *và m cạnh. Khi đó* $m \leq 3n - 6$.
- ***Chứng minh.*** Đối với đồ thị G có f diện, từ bổ đề về các cái bắt tay, suy ra $2m$ = (tổng bậc của các diện) $\geq 3f$ (bởi vì bậc của mỗi diện của đơn đồ thị ít nhất là 3), do đó $f \leq 2/3\ m$.
- Kết hợp với công thức Euler

$$n - m + f = 2$$

ta thu được

$$m - n + 2 \leq 2/3\ m.$$

Từ đó suy ra

$$m \leq 3n - 6.$$

K_5 không là đồ thị phẳng

- **Hệ quả 2. K_5 không là đồ thị phẳng.**
- ***Chứng minh.*** Giả sử K_5 là đồ thị phẳng. Do K_5 có 5 đỉnh và 10 cạnh, nên từ bổ đề 1 suy ra

$$10 (3 \times 5) - 6 = 9.$$

Điều phi lý này đã chứng minh K_5 không là đồ thị phẳng.

- **Chú ý:** $K_{3,3}$ có 6 đỉnh và 9 cạnh, và bất đẳng thức $9 \leq (3 \times 6) - 6 = 12$ là đúng. Sự kiện này cho thấy là ta không thể sử dụng hệ quả 1 để chỉ ra $K_{3,3}$ không là phẳng.
- Ta sẽ phải sử dụng hệ quả khác.

Hệ quả 3

- **Hệ quả 3.** *Giả sử G là đơn đồ thị phẳng liên thông với n đỉnh và m cạnh và không chứa tam giác. Khi đó* $m \le 2n - 4$.
- ***Chứng minh.*** Giả sử G có f diện, khi đó từ bổ đề về các cái bắt tay đối với đồ thị phẳng ta có $2m \ge 4f$ (bởi vì bậc của diện của đơn đồ thị không chứa tam giác ít nhất là 4), vì thế $f \le 1/2\, m$.
- Theo công thức Euler ta có

$$n - m + f = 2 \quad \text{hay} \quad m - n + 2 = f.$$

Từ đó ta thu được

$$m - n + 2 \le m/2.$$

Từ đó suy ra

$$m \le 2n - 4.$$

$K_{3,3}$ không là đồ thị phẳng

- **Hệ quả 4.** $K_{3,3}$ không là đồ thị phẳng.
- **Chứng minh.** Giả sử $K_{3,3}$ là phẳng. Do $K_{3,3}$ có 6 đỉnh, 9 cạnh và không chứa tam giác, nên từ hệ quả 3 suy ra

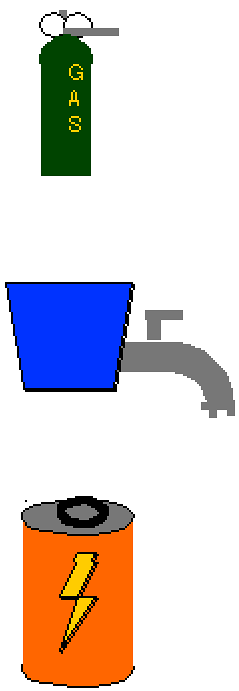
$$9 \le (2\times 6) - 4 = 8.$$

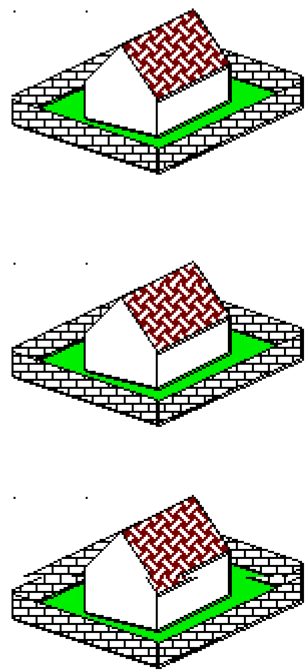
Điều phi lý này đã chứng tỏ $K_{3,3}$ không là đồ thị phẳng.

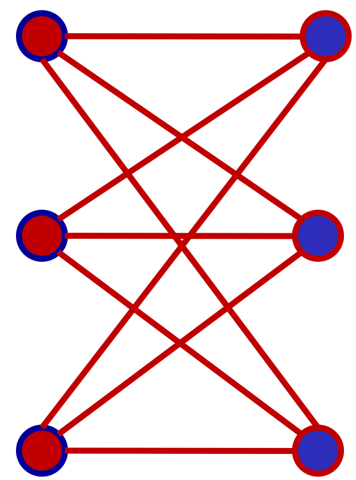
- Kết quả trên đã giải quyết bài toán đố hóc búa về: ba căn nhà và 3 nguồn cung cấp năng lượng

Bài toán xây dựng hệ thống cung cấp năng lượng

- Tìm cách xây dựng hệ thống đường ống nối 3 nguồn cung cấp khí ga, nước và điện cho 3 ngôi nhà sao cho chúng không cắt nhau:







Chứng minh Q_4 không phẳng

Ta chứng minh Q_4 không là đồ thị phẳng.

- Trước hết ta tính số đỉnh và cạnh:

  $|V| = 16$ (gấp đôi số đỉnh của 3-cube)

  $|E| = 32$ (hai lần số cạnh của 3-cube cộng với số đỉnh của 3-cube)
- Bây giờ, giả sử 4-cube là đồ thị phẳng, khi đó theo hệ quả 3 ta phải có:

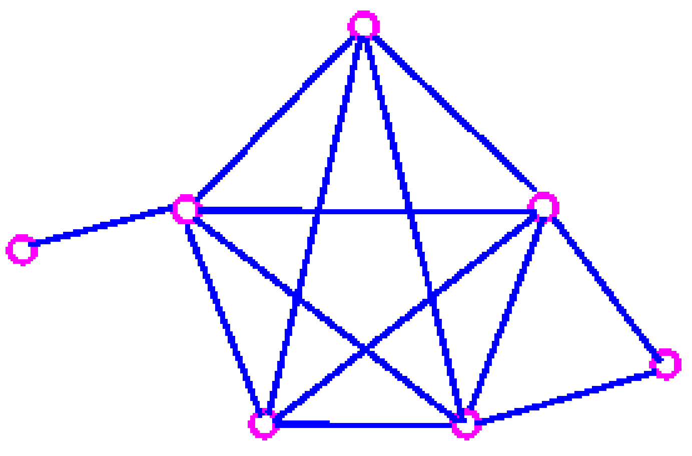
$$32 = m \leq 2n - 4 = 2*16 - 4 = 28 \text{ ?!}$$

Tổng quát: *Q_n không là đồ thị phẳng khi $n \geq 4$.*

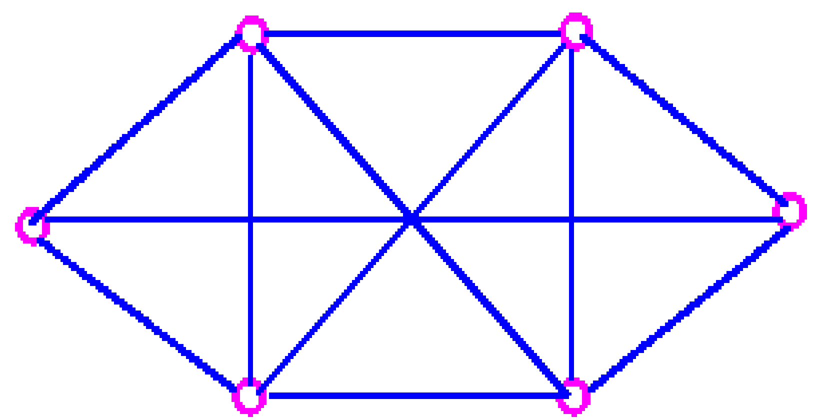
Nhận biết đồ thị phẳng

- Các hệ quả 1 và 3 là các điều kiện cần để đồ thị là phẳng và vì thế chỉ có thể sử dụng để chỉ ra một đồ thị không phải là phẳng. Có nhiều đồ thị thoả mãn các hệ quả này nhưng không phải là phẳng. Vì thế ta cần đưa ra tiêu chuẩn nhận biết đồ thị phẳng. Ta bắt đầu bằng một số nhận xét
- **Nhận xét 1**
  - Không phải mọi đồ thị là phẳng.
  - Ví dụ, ta đã chứng minh K_5 và $K_{3,3}$ không phẳng.
- **Nhận xét 2**
  - Nếu *G* là đồ thị phẳng thì mọi đồ thị con của nó cũng là phẳng;
  - Ta thường sử dụng dạng phủ định
  - **Nhận xét 2a:** Nếu *G* chứa đồ thị không phẳng như đồ thị con thì *G* không là đồ thị phẳng

- **Ví dụ:** Đồ thị G_1 chứa K_5 như là đồ thị con, còn đồ thị G_2 chứa $K_{3,3}$ như là đồ thị con, nên G_1 và G_2 không là đồ thị phẳng:



G_1

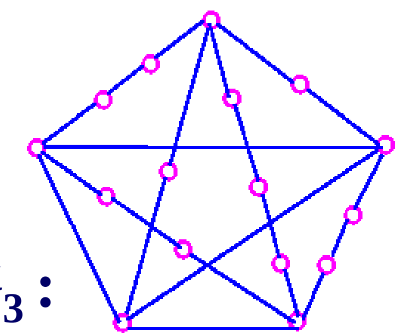


G_2

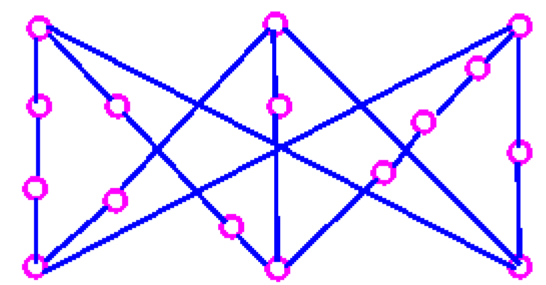
Nhận xét

- **Nhận xét 3.**
  - Nếu G là phẳng thì mọi cách chia cạnh của G đều là đồ thị phẳng.
  - **Nhận xét 3a:** Nếu G thu được bằng cách chia cạnh của một đồ thị không phẳng thì nó không là đồ thị phẳng
- **Ví dụ:** Đồ thị G_3 thu được từ K_5 còn G_4 thu được từ $K_{3,3}$

G_3 :



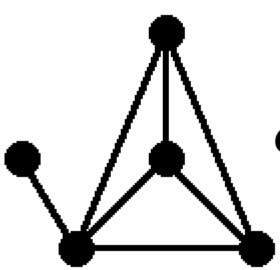
G_4:



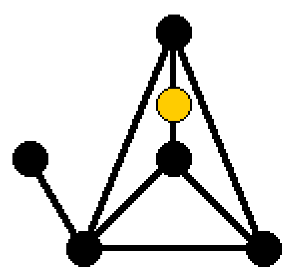
- Từ nhận xét (2a) và (3a) ta suy ra nếu đồ thị G chứa đồ thị con thu được bằng phép chia cạnh của K_5 hoặc $K_{3,3}$ thì G không phẳng..

Nhận biết đồ thị phẳng

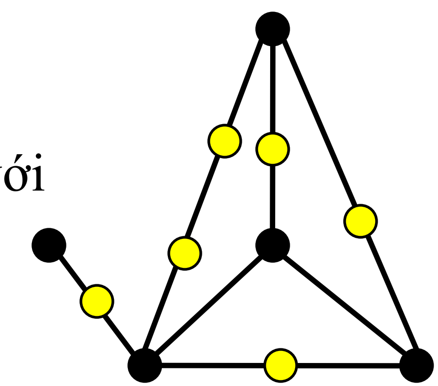
- ***Định nghĩa.*** Ta gọi **phép chia cạnh** (*u*,*v*) của đồ thị *G* là việc thêm vào *G* một đỉnh *w*, loại bỏ cạnh (*u*,*v*) và thêm vào hai cạnh (*u*,*w*) và (*w*,*v*).
- ***Định nghĩa.*** Hai đồ thị *G* và *H* được gọi là đồng phôi (homeomorphic) nếu ta có thể thu được chúng từ đồ thị nào đó bởi các phép chia cạnh.
- **Ví dụ:**



đồng phôi với

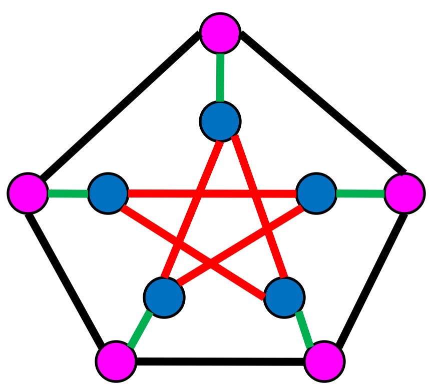


đồng phôi với

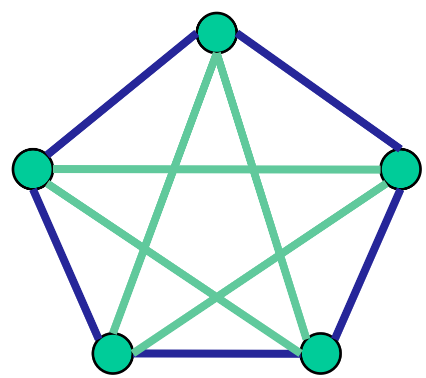


Định lý Kuratowski

- ***Định lý Kuratowski* (1930).** Đồ thị *G* là đồ thị phẳng khi và chỉ khi nó không chứa đồ thị con đồng phôi với K_5 hoặc $K_{3,3}$.
- **Ví dụ:** Đồ thị Petersen không là đồ thị phẳng bởi vì nó là đồng phôi với đồ thị K_5



Đồ thị Petersen



K_5

K. Kuratowski
1896-1980
Poland

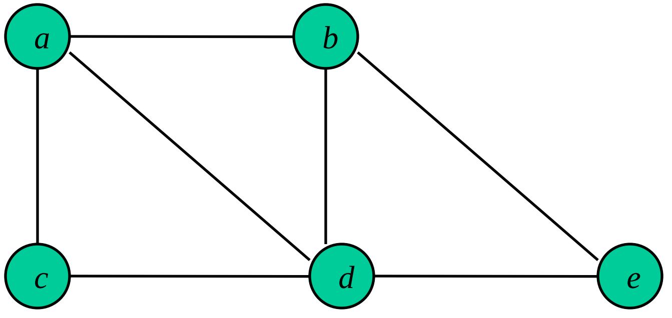
Đồ thị Euler

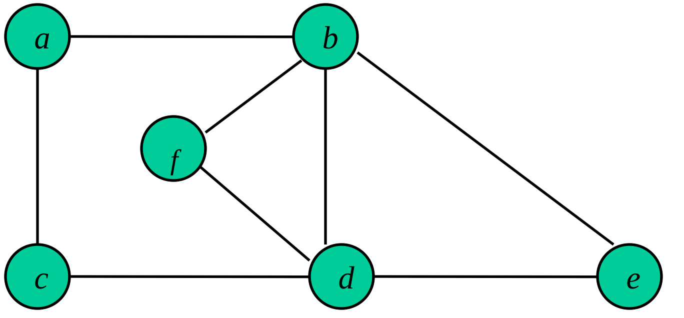
- Định nghĩa
- Nhận biết đồ thị Euler

Đồ thị Euler

- ***Chu trình Euler*** trong đồ thị *G* là chu trình đi qua mỗi cạnh của *G* đúng một lần.
- ***Đường đi Euler*** trong đồ thị *G* là đường đi qua mỗi cạnh của *G* đúng một lần.
- Đồ thị có chu trình Euler được gọi là ***đồ thị Euler***.
- Đồ thị có đường đi Euler được gọi là ***đồ thị nửa Euler.***
- Rõ ràng mọi đồ thị Euler đều là nửa Euler.

Ví dụ





Đồ thị nửa Euler

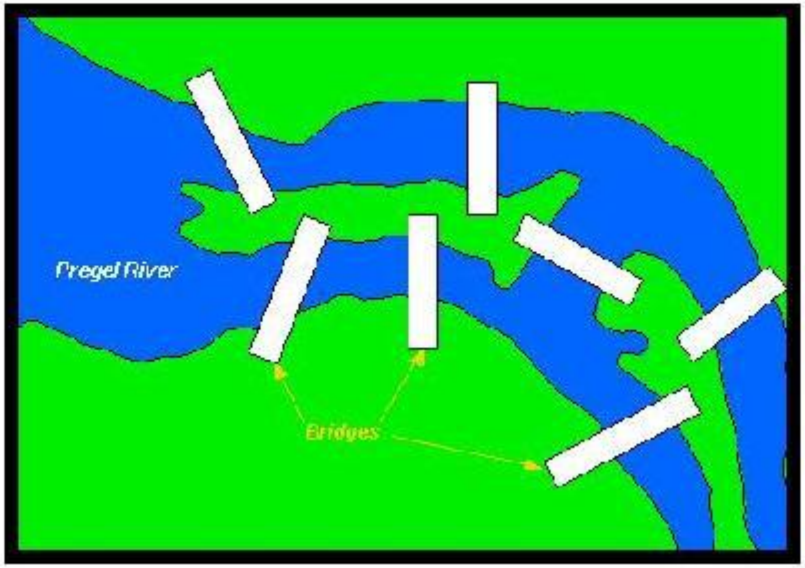
a, c, d, b, e, d, a, b

Đồ thị Euler

a, c, d, e, b, d, f, b, a

Bài toán về 7 cái cầu ở Königsberg

- Hiện nay là Kaliningrad (thuộc Nga)
- Sông Pregel

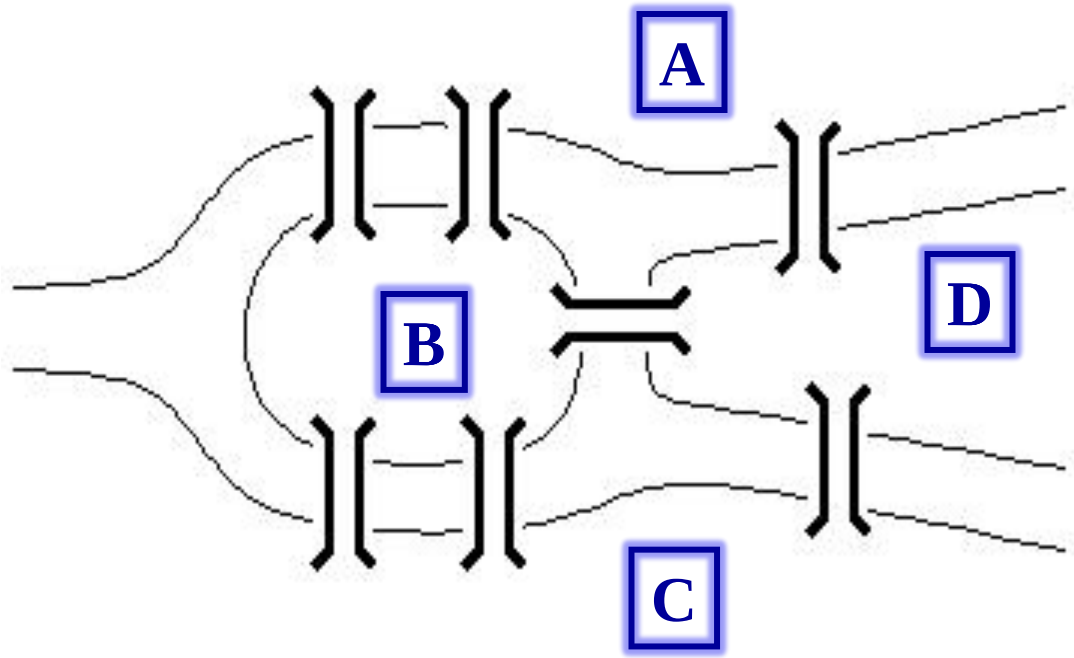


Leonhard Euler

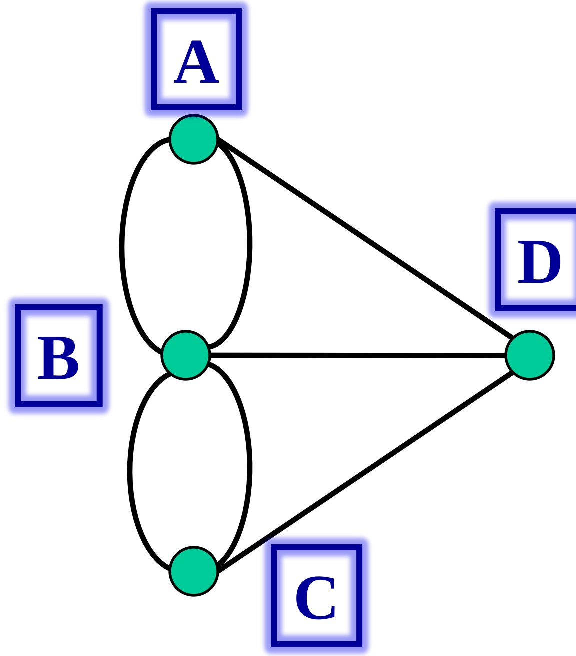
1707-1783

Bài toán về 7 cái cầu ở Königsberg

- Tồn tại hay chăng cách đi qua tất cả 7 cái cầu mỗi cái đúng một lần rồi lại quay về vị trí xuất phát?



Sơ đồ 7 cái cầu

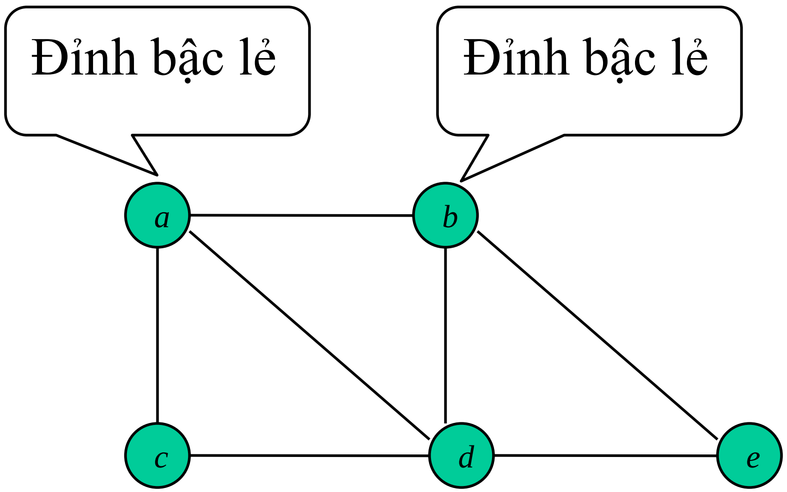


Đa đồ thị vô hướng tương ứng

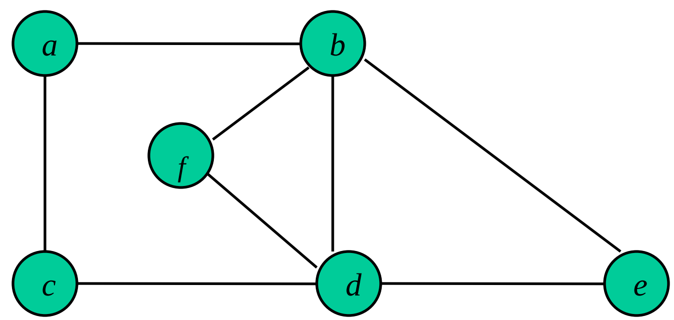
Định lý Euler

- **Định lý:** Đa đồ thị vô hướng liên thông có chu trình Euler khi và chỉ khi nó không có đỉnh bậc lẻ.
- **Proof:**
    - (→) Đi vòng quanh chu trình Euler để tính bậc của các đỉnh, mỗi lần đi qua một đỉnh bậc của nó tăng lên 2.
    - (←) Xem trong giáo trình.
- **Định lý:** Đa đồ thị vô hướng liên thông có đường đi Euler khi và chỉ khi nó có không quá 2 đỉnh bậc lẻ.
  – Một đỉnh sẽ là đỉnh xuất phát, còn đỉnh kia sẽ là đỉnh kết thúc của đường đi Euler.

Ví dụ



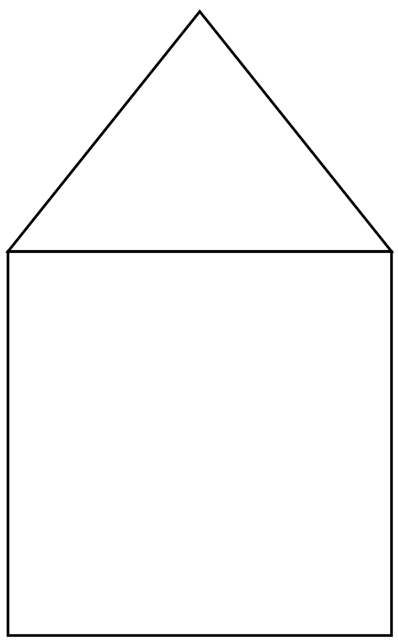
Đồ thị nửa Euler

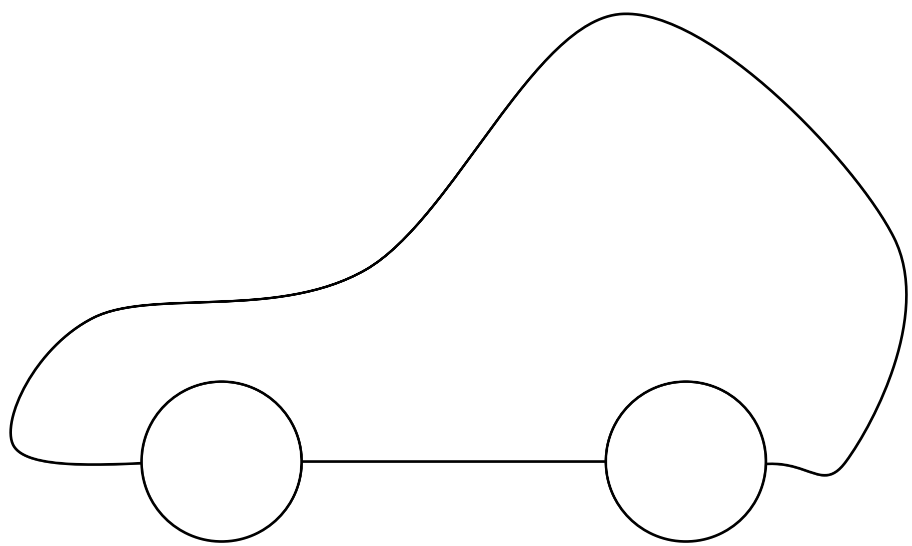


Đồ thị Euler

Vẽ một nét

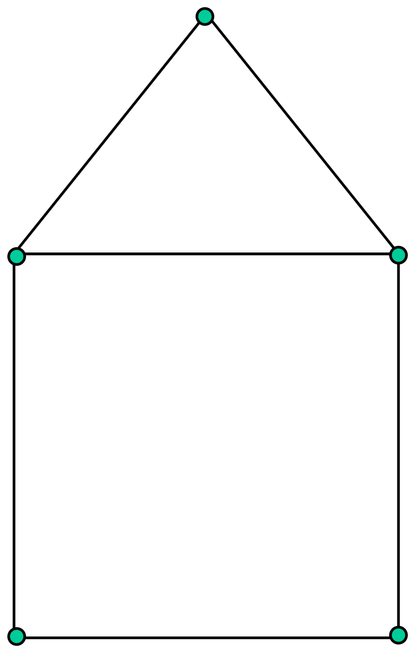
Hình nào trong các hình sau đây có thể tô bởi bút chì mà không được nhấc bút khỏi mặt giấy cũng như không được tô lại bất cứ đoạn nào (vẽ bởi một nét)?

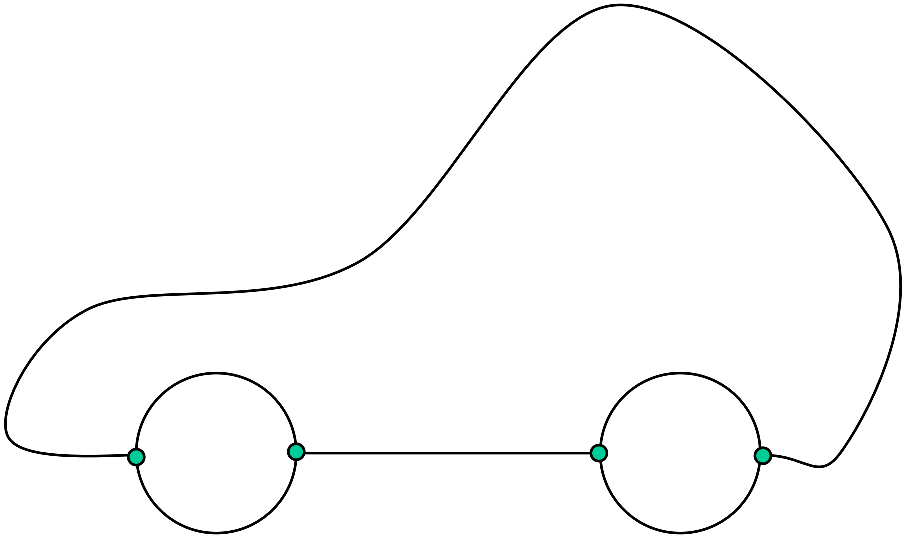




Vẽ một nét

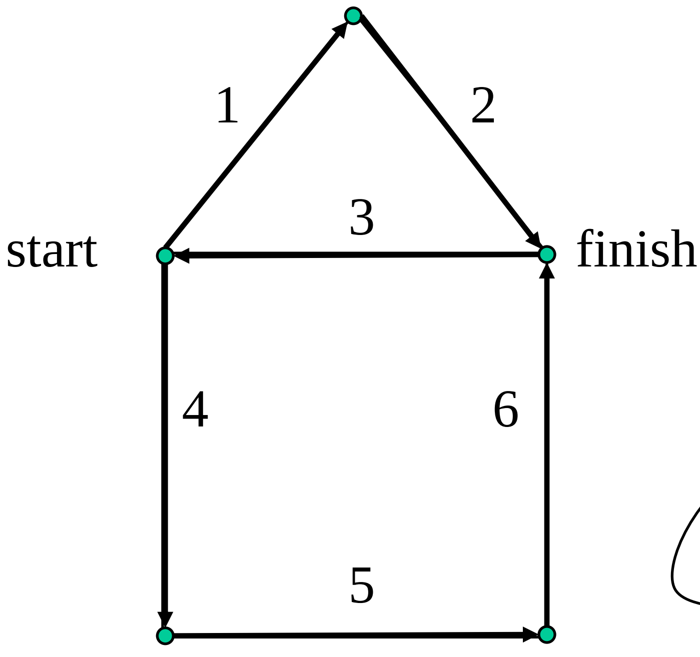
Trong ngôn ngữ đồ thị: Đồ thị nào trong hai đồ thị sau đây có đường đi Euler?

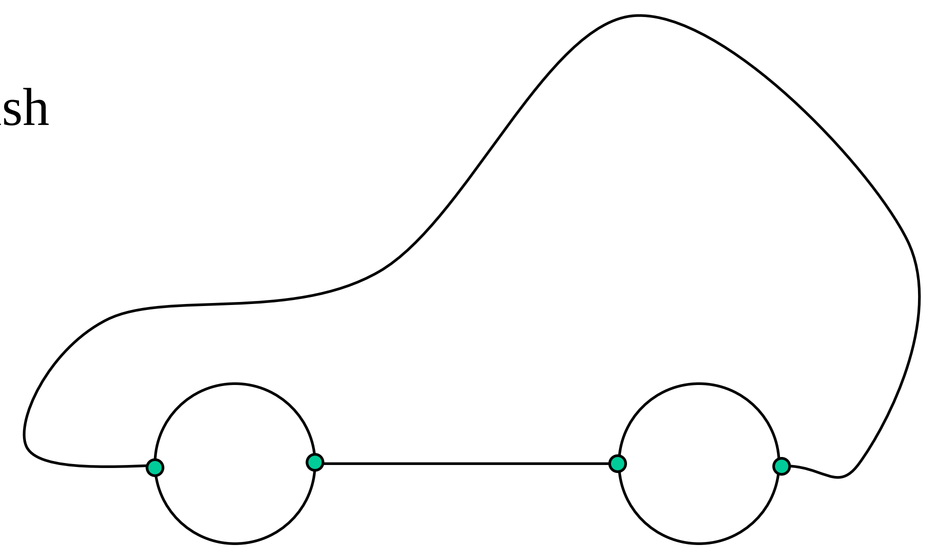




Vẽ một nét – Đường đi Euler

Trả lời: Ngôi nhà vẽ được bởi một nét, còn ôtô thì không thể.





Đồ thị Hamilton

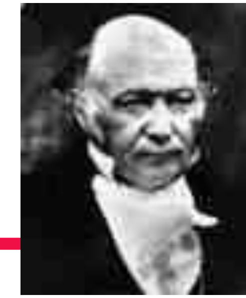
- Định nghĩa
- Nhận biết đồ thị Hamilton

Đồ thị Hamilton

- ***Chu trình Hamilton*** trong đồ thị *G* là chu trình đi qua mỗi đỉnh của *G* đúng một lần.
- ***Đường đi Hamilton*** trong đồ thị G là đường đi qua mỗi đỉnh của *G* đúng một lần.
- Đồ thị có chu trình Hamilton được gọi là ***đồ thị Hamilton***.
- Đồ thị có đường đi Hamilton được gọi là ***đồ thị nửa Hamilton.***
- Rõ ràng mọi đồ thị Hamilton đều là nửa Hamilton

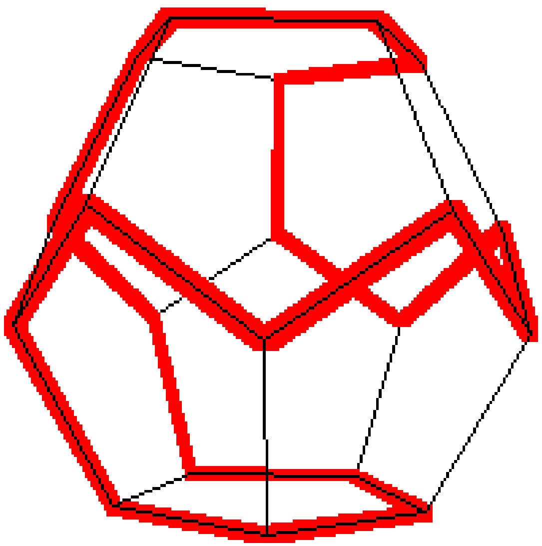
Trò chơi vòng quanh thế giới

(Round-the-World Puzzle)

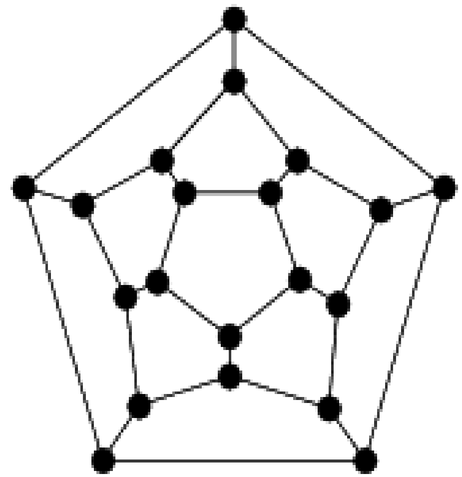


Sir William Rowan Hamilton 1805-1865

- Bạn có thể chỉ ra cách đi qua tất cả các đỉnh của dodecahedron (thập nhị diện) mỗi đỉnh đúng một lần?`



Dodecahedron puzzle

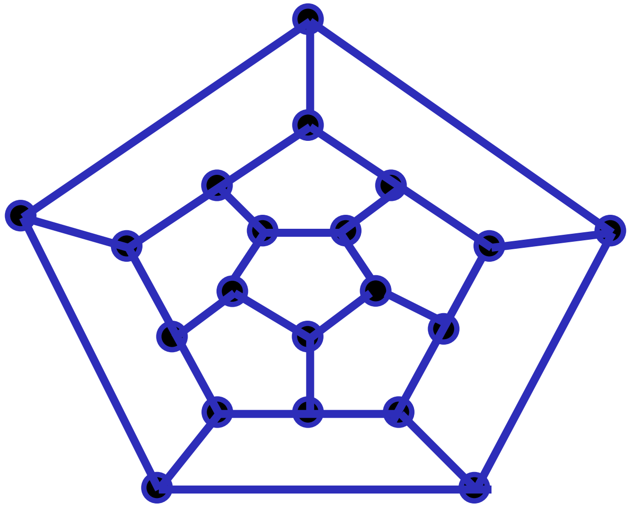


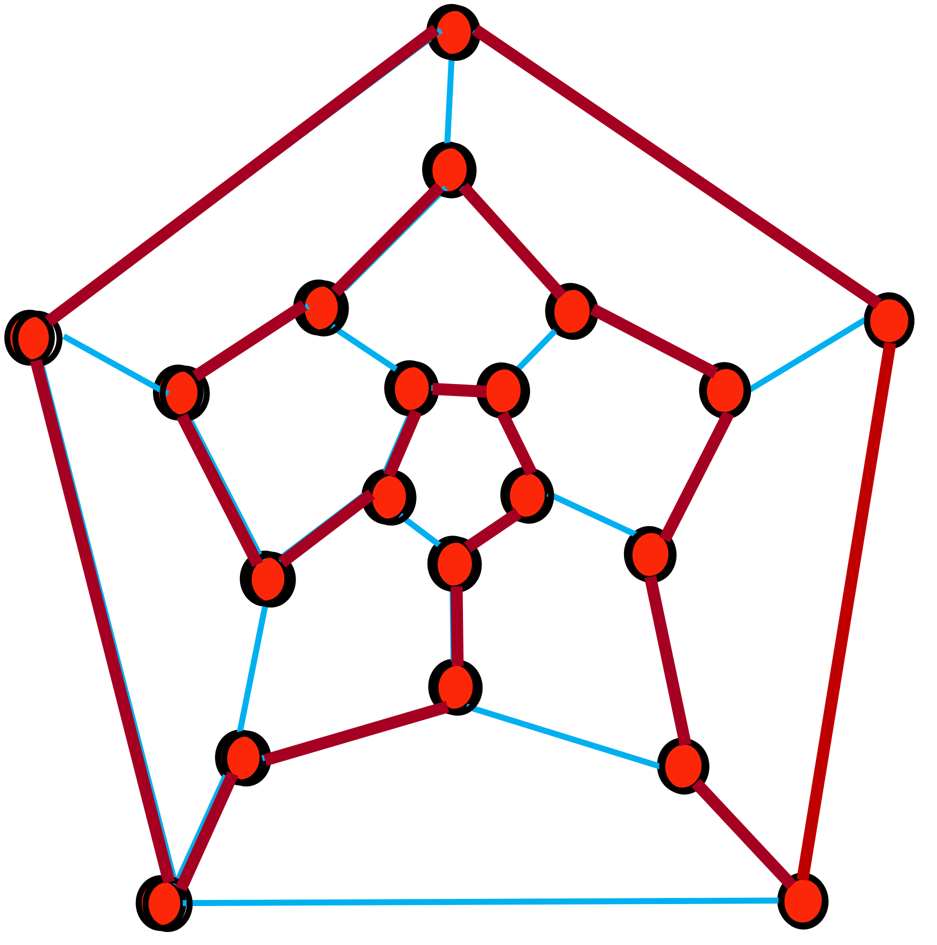
Đồ thị tương ứng

Hộp trò chơi

Ví dụ

- Đồ thị Hamilton





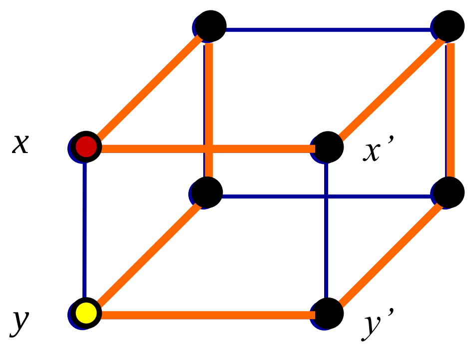
Ví dụ

Ví dụ: CM Q_n ($n \geq 3$) là đồ thị Hamilton.

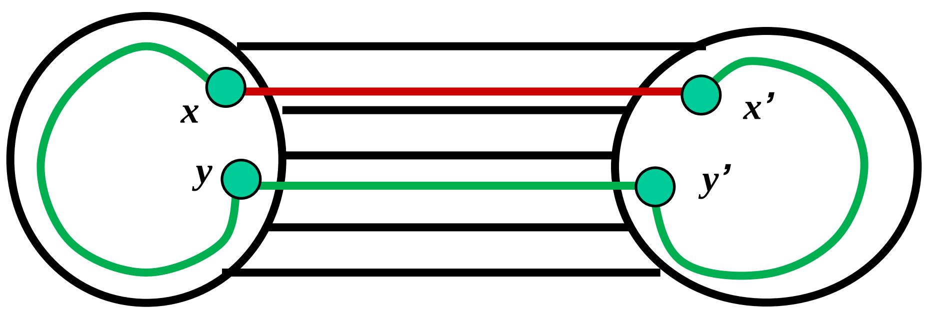
Chứng minh. Qui nạp theo ***n*****.**

Cơ sở: *n*=3 đúng

Chuyển qui nạp: Giả sử Q_{n-1} là hamilton. Xét Q_n:



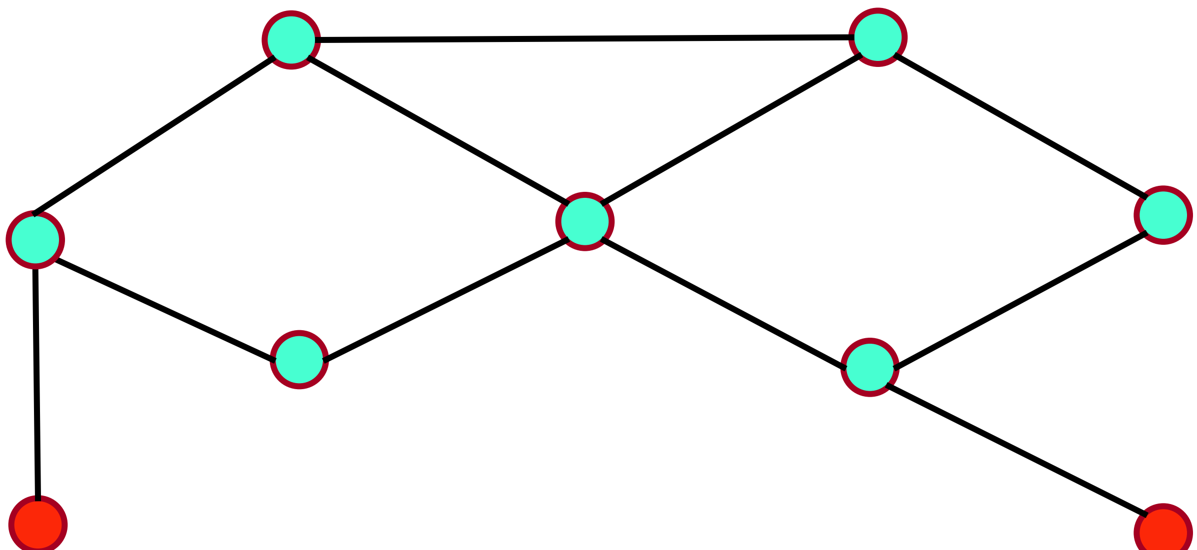
3 - cube



(n –1)-cube (n –1)-cube

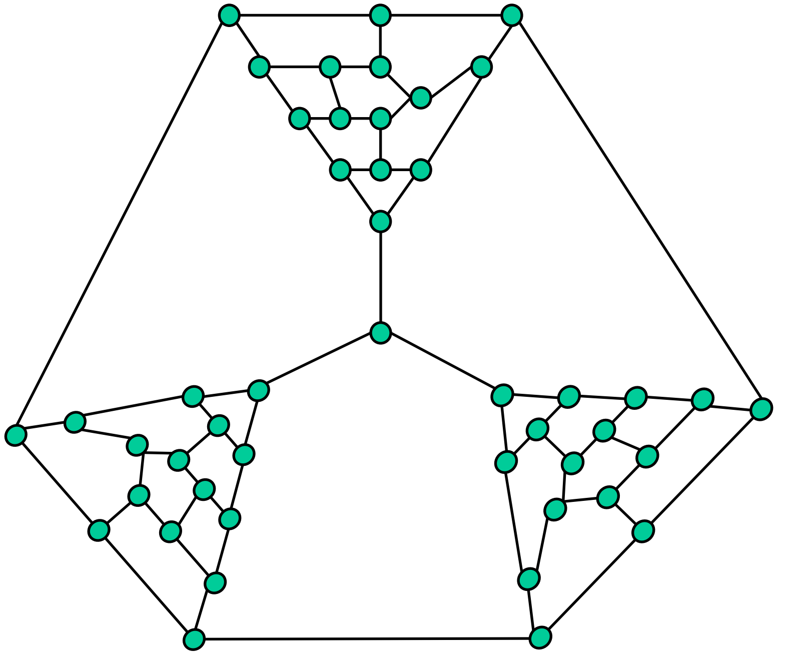
Đồ thị Hamilton

- Đồ thị có hai đỉnh bậc 1 ⇒ không là đồ thị Hamilton



- Dễ thấy đồ thị trên là đồ thị nửa Hamilton

Tutte Graph

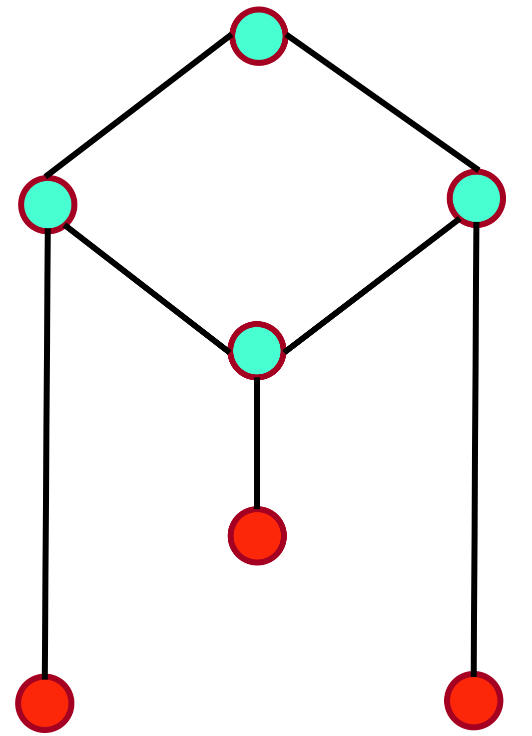


1. Đồ thị Tutte là 3-liên thông và chính qui bậc 3.
2. Đồ thị Tutte không là đồ thị Hamilton.

Đồ thị không là nửa Hamilton

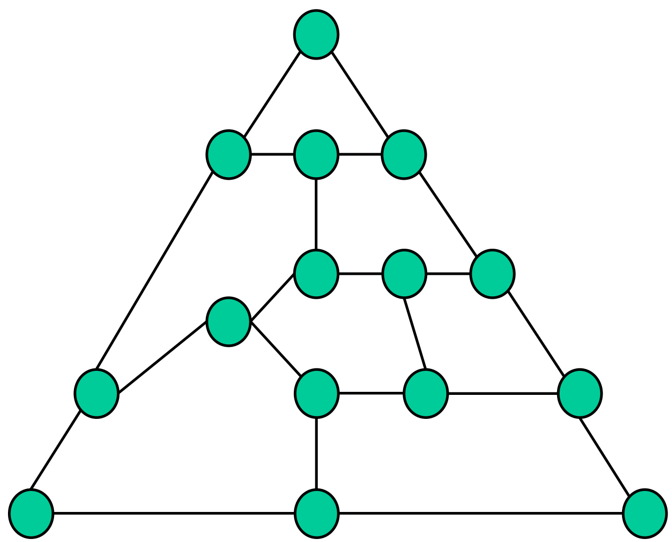
- Các đỉnh bậc 1 phải là đỉnh bắt đầu hoặc kết thúc của đường đi Hamilton.

Đồ thị có ba đỉnh bậc 1
$\Rightarrow$ không là nửa Hamilton



Đồ thị không là nửa Hamilton

- Đồ thị sau đây không là nửa Hamilton.



- ***Chú ý:*** Phần khó nhất trong chứng minh đồ thị Tutte không là Hamilton dựa vào kết quả này.

Định lý về sự tồn tại đường đi Hamilton

- **Định lý Dirac**: Nếu G là đơn đồ thị vô hướng liên thông với $n \geq 3$ đỉnh, và $\forall v \deg(v) \geq n/2$, thì G có chu trình Hamilton.

- **Định lý Ore:** Nếu G đơn đồ thị vô hướng liên thông với $n \geq 3$ đỉnh, và $\deg(u) + \deg(v) \geq n$ với mọi cặp đỉnh không kề nhau u, v, thì G có chu trình Hamilton.

Paul Adrien Maurice Dirac
1902 - 1984
(USA)

Oystein Ore
1899 - 1968
(Norway)

HAM-CIRCUIT là NP-đầy đủ

- Gọi HAM-CIRCUIT là bài toán:
  – Cho đơn đồ thị vô hướng *G*, hỏi *G* có chứa chu trình Hamilton hay không?
- Bài toán này được chứng minh là thuộc lớp bài toán *NP-đầy đủ*!
  – Có nghĩa là, nếu như tìm được thuật toán để giải nó trong thời gian đa thức, thì thuật toán này có thể sử dụng để giải mọi bài toán thuộc lớp NP trong thời gian đa thức.

Chương 1
CÁC KHÁI NIỆM CƠ BẢN

1.1. Đồ thị trong thực tế

1.2. Các loại đồ thị

1.3. Bậc của đỉnh

1.4. Đồ thị con

1.5. Đồ thị đẳng cấu

1.6. Đường đi và chu trình

1.7. Tính liên thông

1.8. Một số loại đồ thị đặc biệt

1.9. Tô màu đồ thị

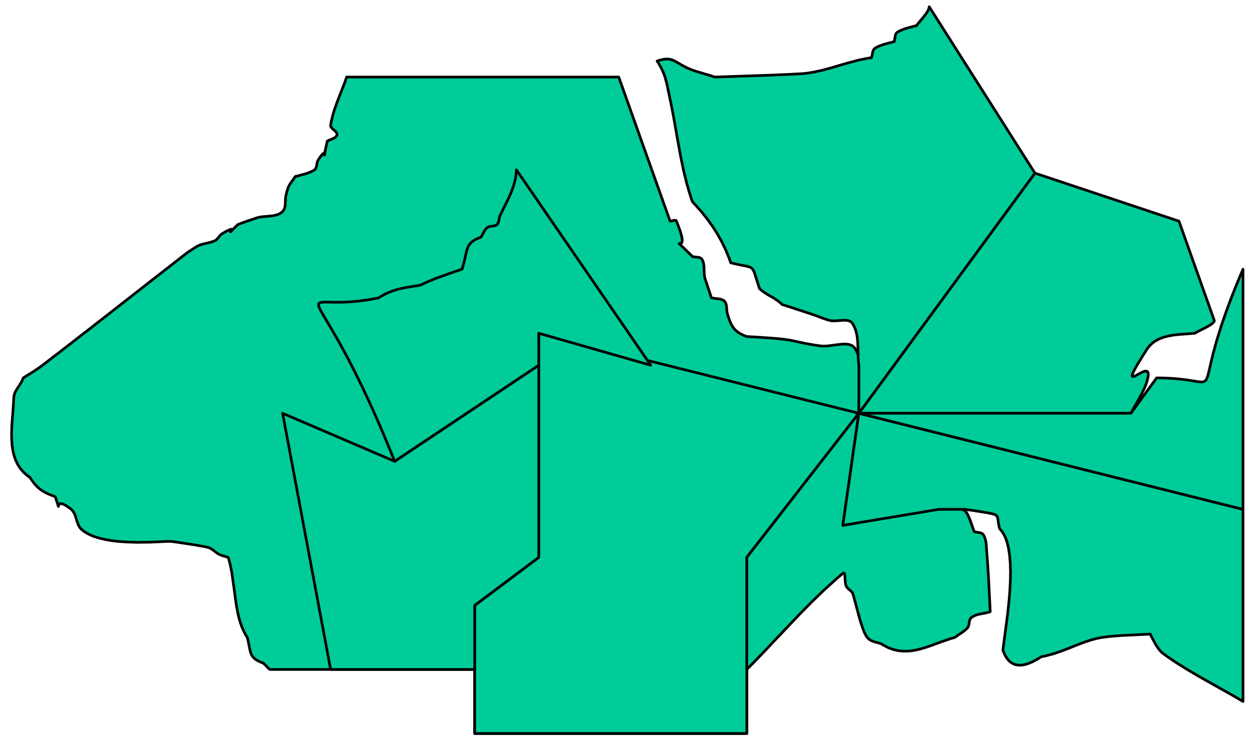
Tô màu đồ thị

Graph Coloring

- *Tô màu đỉnh* (Vertex Coloring)
- *Tô màu cạnh* (Edge Coloring)

Tô màu đồ thị - Graph Coloring

Xét bản đồ



Tô màu bản đồ - Map Coloring

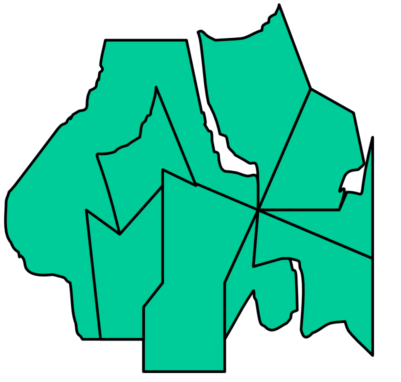
Ta muốn nhận biết các nước bằng cách tô màu.

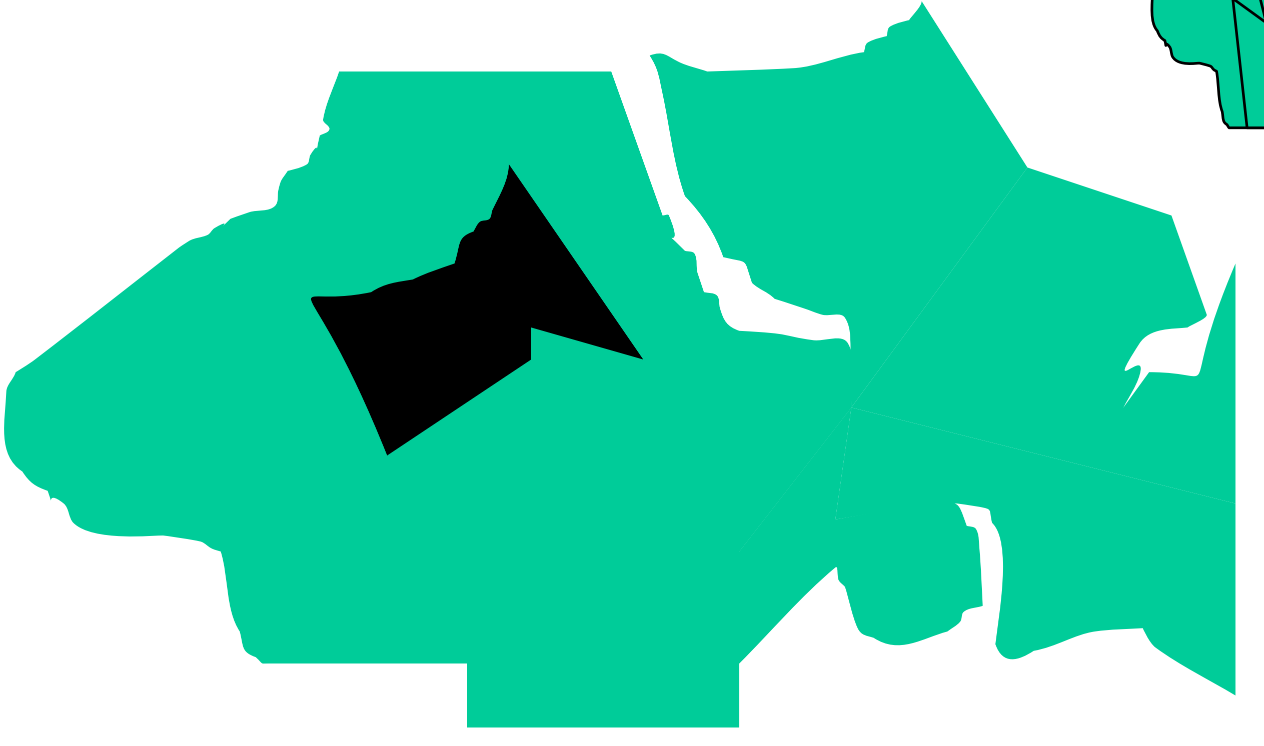
Rõ ràng: Tô bởi 1 màu là không thể phân biệt được:

Map Coloring

Bổ sung thêm 1 màu nữa.

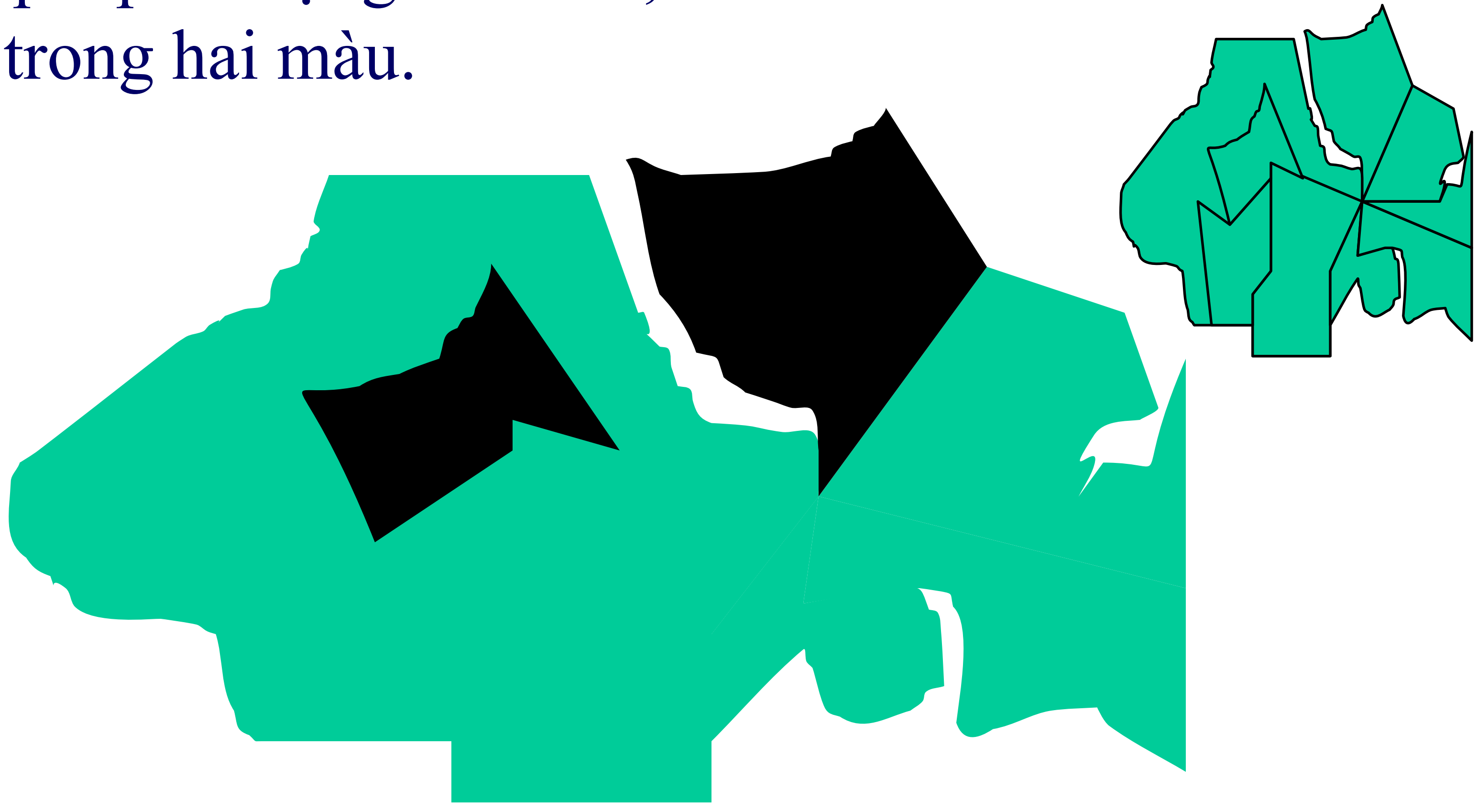
Thử tô các nước bởi 2 màu.





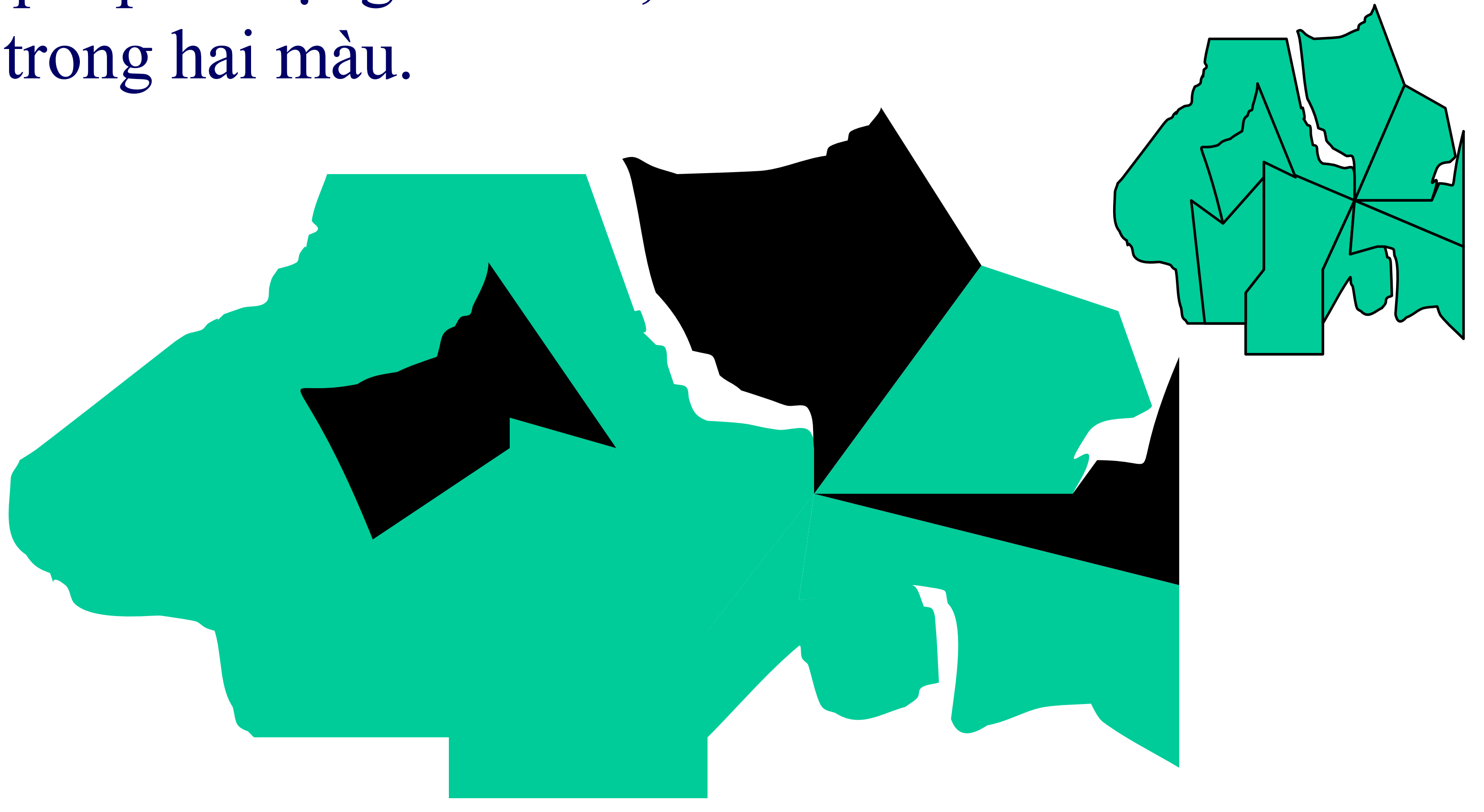
Map Coloring

Cho phép sử dụng hai màu, ta thử tô mỗi nước bởi một trong hai màu.



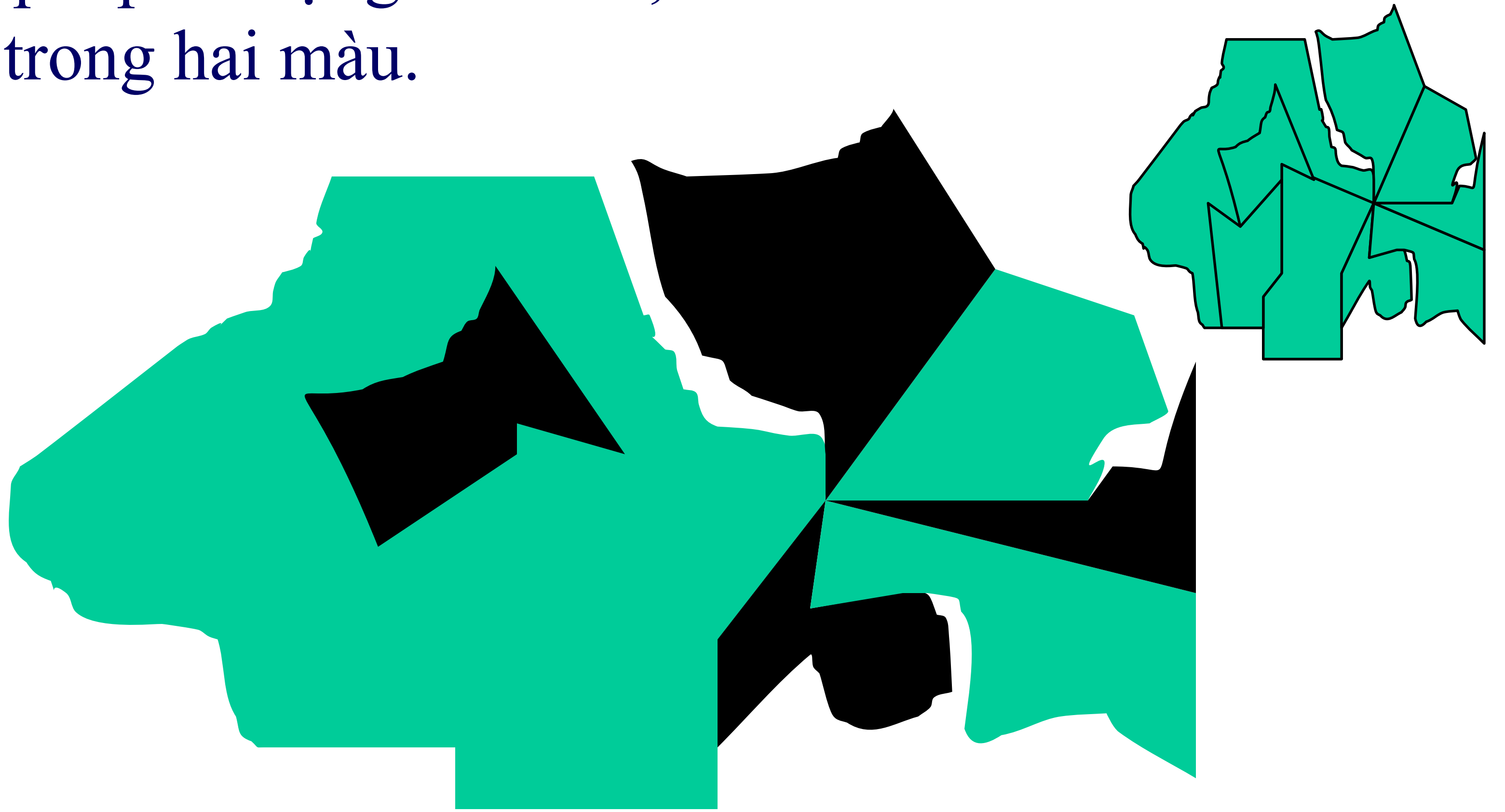
Map Coloring

Cho phép sử dụng hai màu, ta thử tô mỗi nước bởi một trong hai màu.



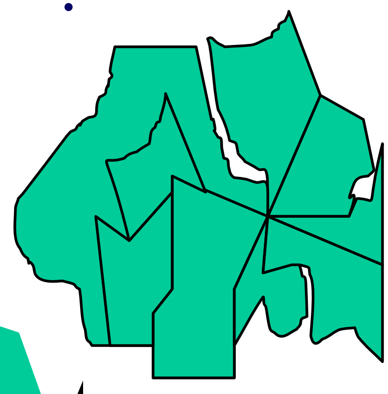
Map Coloring

Cho phép sử dụng hai màu, ta thử tô mỗi nước bởi một trong hai màu.



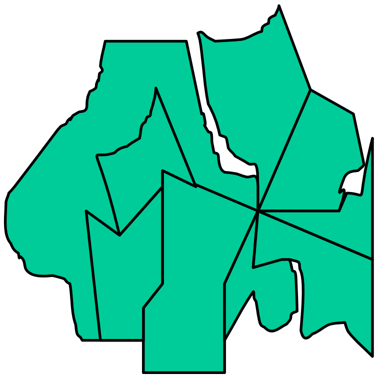
Map Coloring

Hai nước bị tô bởi cùng màu. Không phân biệt được danh giới.



Map Coloring

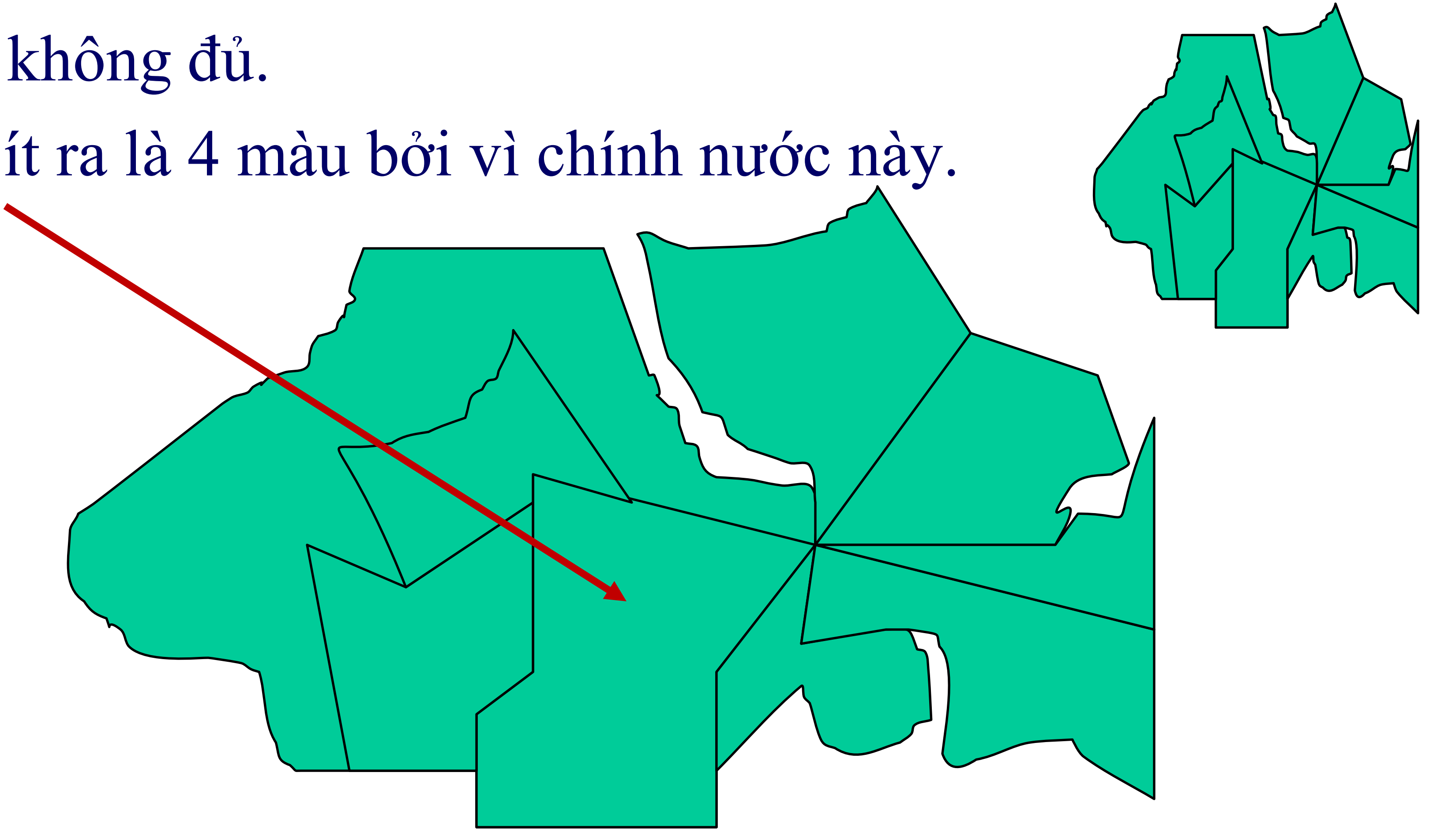
Vậy thì thêm 1 màu nữa:



Map Coloring

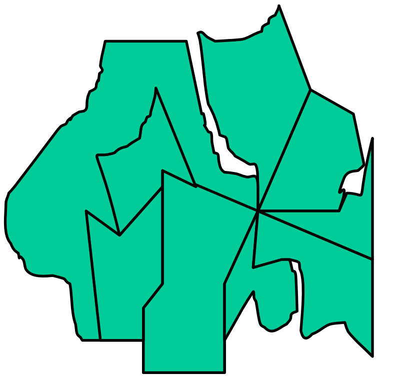
Vẫn không đủ.

Cần ít ra là 4 màu bởi vì chính nước này.



Map Coloring

Với 4 màu, có thể tô được.



4-Color Theorem

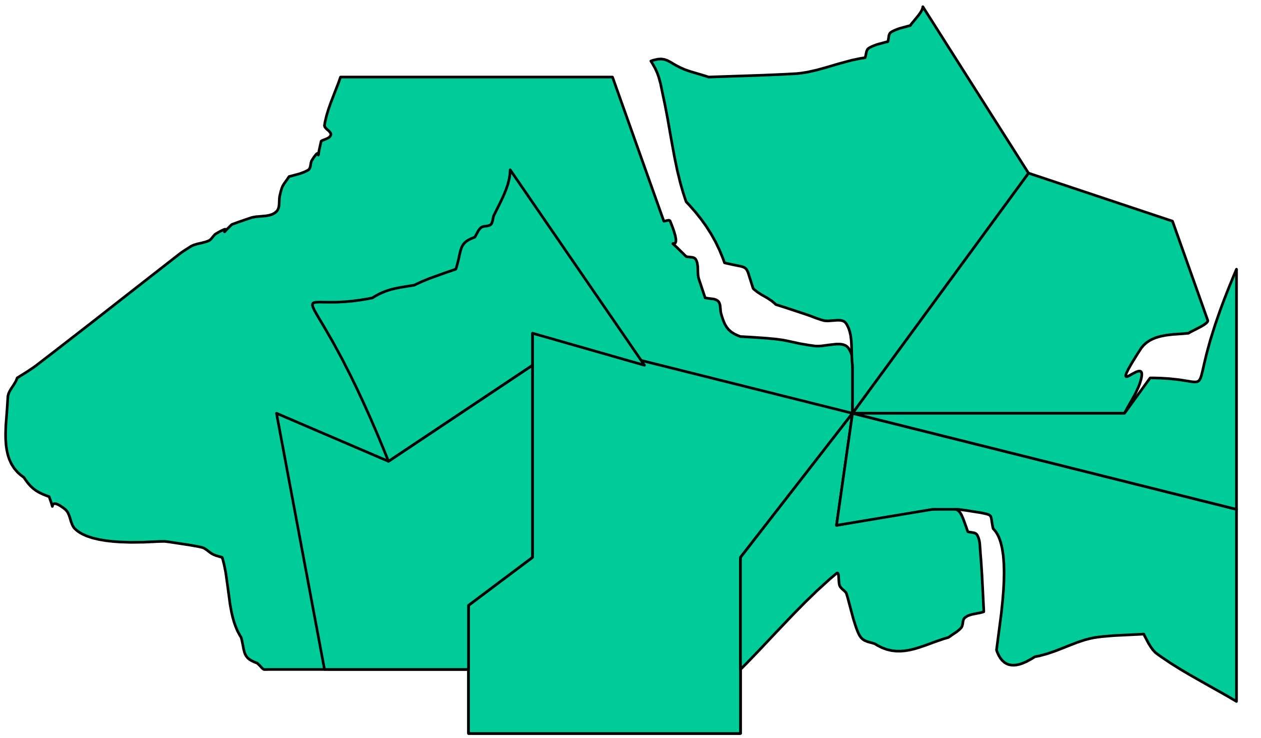
Định lý 4 màu: *Mọi bản đồ hành chính đều có thể tô bởi bốn màu sao cho không có 2 đơn vị hành chính có chung biên giới nào bị tô bởi cùng màu.*

Proof.

Năm 1976, Haken và Appel chứng minh được định lý 4 màu "bằng máy tính". (Thực hiện kiểm tra tô bởi 4 màu gần 2000 bản đồ đặc biệt bằng máy tính.)

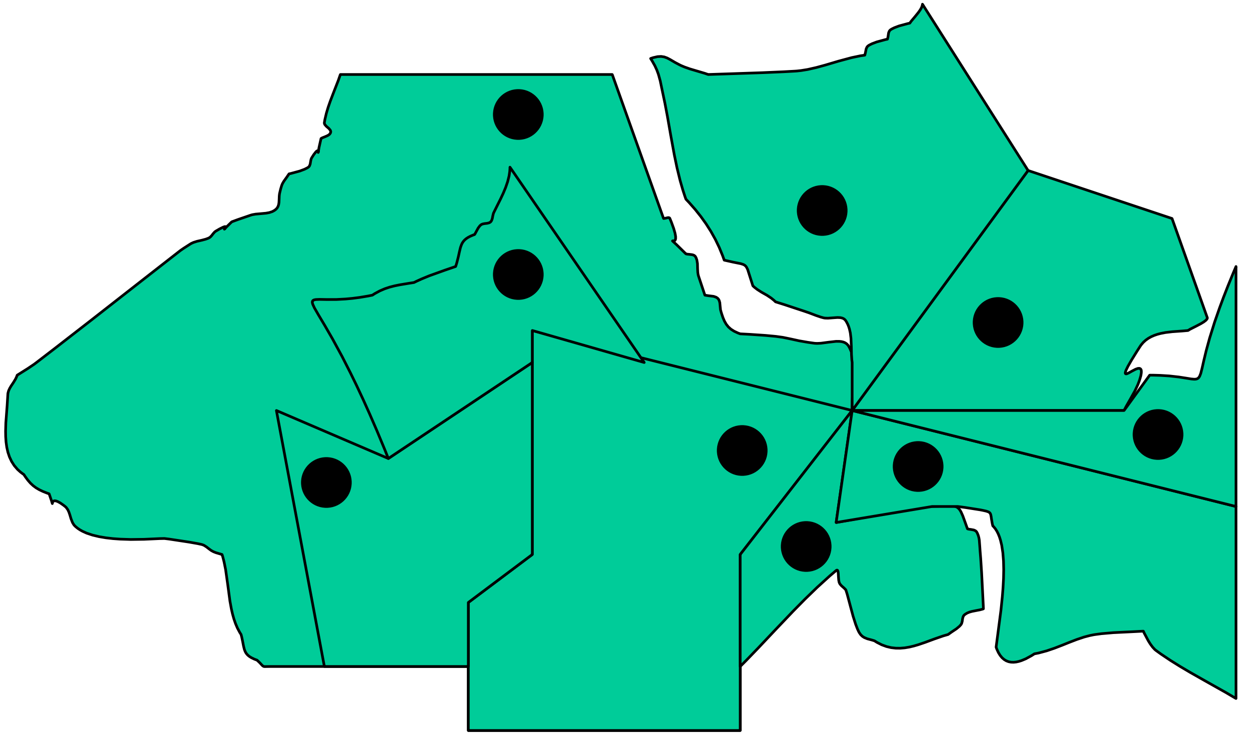
Từ tô màu bản đồ đến tô màu đồ thị

Bài toán tô màu bản đồ có thể dẫn về bài toán tô màu đồ thị:



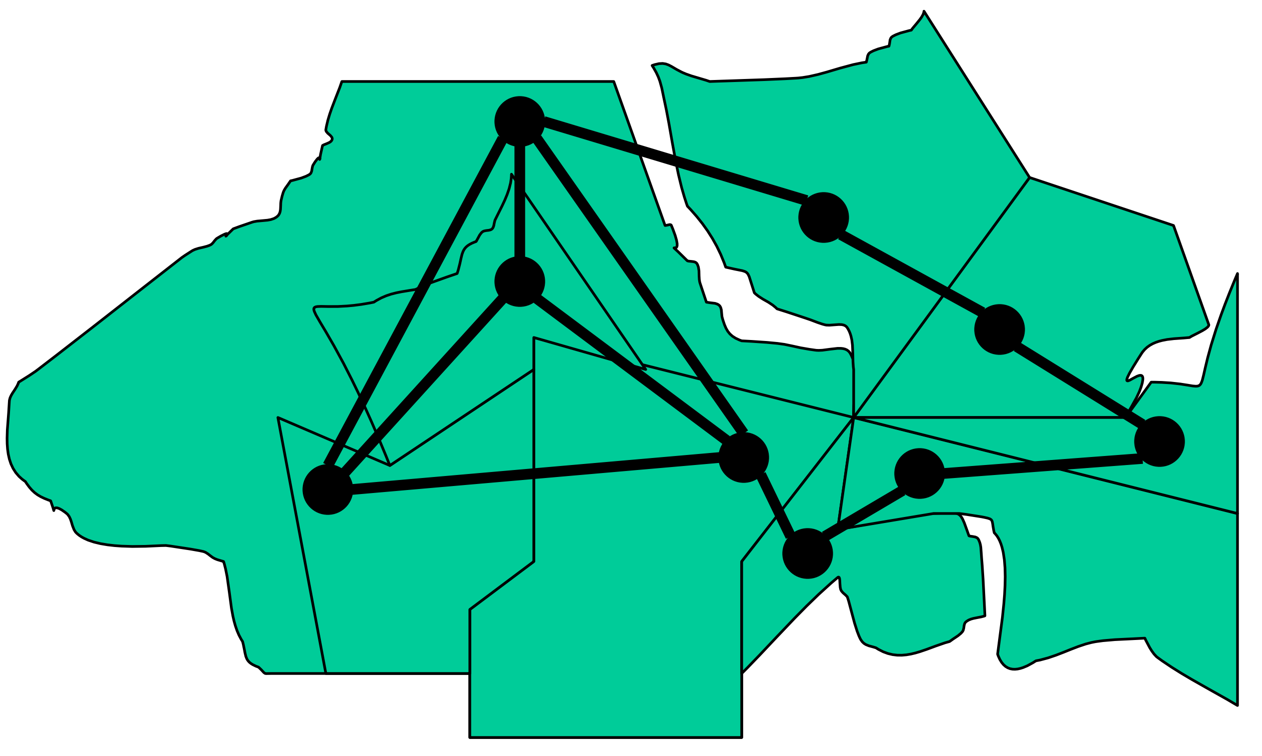
Từ tô màu bản đồ đến tô màu đồ thị

Mỗi vùng đặt tương ứng với một đỉnh:



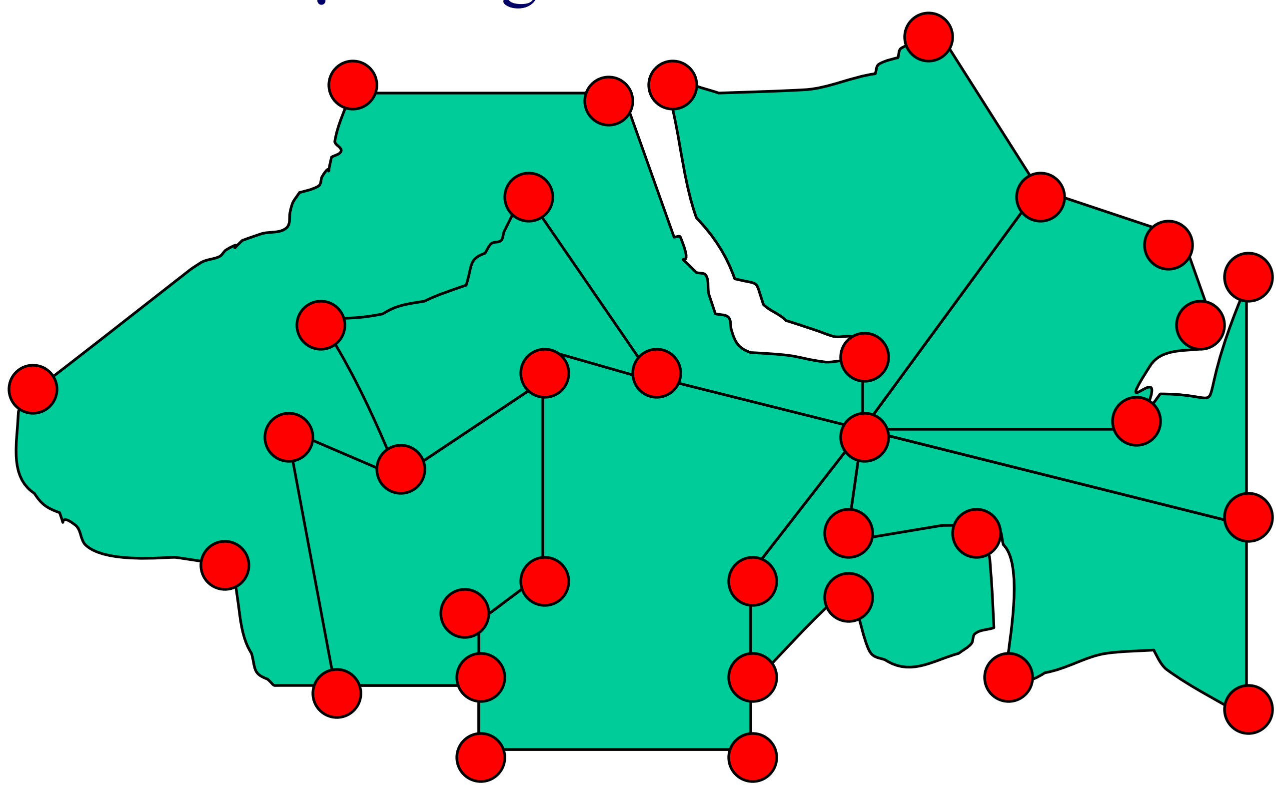
Từ tô màu bản đồ đến tô màu đồ thị

Hai vùng có chung biên giới có cạnh nối:



Từ tô màu bản đồ đến tô màu đồ thị

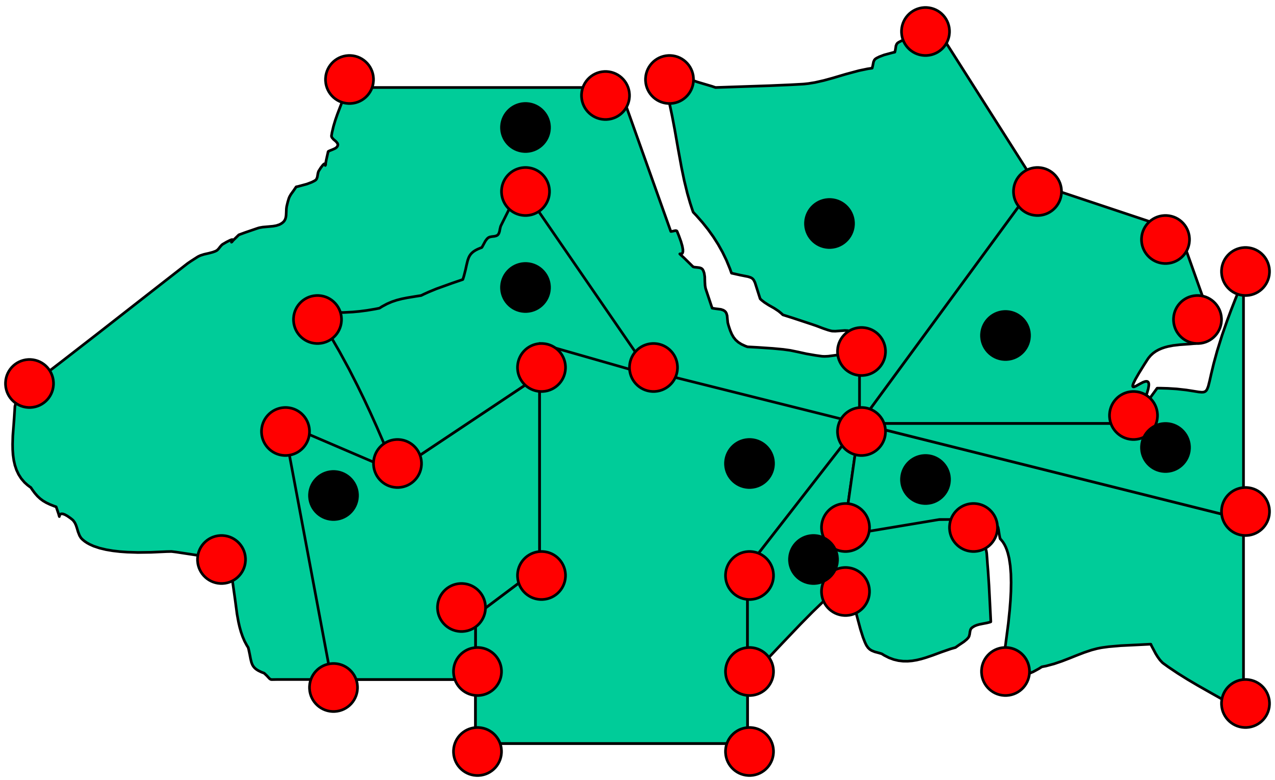
Thực ra, ta cũng có thể xem bản đồ là đồ thị và khi đó sẽ xét đồ thị đối ngẫu của nó.



Từ tô màu bản đồ đến tô màu đồ thị

Đồ thị đối ngẫu (*Dual Graphs*) :

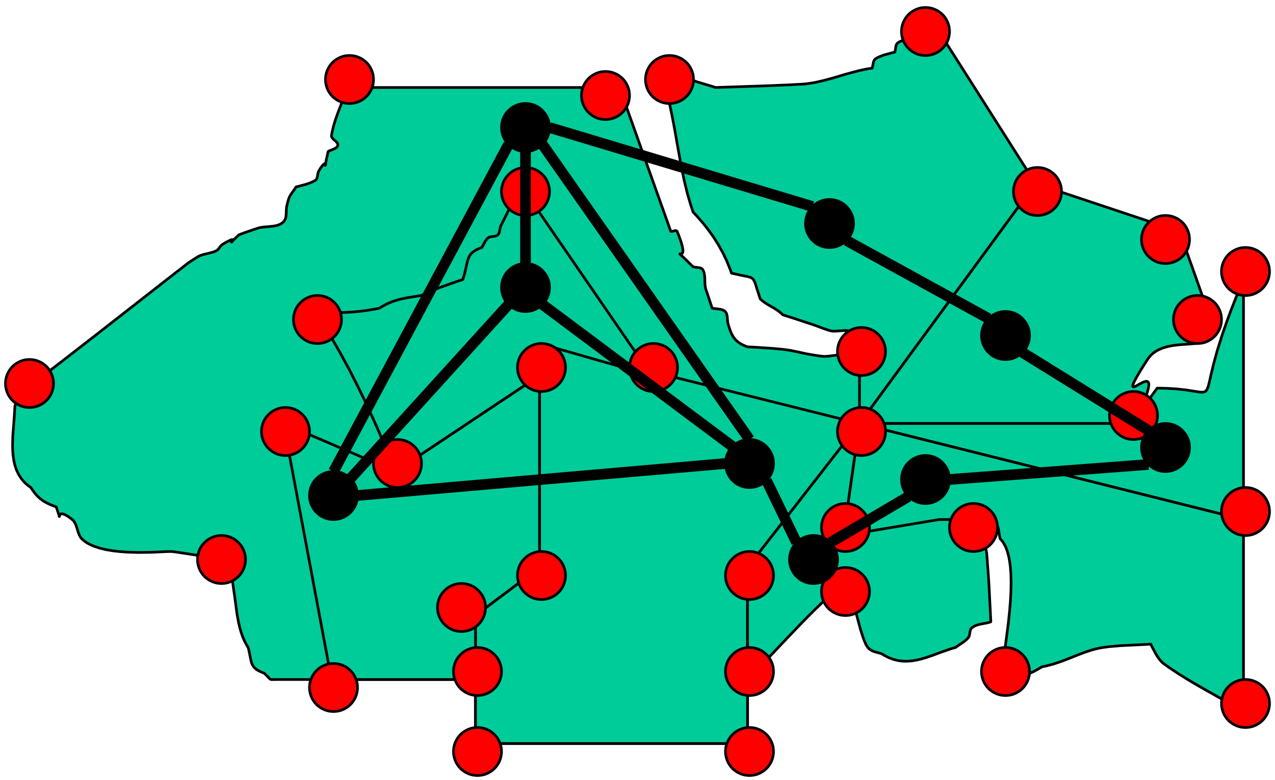
1) Đặt mỗi miền ứng với 1 đỉnh:



Từ tô màu bản đồ đến tô màu đồ thị

Đồ thị đối ngẫu (*Dual Graphs*) :

2) Nối các đỉnh bởi các cạnh:



Định nghĩa đồ thị đối ngẫu

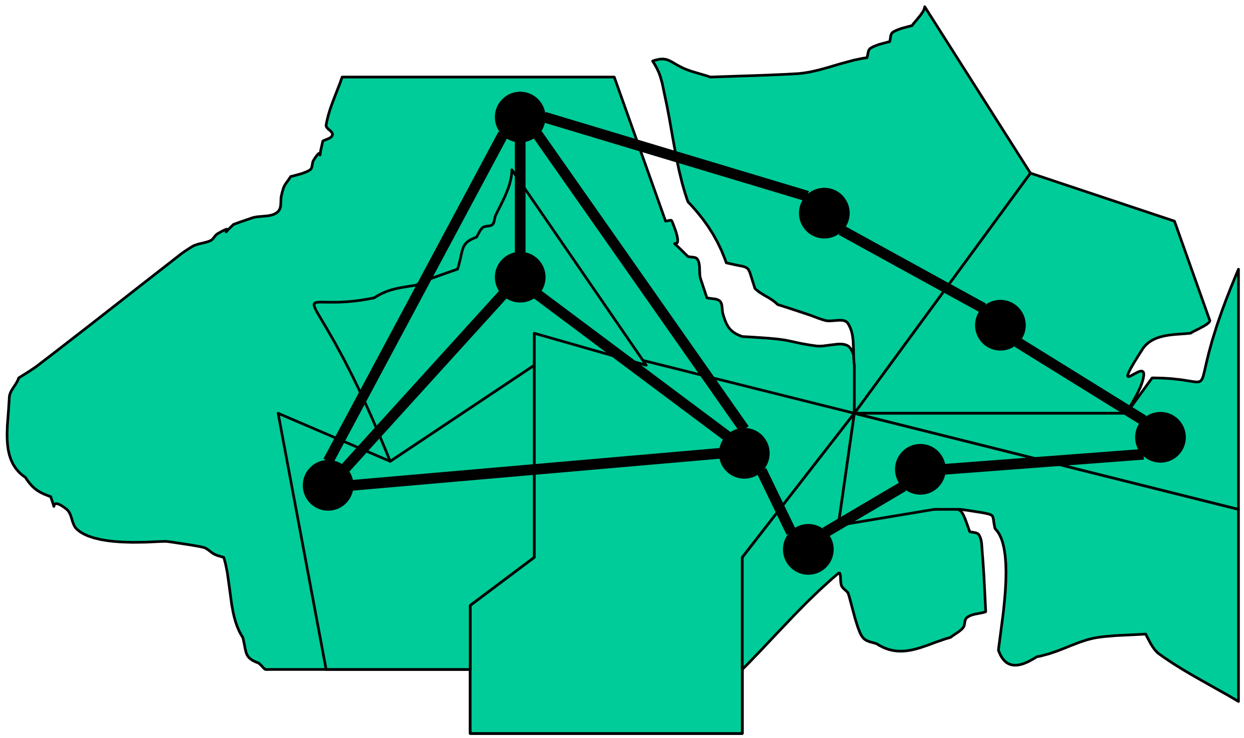
Định nghĩa:

Đồ thị đối ngẫu G^* của đồ thị phẳng $G = (V, E)$ với tập các miền R là đồ thị với tập đỉnh và cạnh được xây dựng như sau

- Tập đỉnh của G^*: $V(G^*) = R$
- Tập cạnh của G^*: $E(G^*)$ = tập các cạnh dạng (F_1,F_2) trong đó 2 miền F_1 và F_2 có cạnh chung.

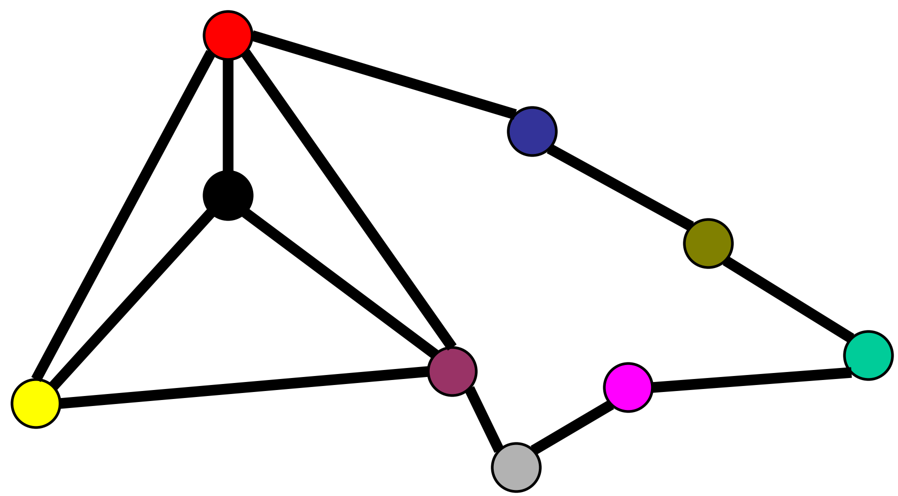
Từ tô màu bản đồ đến tô màu đồ thị

Như vậy đồ thị đối ngẫu là:



Từ tô màu bản đồ đến tô màu đồ thị

Tô màu miền tương đương với tô màu đỉnh của đồ thị đối ngẫu.



Định nghĩa sắc số

Định nghĩa: Giả sử c là số nguyên dương. Đơn đồ thị vô hướng được gọi là *tô được bởi c màu* nếu có thể tô các đỉnh của nó bởi c màu sao cho không có hai đỉnh kề nhau nào bị tô bởi cùng một màu.

Sắc số (***chromatic number***) của đồ thị G, ký hiệu bởi $\chi(G)$, là số c nhỏ nhất sao cho có thể tô được G bởi c màu.

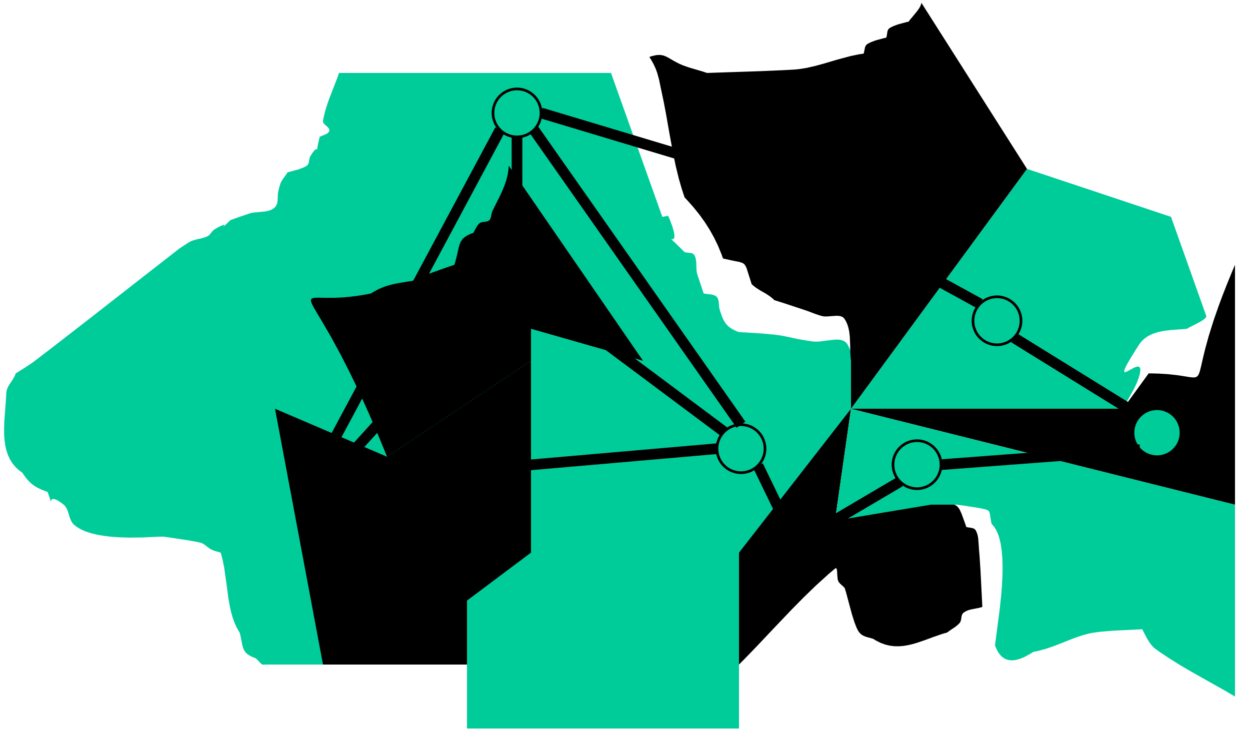
Ví dụ:

Ta có $\chi(G) = 2$, nếu G là đồ thị hai phía. Dễ thấy điều ngược lại cũng đúng.

Rõ ràng $\chi(G) \geq \Delta(G)$.

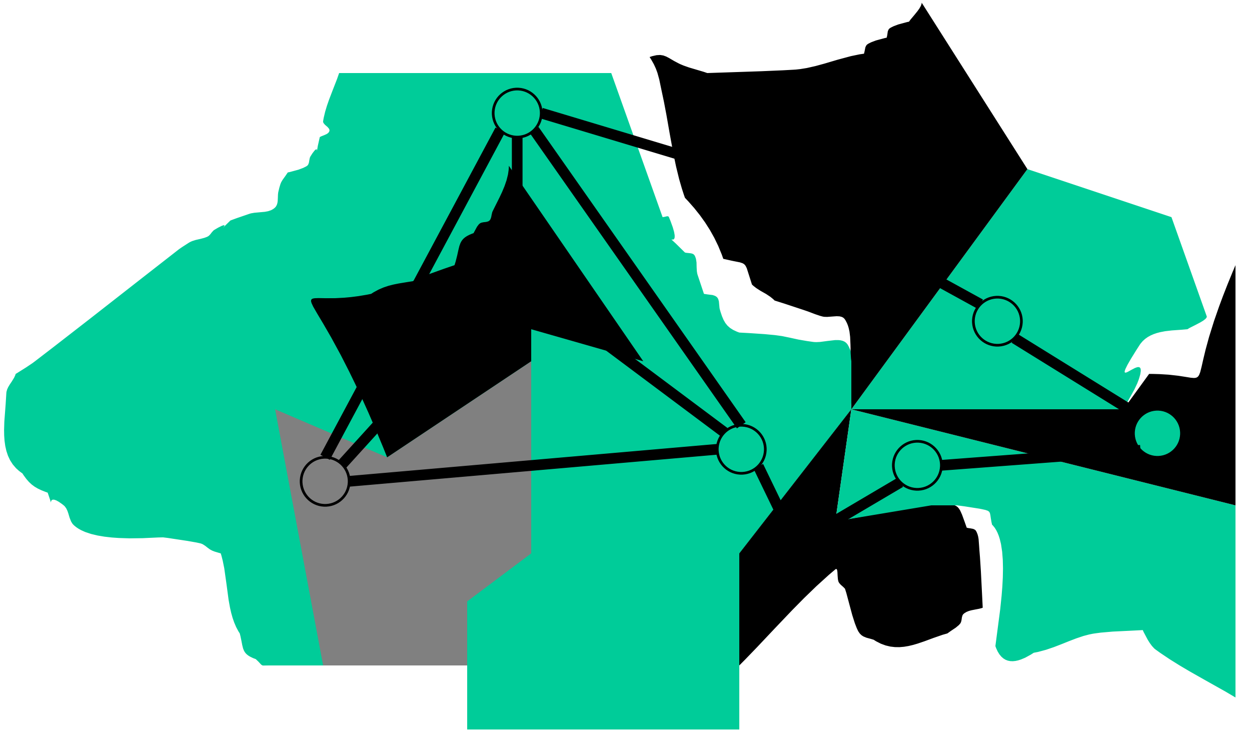
Từ tô màu bản đồ đến tô màu đồ thị

Bản đồ không tô được bởi 2 màu, vì thế đồ thị đối ngẫu không tô được bởi 2 màu:



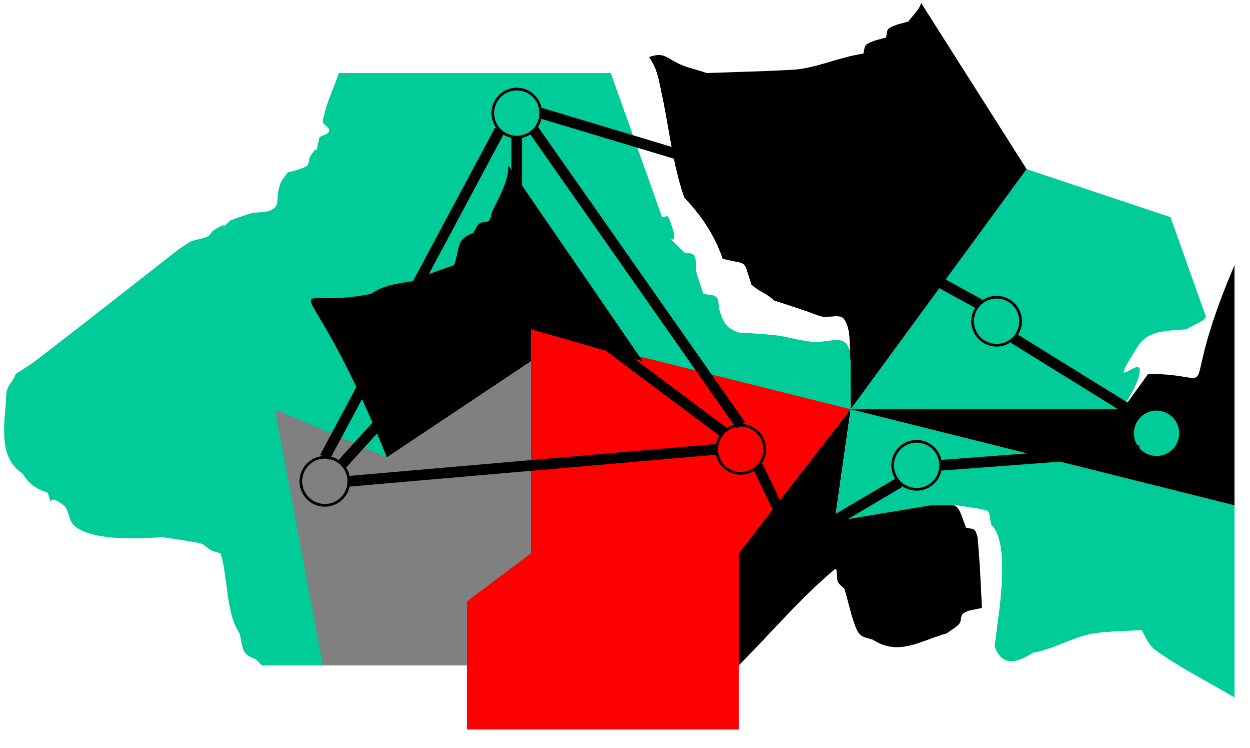
Từ tô màu bản đồ đến tô màu đồ thị

Bản đồ không tô được bởi 3 màu, vì thế đồ thị đối ngẫu không tô được bởi 3 màu:



Từ tô màu bản đồ đến tô màu đồ thị

Đồ thị tô được bởi 4 màu vì thế bản đồ cũng tô được bởi 4 màu:

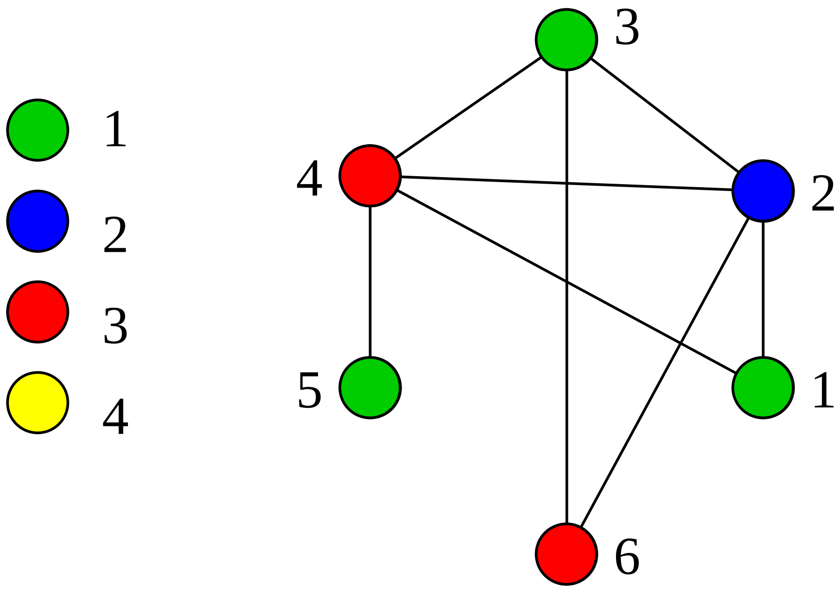


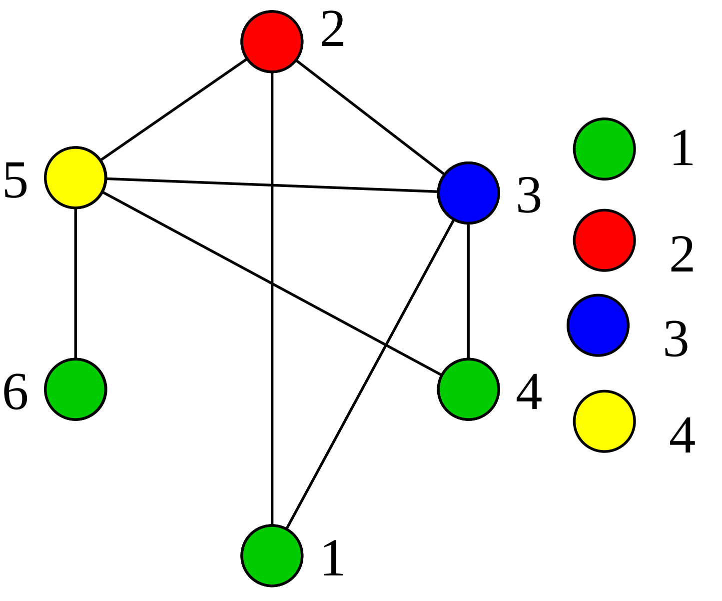
Định lý 4 màu trong ngôn ngữ đồ thị

Định lý. Mọi đồ thị phẳng đều tô được bởi 4 màu.

Thuật toán tham lam

- **Khởi tạo.** Sắp xếp các đỉnh của đồ thị theo thứ tự $v_1, v_2, \ldots, v_n$
- **Bước i (i=1, 2,..., n):** Tô đỉnh v_i bởi màu có chỉ số nhỏ nhất trong số các màu chưa được sử dụng để tô các đỉnh kề của nó.





- **Chú ý:** Kết quả thực hiện thuật toán là phụ thuộc vào trình tự sắp xếp các đỉnh của đồ thị.

Cận trên cho sắc số

Định lý. Đối với đơn đồ thị vô hướng bất kỳ G ta có

$$\chi(G) \leq \Delta(G)+1.$$

Chứng minh

- Trong dãy đỉnh, mỗi đỉnh có nhiều nhất $\Delta(G)$ đỉnh kề.
- Do đó, thuật toán tham lam không thể sử dụng nhiều hơn $\Delta(G)+1$ màu.

Một cận trên tốt hơn được cho trong định lý sau đây

Định lý Brook (1941). Giả sử G là đơn đồ thị vô hướng liên thông khác với đồ thị đầy đủ và chu trình độ dài lẻ. Khi đó

$$\chi(G) \leq \Delta(G).$$

Tô màu đồ thị và Lập lịch

Ví dụ:

Ta cần lập lịch thi kết thúc môn học cho các chuyên đề có mã số:

1007, 3137, 3157, 3203, 3261, 4115, 4118, 4156

- Giả sử các môn học sau không có sinh viên nào đồng thời cùng thi (do điều kiện tiên quyết) :

1007-3137

1007-3157, 3137-3157

1007-3203

1007-3261, 3137-3261, 3203-3261

1007-4115, 3137-4115, 3203-4115, 3261-4115

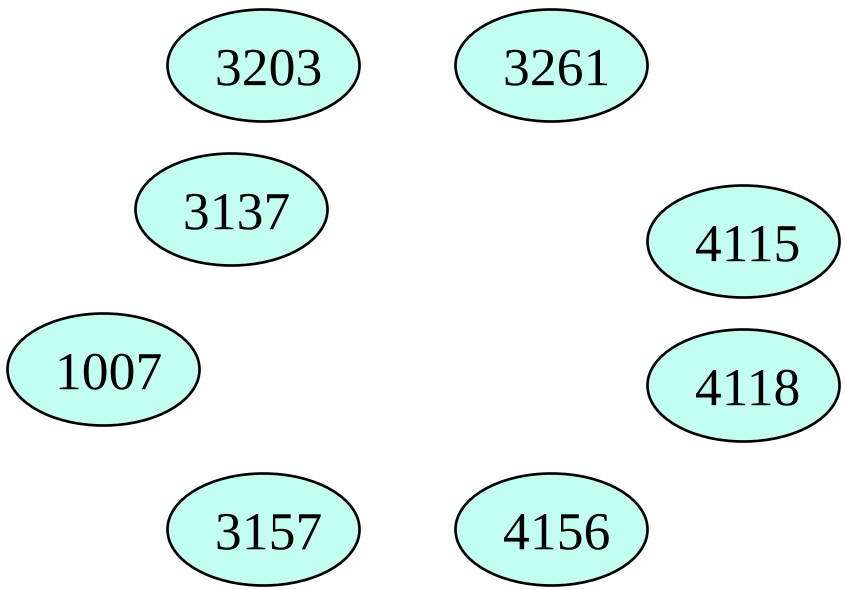
1007-4118, 3137-4118

1007-4156, 3137-4156, 3157-4156

Hỏi lịch thi gồm ít nhất bao nhiêu ngày? (Lịch thi phải đảm bảo mỗi sinh viên trong một ngày phải thi không quá 1 môn)

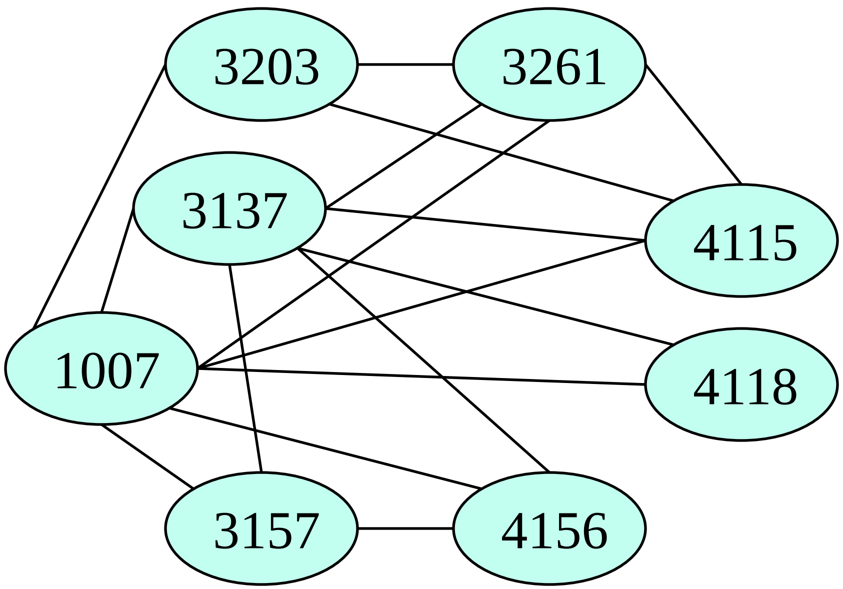
Tô màu đồ thị và Lập lịch

- Đưa bài toán về bài toán tô màu đồ thị: Các đỉnh tương ứng với các môn học, cạnh nối có giữa hai đỉnh nếu hai môn tương ứng có thể có chung sinh viên dự thi (vì thế không được tổ chức đồng thời):



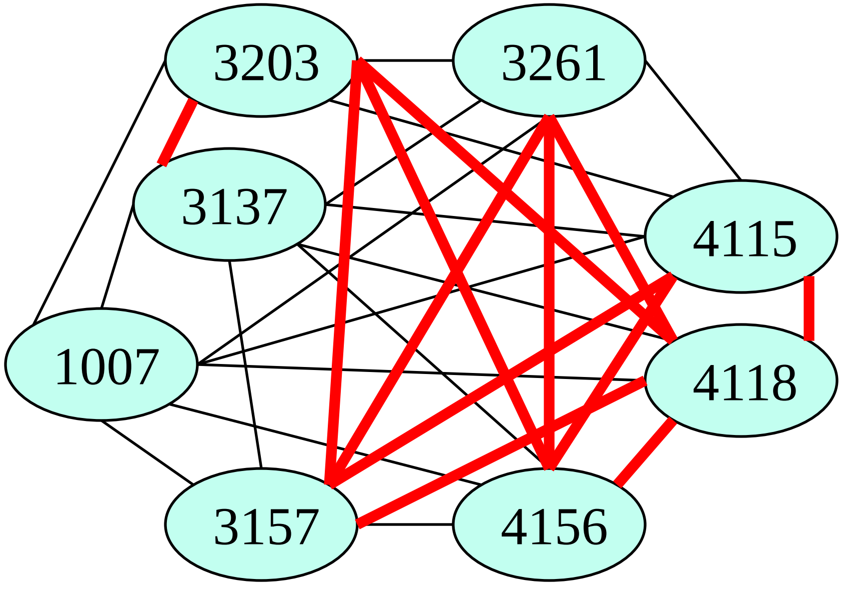
Tô màu đồ thị và Lập lịch

- Trước hết ta đưa vào cạnh nối giữa các môn chắc chắc không có chung sinh viên (từ điều kiện tiên quyết)…



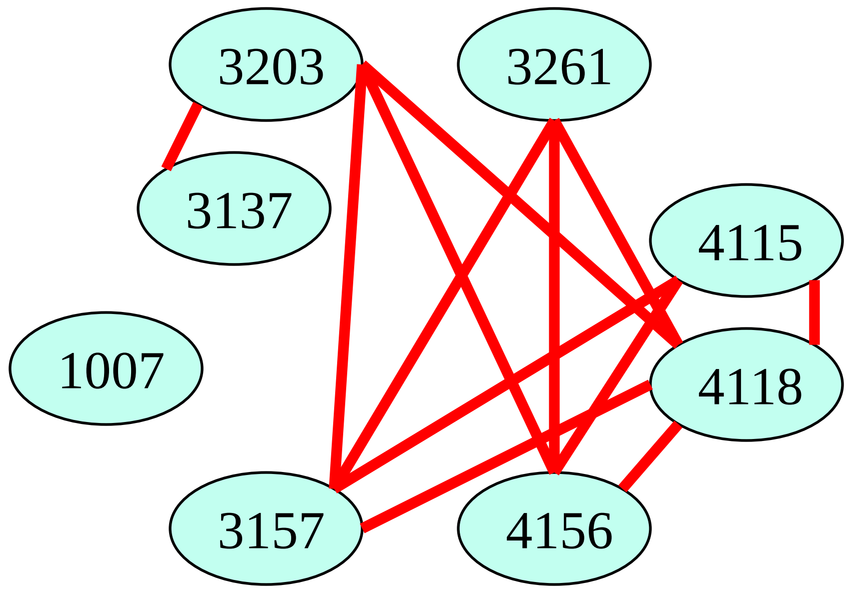
Tô màu đồ thị và Lập lịch

…và sau đó xây dựng đồ thị bù (complementary graph):



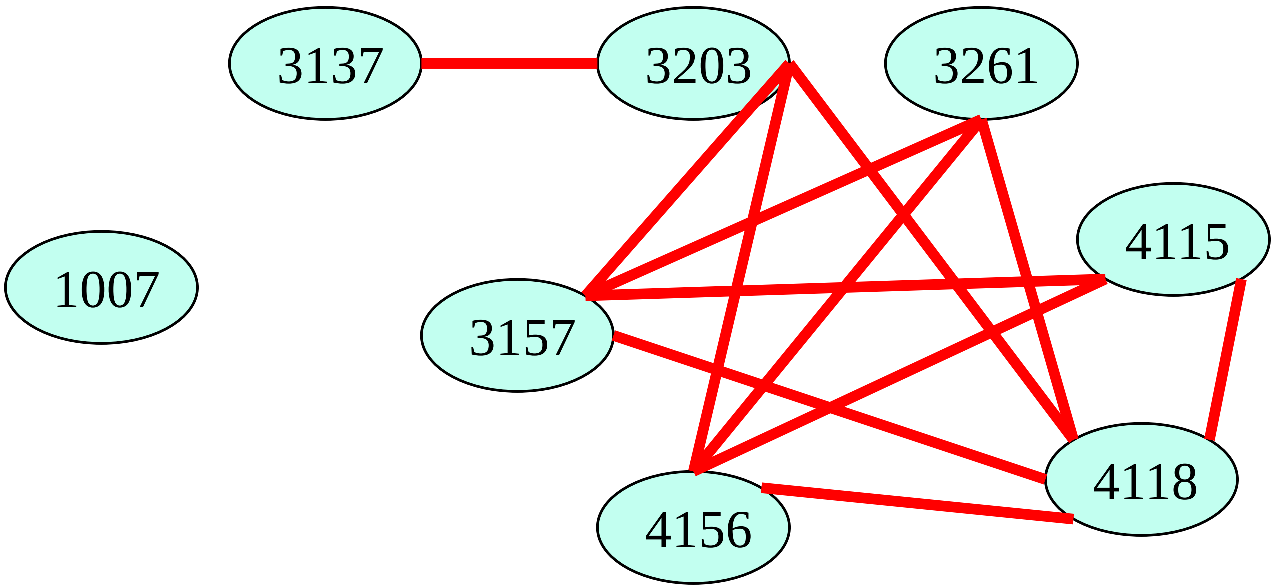
Tô màu đồ thị và Lập lịch

…và sau đó làm việc với đồ thị bù (chỉ các môn học có cạnh nối mới có thể có chung sinh viên):



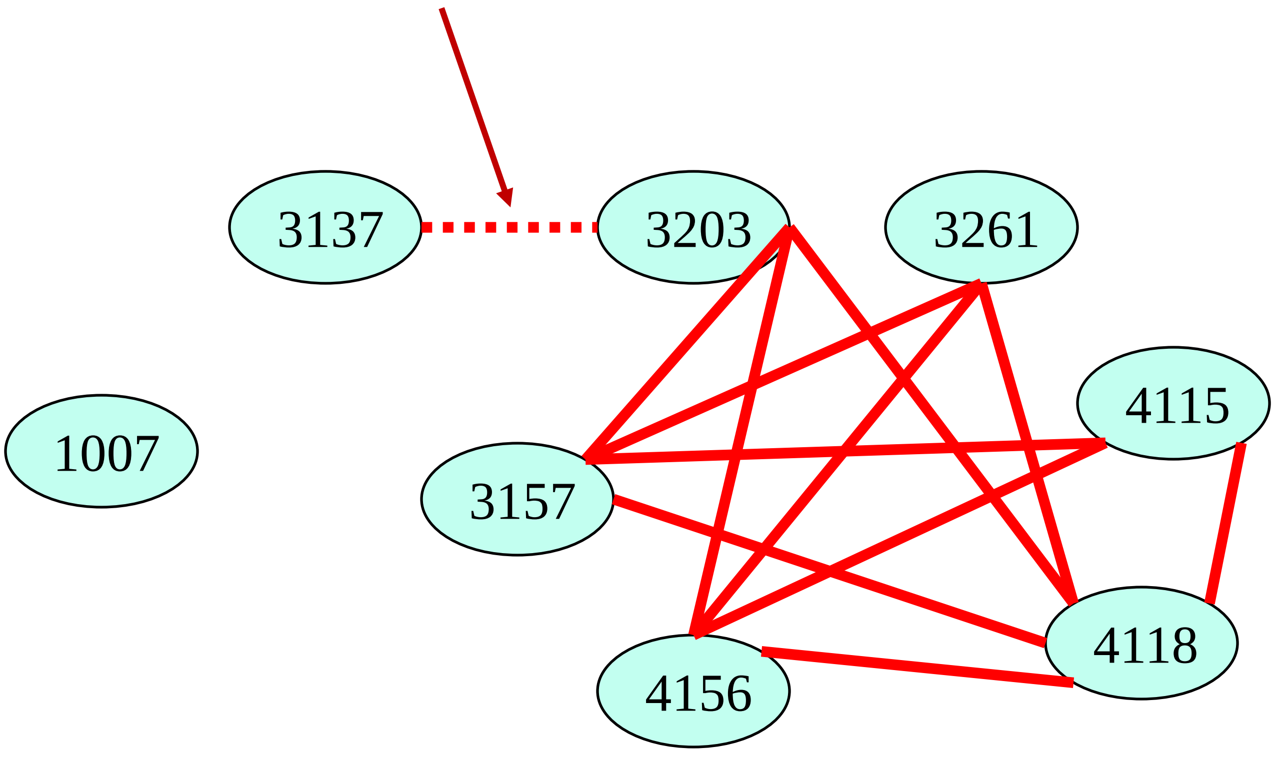
Tô màu đồ thị và Lập lịch

Vẽ lại:



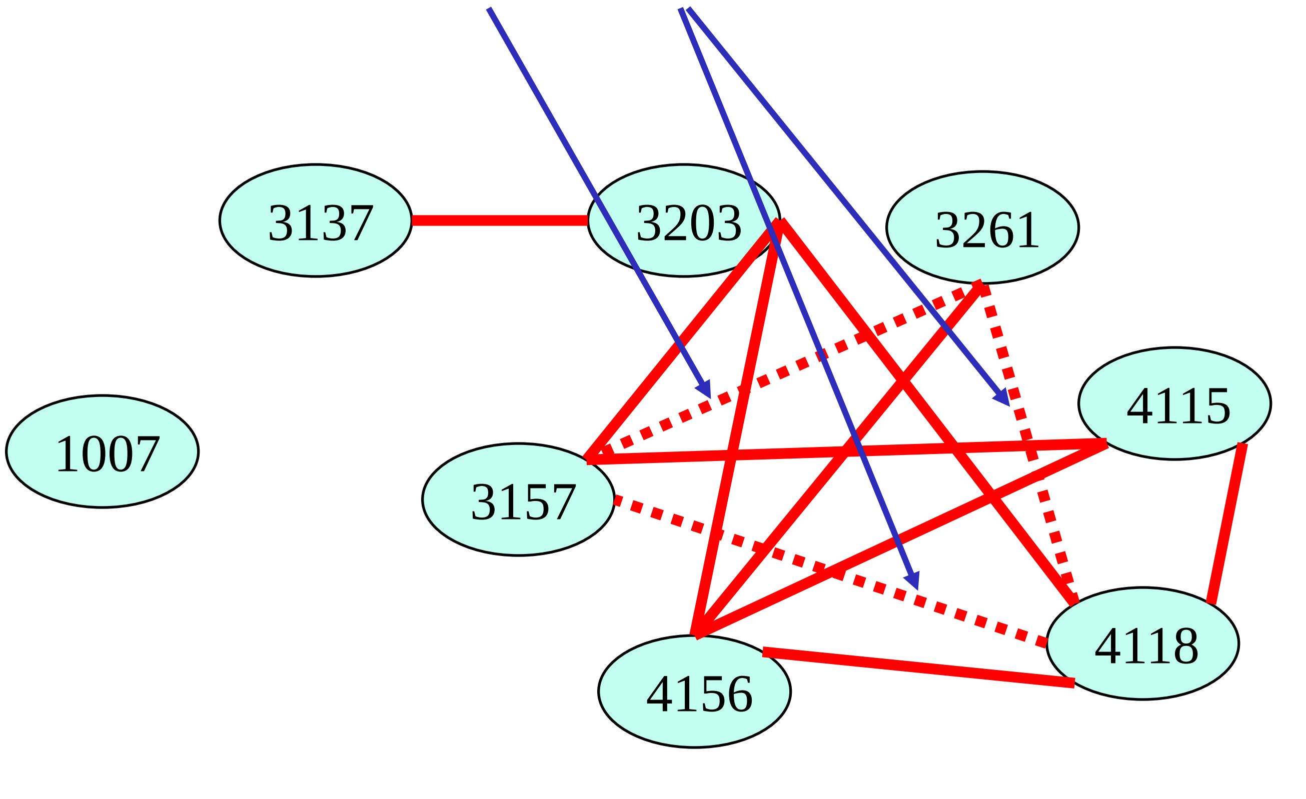
Tô màu đồ thị và Lập lịch

Không thể tô bởi 1 màu vì cạnh này



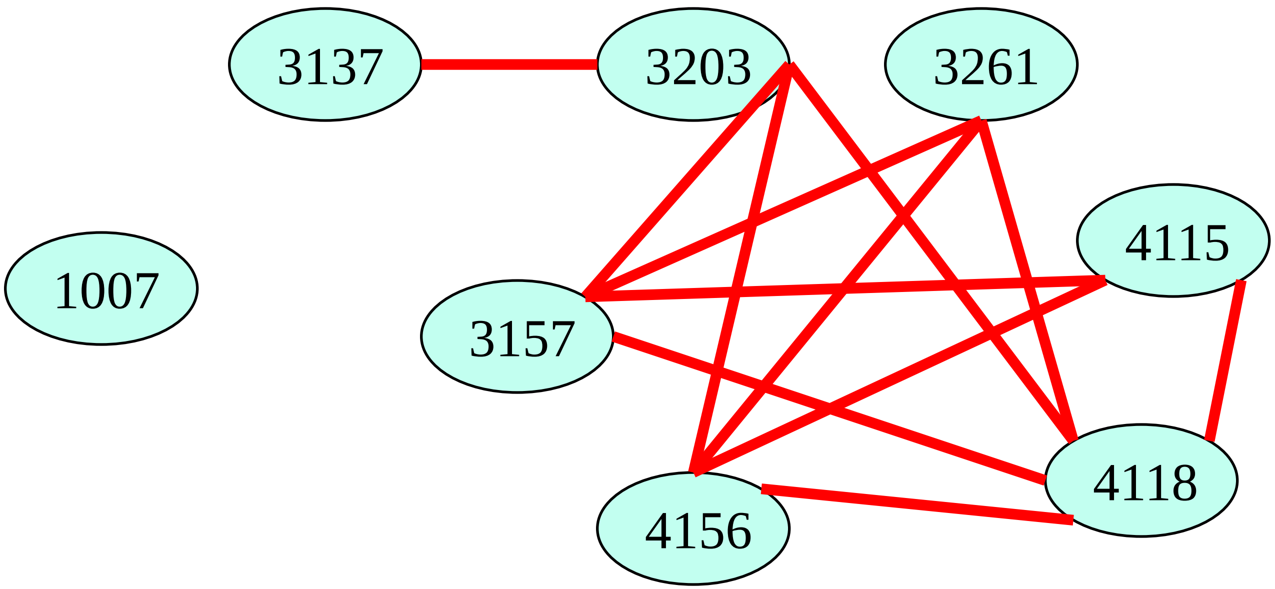
Tô màu đồ thị và Lập lịch

2 màu không đủ tô vì có tam giác này



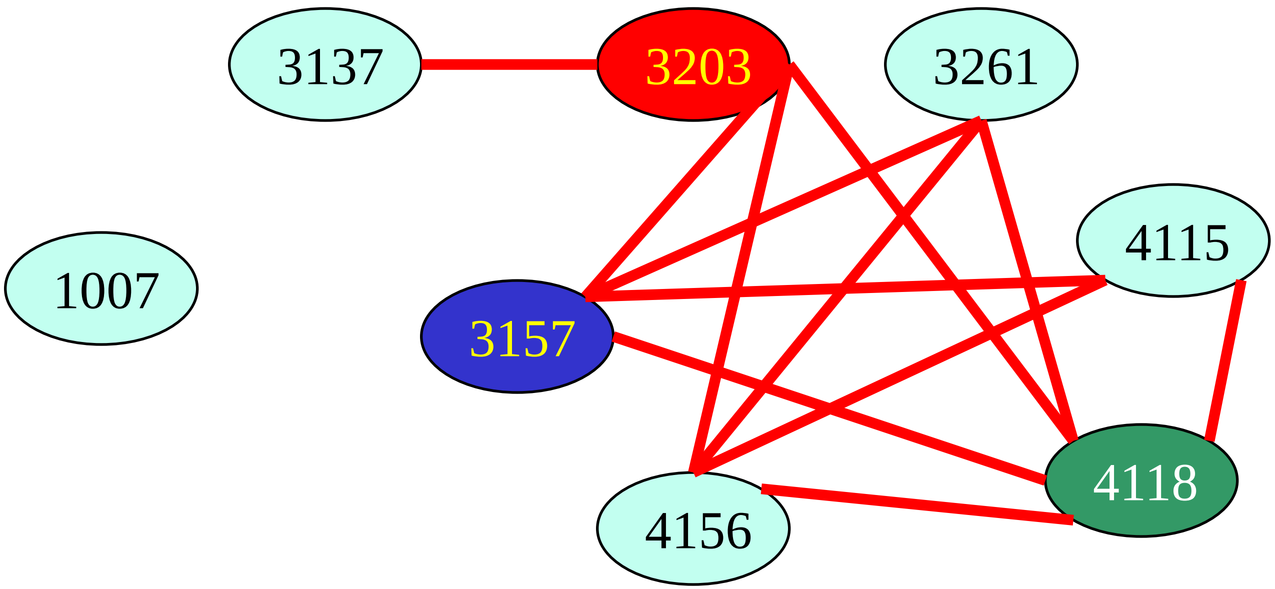
Tô màu đồ thị và Lập lịch

3 màu là đủ tô tam giác này. Ta tô bởi ba màu Red, Green, Blue.



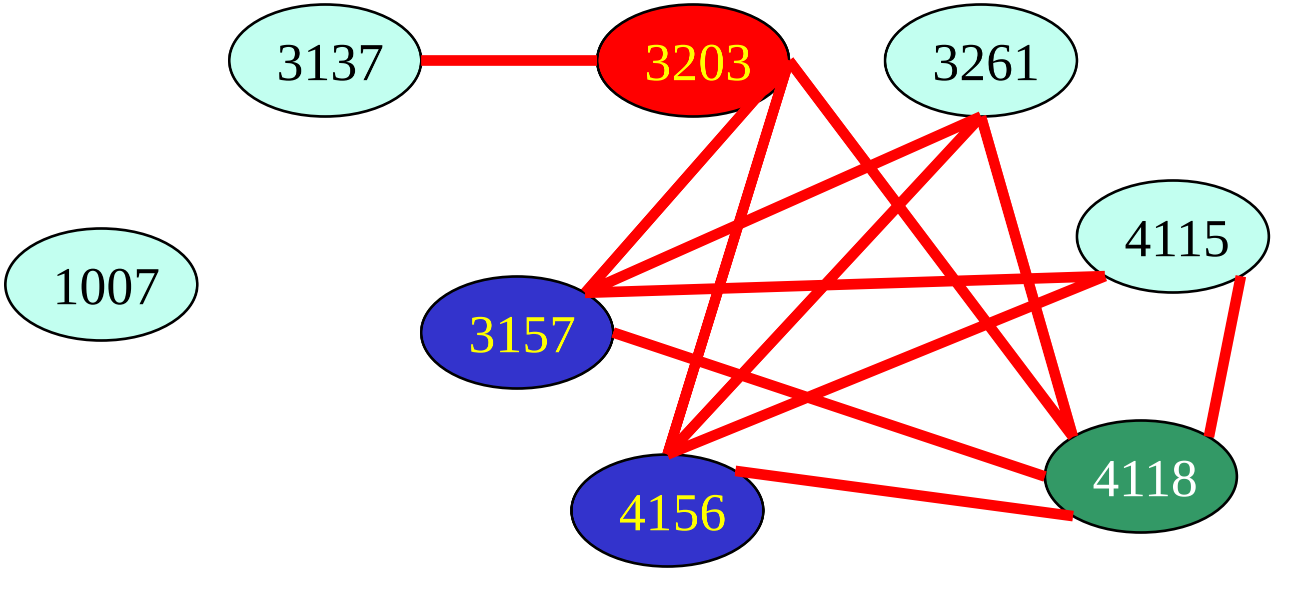
Tô màu đồ thị và Lập lịch

3203-Red, 3157-Blue, 4118-Green:



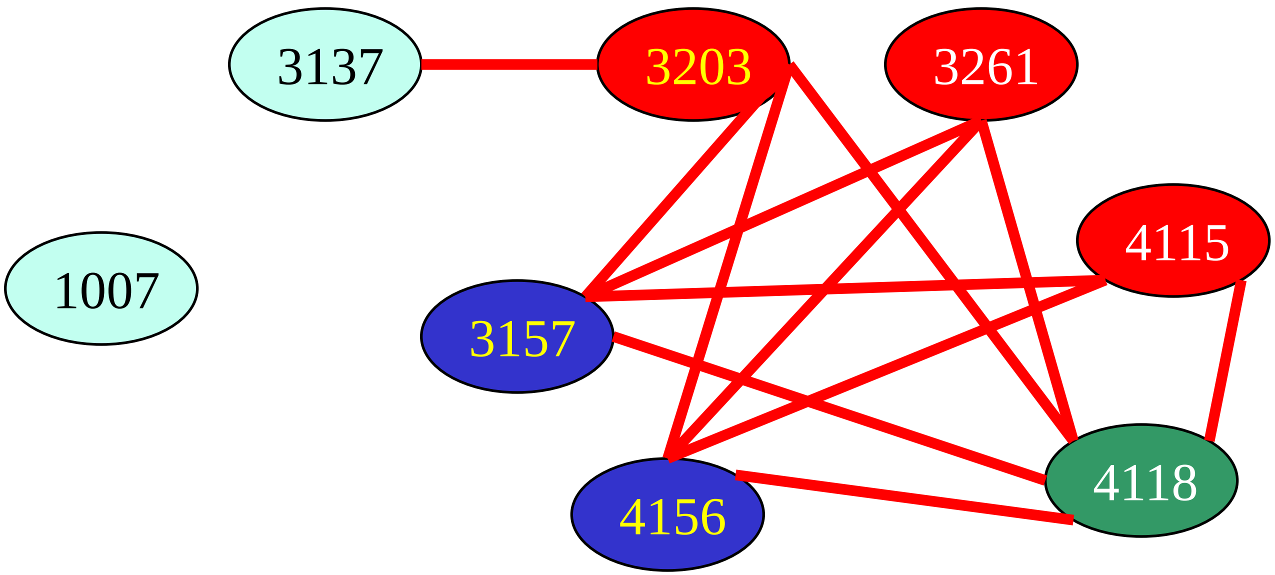
Tô màu đồ thị và Lập lịch

do đó 4156 phải tô bởi Blue:



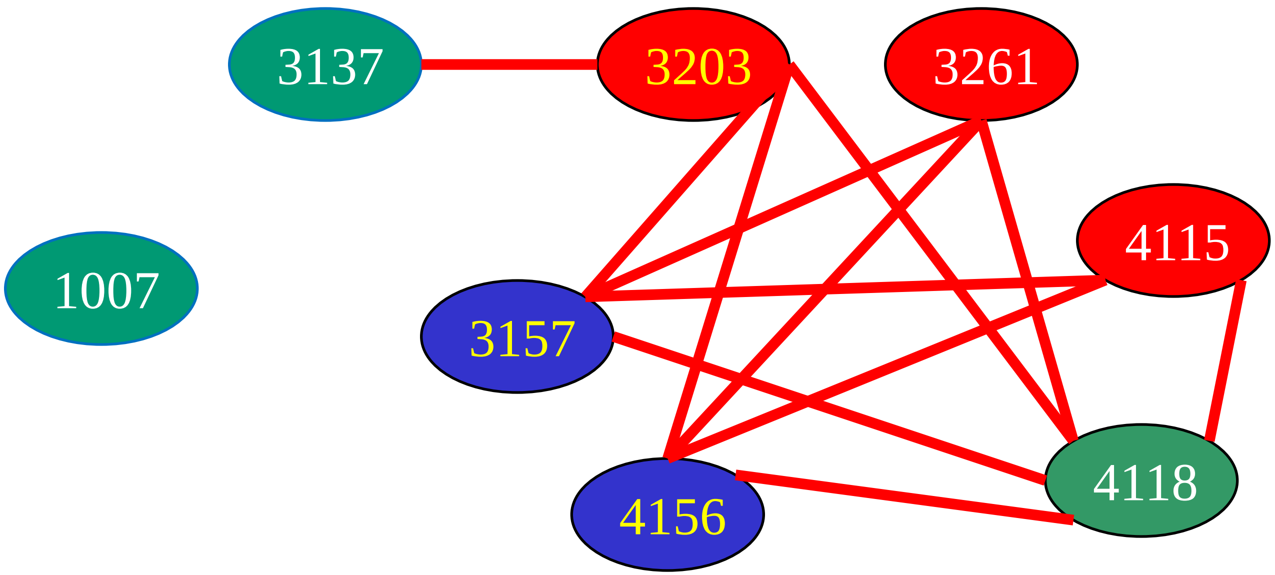
Tô màu đồ thị và Lập lịch

vì thế 3261 và 4115 phải là Red.



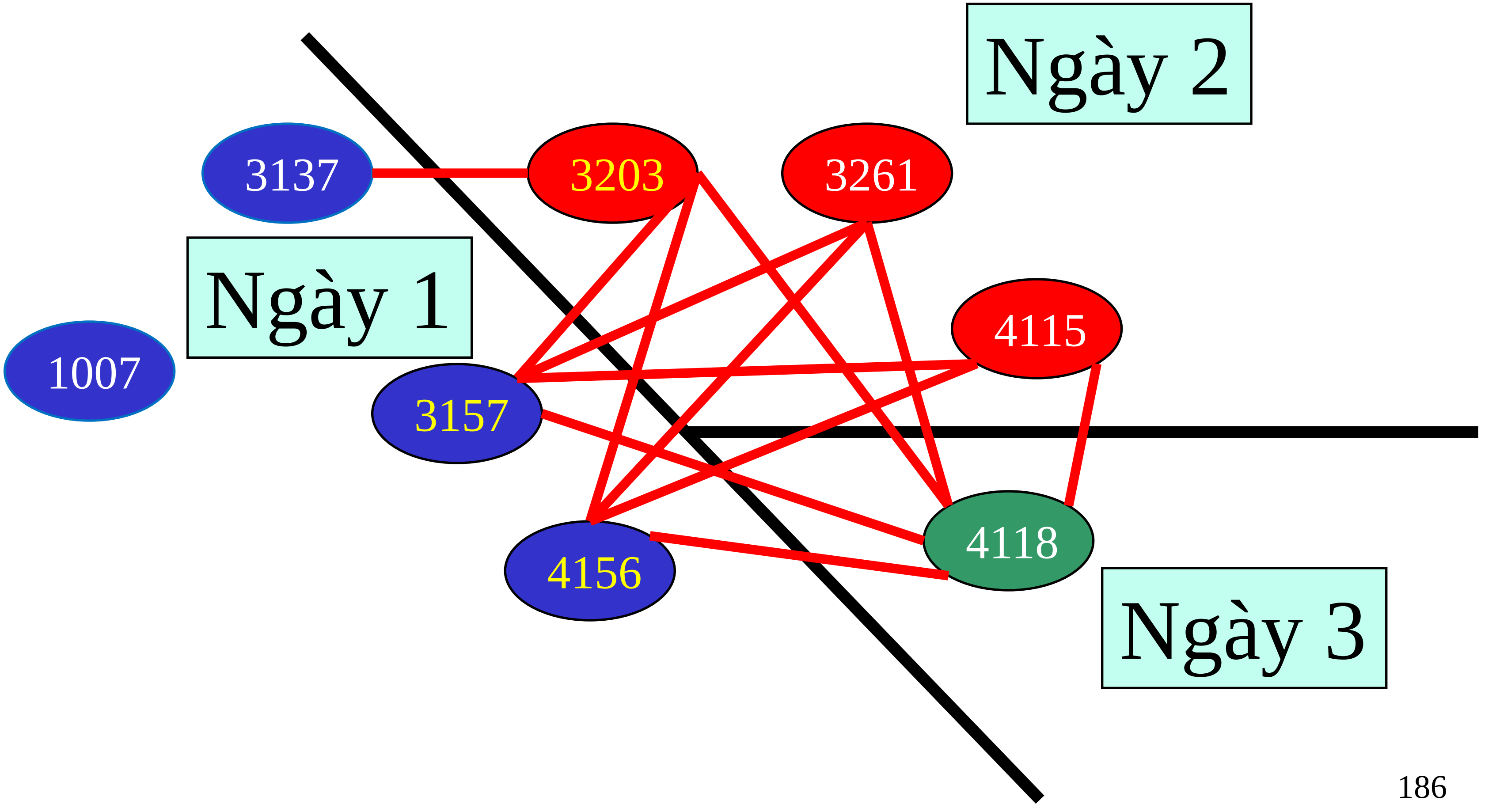
Tô màu đồ thị và Lập lịch

3137 và 1007 dễ dàng tô.



Tô màu đồ thị và Lập lịch

Vậy cần 3 ngày:



Tô màu cạnh

- Ở phần trên ta xét tô màu đỉnh của đồ thị. Một cách hoàn toàn tương tự, ta có thể phát biểu bài toán tô màu cạnh của đồ thị.
- **Định nghĩa.** Ta gọi một phép tô màu cạnh của đơn đồ thị vô hướng $G=(V,E)$ là phép gán cho mỗi cạnh của đồ thị một màu sao cho không có hai cạnh có chung đỉnh nào bị tô bởi một màu.
- Số màu ít nhất cần sử dụng để tô màu các cạnh của đồ thị G được gọi là sắc số cạnh và ký hiệu là $\chi'(G)$

Tô màu cạnh

- **Định lý Vizing.** Đối với đơn đồ thị vô hướng G ta có

$$\Delta(G) \leq \chi'(G) \leq \Delta(G)+1.$$

- **Chứng minh.**
  - Vế trái của bất đẳng thức là hiển nhiên
  - Vế phải có thể chứng minh bằng qui nạp

- **Định lý.** Đối với đơn đồ thị hai phía G ta có

$$\chi'(G) = \Delta(G).$$

- **Chứng minh.** Thuật toán tô màu α/β

Thuật toán tô màu α/β

- Ký hiệu $C = \{1, 2, \ldots, \Delta(G)\}$ là tập màu được sử dụng.
- Lần lượt tô màu các cạnh của đồ thị theo qui tắc sau:
- Giả sử ta đang xét việc tô màu cạnh $e=(u,v)$. Ký hiệu $M(z)$ là tập màu đã dùng để tô các cạnh kề của đỉnh z. Rõ ràng $|M(u)| < \Delta(G)$ và $|M(v)| < \Delta(G)$. Có hai tình huống:
- 1) Nếu tìm được màu $c \in C \setminus (M(u) \cup M(v))$ thì có thể dùng màu c để tô màu cạnh e.

Thuật toán tô màu α/β

2) Không tìm được màu $c \in C \setminus (M(u) \cup M(v))$. Do $|M(u)| < \Delta(G)$ và $|M(v)| < \Delta(G)$ suy ra phải tìm được α là màu chưa được dùng để tô bất cứ cạnh nào kề với u nhưng đã được dùng để tô cạnh kề với v, và β là màu chưa được dùng để tô bất cứ cạnh nào kề với v nhưng đã được dùng để tô cạnh kề với u. Khi đó xuất phát từ u ta đi theo cạnh màu β ta đến đỉnh v_1, nếu trong số các cạnh kề v_1 đã có cạnh được tô màu α thi đi theo cạnh này ta đến đỉnh v_2, … Gọi đường đi tìm được là P. Lật ngược màu α/ β của các cạnh trên đường đi này, khi đó cách tô màu cạnh vẫn hợp lệ, đồng thời màu β có thể được dùng để tô cạnh e.

Chương 2

BIỂU DIỄN ĐỒ THỊ

Representations of Graphs

Biểu diễn đồ thị

- Có nhiều cách biểu diễn. Việc lựa chọn cách biểu diễn phụ thuộc vào từng bài toán cụ thể cần xét, thuật toán cụ thể cần cài đặt.
- Có hai vấn đề chính cần quan tâm khi lựa chọn cách biểu diễn:
  – Bộ nhớ mà cách biểu diễn đó đòi hỏi
  – Thời gian cần thiết để trả lời các truy vấn thường xuyên đối với đồ thị trong quá trình xử lý đồ thị:
    - Chẳng hạn:
      – Có cạnh nối hai đỉnh *u*, *v* ?
      – Liệt kê các đỉnh kề của đỉnh *v* ?

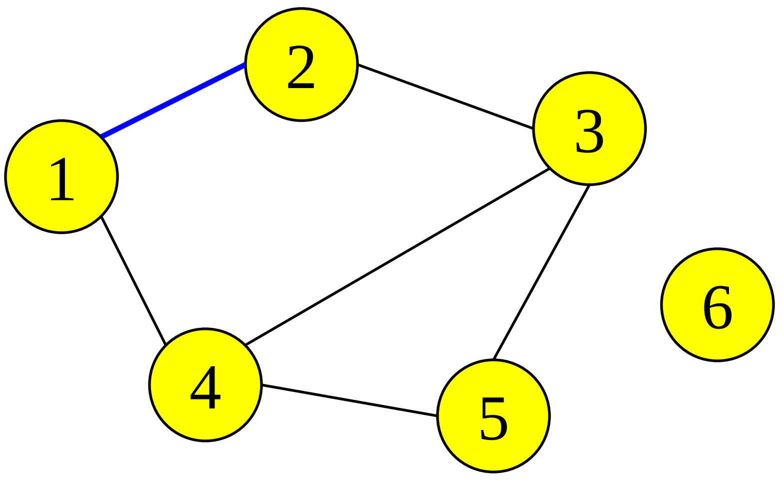
Ma trận kề
(Adjacency Matrix)

- $|V| \times |V|$ ma trận A.
- Các đỉnh được đánh số từ 1 đến $|V|$ theo 1 thứ tự nào đó.
- A xác định bởi:

$$A[i,j] = a_{ij} = \begin{cases} 1 & \text{nÕu } (i,j) \in E \\ 0 & \text{nÕu tr¸i l¹i} \end{cases}$$

- $n = |V|$; $m = |E|$

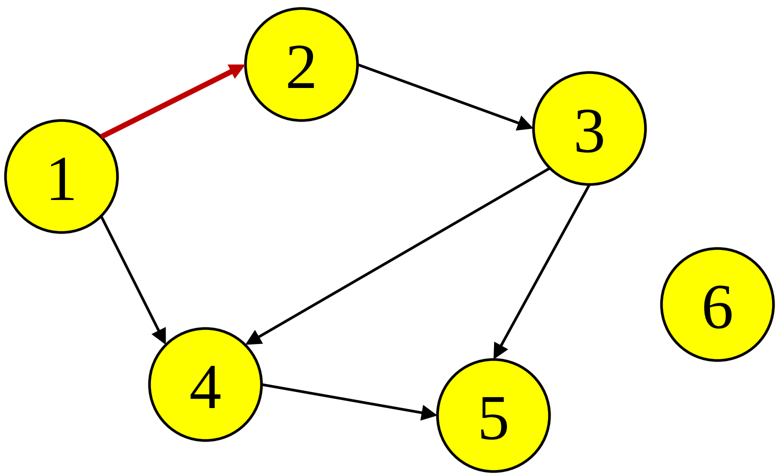
Ma trận kề của đồ thị vô hướng



$$
\begin{array}{c c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} &
\begin{pmatrix}
0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

$$
A[u,v] = \begin{cases} 1 \text{ nếu } (u,v) \in E \\ 0 \text{ nếu trái lại} \end{cases}
$$

Ma trận kề của đồ thị có hướng



$$\begin{array}{c c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}
\end{array}$$

$$A[u,v] = \begin{cases} 1 \text{ nếu } (u,v) \in E \\ 0 \text{ nếu trái lại} \end{cases}$$

Tính chất của ma trận kề

– Gọi A là ma trận kề của đồ thị vô hướng:
  - A là ma trận đối xứng: $A = A^T$ $(a_{ij} = a_{ji})$
  - deg(v) = Tổng các phần tử trên dòng v của A
  - Nếu ký hiệu $A^k = (a^{(k)}[u,v])$ thì $a^{(k)}[u,v]$ là số lượng đường đi từ u đến v đi qua không quá k-1 đỉnh trung gian.

– Khái niệm ma trận kề có thể mở rộng để biểu diễn đa đồ thị vô hướng: a_{uv} – số lượng cạnh nối hai đỉnh u và v.

Phân tích chi phí

- Bộ nhớ (Space)
  - $|V|^2$ bits
  - $(|V|^2 + |V|)/2$ (nếu là đồ thị vô hướng, nhưng khó cài đặt).
  - Các thông tin bổ sung, chẳng hạn chi phí trên cạnh, cần được cất giữ dưới dạng ma trận. Một cách làm khác là cất giữ con trỏ đến các thông tin này.
- Thời gian trả lời các truy vấn
  - Hai đỉnh i và j có kề nhau? $O(1)$
  - Bổ sung hoặc loại bỏ cạnh $O(1)$
  - Bổ sung đỉnh: tăng kích thước ma trận
  - Liệt kê các đỉnh kề của v $O(|V|)$ (ngay cả khi v là đỉnh cô lập).

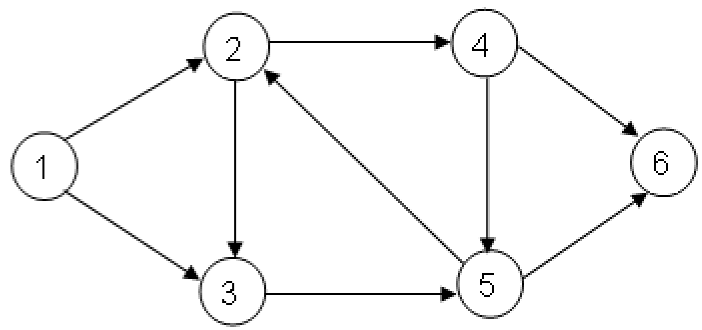
Ma trận liên thuộc đỉnh cạnh

- XÐt $G = (V, E)$, ($V = \{1, 2, ..., n\}$, $E = \{e_1, e_2, ..., e_m\}$), lµ ®¬n ®å thÞ cã h-íng.
- Ma trËn liªn thuéc ®Ønh c¹nh $A = (a_{ij}$: $i = 1, 2, ..., n$; $j = 1, 2, ..., m$), víi

$$a_{ij} = \begin{cases} 1, & \text{nếu đỉnh } i \text{ là đỉnh đầu của cung } e_j \\ -1, & \text{nếu đỉnh } i \text{ là đỉnh cuối của cung } e_j \\ 0, & \text{nếu đỉnh } i \text{ không là đầu mút của cung } e_j \end{cases}$$

- Ma trËn liªn thuéc ®Ønh-c¹nh lµ mét trong nh÷ng c¸ch biÓu diÔn rÊt hay ®-îc sö dông trong c¸c bµi to¸n liªn quan ®Õn ®å thÞ cã h-íng mµ trong ®ã ph¶i xö lý c¸c cung cña ®å thÞ.

Ma trận liên thuộc đỉnh cạnh



$$
A = \begin{array}{cc}
 & \begin{array}{ccccccccc} (1,2) & (1,3) & (2,3) & (2,4) & (3,5) & (4,5) & (4,6) & (5,2) & (5,6) \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} &
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 \\
0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1
\end{bmatrix}
\end{array}
$$

Ma trận trọng số

- Trong tr-êng hîp ®å thÞ cã träng sè trªn c¹nh, thay v× ma trËn kÒ, ®Ó biÓu diÔn ®å thÞ ta sö dông **ma trËn träng sè**

$$C = c[i, j], \quad i, j = 1, 2, ..., n,$$

víi

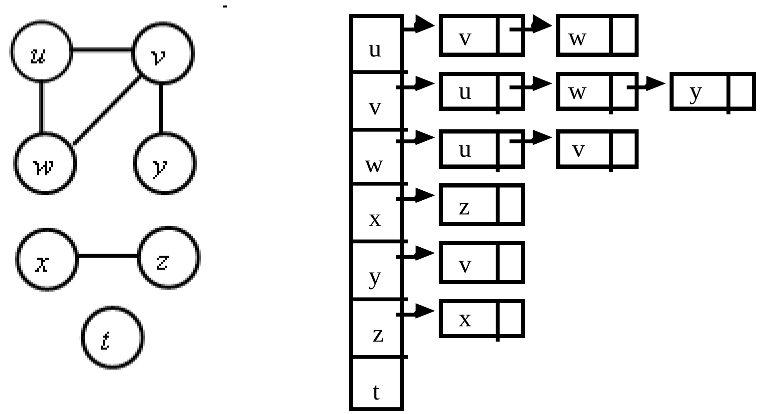
$$c[i,j] = \begin{cases} c(i,j), & \text{nÕu } (i,j) \in E \\ \theta, & \text{nÕu } (i,j) \notin E, \end{cases}$$

trong ®ã θ lµ gi¸ trÞ ®Æc biÖt ®Ó chØ ra mét cÆp (i,j) kh«ng lµ c¹nh, tuú tõng tr-êng hîp cô thÓ, cã thÓ ®-îc ®Æt b»ng mét trong c¸c gi¸ trÞ sau: 0, $+\infty$, $-\infty$.

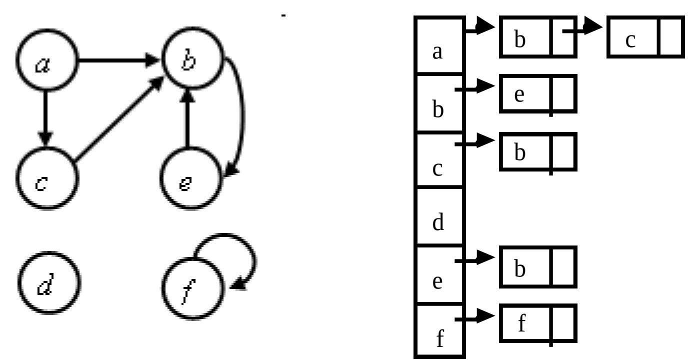
Danh sách kề

- **Danh sách kề** (Adjacency Lists): Với mỗi đỉnh *v* cất giữ danh sách các đỉnh kề của nó.
  - Là mảng *Ke* gồm |*V*| danh sách.
  - Mỗi đỉnh có một danh sách.
  - Với mỗi *u* ∈ *V*, *Ke*[*u*] bao gồm tất cả các đỉnh kề của *u*.
- **Ví dụ:**

Đồ thị vô hướng

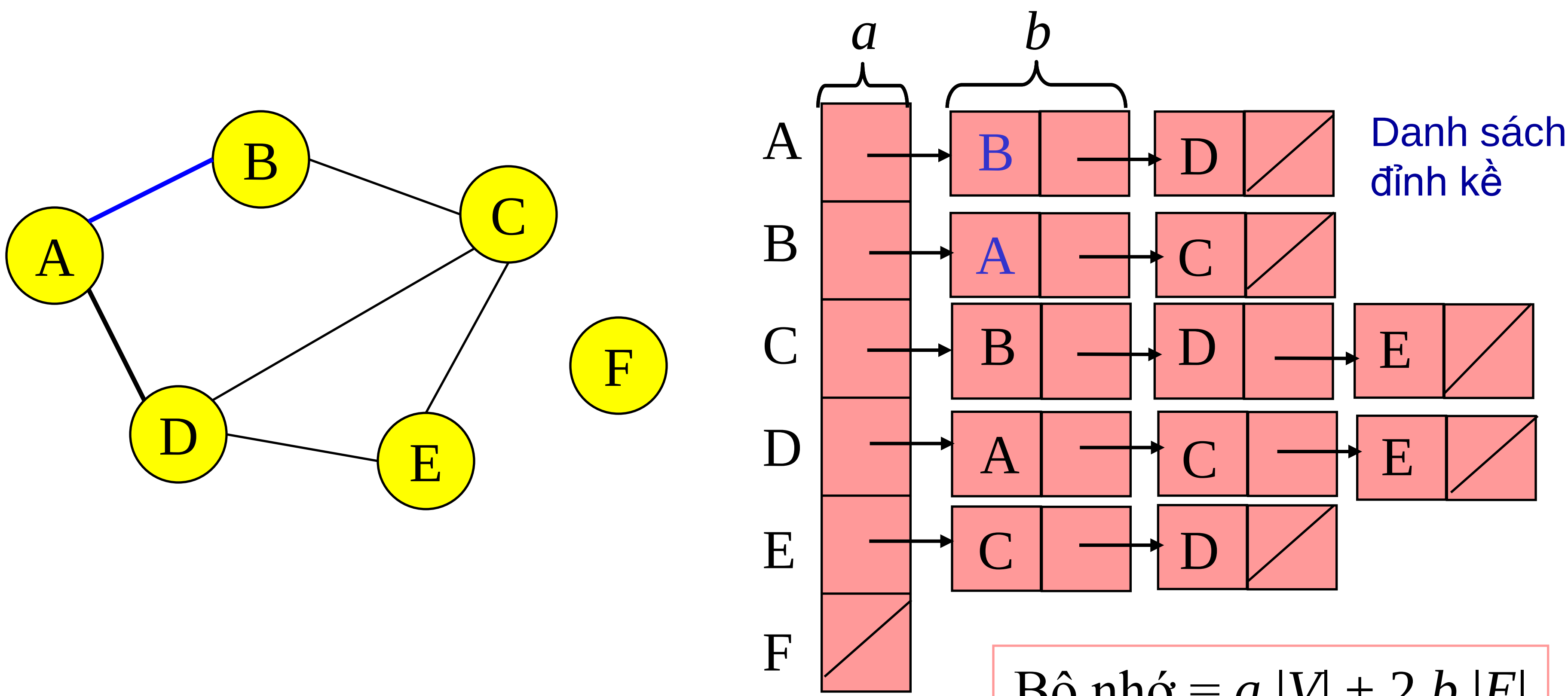


Đồ thị có hướng



Danh sách kề của đồ thị vô hướng

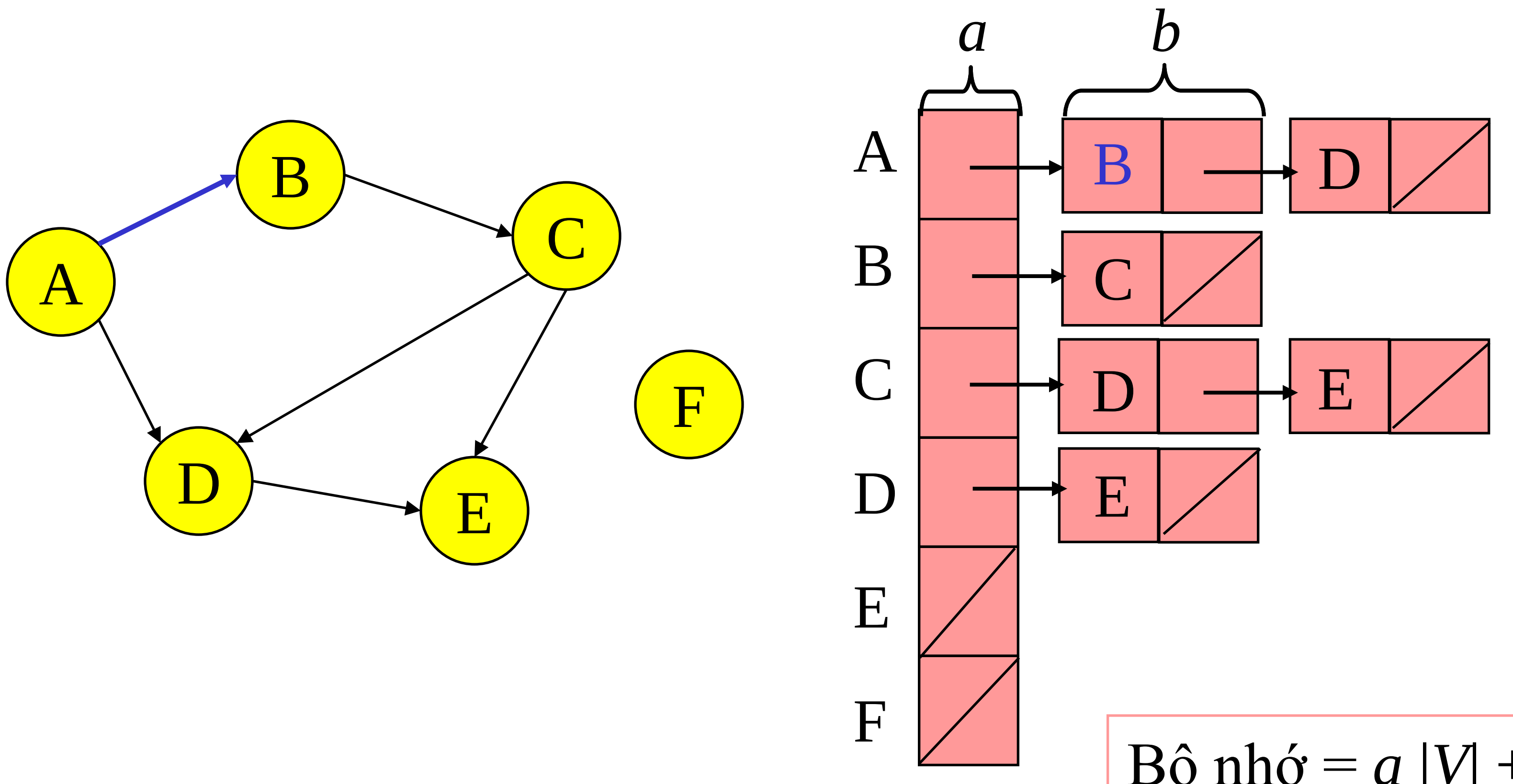
Với mỗi $v \in V$, Ke(v) = danh sách các đỉnh u: $(v, u) \in E$



Bộ nhớ = $a\,|V| + 2\,b\,|E|$

Danh sách kề của đồ thị có hướng

Với mỗi $v \in V$, Ke(v) = { u: (v, u) $\in E$ }



Bộ nhớ = $a\,|V| + b\,|E|$

Yêu cầu bộ nhớ

- Tổng cộng bộ nhớ: $\Theta(|V|+|E|)$
- Thường là nhỏ hơn nhiều so với $|V|^2$, nhất là đối với đồ thị thưa (sparse graph).
- Đồ thị thưa là đồ thị mà $|E| = k\,|V|$ với $k < 10$.
- ***Chú ý:***
  - ***Phần lớn các đồ thị trong thực tế ứng dụng là đồ thị thưa!***
  - ***Cách biểu diễn này được sử dụng nhiều nhất trong ứng dụng***

Biểu diễn đồ thị

- <u>Thời gian trả lời các truy vấn:</u>
  - Thêm cạnh $O(1)$
  - Xoá cạnh Duyệt qua danh sách kề của mỗi đầu mút.
  - Thêm đỉnh Phụ thuộc vào cài đặt.
  - Liệt kê các đỉnh kề của *v*: O(<số đỉnh kề>) (tốt hơn ma trận kề)
  - Hai đỉnh *i*, *j* có kề nhau?
    - Tìm kiếm trên danh sách: $\Theta(\text{degree}(i))$. Đánh giá trong tình huống tồi nhất là $O(|V|)$ => không hiệu quả (tồi hơn ma trận kề)

Chương 3

C¸c thuËt to¸n duyÖt ®å thÞ (Graph Searching, Graph Traversal)

Các thuật toán duyệt đồ thị

- Duyệt đồ thị: Graph Searching hoặc Graph Traversal
  - Duyệt qua mỗi đỉnh và mỗi cạnh của đồ thị
- Ứng dụng:
  - Cần để khảo sát các tính chất của đồ thị
  - Là thành phần cơ bản của nhiều thuật toán trên đồ thị
- Hai thuật toán duyệt cơ bản:
  - Tìm kiếm theo chiều rộng (Breadth First Search – BFS)
  - Tìm kiếm theo chiều sâu (Depth First Search – DFS)

Ý tưởng chung của các thuật toán duyệt

Ý tưởng chung:

- Trong quá trình thực hiện thuật toán, mỗi đỉnh ở một trong ba trạng thái:
  – Chưa thăm, thể hiện bởi màu trắng
  – Đã thăm (nhưng chưa duyệt xong), thể hiện bởi màu xám
  – Đã duyệt xong, thể hiện bởi màu đen
- Trạng thái của đỉnh sẽ biến đổi theo qui tắc sau:
  – Thoạt đầu mỗi đỉnh đều có màu trắng (chưa thăm - not visited).
  – Đỉnh đã được thăm sẽ chuyển thành màu xám (trở thành đã thăm nhưng chưa duyệt xong - visited).
  – Khi tất cả các đỉnh kề của một đỉnh *v* là đã được thăm, đỉnh *v* sẽ có màu đen (đã duyệt xong – discovered).

Tìm kiếm theo chiều rộng

Breadth-first Search (BFS)

Tìm kiếm theo chiều rộng
Breadth-first Search

- **Input:** Đồ thị $G = (V, E)$, vô hướng hoặc có hướng.
- **Output:**
  - $d[v]$ = khoảng cách (độ dài của đường đi ngắn nhất) từ s (là đỉnh xuất phát tìm kiếm) đến v, với mọi $v \in V$. $d[v] = \infty$ nếu v không đạt tới được từ s.
  - $\pi[v] = u$ đỉnh đi trước v trong đường đi từ s (là đỉnh xuất phát tìm kiếm) đến v có độ dài $d[v]$.
  - Xây dựng cây BFS với gốc tại s chứa tất cả các đỉnh đạt tới được từ s.

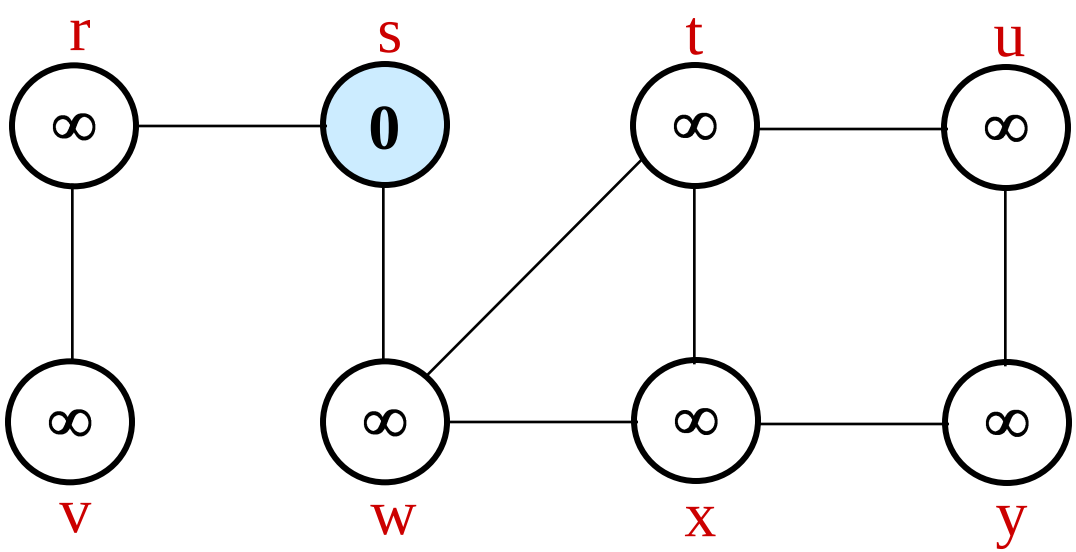
```
Procedure BFS(s);
(* Tìm kiếm theo chiều rộng bắt đầu từ đỉnh s *)
begin
     color[s] ← gray;
     d[s] ← 0;  π[s] ← nil;
     Q ← ∅; enqueue(Q,s); (* Nạp s vào Q *)
     while Q ≠ ∅ do
     begin
        u ← dequeue(Q);  (* Lấy u từ Q *)
        for  v ∈Adj[u] do
           if color[v] = white then
           begin
              color[v] ← gray;
              d[v] ← d[u] + 1; π[v] ← u;
              enqueue(Q,v)   (* Nạp v vào Q *)
           end;
        color[u] ← black
     end;
end;
BEGIN (* Main Program*)
     for  v  ∈ V do (* Khởi tạo *)
     begin
        color[v] ← white;  d[v] ← ∝;   π[v] ← nil;
     end;
     for  v  ∈ V do
       if color[v]=white then  BFS(v);
END.
```

Trắng: chưa thăm
xám: đã thăm
đen: đã duyệt xong

Q: hàng đợi các đỉnh được thăm
color[*v*]: màu của đỉnh v
d[*v*]: khoảng cách từ s đến v
π[*u*]: đỉnh đi trước v

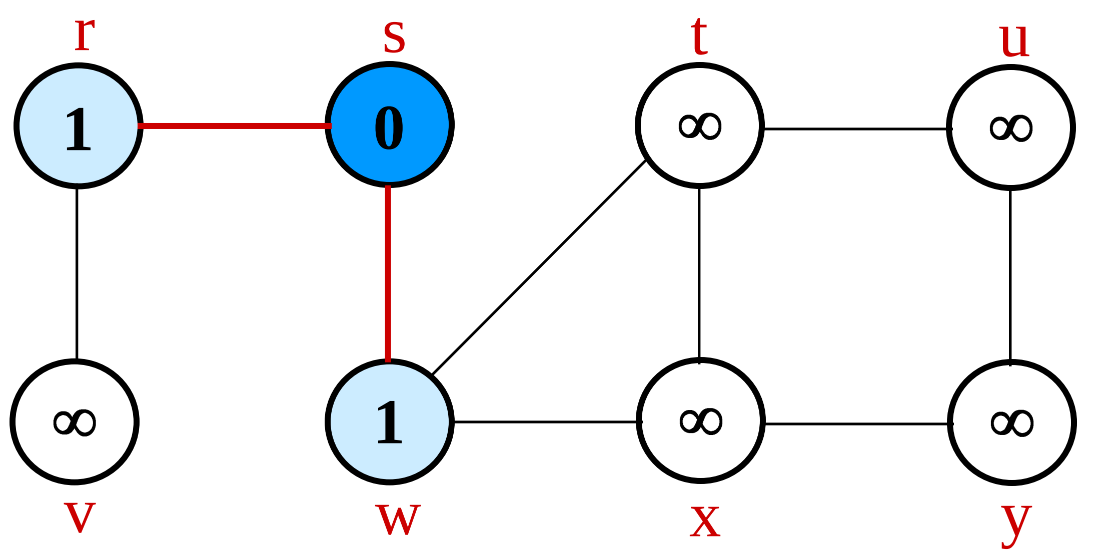
Ví dụ: xem minh hoạ

Ví dụ (BFS)



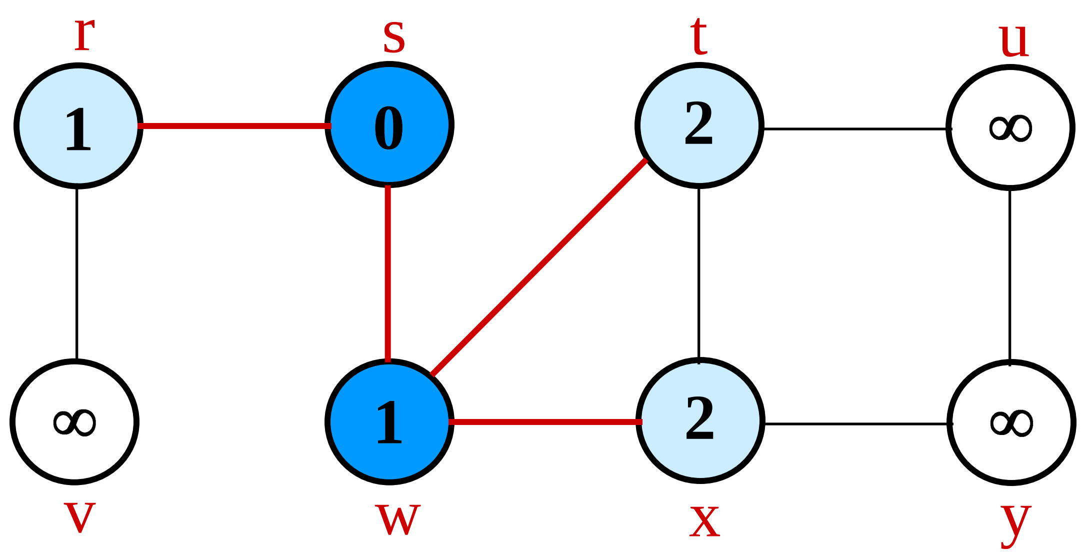
Q: s
0

Ví dụ (BFS)



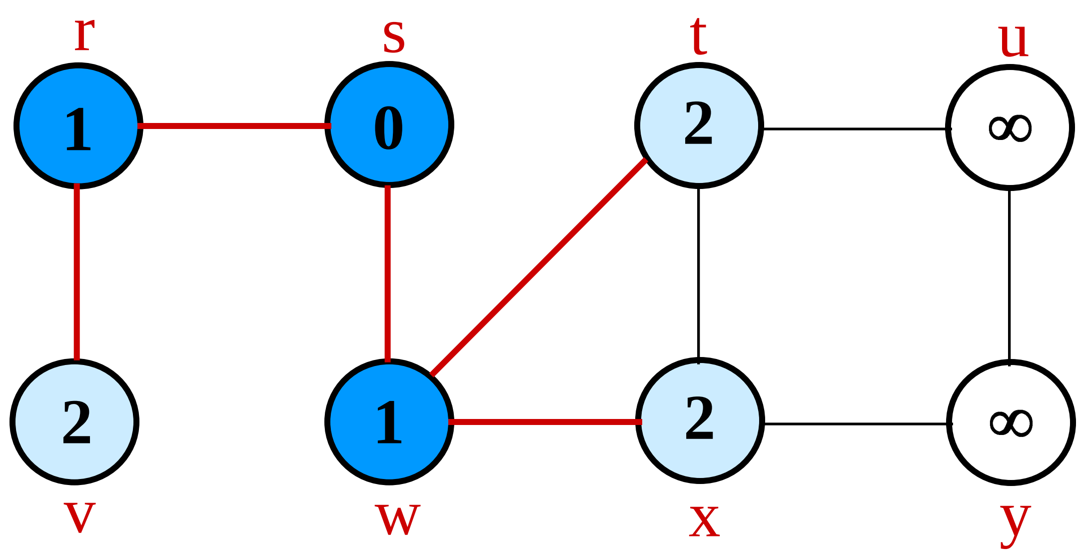
Q:	w	r
	1	1

Ví dụ (BFS)



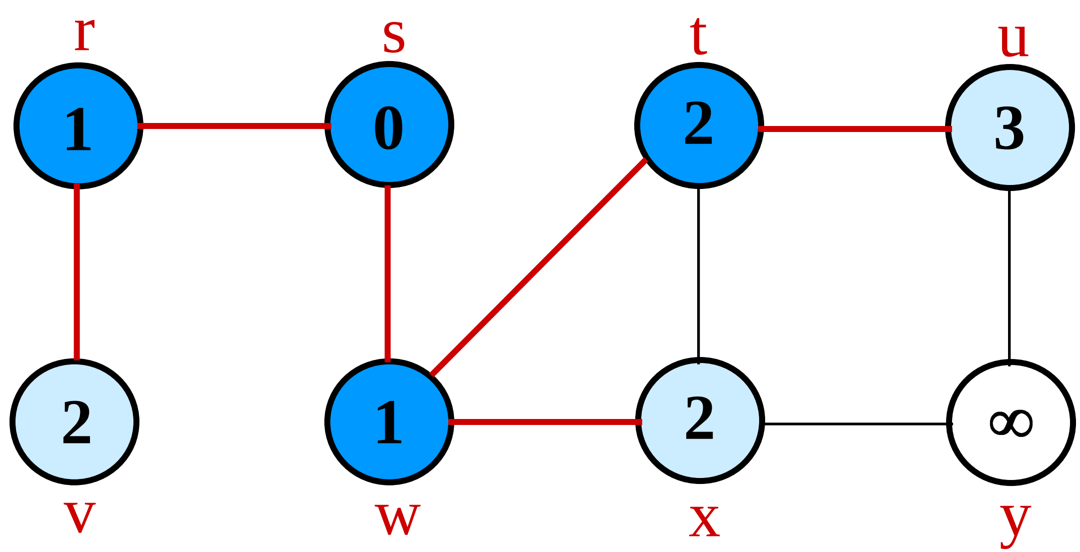
Q:	r	t	x
	1	2	2

Ví dụ (BFS)



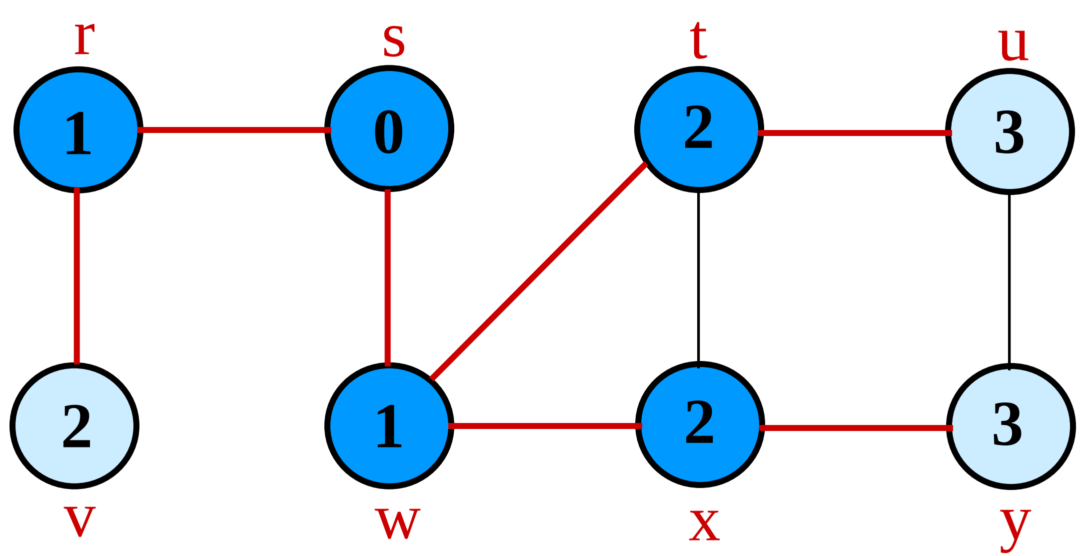
Q: t x v
2 2 2

Ví dụ (BFS)



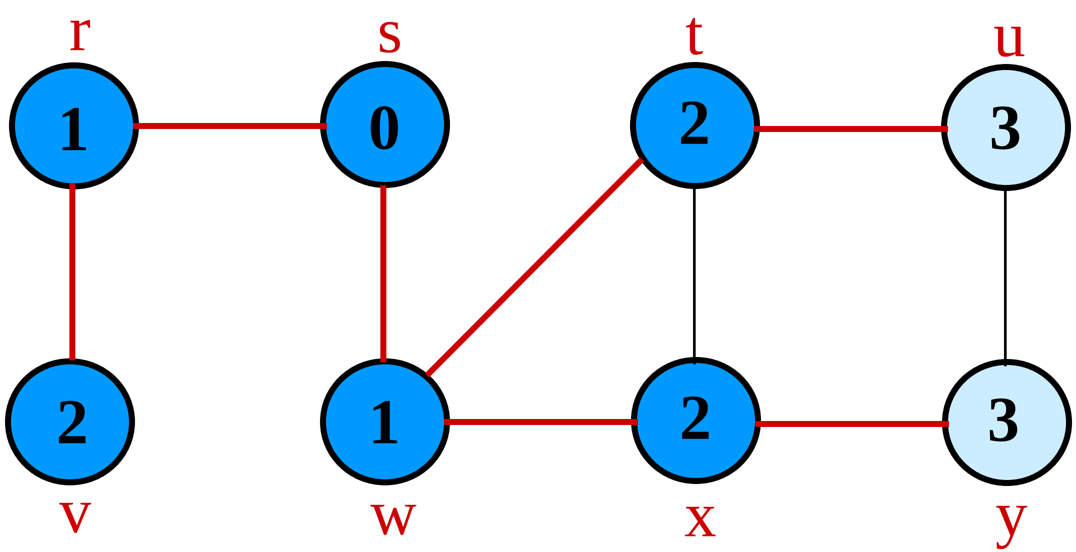
Q:	x	v	u
	2	2	3

Ví dụ (BFS)



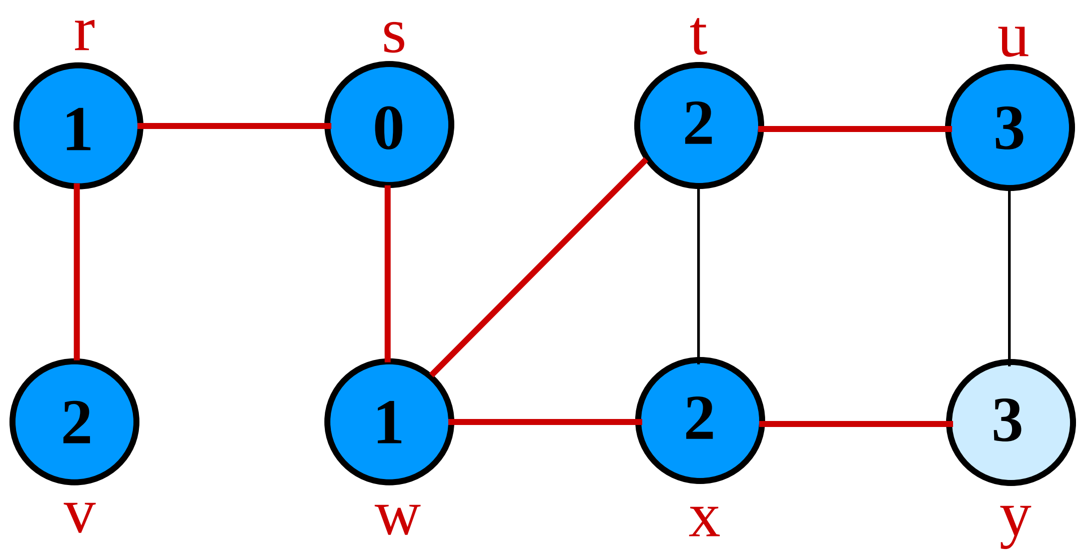
Q:	v	u	y
	2	3	3

Ví dụ (BFS)



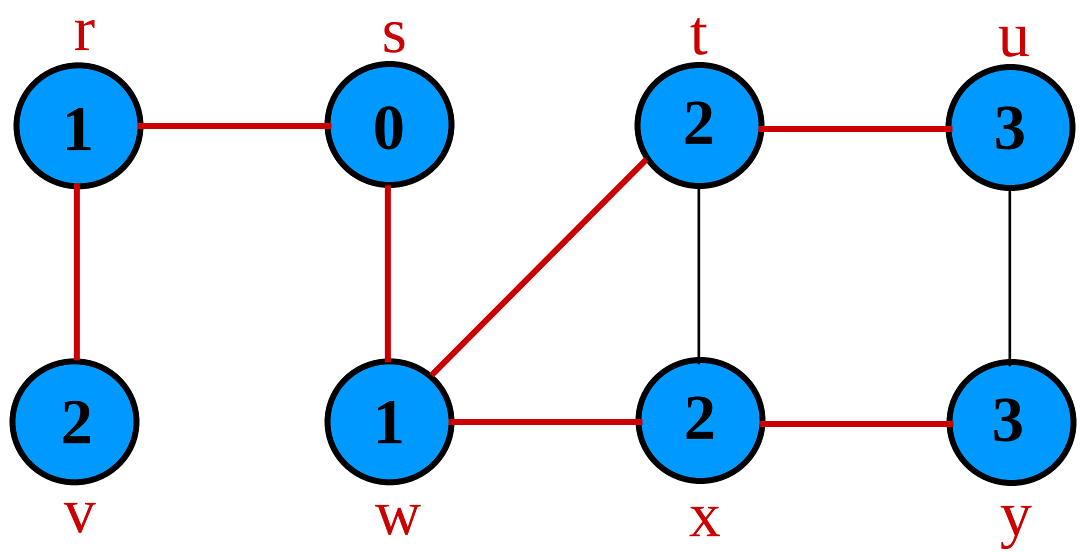
Q: u y
3 3

Ví dụ (BFS)



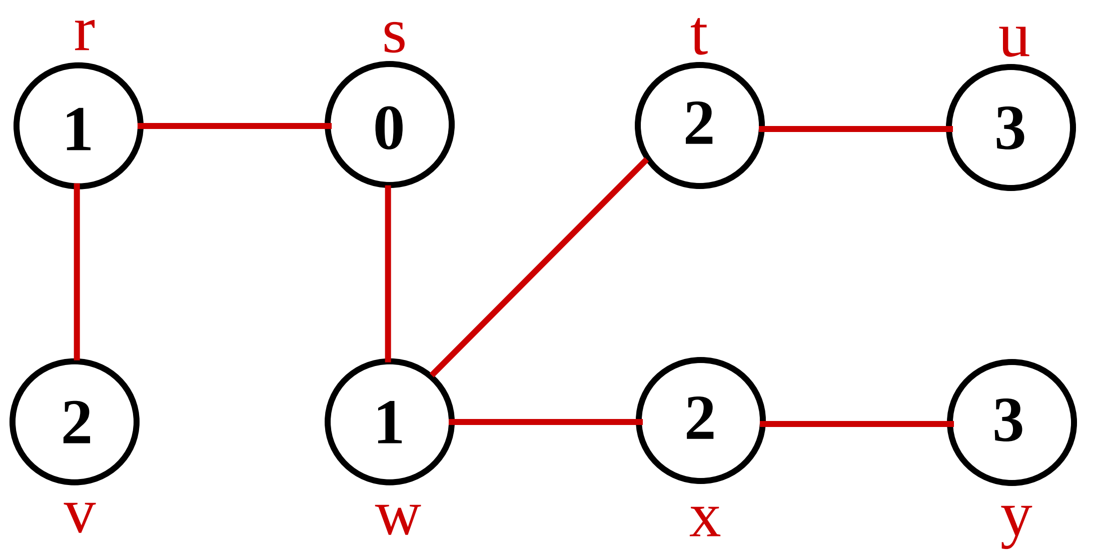
Q: y
3

Ví dụ (BFS)



Q: $\varnothing$

Ví dụ (BFS)



Cây BFS(s)

Phân tích BFS

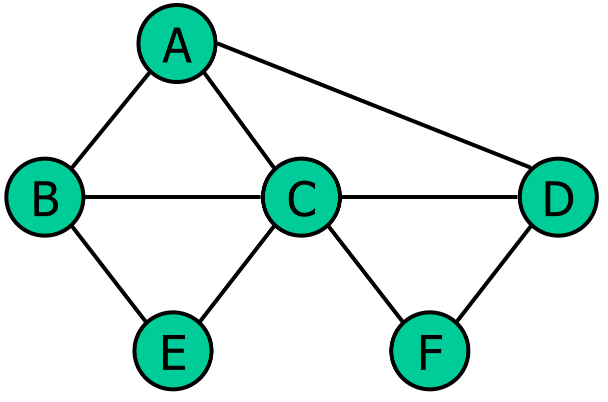
- Việc khởi tạo đòi hỏi $O(|V|)$.
- Vòng lặp duyệt
  - Mỗi đỉnh được nạp vào và loại ra khỏi hàng đợi một lần, mỗi thao tác đòi hỏi thời gian $O(1)$. Như vậy tổng thời gian làm việc với hàng đợi là $O(V)$.
  - Danh sách kề của mỗi đỉnh được duyệt qua đúng một lần. Tổng độ dài của tất cả các danh sách kề là $\Theta(|E|)$.
- Tổng cộng ta có thời gian tính của BFS(s) là $O(|V|+|E|)$,là tuyến tính theo kích thước của danh sách kề biểu diễn đồ thị.

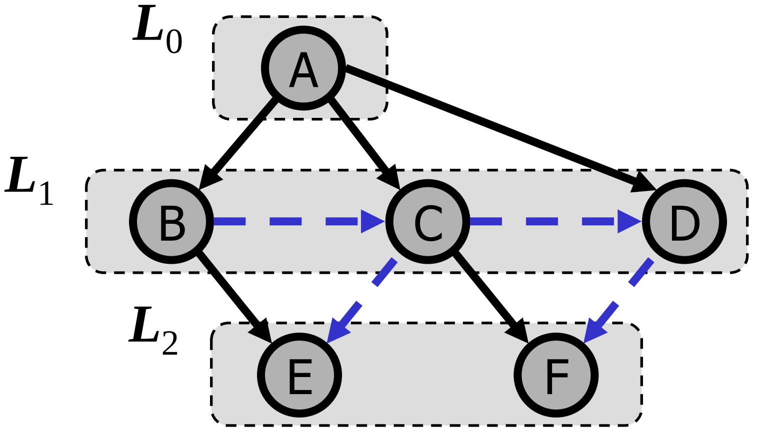
Cây BFS(s)

- Đối với đồ thị $G = (V, E)$ và đỉnh s. Thực hiện BFS(s), xét đồ thị con $G_\pi = (V_\pi, E_\pi)$ trong đó
  - $V_\pi = \{v \in V : \pi[v] \neq \text{NIL}\} \cup \{s\}$
  - $E_\pi = \{(\pi[v], v) \in E : v \in V_\pi \setminus \{s\}\}$
- $G_\pi = (V_\pi, E_\pi)$ là cây và được gọi là cây BFS(s)
- Các cạnh trong E_π được gọi là cạnh của cây. $|E_\pi| = |V_\pi| - 1$.
- BFS(s) cho phép đến thăm tất cả các đỉnh đạt tới được từ s.
- Trình tự thăm các đỉnh khi thực hiện BFS(s): Đầu tiên đến thăm các đỉnh đạt được từ s bởi đường đi qua 1 cạnh, sau đó là thăm các đỉnh đạt được từ s bởi đường đi qua 2 cạnh, …Do đó nếu đỉnh t được thăm trong BFS(s) thì nó sẽ được thăm theo đường đi ngắn nhất theo số cạnh.

BFS – Loang trên đồ thị

- Thứ tự thăm đỉnh nhờ thực hiện BFS(A)





Ứng dụng trực tiếp cuả BFS

- Sử dụng BFS để kiểm tra tính liên thông của đồ thị vô hướng:
  - Mỗi lần gọi đến BFS ở trong chương trình chính sẽ sinh ra một thành phần liên thông
- Xét sự tồn tại đường đi từ đỉnh *s* đến đỉnh *t*:
  - Thực hiện BFS(s).
  - Nếu π[t] = NIL thì không có đường đi, trái lại ta có đường đi

$$t \Leftarrow \pi[t] \Leftarrow \pi[\pi[t]] \Leftarrow \ldots \Leftarrow s$$

- Chú ý: BFS tìm được đường đi ngắn nhất theo số cạnh.

Tìm kiếm theo chiều sâu

Depth-first Search (DFS)

Ý tưởng của tìm kiếm theo chiều sâu

- Ta sÏ b¾t ®Çu t×m kiÕm tõ mét ®Ønh *s* nµo ®ã cña ®å thÞ. Sau ®ã chän *u* lµ mét ®Ønh tuú ý kÒ víi *s* vµ lÆp l¹i qu¸ tr×nh ®èi víi *u*.
- ë b-íc tæng qu¸t, gi¶ sö ta ®ang xÐt ®Ønh *v:*
  - NÕu nh- trong sè c¸c ®Ønh kÒ víi *v* t×m ®-îc ®Ønh *w* lµ ch-a ®-îc thăm th× ta sÏ thăm ®Ønh nµy (nã sÏ trë thµnh ®· thăm nhưng chưa duyệt xong) vµ b¾t ®Çu tõ nã ta sÏ tiÕp tôc qu¸ tr×nh t×m kiÕm.
  - NÕu nh- kh«ng cßn ®Ønh nµo kÒ víi *v* lµ ch-a thăm th× ta sÏ nãi r»ng ®Ønh nµy lµ ®· duyÖt xong vµ quay trë l¹i tiÕp tôc t×m kiÕm tõ ®Ønh mµ tr-íc ®ã ta ®Õn ®-îc ®Ønh *v* (nÕu *v* = *s*, th× kÕt thóc t×m kiÕm).
- Cã thÓ nãi n«m na lµ t×m kiÕm theo chiÒu s©u b¾t ®Çu tõ ®Ønh *s* ®-îc thùc hiÖn trªn c¬ së t×m kiÕm theo chiÒu s©u tõ tÊt c¶ c¸c ®Ønh ch-a thăm kÒ víi *s*.

Mô tả DFS

- **Input:** Đồ thị $G = (V, E)$ cho bởi danh sách kề
- **Output:**
  - **2 mốc thời gian cho mỗi đỉnh (là các số nguyên trong khoảng** 1 và $2|V|$).
    - $d[v]$ = ***thời điểm bắt đầu thăm*** (v chuyển từ trắng sang xám)
    - $f[v]$ = ***thời điểm kết thúc thăm*** (v chuyển từ xám sang đen)
  - $\pi[v]$: đỉnh đi trước v – tức là đỉnh mà từ đó ta đến thăm v.
- Sử dụng biến color để ghi nhận trạng thái của các đỉnh như đã mô tả.

Depth-First Search: Code

```
DFS(G)
BEGIN
    for v∈V do
    begin
        color[v] = WHITE;
        π[v] = NIL
    end;
    time = 0;
    for u∈V do
    begin
        if (color[u]= WHITE) then
            DFS(u);
    end;
END.
```

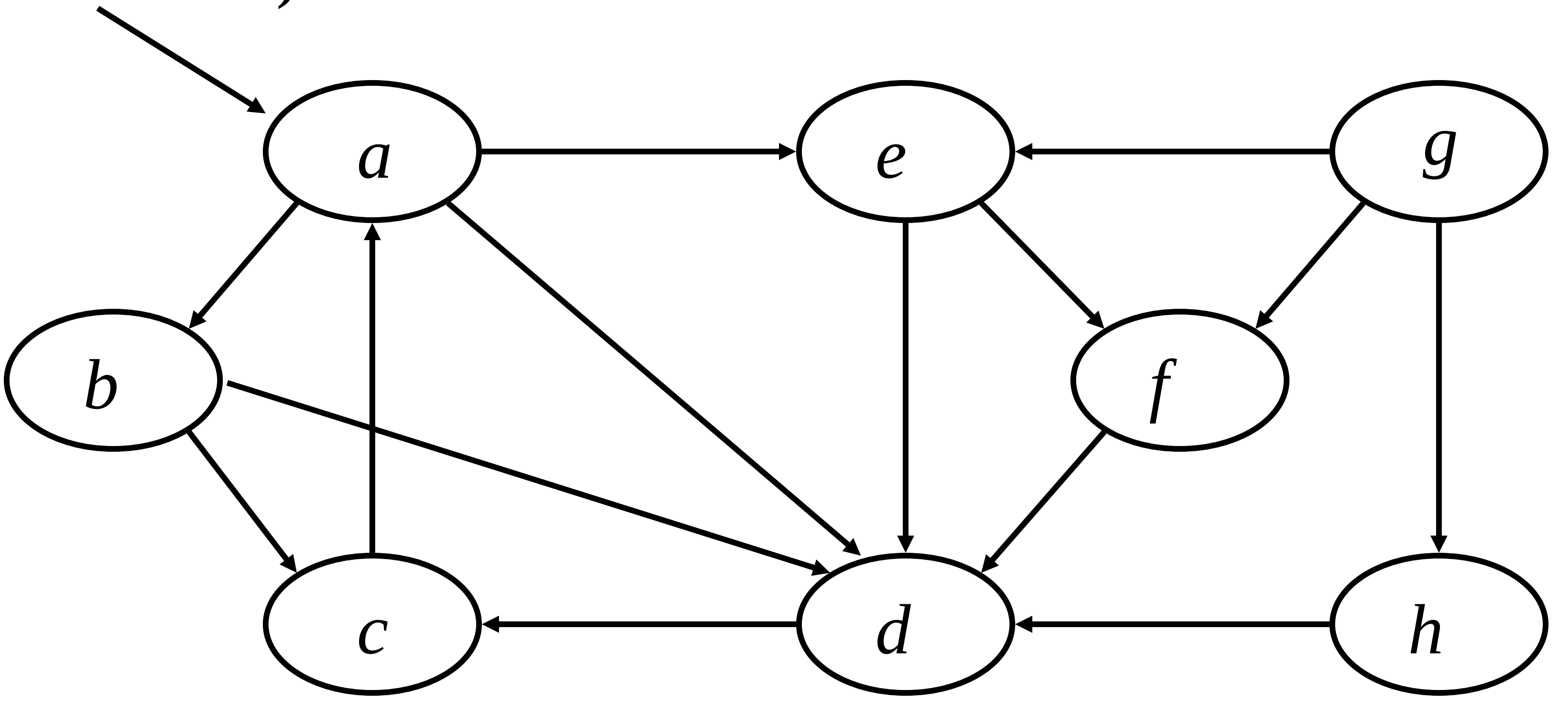
```
procedure DFS(u);
begin
    color[u] = GRAY;
    time = time+1;
    d[u] = time;
    for v ∈ Ke(u)do
        if (color[v]= WHITE)then
        begin
            π[v] = u;
            DFS(v);
        end;
    color[u] = BLACK;
    time = time+1;
    f[u] = time;
end;
```

Phân tích thuật toán DFS

- Mỗi đỉnh được thăm đúng 1 lần, việc thăm mỗi đỉnh đòi hỏi chi phí thời gian $O(1)$, suy ra thao tác thăm đỉnh đòi hỏi thời gian $O(|V|)$.
- Vòng lặp trong DFS(u) thực hiện việc duyệt cạnh của đồ thị
  - Mỗi cạnh được duyệt qua đúng một lần nếu đồ thị là có hướng và 2 lần nếu đồ thị là vô hướng
  - Như vậy tổng số lần lặp là $O(|E|)$.
- Vậy, thuật toán có thời gian $O(|V|+|E|)$
- Đối với đồ thị, thuật toán có đánh giá như vậy gọi là ***thuật toán thời gian tuyến tính***

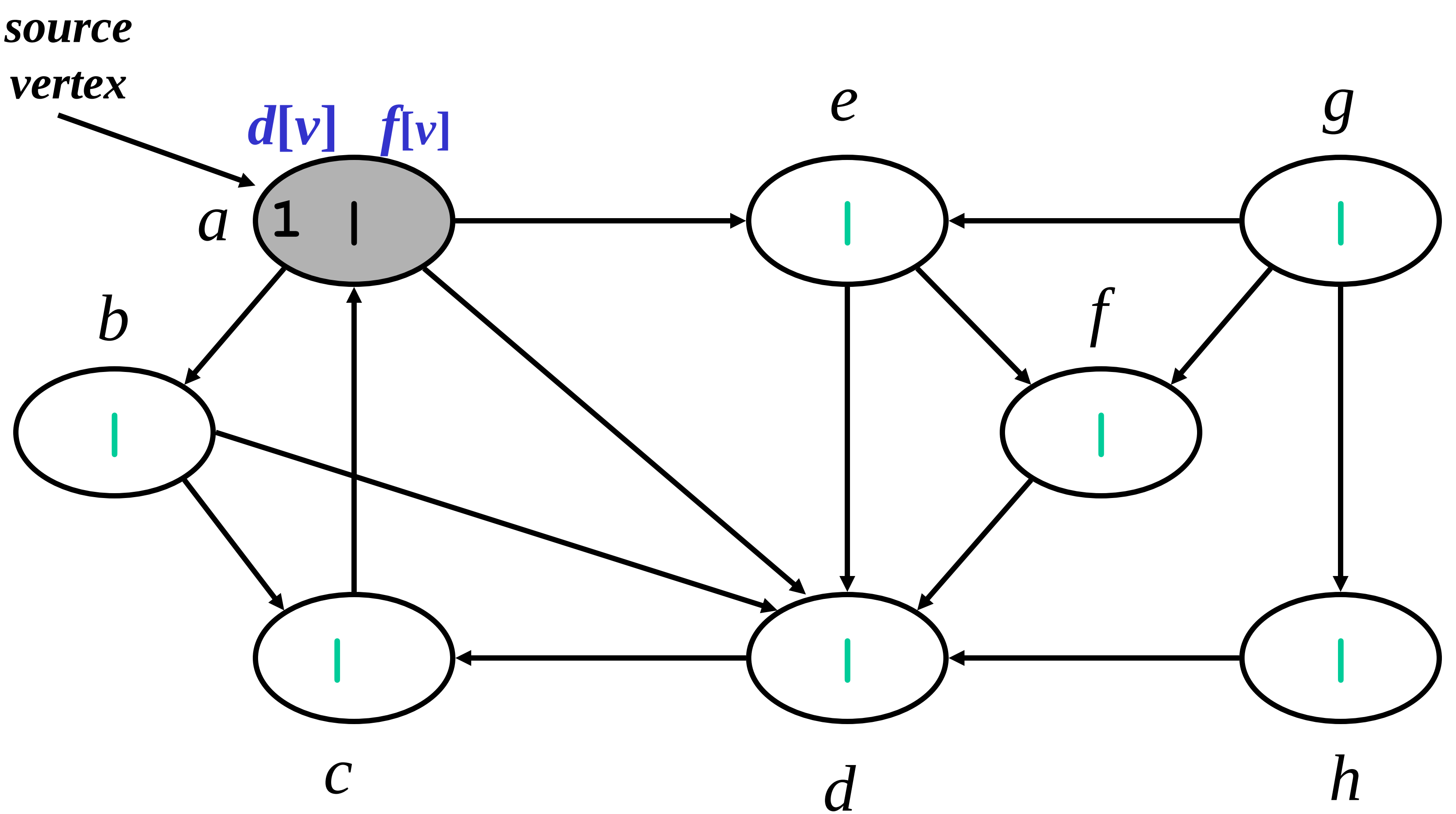
Ví dụ: DFS

Đỉnh xuất phát tìm kiếm (Source vertex)

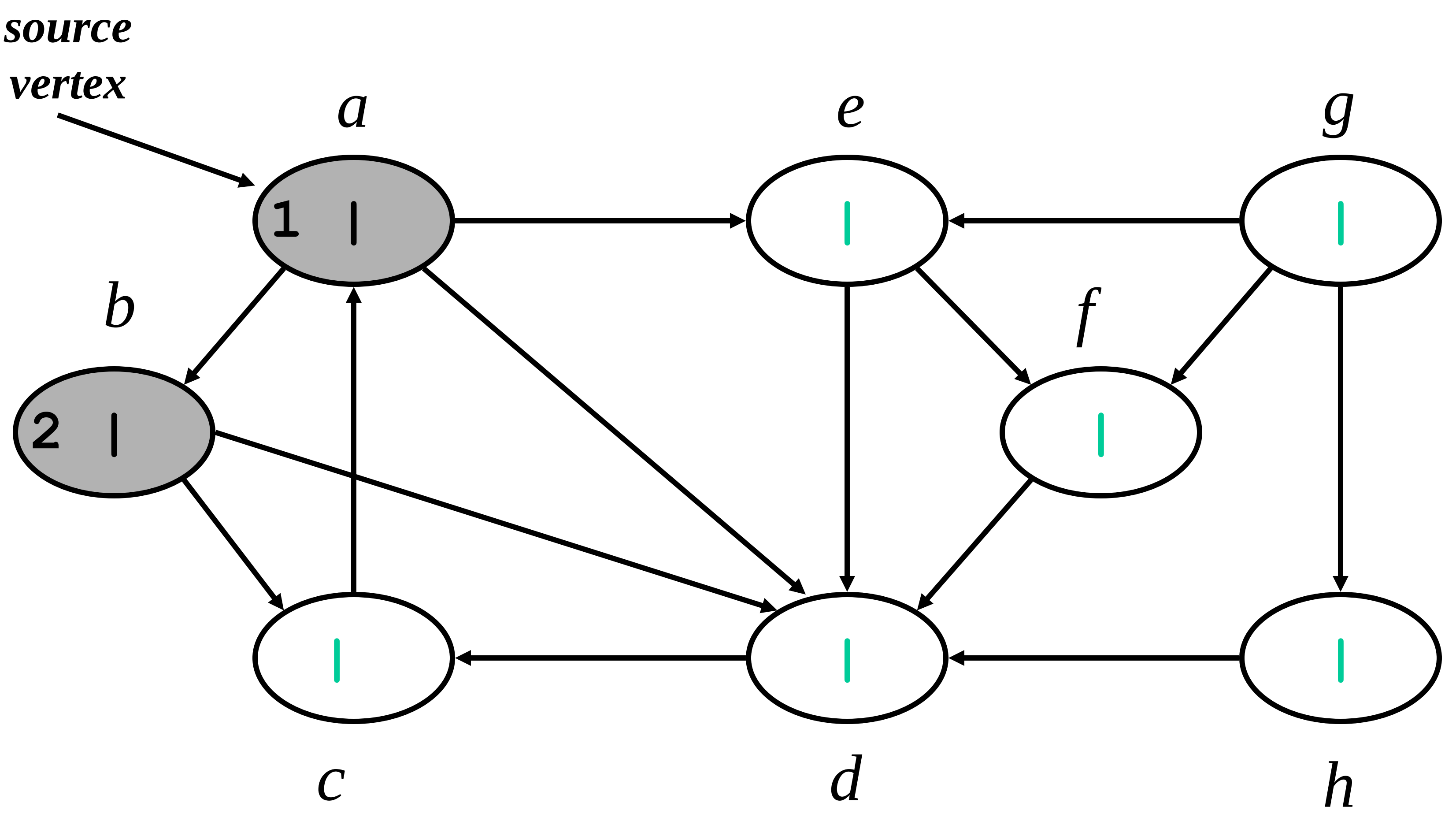


Để hoạt động của thuật toán là xác định, giả thiết rằng ta duyệt các đỉnh trong danh sách kề của một đỉnh theo thứ tự từ điển

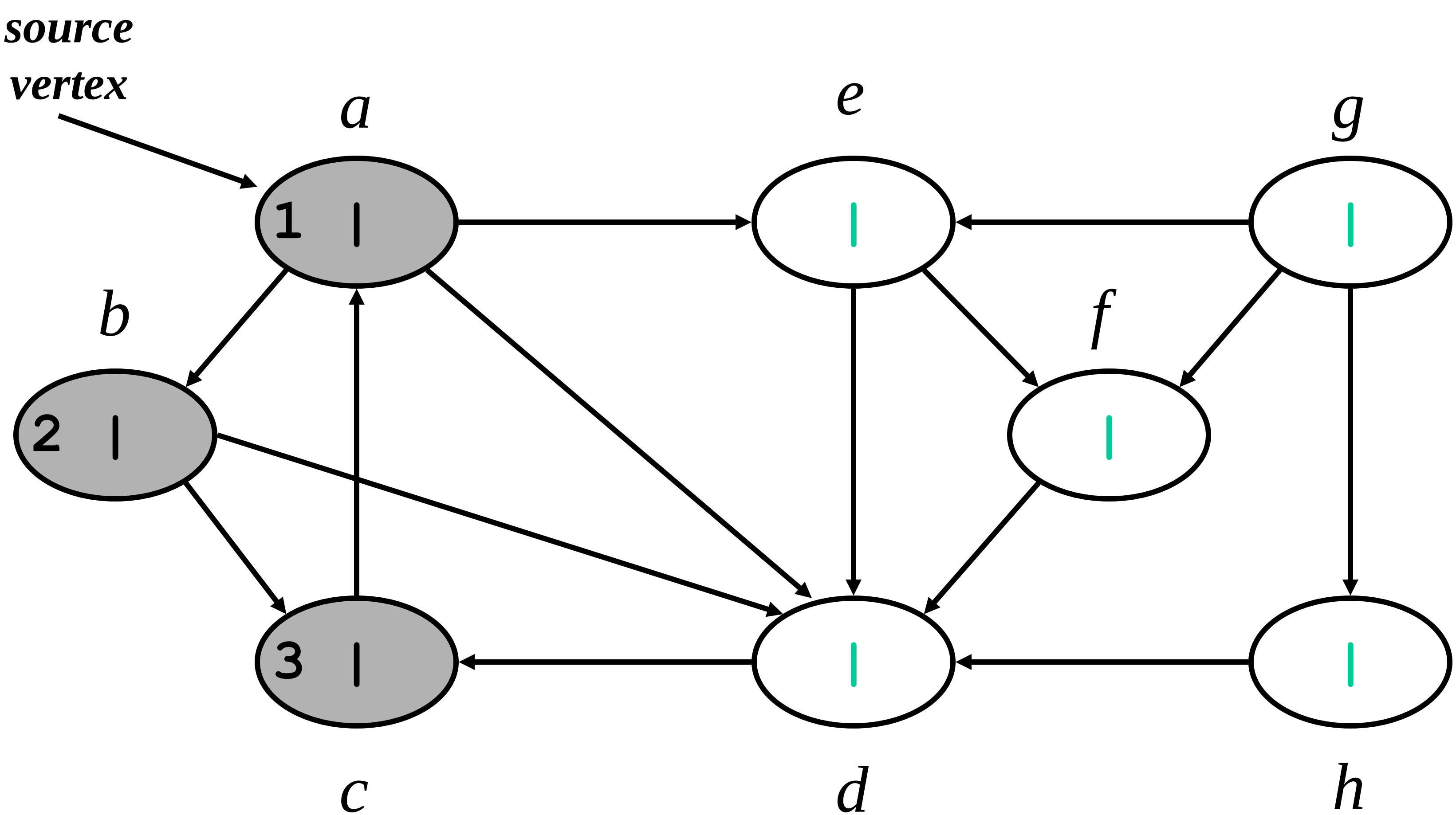
Ví dụ: DFS



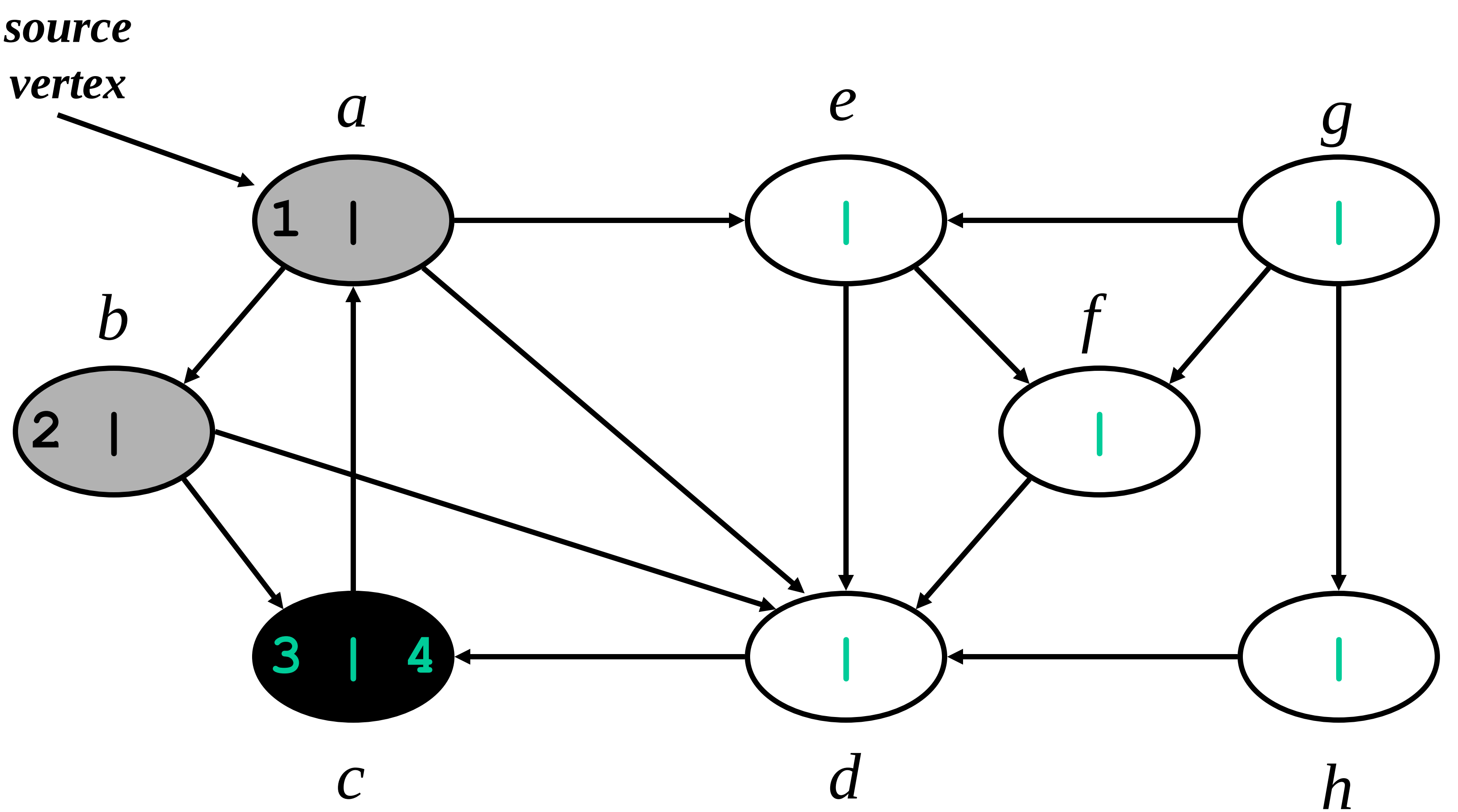
Ví dụ: DFS



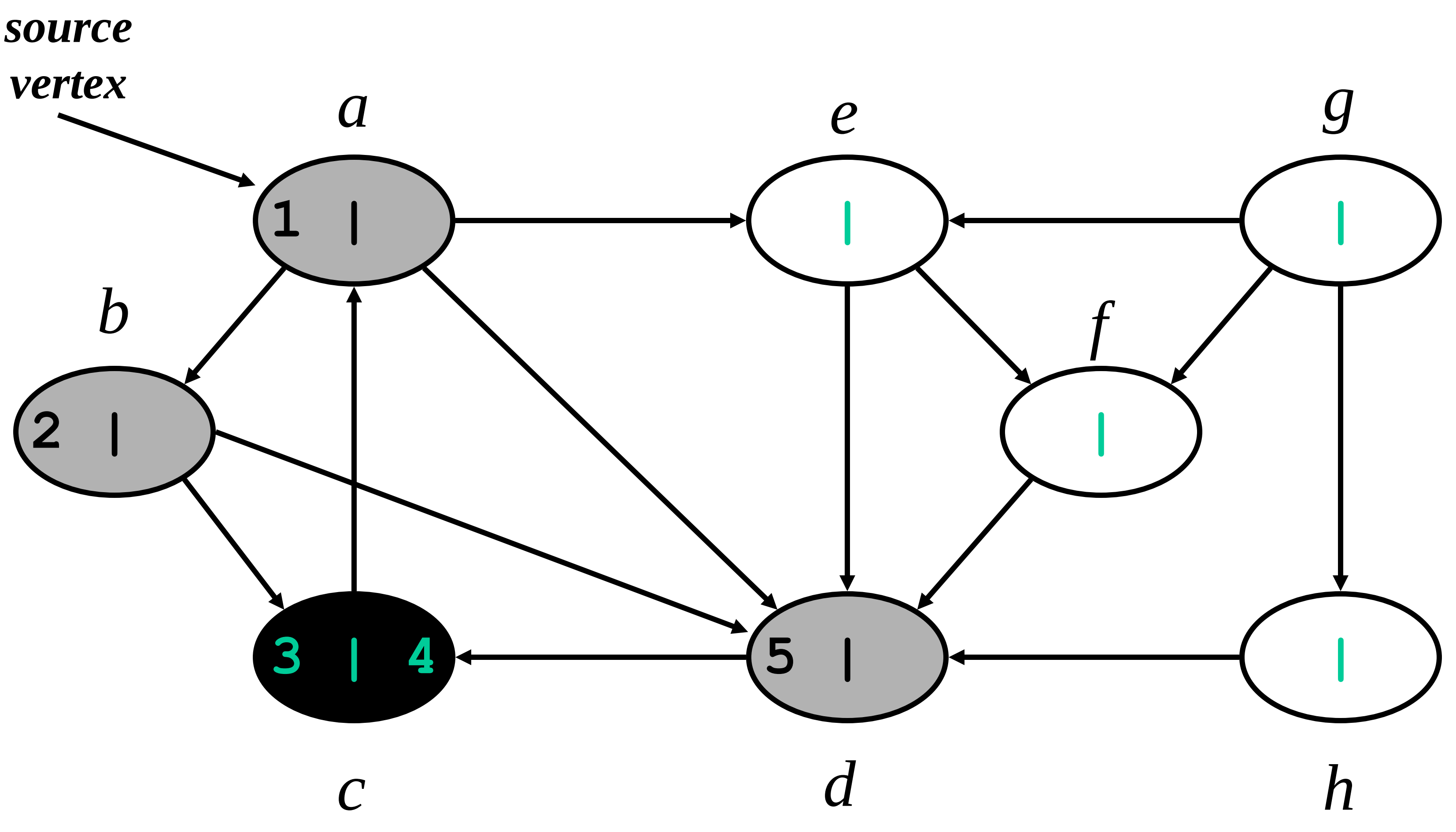
Ví dụ: DFS



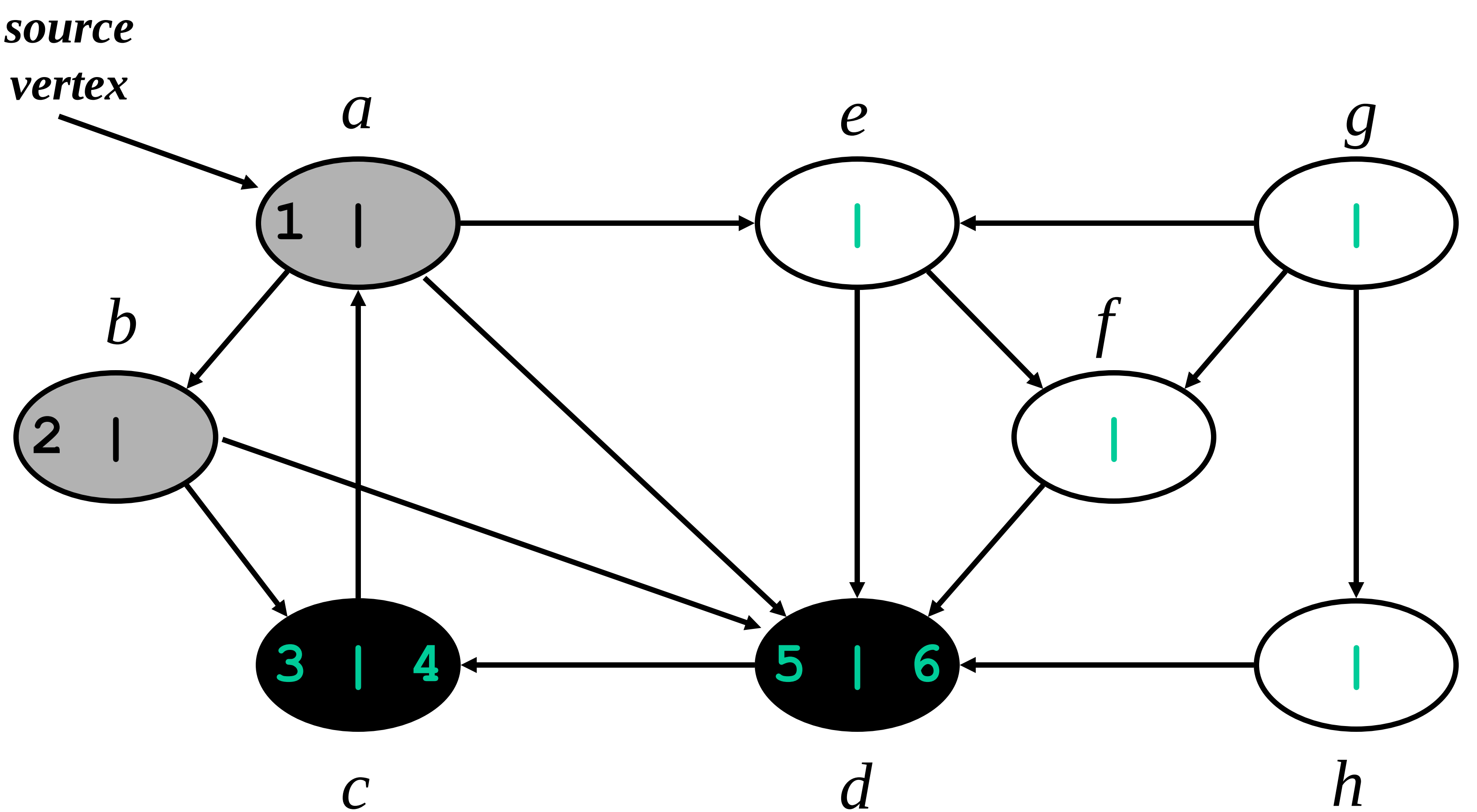
Ví dụ: DFS



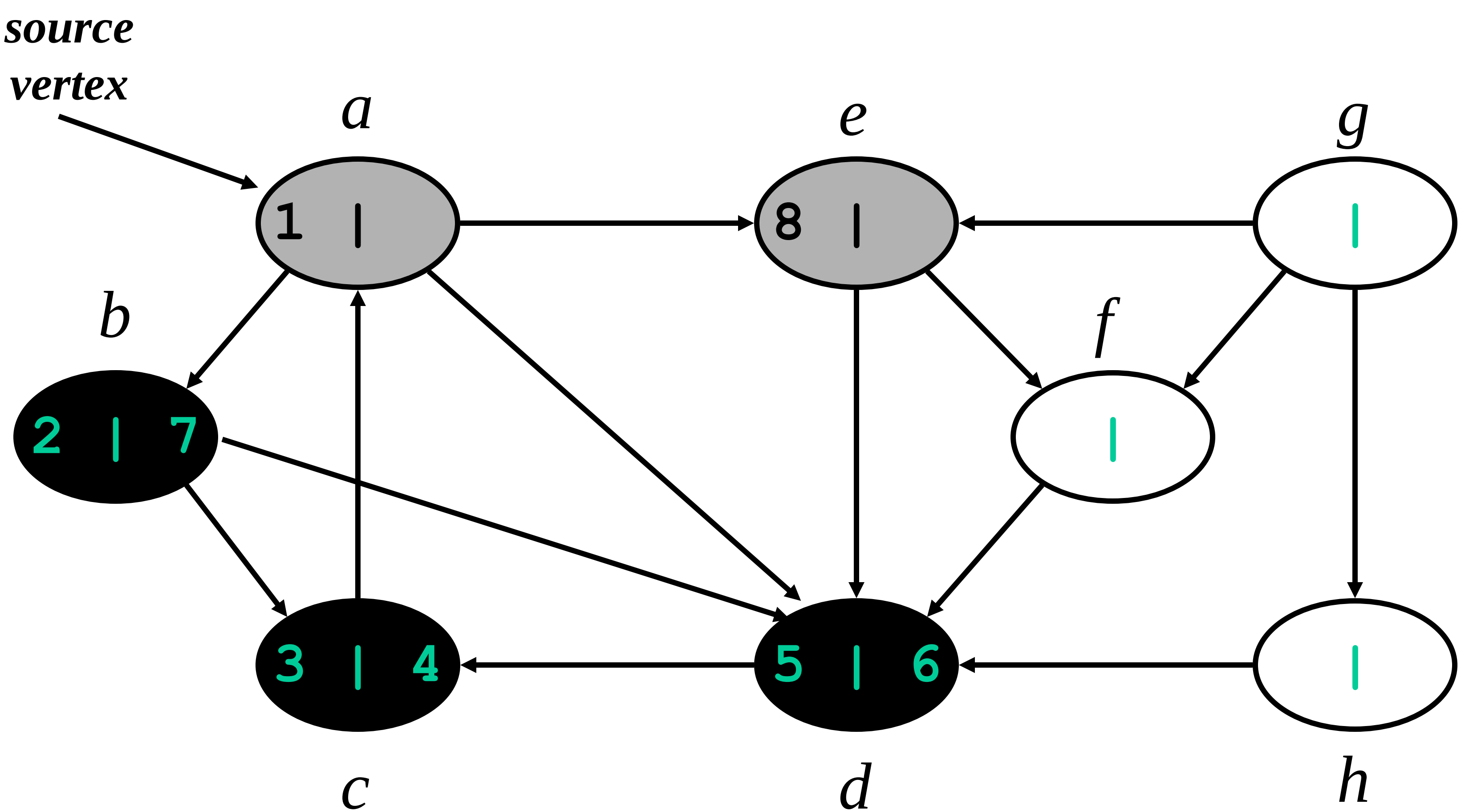
Ví dụ: DFS



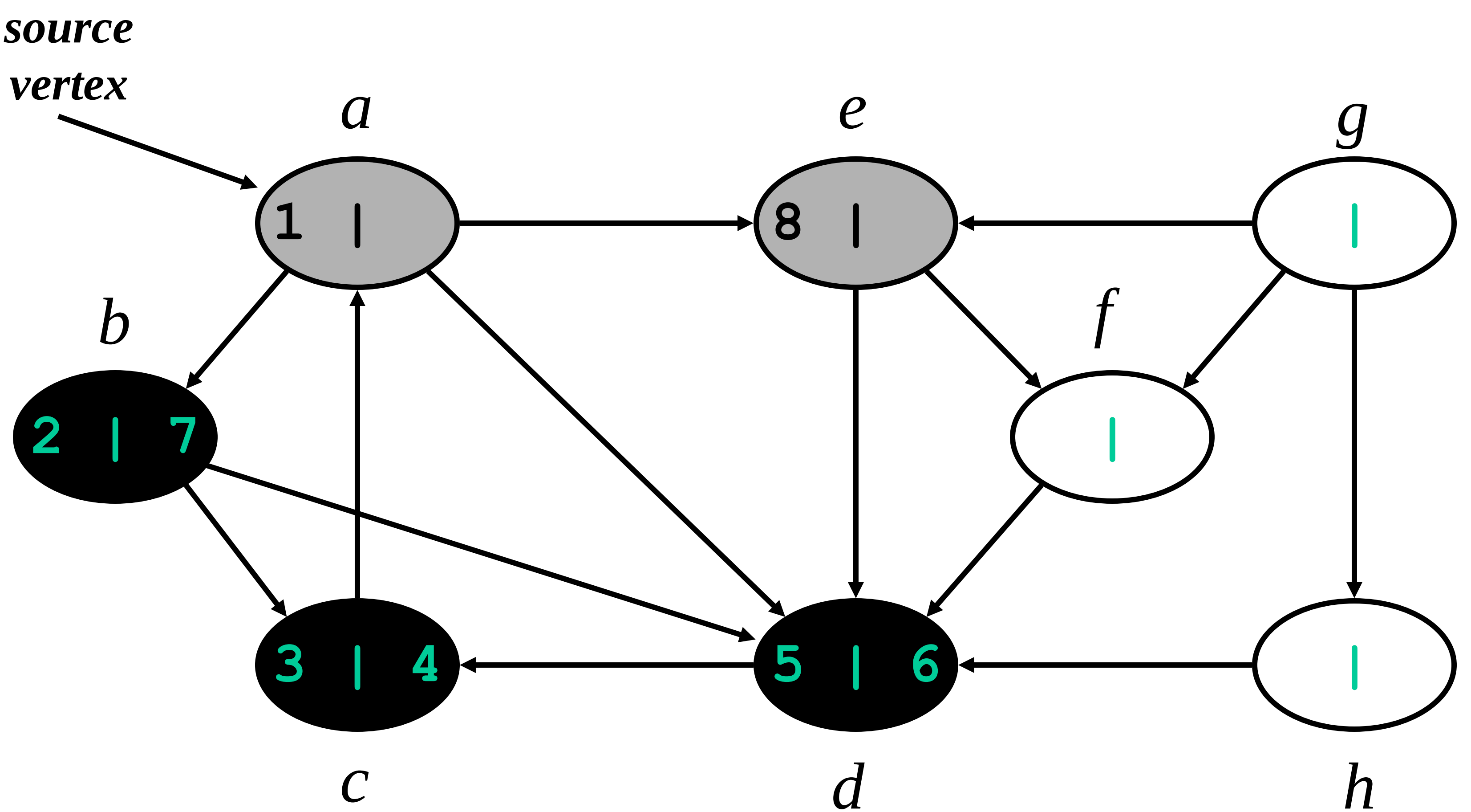
Ví dụ: DFS



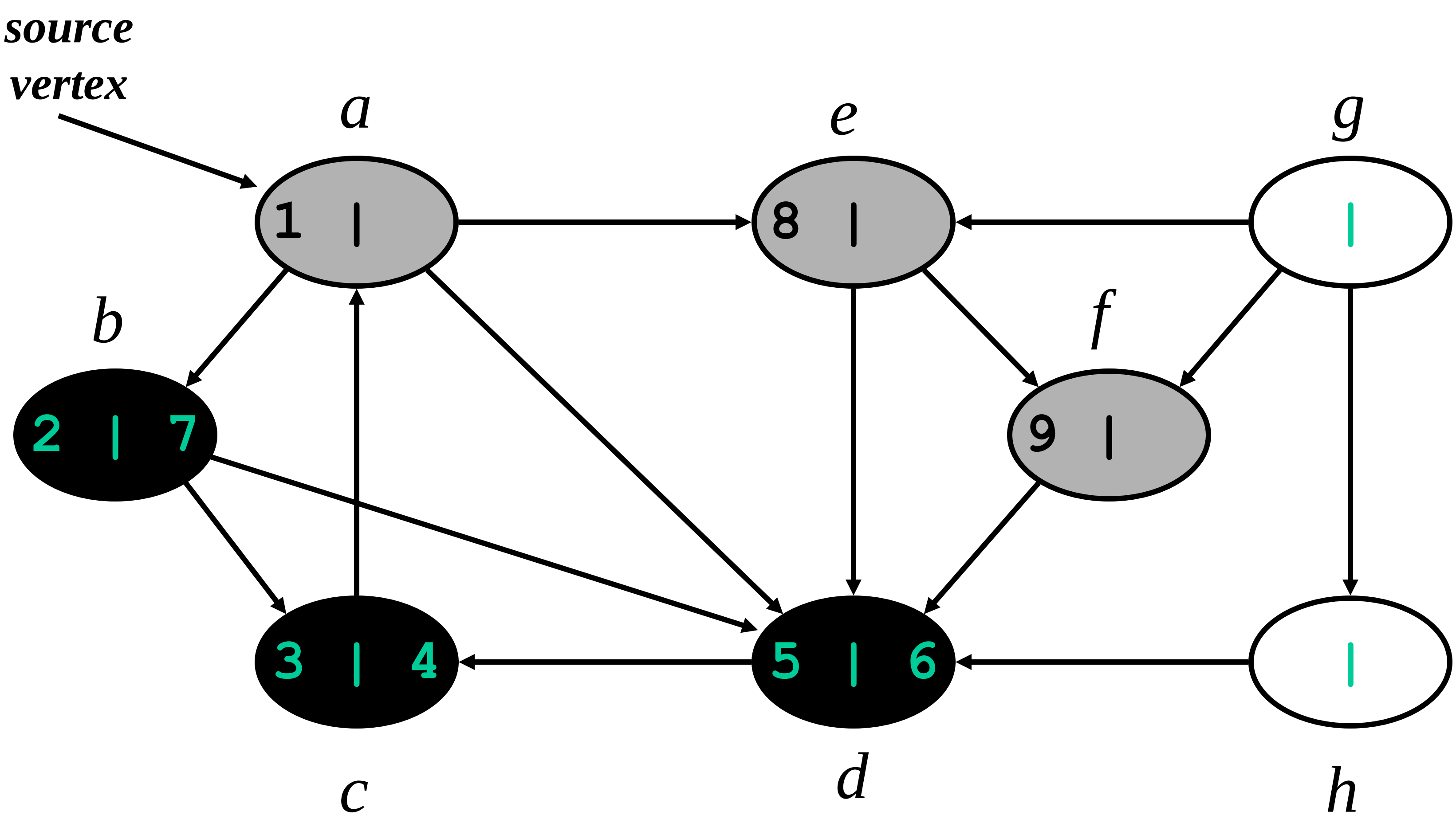
Ví dụ: DFS



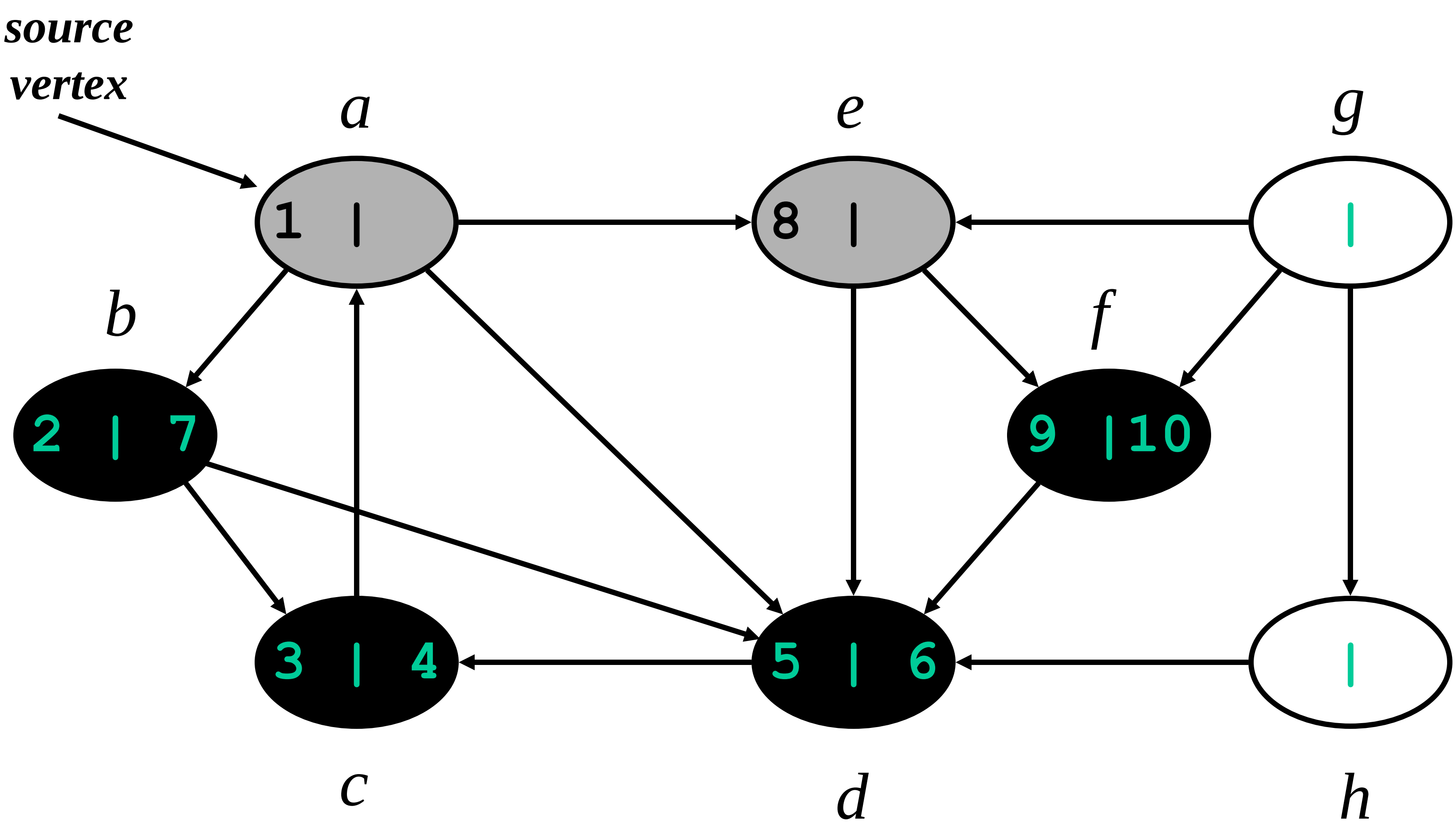
Ví dụ: DFS



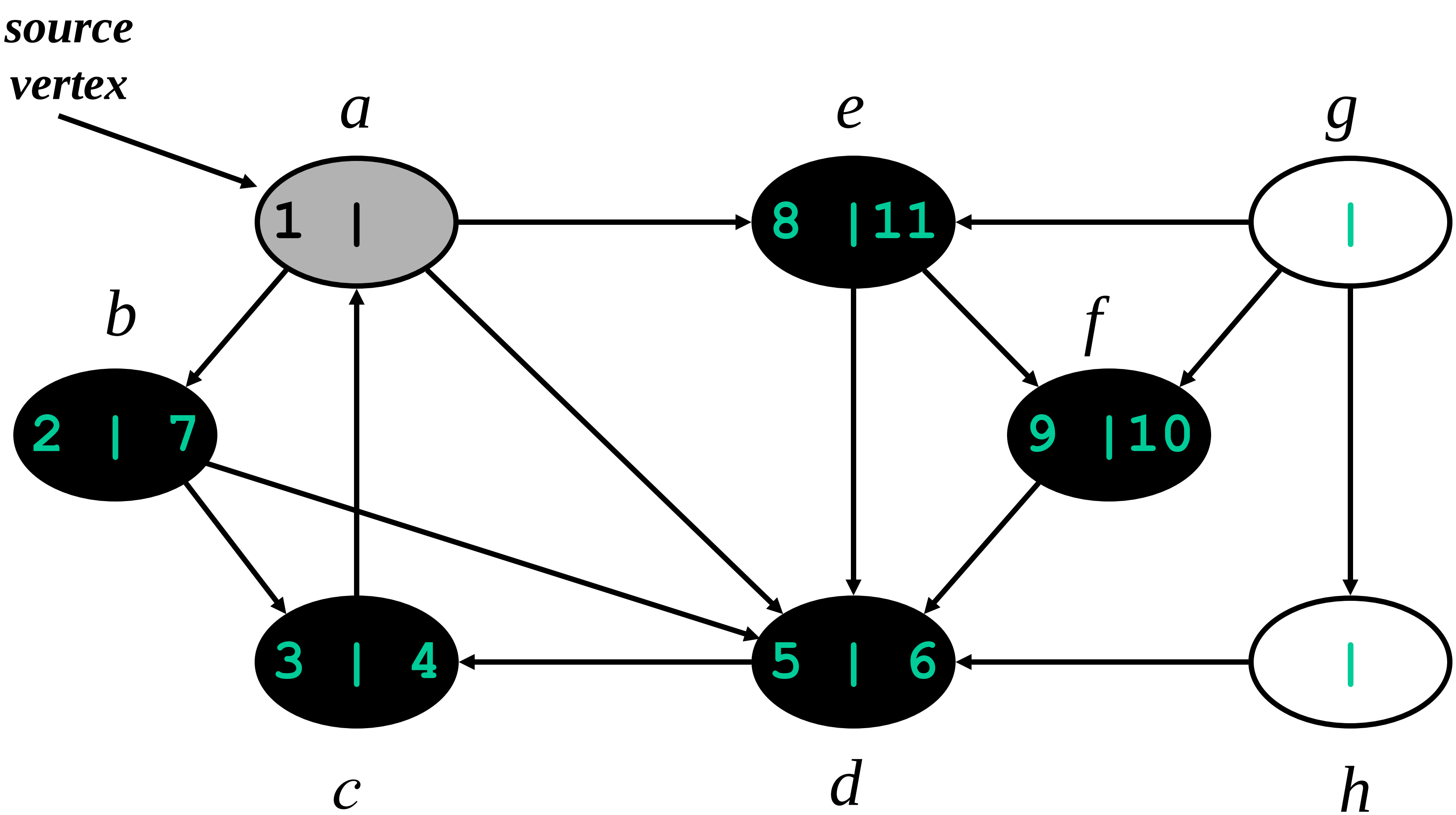
Ví dụ: DFS



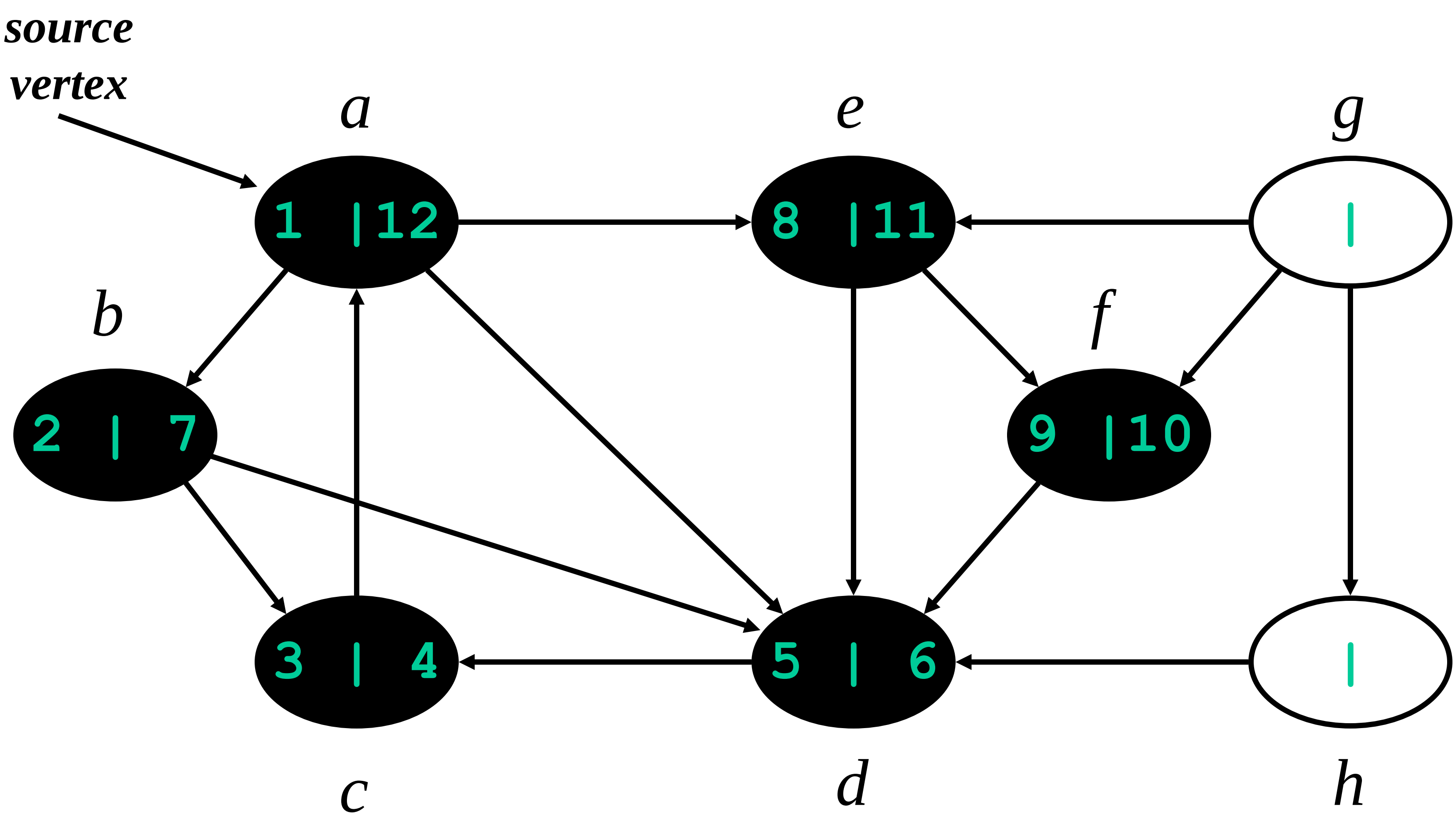
Ví dụ: DFS



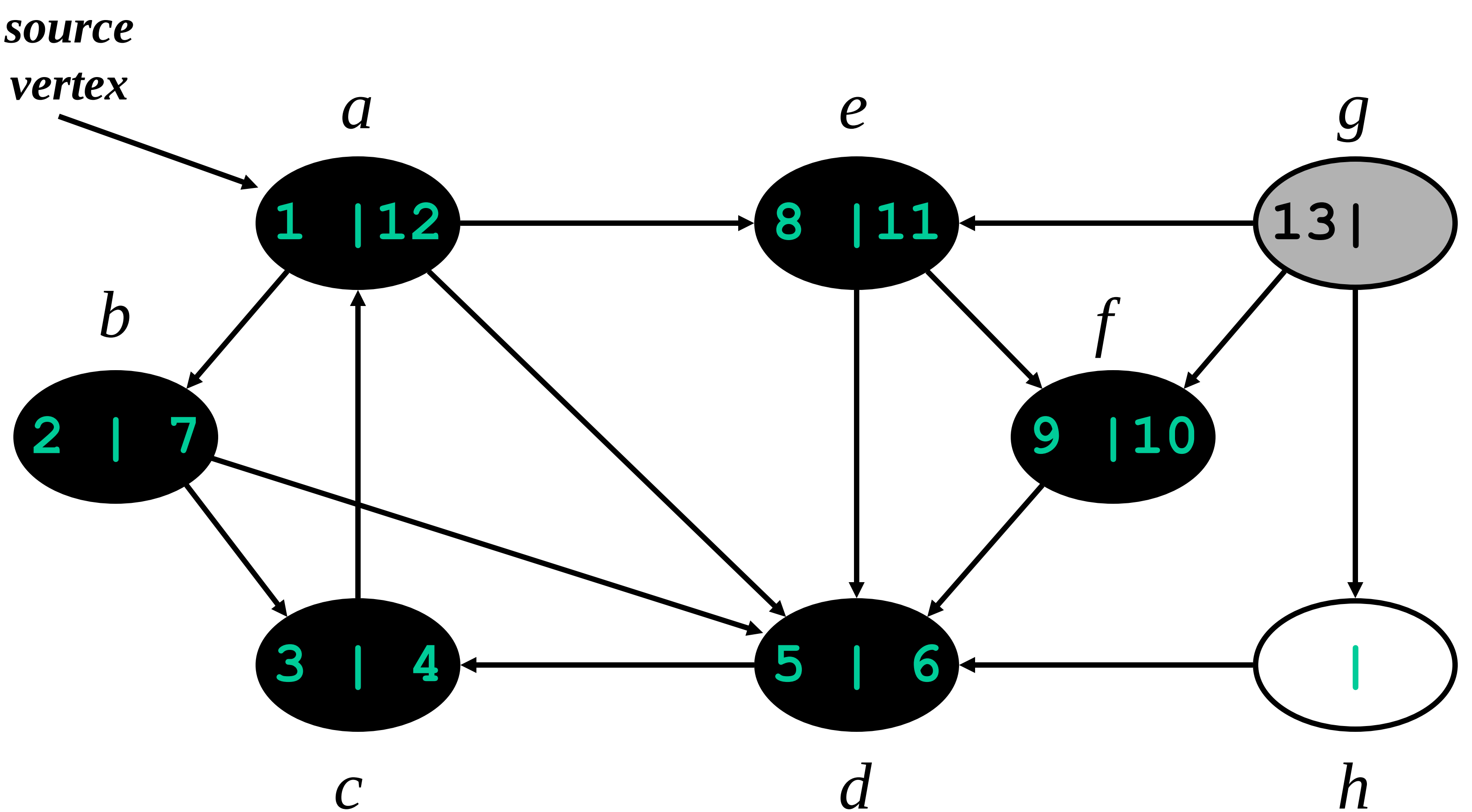
Ví dụ: DFS



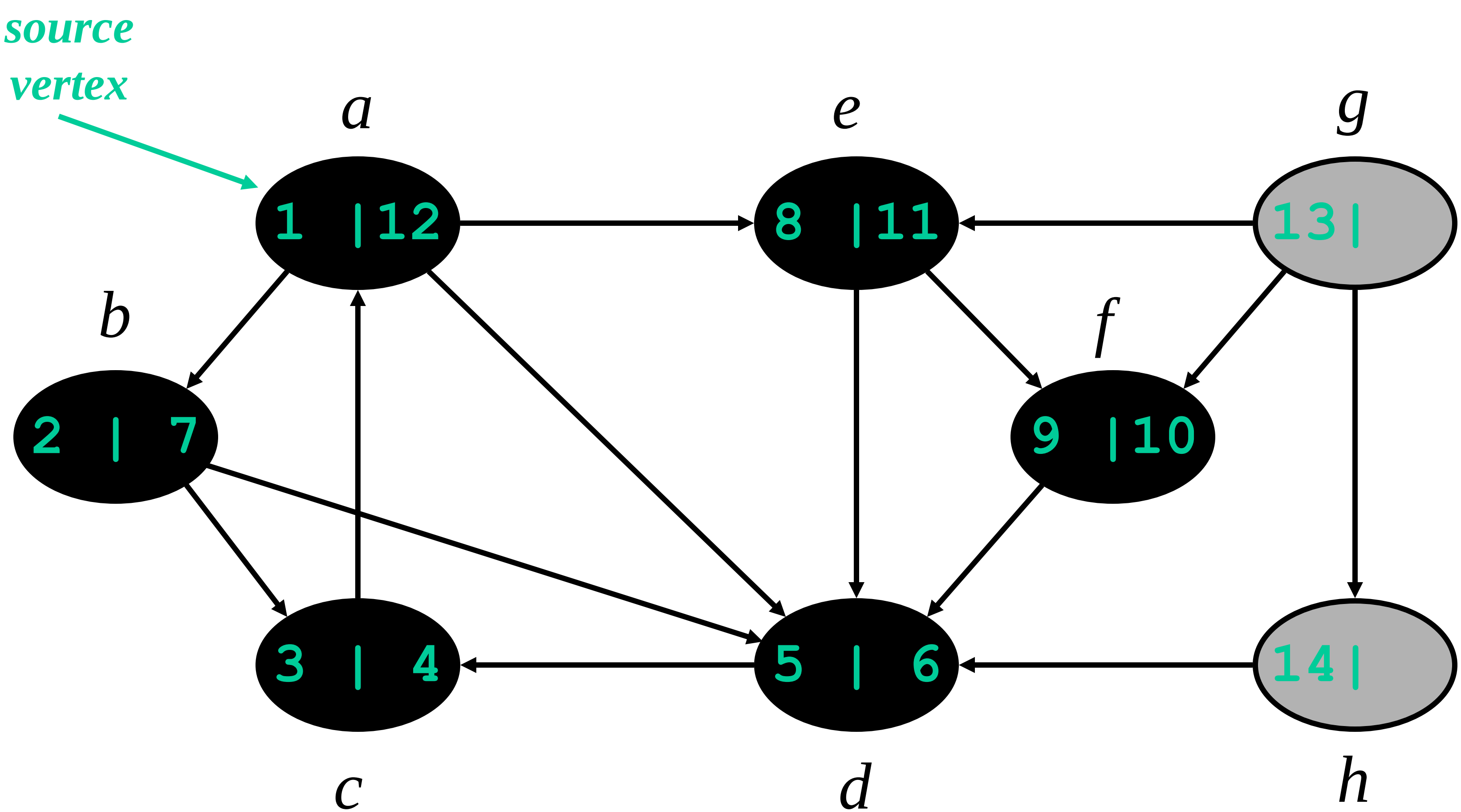
Ví dụ: DFS



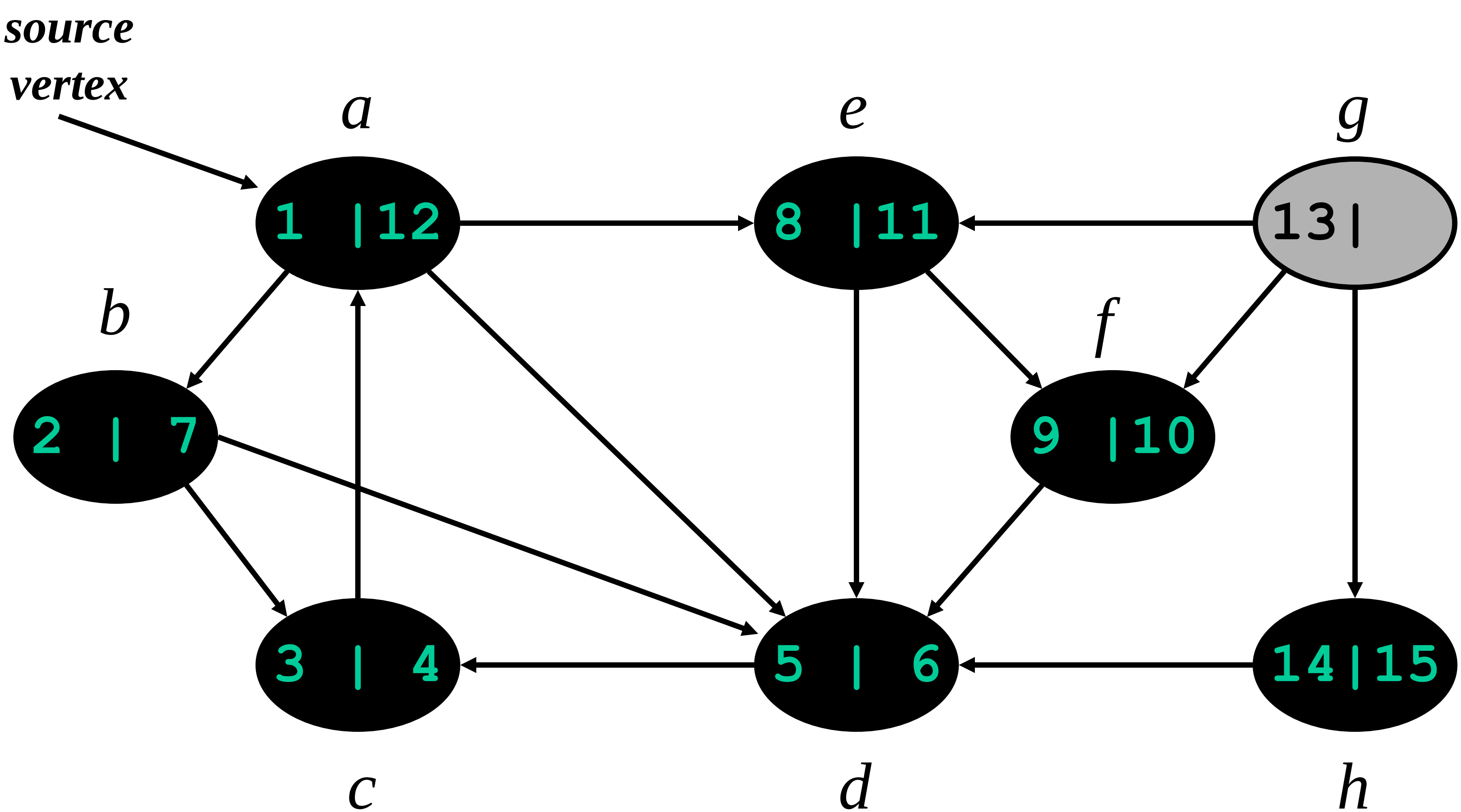
Ví dụ: DFS



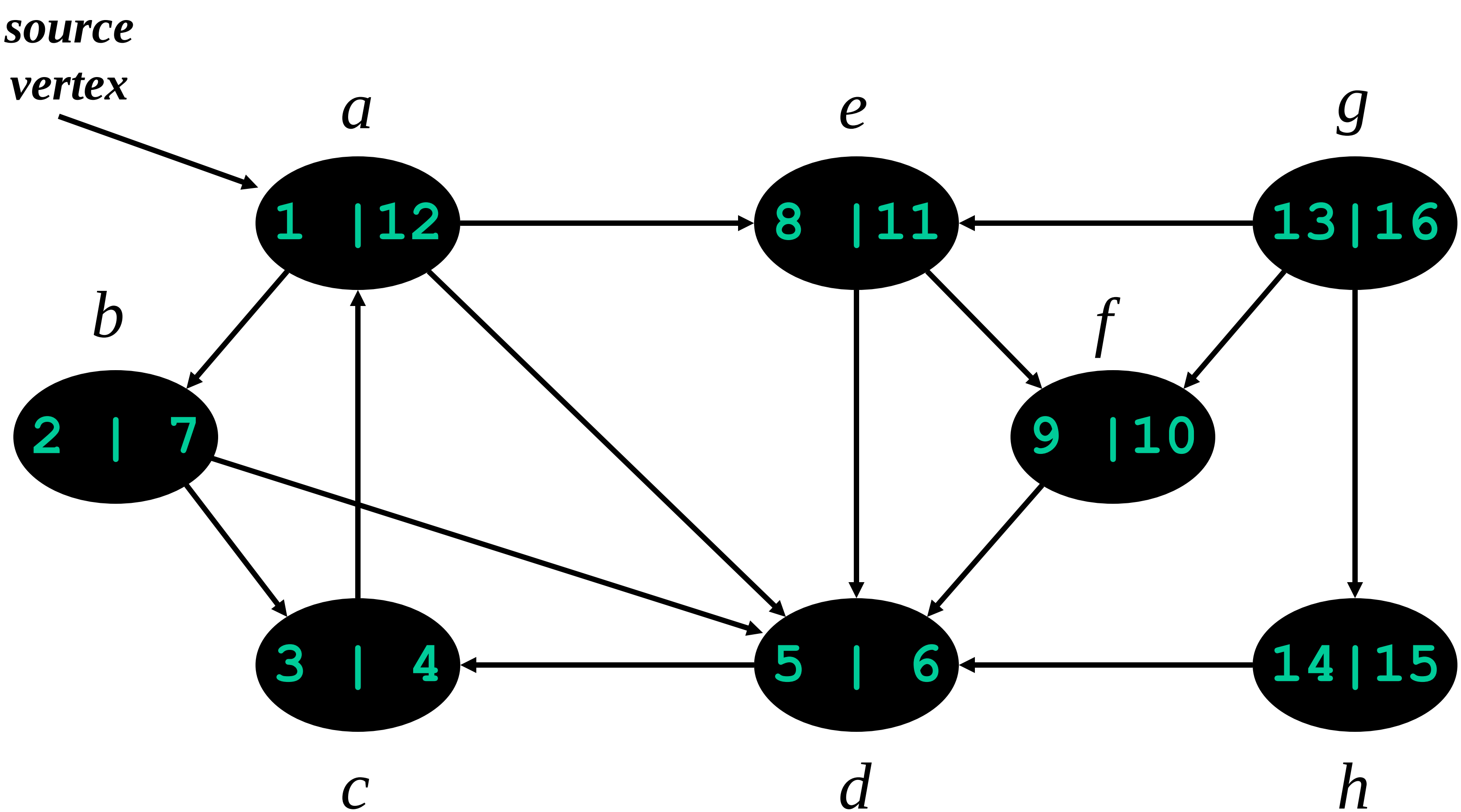
Ví dụ: DFS



Ví dụ: DFS



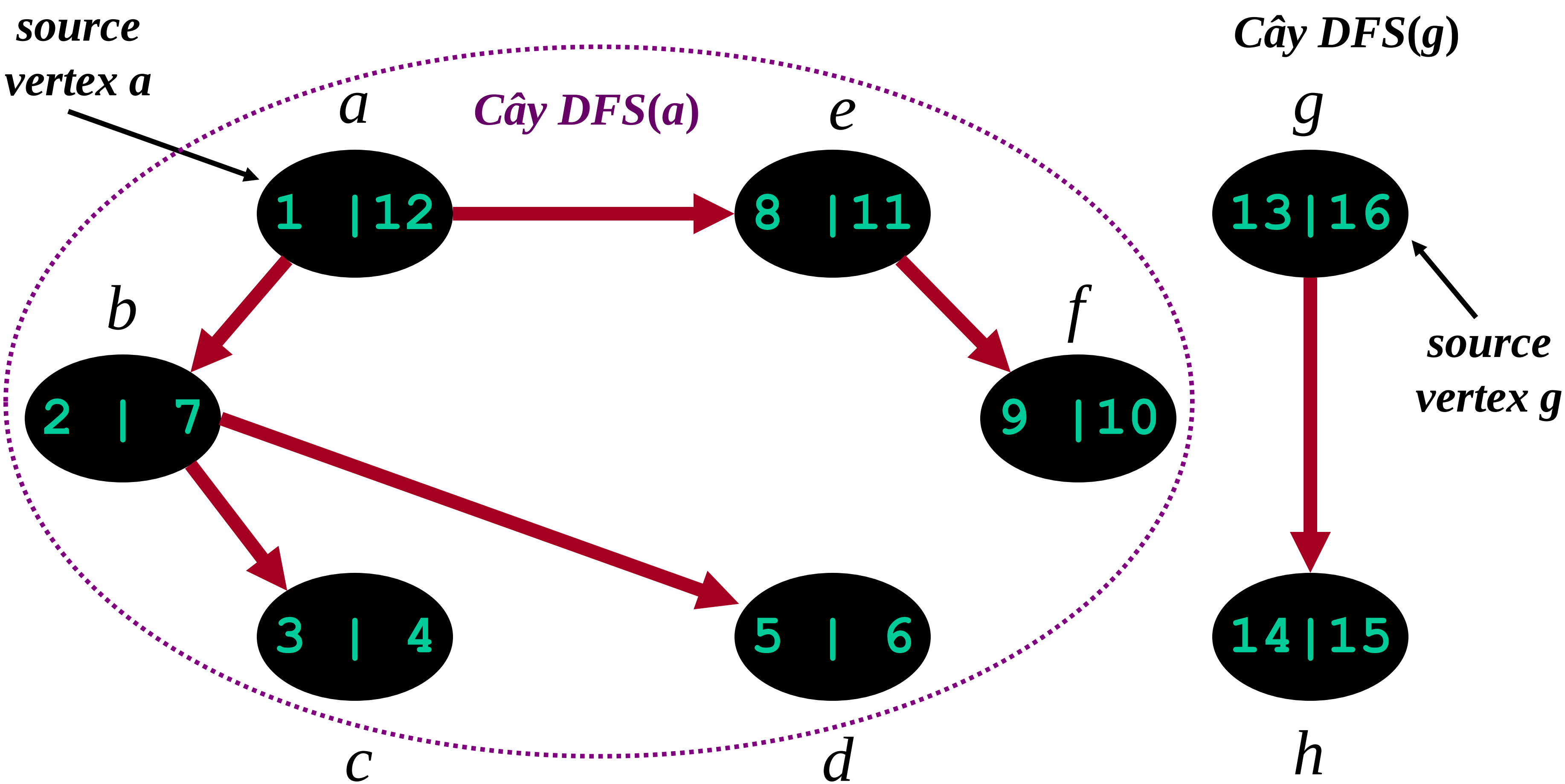
Ví dụ: DFS



DFS: Các loại cạnh

- DFS(G) sinh ra một cách phân loại các cạnh của đồ thị đã cho:
  - *Cạnh của cây* (*Tree edge*): là cạnh mà theo đó từ một đỉnh ta đến thăm một đỉnh mới (cạnh đi vào đỉnh trắng)
    - Các cạnh này tạo thành một rừng gọi là ***rừng tìm kiếm DFS***.
    - Các đỉnh được thăm khi thực hiện DFS(v) và các cạnh của cây tạo thành cây được gọi là ***cây DFS(v)***

Ví dụ: Rừng DFS

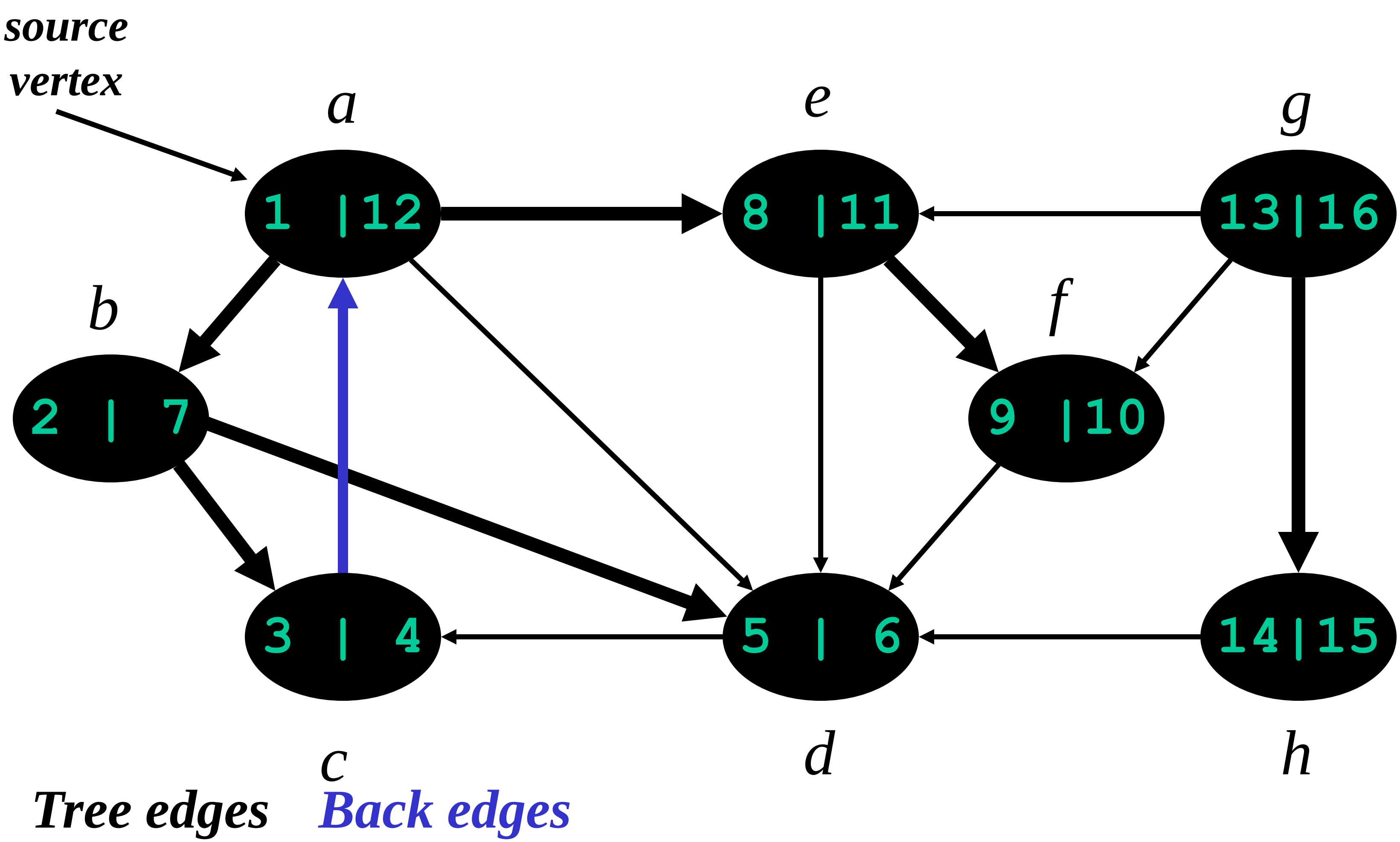


Tree edges

DFS: Cạnh ngược

- DFS tạo ra một cách phân loại các cạnh của đồ thị đã cho:
  – *Cạnh của cây* (*Tree edge*): là cạnh mà theo đó từ một đỉnh ta đến thăm một đỉnh mới (cạnh đi vào đỉnh trắng)
  – *Cạnh ngược* (*Back edge*): đi từ con cháu (descendent) đến tổ tiên (ancestor)
    - Đi vào đỉnh xám (đi từ đỉnh xám đến đỉnh xám)

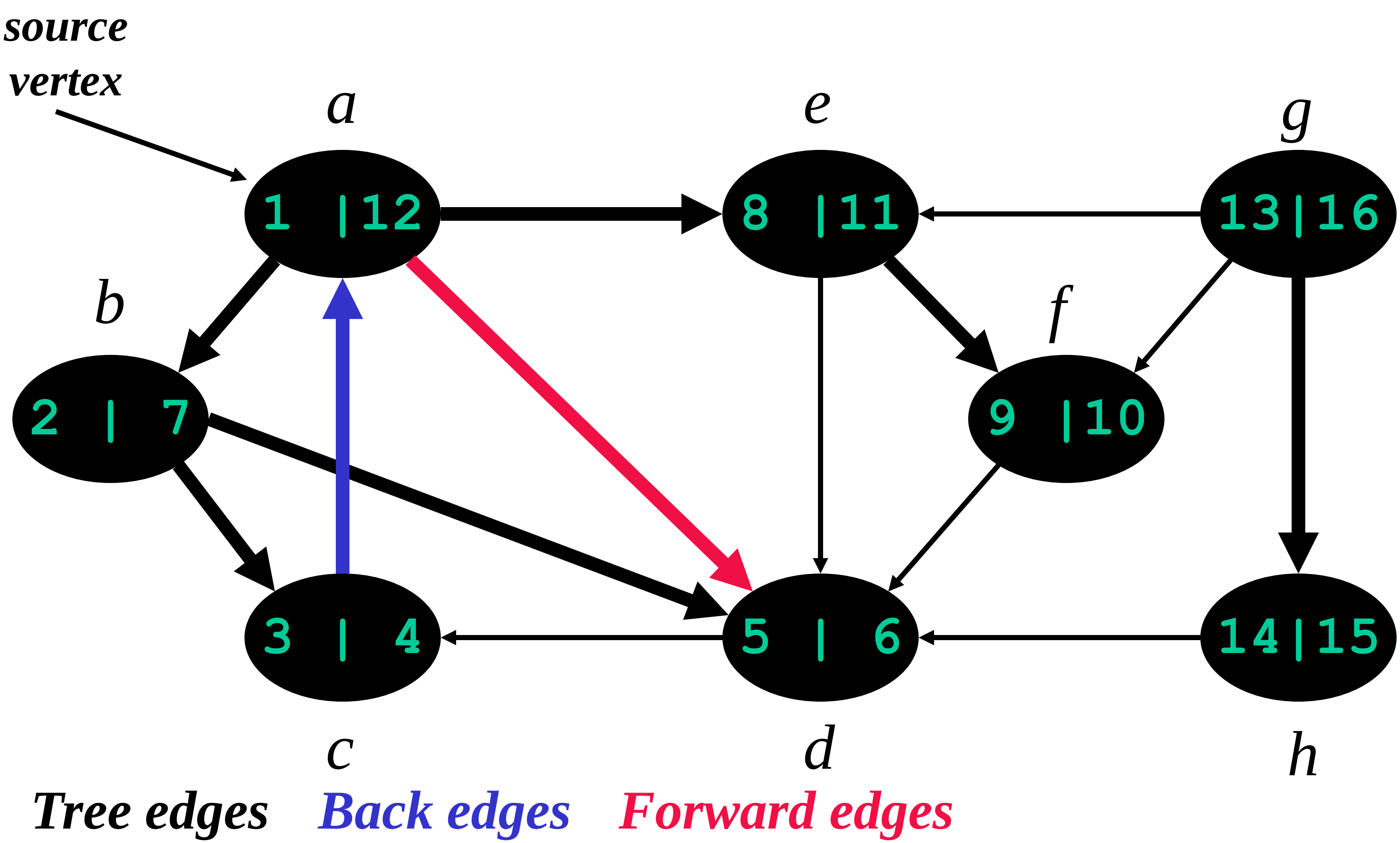
Ví dụ: DFS Cạnh ngược



DFS: Cạnh tới

- DFS tạo ra một cách phân loại các cạnh của đồ thị đã cho:
  - *Cạnh của cây* (*Tree edge*): là cạnh mà theo đó từ một đỉnh ta đến thăm một đỉnh mới (cạnh đi vào đỉnh trắng)
  - *Cạnh ngược* (*Back edge*): đi từ con cháu (descendent) đến tổ tiên (ancestor)
  - *Cạnh tới* (*Forward edge*): đi từ tổ tiên đến con cháu
    - Không là cạnh của cây
    - Đi từ đỉnh xám đến đỉnh đen

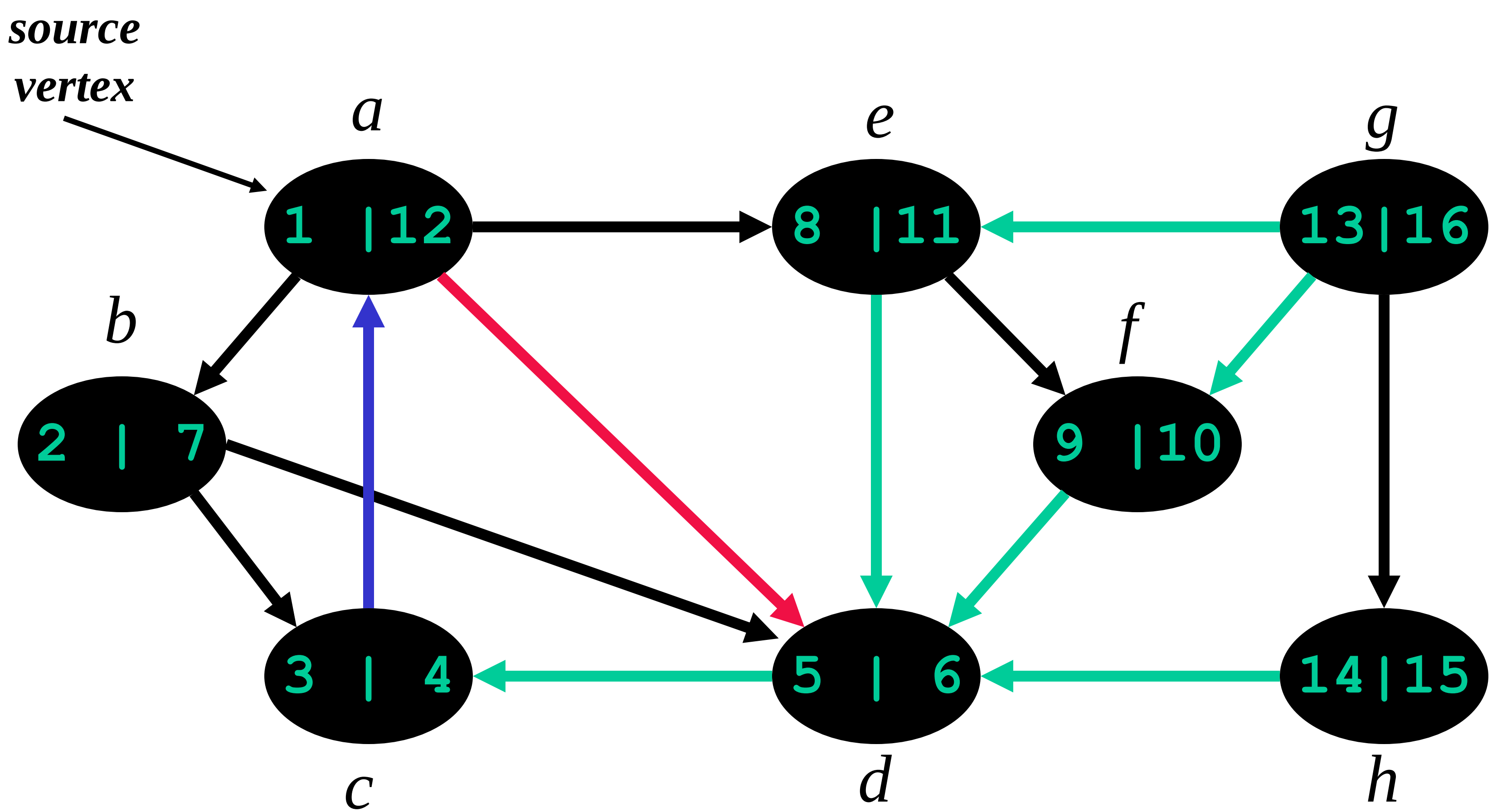
Ví dụ: DFS Cạnh tới



DFS: Cạnh vòng

- DFS tạo ra một cách phân loại các cạnh của đồ thị đã cho:
  – *Cạnh của cây* (*Tree edge*): là cạnh mà theo đó từ một đỉnh ta đến thăm một đỉnh mới (cạnh đi vào đỉnh trắng)
  – *Cạnh ngược* (*Back edge*): đi từ con cháu (descendent) đến tổ tiên (ancestor)
  – *Cạnh tới* (*Forward edge*): đi từ tổ tiên đến con cháu
  – *Cạnh vòng* (*Cross edge*): cạnh nối hai đỉnh không có quan hệ họ hàng
    - Không là cạnh của cây, và giống như cạnh vòng cũng
    - Đi từ đỉnh xám đến đỉnh đen

Ví dụ: DFS Cạnh vòng



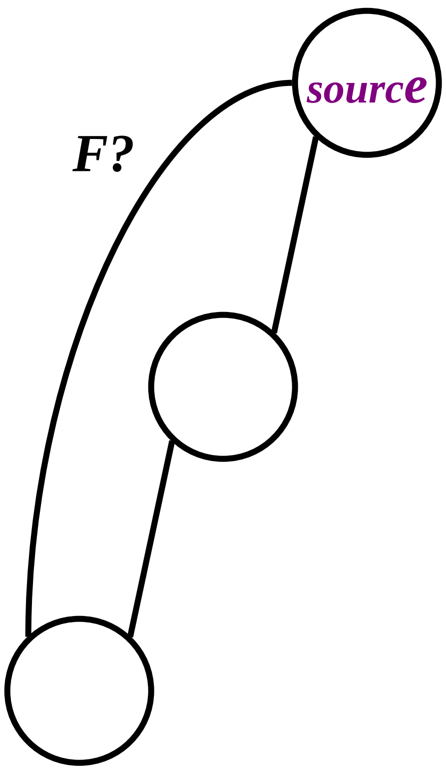
Tree edges ***Back edges*** ***Forward edges*** ***Cross edges***

DFS: Các loại cạnh

- DFS tạo ra một cách phân loại các cạnh của đồ thị đã cho:
  - *Tree edge*: cạnh theo đó từ một đỉnh đến thăm đỉnh mới (trắng)
  - *Back edge*: đi từ con cháu đến tổ tiên
  - *Forward edge*: đi từ tổ tiên đến con cháu
  - *Cross edge*: giữa hai đỉnh không có họ hàng
- **Chú ý:** Cạnh của cây & cạnh ngược là quan trọng; nhiều thuật toán không đòi hỏi phân biệt cạnh tới và cạnh vòng

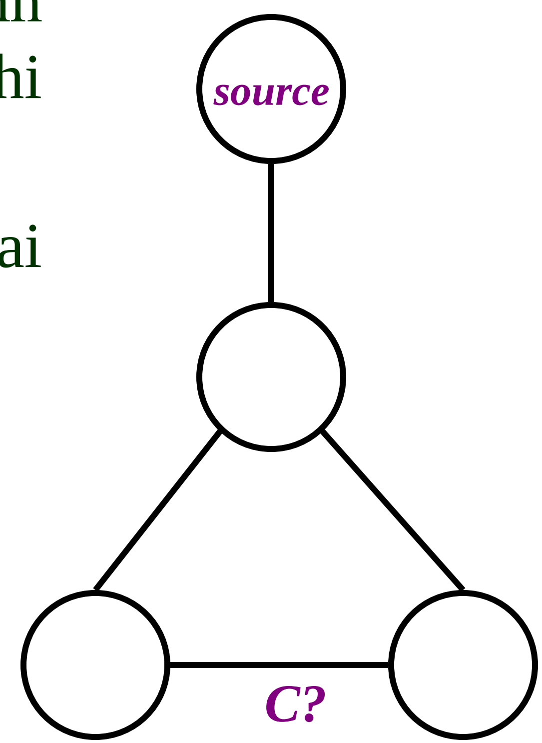
DFS: Các loại cạnh

- ***Định lý:*** *Nếu G là đồ thị vô hướng, thì DFS chỉ sản sinh ra cạnh của cây và cạnh ngược.*
- Chứng minh bằng phản chứng:
  - Giả sử có cạnh tới (forward edge)
  - Nhưng khi đó *F* phải là cạnh ngược (back edge)?!



DFS: Các loại cạnh

- Giả sử có cạnh vòng (cross edge)
  - Khi đó *C* không thể là cạnh vòng bởi vì:
  - Nó phải được khảo sát từ một trong hai đỉnh đầu mút và trở thành cạnh của cây trước khi đỉnh kia được khảo sát
  - Do đó bức tranh bên là không đúng…cả hai cạnh bên không thể là cạnh của cây



DFS: Phân biệt các loại cạnh

- Dễ dàng phân biệt các loại cạnh nhê ph©n tÝch mµu cña c¸c ®Ønh vµ/hoÆc xÐt c¸c gi¸ trÞ cña c¸c mèc thêi gian *d* vµ *f*.
  – Khi ta duyệt cạnh *e*=(*u*, *v*) từ đỉnh *u*, căn cứ vào màu của *v* ta có thể biết cạnh này thuộc loại cạnh nào:
    1. WHITE cho biết *e* là cạnh của cây
    2. GRAY cho biết *e* là cạnh ngược
    3. BLACK cho biết *e* là cạnh tới hoặc vòng

Phân tích DFS

(* Main Program*)

```
for u ∈ V do
    color[u] ← white
    π[u] ← NIL
time ← 0
for u ∈ V do
    if color[u] = white
        then DFS-Visit(u)
```

Các biến time, color, *π là toàn cục* .

<u>DFS-Visit(*u*)</u>

```
color[u] ← GRAY  (* Thăm đỉnh u *)
time ← time + 1
d[u] ← time
for each v ∈ Adj[u] do
    if color[v] = WHITE
        then π[v] ← u
            DFS-Visit(v)
color[u] ← BLACK (* Đỉnh u đã duyệt xong *)
f[u] ← time ← time + 1
```

Phân tích DFS

- Vòng lặp trên các dòng 1-2 và 5-7 đòi hỏi thời gian $\Theta(|V|)$, chưa tính thời gian thực hiện lệnh DFS(v).
- DFS(v) thực hiện đối với mỗi đỉnh trắng $v \in V$ và ngay sau khi được thăm nó được tô màu xám. Các dòng 3-6 của DFS(v) sẽ thực hiện |Adj[v]| lần. Vậy thời gian tổng cộng của DFS(v) là $\sum_{v \in V} |\mathrm{Adj}[v]| = \Theta(|E|)$
- Do đó thời gian của DFS là $\Theta(|V|+|E|)$.
- Thuật toán trên đồ thị có đánh giá thời gian như trên gọi là thuật toán thời gian tuyến tính

CÁC ỨNG DỤNG CỦA DFS

- Tính liên thông của đồ thị
- Tìm đường đi từ *s* đến *t*
- Phát hiện chu trình
- Kiểm tra tính liên thông mạnh
- Định hướng đồ thị

Bài toán về tính liên thông

- **Bài toán:** Cho đồ thị vô hướng $G = (V,E)$. Hỏi đồ thị gồm bao nhiêu thành phần liên thông, và từng thành phần liên thông gồm các đỉnh nào?
- Giải: Sử dụng DFS (BFS) :
  – Mỗi lần gọi đến DFS (BFS) ở trong chương trình chính sẽ sinh ra một thành phần liên thông

DFS giải bài toán liên thông

(* Main Program*)

1. **for** $u \in V$ **do**
2. $id[u] \leftarrow 0$
3. $cnt \leftarrow 0$ (* cnt – số lượng tplt *)
4. **for** $u \in V$ **do**
5. **if** $id[u] = 0$
6. **then** $cnt \leftarrow cnt + 1$
7. DFS-Visit(u)

DFS-Visit(u)

1. $id[u] \leftarrow cnt$
2. **for** each $v \in Adj[u]$ **do**
3. **if** $id[v] = 0$
4. **then** DFS-Visit(v)

Tìm đường đi

- Bài toán tìm đường đi
  - Input: Đồ thị $G = (V,E)$ xác định bởi danh sách kề và hai đỉnh *s*, *t*.
  - Đầu ra: Đường đi từ đỉnh *s* đến đỉnh *t*, hoặc khẳng định không tồn tại đường đi từ *s* đến *t*.
- Thuật toán: Thực hiện DFS(s) (hoặc BFS(s)).
  - Nếu $\pi[t]$ = NIL thì không có đường đi, trái lại ta có đường đi

$$t \leftarrow \pi[t] \leftarrow \pi[\pi[t]] \leftarrow \ldots \leftarrow s$$

DFS giải bài toán đường đi

(* Main Program*)

1. **for** $u \in V$ **do**
2. $\quad$ *color*[u] $\leftarrow$ white
3. $\quad$ $\pi[u] \leftarrow$ NIL
4. DFS-Visit(s)

DFS-Visit(u)

1. $\quad$ *color*[u] $\leftarrow$ GRAY (* Thăm đỉnh u *)

2. $\quad$ **for** each $v \in Adj[u]$ **do**

3. $\quad\quad$ **if** *color*[v] = WHITE

4. $\quad\quad\quad$ **then** $\pi[v] \leftarrow u$

5. $\quad\quad\quad\quad$ DFS-Visit(v)

DFS và Chu trình

- **Bài toán:** Cho đồ thị *G*=(*V*,*E*). Hỏi *G* có chứa chu trình hay không?
- ***Định lý:*** *Đồ thị G là không chứa chu trình khi và chỉ khi trong quá trình thực hiện DFS ta không phát hiện ra cạnh ngược.*
- Chứng minh:
  - Nếu *G* không chứa chu trình thì rõ ràng không có cạnh ngược (bởi vì sự tồn tại cạnh ngược dẫn đến phát hiện chu trình)
  - Nếu không có cạnh ngược thì *G* là không chứa chu trình (acyclic). Thực vậy
    - Không có cạnh ngược tức là chỉ có cạnh của cây
    - Nếu chỉ có cạnh của cây thì *G* chỉ là cây hoặc rừng
    - Vậy *G* không chứa chu trinh
- Như vậy DFS có thể áp dụng để giải bài toán đặt ra.

DFS và chu trình

- ***Cần phải điều chỉnh như thế nào để phát hiện chu trình?***

(* Main Program*)

1. **for** $u \in V$ **do**
2. *color*[u] ← white
3. $\pi[u]$ ← NIL
4. *time* ← 0
5. **for** $u \in V$ **do**
6. **if** *color*[u] = white
7. **then** DFS-Visit(u)

DFS(u)

1. *color*[u] ← GRAY (* Thăm đỉnh u *)
2. *time* ← *time* + 1
3. $d[u]$ ← *time*
4. **for** each $v \in Adj[u]$ **do**
5. **if** *color*[v] = WHITE
6. **then** $\pi[v] \leftarrow u$
7. DFS-Visit(v)
8. *color*[u] ← BLACK (* Đỉnh u đã duyệt xong *)
9. $f[u] \leftarrow time \leftarrow time + 1$

DFS và chu trình

- **Câu hỏi:** Thời gian tính là bao nhiêu?
- **Trả lời:** Chính là thời gian thực hiện *DFS*: $O(|V|+|E|)$.
- **Câu hỏi:** Nếu *G* là đồ thị vô hướng thì có thể đánh giá thời gian tính sát hơn nữa được không?
- **Trả lời:** Thuật toán có thời gian tính $O(|V|)$, bởi vì:
  – Trong một rừng (đồ thị không chứa chu trình) $|E| \leq |V| - 1$
  – Vì vậy nếu đồ thị có $|V|$ cạnh thì chắc chắn nó chứa chu trình, và thuật toán kết thúc.

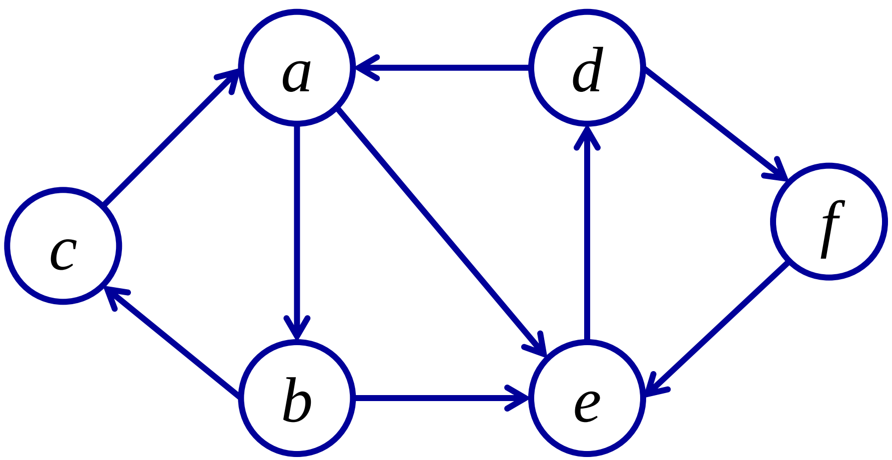
Kiểm tra tính liên thông mạnh

- **Bài toán:** Hỏi đồ thị có hướng *G* có là liên thông mạnh?
- ***Mệnh đề:*** *Đồ thị có hướng G=(V,E) là liên thông mạnh khi và chỉ khi luôn tìm được đường đi từ một đỉnh v đến tất cả các đỉnh còn lại và luôn tìm được đường đi từ tất cả các đỉnh thuộc V \ {v} đến v.*
- **Chứng minh:** Hiển nhiên

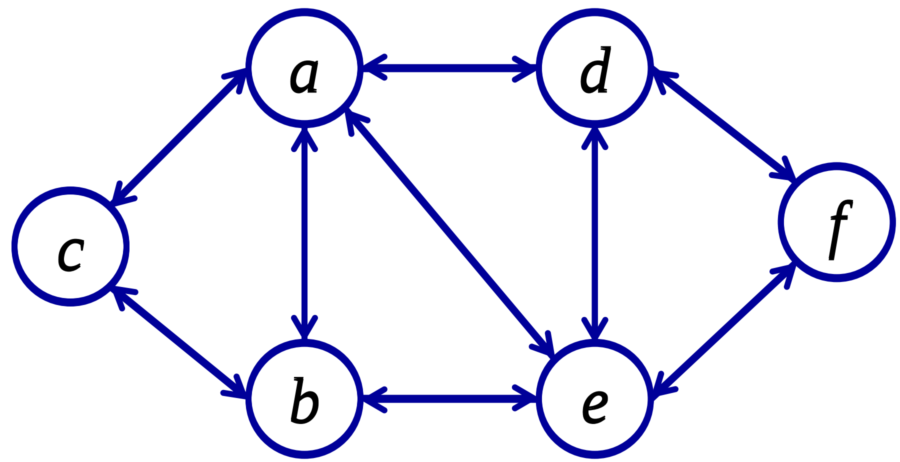
Thuật toán kiểm tra tính liên thông mạnh

- Thuật toán.
  - Chän $v \in V$ lµ mét ®Ønh tuú ý
  - Thùc hiÖn DFS(v) trªn G. NÕu tån t¹i ®Ønh u kh«ng ®-îc th¨m th× G kh«ng liªn th«ng m¹nh vµ thuËt to¸n kÕt thóc. Tr¸i l¹i thùc hiÖn tiÕp
  - Thùc hiÖn DFS(v) trªn $G^T = (V, E^T)$, víi E^T thu ®-îc tõ E bëi viÖc ®¶o ng-îc h-íng c¸c cung. NÕu tån t¹i ®Ønh u kh«ng ®-îc th¨m th× G kh«ng liªn th«ng m¹nh, nÕu tr¸i l¹i G lµ liªn th«ng m¹nh.
- Thời gian tính: $O(|V|+|E|)$

Thuật toán kiểm tra tính liên thông mạnh



Đồ thị G

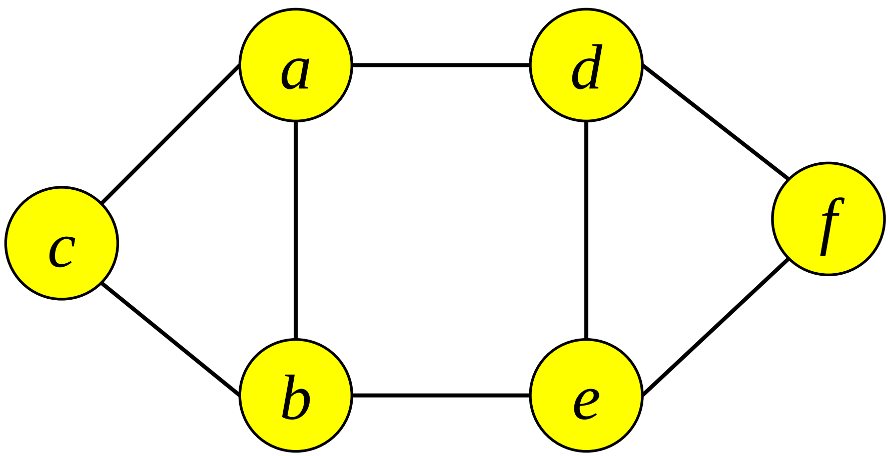


Đồ thị G^T

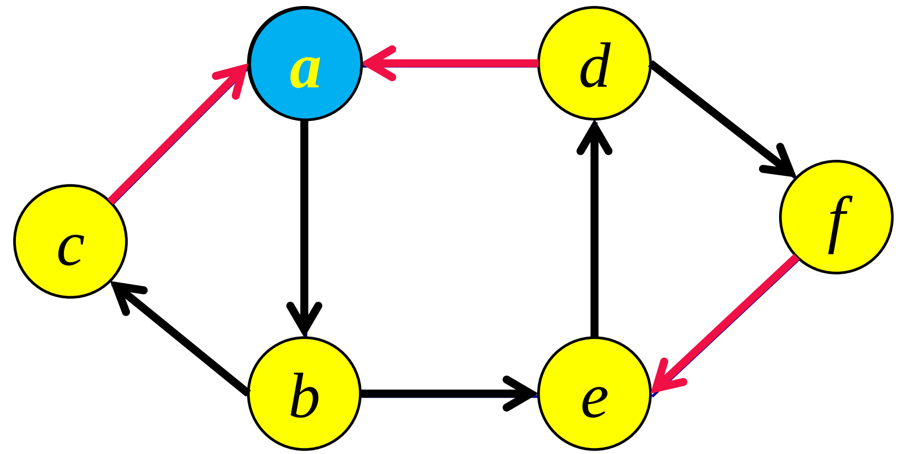
Định hướng đồ thị

- **Bµi to¸n:** Cho ®å thÞ v« h-íng liªn th«ng *G*= (*V*, *E*). H·y t×m c¸ch ®Þnh h-íng c¸c c¹nh cña nã ®Ó thu ®-îc ®å thÞ cã h-íng liªn th«ng m¹nh hoÆc tr¶ lêi *G* lµ kh«ng ®Þnh h-íng ®-îc.
- **ThuËt to¸n ®Þnh h-íng δ:** Trong qu¸ tr×nh thùc hiÖn DFS(G) ®Þnh h-íng c¸c c¹nh cña c©y DFS theo chiÒu tõ tæ tiªn ®Õn con ch¸u, c¸c c¹nh ng-îc theo h-íng tõ con ch¸u ®Õn tæ tiªn. Ký hiÖu ®å thÞ thu ®-îc lµ *G*(δ**)**
- **Bæ ®Ò. *G* lµ ®Þnh h-íng ®-îc khi vµ chØ khi *G*(δ) lµ liªn th«ng m¹nh.**

Ví dụ: Định hướng đồ thị



Đồ thị *G*



Đồ thị *G*(δ)

Questions?