KHÁI NIỆM CƠ BẢN TRONG LÝ THUYẾT ĐỒ THỊ

Đặng Nguyễn Đức Tiến
Đặng Trần Minh Hậu
Nguyễn Ngọc Thảo

Nội dung bài giảng

- Các khái niệm cơ bản về đồ thị
- Một số mô hình đồ thị phổ biến
- Đơn đồ thị đặc biệt
- Biểu diễn đồ thị
- Đồ thị con – Đồ thị bộ phận – Đồ thị đẳng cấu
- Các chiến lược duyệt đồ thị
- Tính liên thông của đồ thị

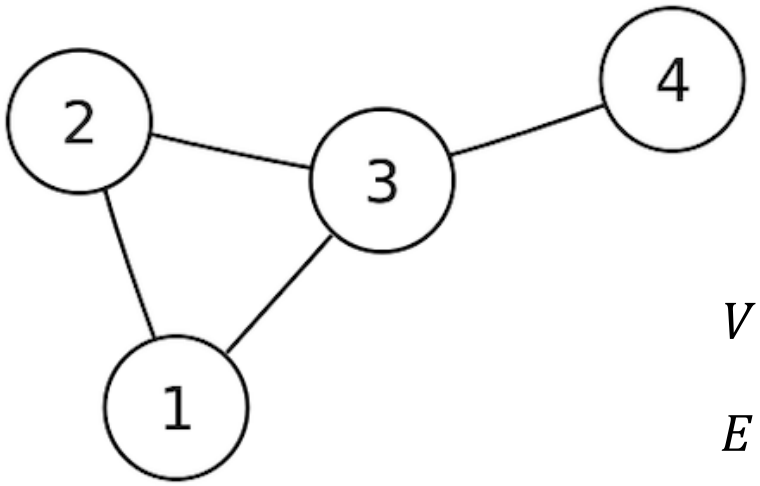
Các khái niệm về đồ thị

Định nghĩa đồ thị

- Đồ thị (graph) là một cấu trúc rời rạc gồm các đỉnh và các cạnh nối những đỉnh đó.
  - **Quy ước:** Ta chỉ làm việc với đồ thị hữu hạn, tức là có tập đỉnh hữu hạn và tập cạnh hữu hạn.
- Ký hiệu biểu diễn đồ thị: $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, trong đó $\boldsymbol{V}$ là tập đỉnh (vertex) và $\boldsymbol{E}$ là tập cạnh (edge).
- Một cạnh nối hai đỉnh x và y được biểu diễn thành một cặp có thứ tự (ordered pair) (x, y) hoặc không có thứ tự $\{x, y\}$.

Tính chất hướng của đồ thị

- Đồ thị vô hướng (undirected graph) $\boldsymbol{G = (V, E)}$ gồm tập đỉnh $\boldsymbol{V}$ và tập cạnh $\boldsymbol{E}$ chứa các cặp đỉnh **không** có thứ tự từ $\boldsymbol{V}$.
- $\boldsymbol{E \subseteq \{\{x, y\} \mid x, y \in V,\ x \neq y\}}$
  - Một ngoại lệ không thỏa $x \neq y$ là cạnh khuyên (loop).

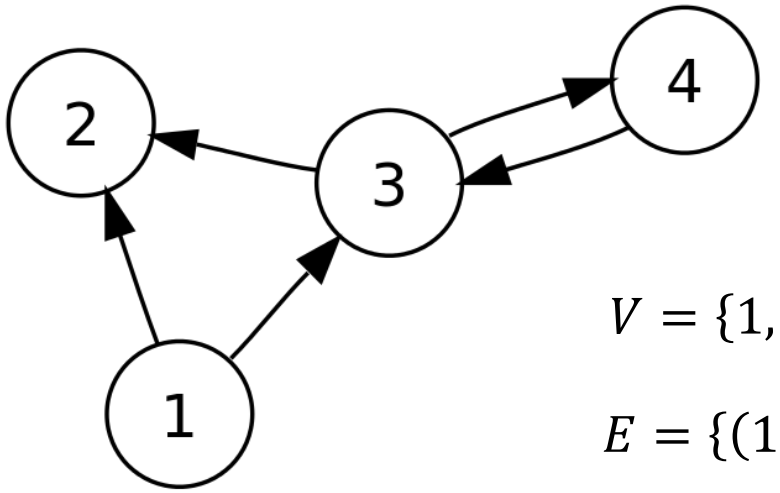


$$V = \{1, 2, 3, 4\}$$

$$E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{3, 4\}\}$$

Tính chất hướng của đồ thị

- Đồ thị có hướng (directed graph, digraph) $\boldsymbol{G = (V, E)}$ gồm tập đỉnh $\boldsymbol{V}$ và tập cạnh $\boldsymbol{E}$ chứa các cặp đỉnh **có** thứ tự từ $\boldsymbol{V}$.
- $\boldsymbol{E \subseteq \{(x, y) \mid (x, y) \in V^2,\ x \neq y\}}$
  - Một ngoại lệ không thỏa $x \neq y$ là cạnh khuyên (loop).



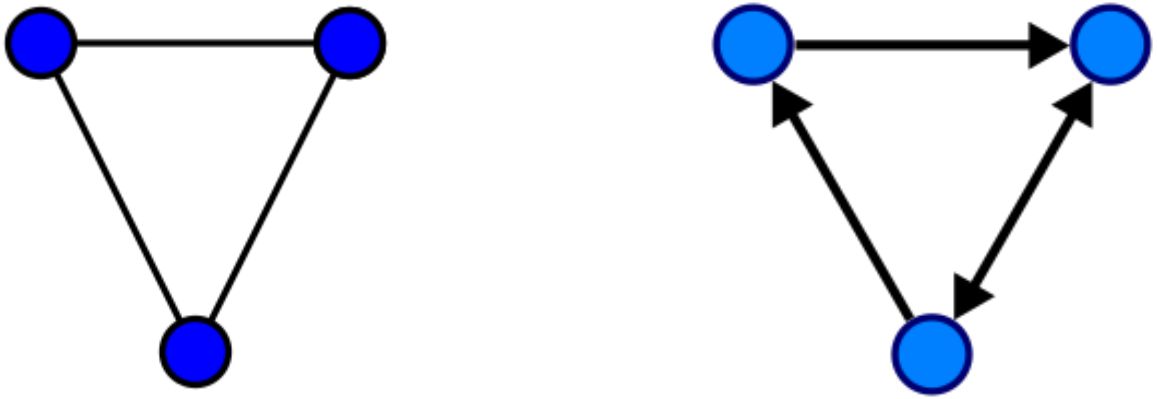
$$V = \{1, 2, 3, 4\}$$

$$E = \{(1, 2), (1, 3), (3, 2), (3, 4), (4, 3)\}$$

Đơn đồ thị

- Đơn đồ thị (simple graph) $\boldsymbol{G = (V, E)}$ gồm tập đỉnh $\boldsymbol{V}$ và tập cạnh $\boldsymbol{E}$ chứa các cặp đỉnh phân biệt có/không có thứ tự từ $\boldsymbol{V}$
- $\boldsymbol{E \subseteq \{\{x, y\} \mid x, y \in V,\ x \neq y\}}$ (vô hướng) hoặc

  $\boldsymbol{E \subseteq \{(x, y) \mid (x, y) \in V^2,\ x \neq y\}}$ (có hướng).



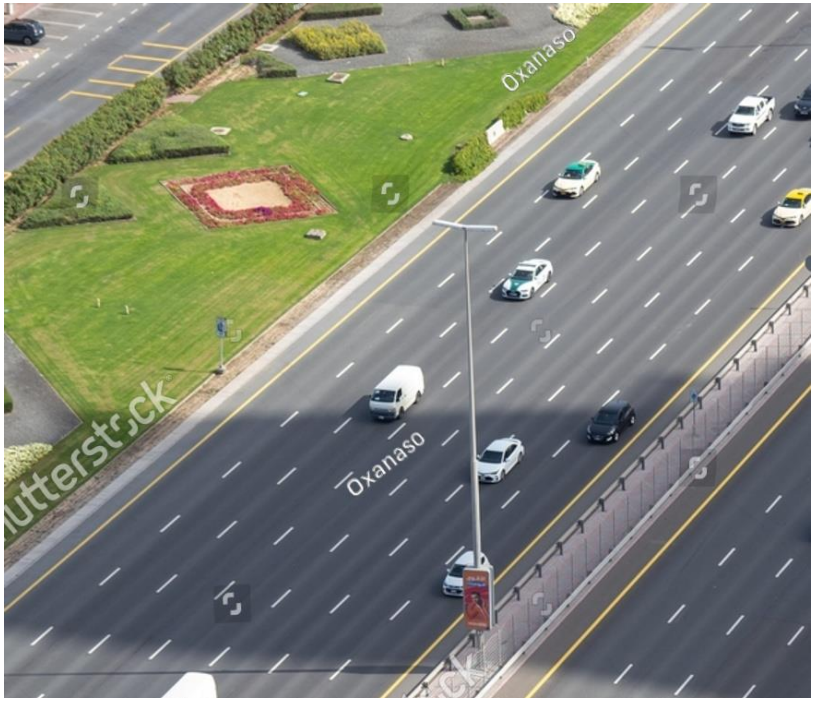
Hình trái: Đơn đồ thị vô hướng (undirected simple graph)
Hình phải: Đơn đồ thị có hướng (directed simple graph)

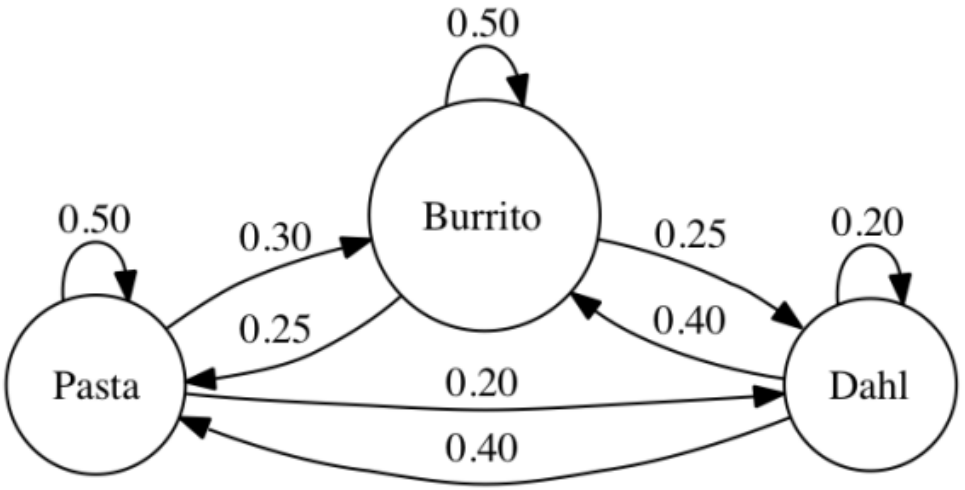
Cạnh khuyên và Cạnh bội

- Ta định nghĩa thêm hàm ánh xạ $\boldsymbol{\phi}: \boldsymbol{E} \rightarrow \{\{\boldsymbol{u}, \boldsymbol{v}\} \mid \boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{V}\}$.
- Cạnh khuyên (loop) là cạnh nối một đỉnh với chính đỉnh đó.
  - Gọi e là cạnh khuyên tại đỉnh u. Ta có $\phi(e) = \{u, u\}$.
- Các cạnh $e_1, e_2 \in E$ là các cạnh bội (multiple edges) nếu $\phi(e_1) = \phi(e_2)$.
- Ta có thể định nghĩa tương tự cho cạnh khuyên có hướng và các cạnh bội có hướng.

Cạnh khuyên và Cạnh bội: Ứng dụng

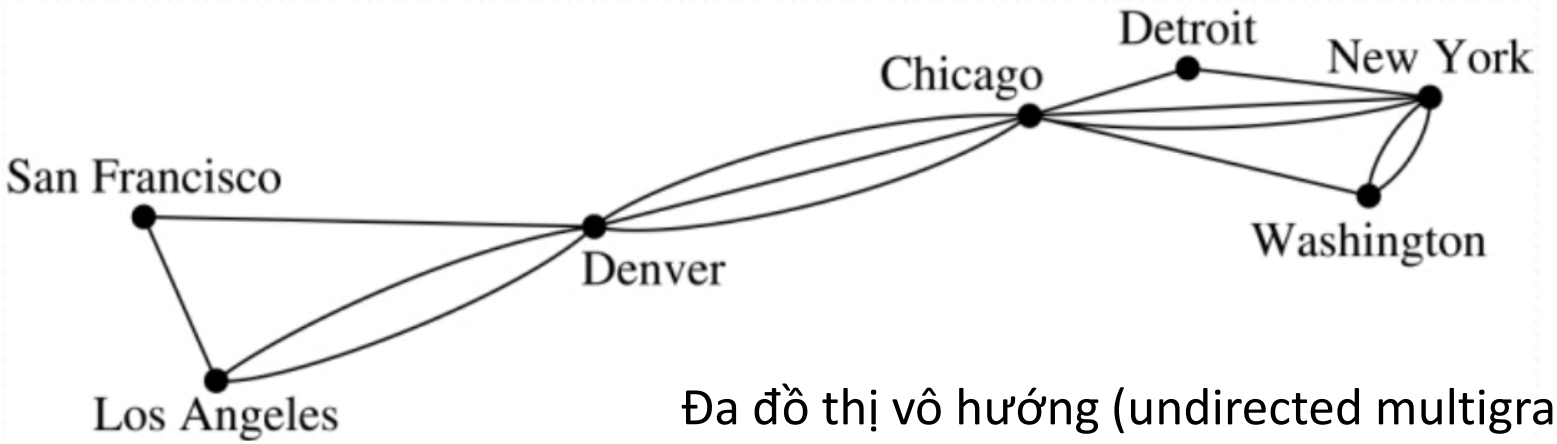
- Cạnh bội: đường cao tốc nhiều làn, mạng song song, v.v.
- Cạnh khuyên: các trạng thái lặp lại trong mô hình Markov, v.v.



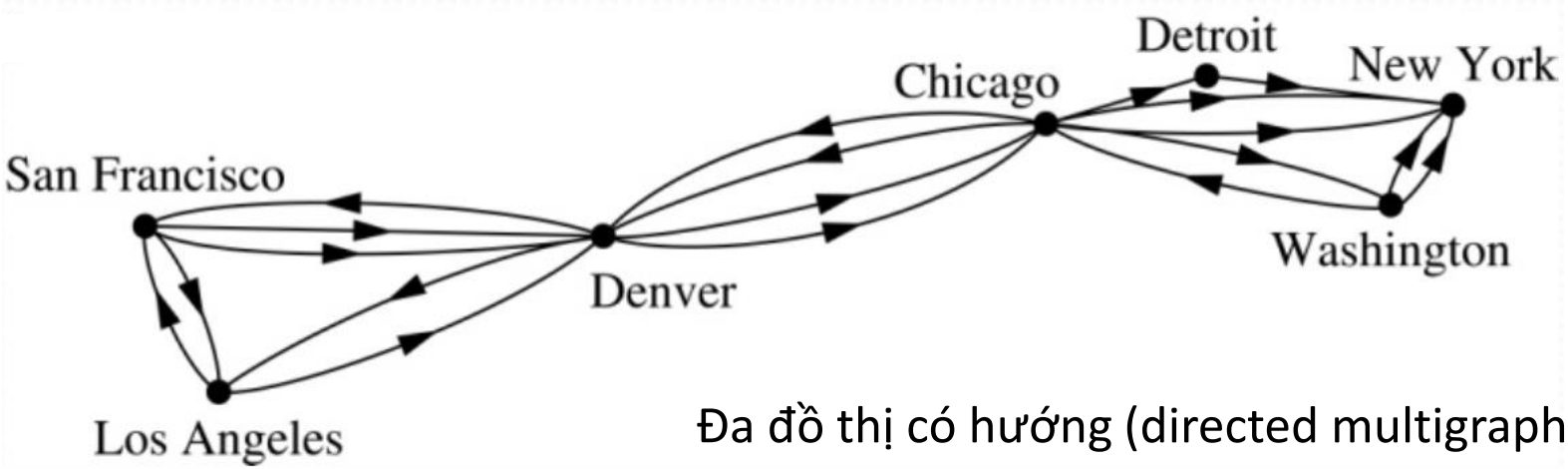


Đa đồ thị

- Đa đồ thị (multigraph) $\boldsymbol{G = (V, E)}$ chứa các cạnh bội.



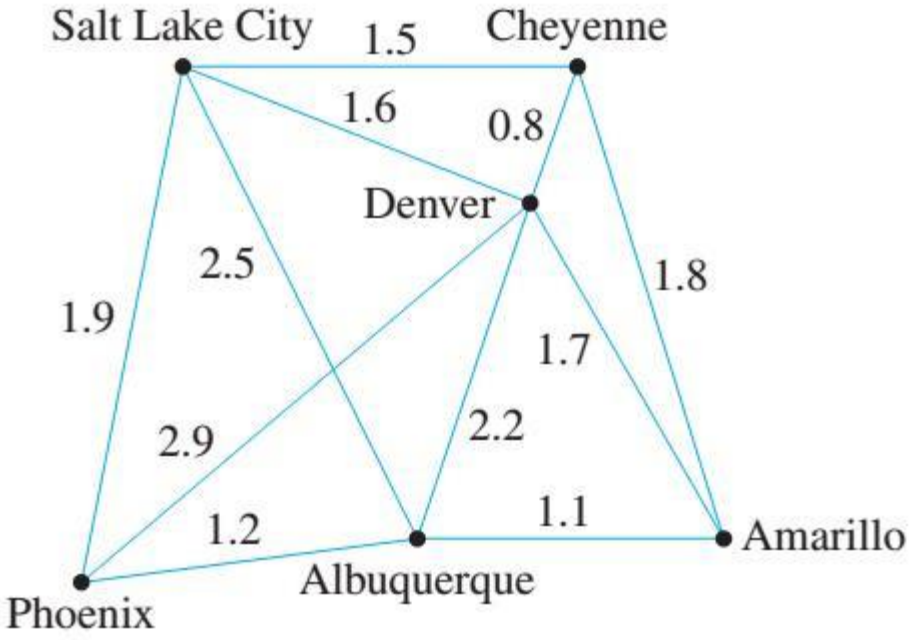
Đa đồ thị vô hướng (undirected multigraph)



Đa đồ thị có hướng (directed multigraph)

Đồ thị có trọng số

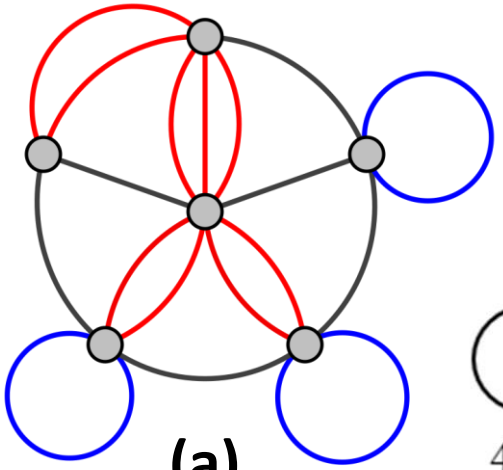
- Đồ thị có trọng số (weighted graph) có các cạnh được gắn nhãn bằng các giá trị số thể hiện cho một ngữ nghĩa nào đó.
  - Ví dụ, độ dài quãng đường bay giữa các thành phố lớn trên thế giới; mức độ tương tác của hai người dùng trong một mạng xã hội.



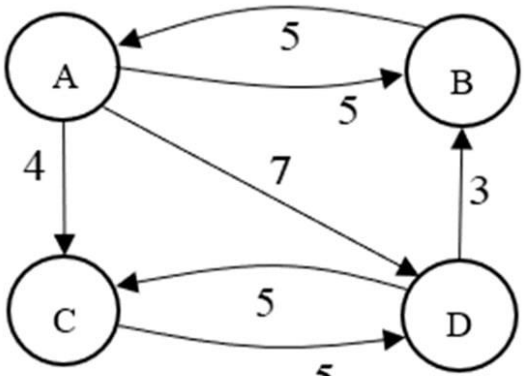
Đồ thị biểu diễn liên kết giữa sáu thành phố ở Hoa Kỳ. Trọng số của mỗi cạnh đại diện cho khoảng cách giữa hai thành phố tương ứng.

Bài tập rèn luyện

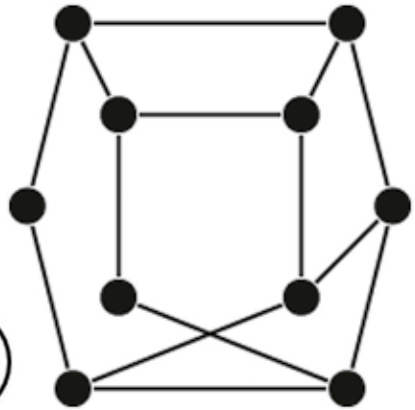
- Xác định loại đồ thị cho mỗi đồ thị được cho dưới đây.
  - Đồ thị có trọng số hay đồ thị không có trọng số?
  - Đồ thị có hướng hay đồ thị vô hướng?
  - Đơn đồ thị hay đa đồ thị?



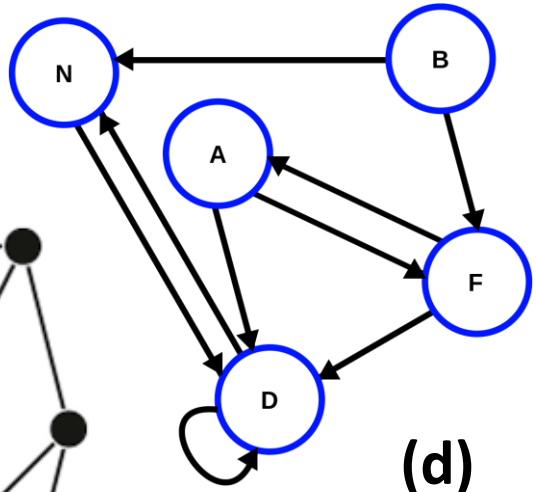
(a)



(b)



(c)



(d)

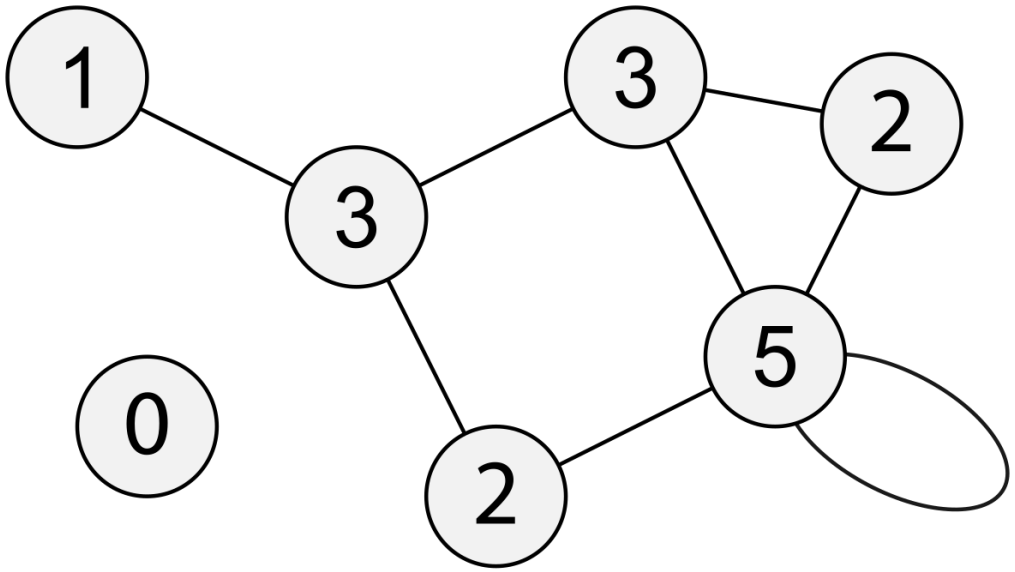
Cạnh và Đỉnh trong đồ thị vô hướng

- Hai đỉnh u và v trong một đồ thị vô hướng $\boldsymbol{G}$ là kề (adjacent) hay láng giềng (neighbor) của nhau nếu $\{u, v\}$ là cạnh của $\boldsymbol{G}$
- u và v gọi là các điểm đầu mút (endpoint) của cạnh $\{u, v\}$.

- Cạnh $e = \{u, v\}$. e được gọi là cạnh liên thuộc (incident) với các đỉnh u và v, hoặc gọi là cạnh nối (connect) u và v.

Bậc của đỉnh trong đồ thị vô hướng

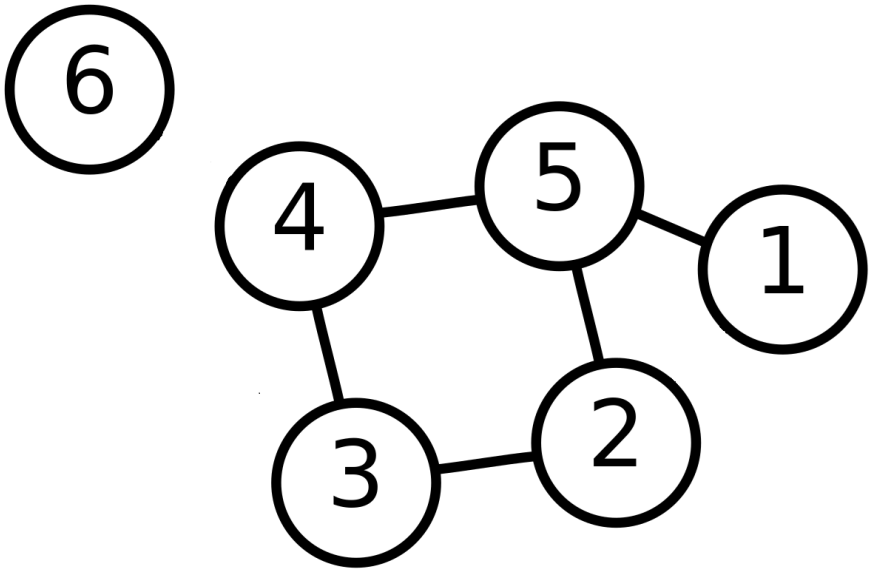
- Bậc (degree) của một đỉnh v trên đồ thị vô hướng là số cạnh liên thuộc với v, ký hiệu $\deg(v)$.
- Ngoại lệ: cạnh khuyên tại một đỉnh sẽ thêm hai bậc vào đỉnh



Các đỉnh đã được gán nhãn bằng bậc của đỉnh.

Đỉnh treo và Đỉnh cô lập

- Đỉnh treo (pendant vertex) là đỉnh có bậc bằng 1.
- Đỉnh cô lập (isolated vertex) là đỉnh có bậc bằng 0.



Đỉnh 1 là đỉnh treo.

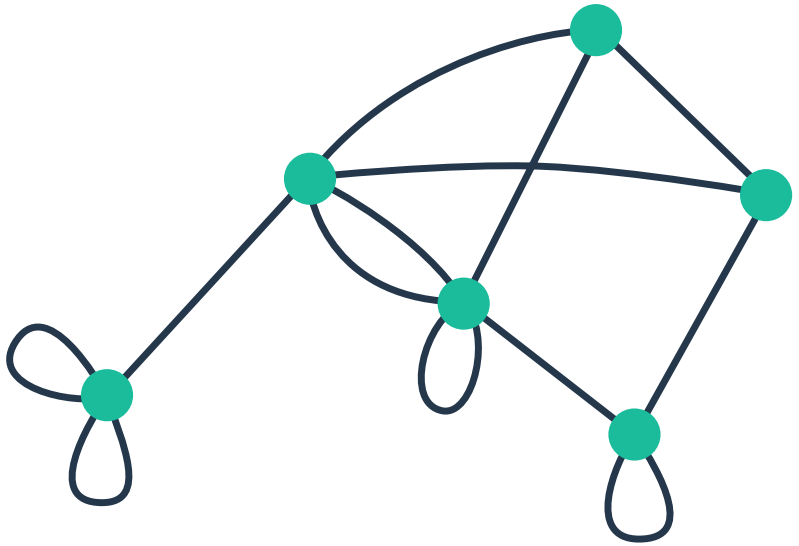
Đỉnh 6 là đỉnh cô lập.

Số cạnh trong đồ thị vô hướng

- Định lý bắt tay (Handshaking Theorem): Cho đồ thị vô hướng $\boldsymbol{G = (V, E)}$ có $|E|$ cạnh. Số cạnh liên hệ bậc của đỉnh như sau

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

  - V là tập đỉnh của $\boldsymbol{G}$, $|E|$ là số cạnh trong $\boldsymbol{G}$, và $\deg(v)$ là bậc của đỉn.h



Đồ thị này
có 13 cạnh.

Giải thích định lý bắt tay

- Định nghĩa hàm $f: E \times V \to \{0, 1, 2\}$ trong đó:
  - $f(e, v) = 0$ nếu e **không** liên thuộc với v.
  - $f(e, v) = 1$ nếu e **không** là cạnh khuyên và e liên thuộc với v.
  - $f(e, v) = 2$ nếu e là cạnh khuyên và e liên thuộc với v.
- Với cạnh $e \in E$ bất kỳ, ta suy ra được từ định nghĩa $\sum_{v \in V} f(e, v) = 2$
- Như vậy, $$\sum_{e \in E} 2 = \sum_{e \in E} \sum_{v \in V} f(e, v)$$

$$\Rightarrow \quad 2|E| = \sum_{e \in E} 2 = \sum_{e \in E} \sum_{v \in V} f(e, v) = \sum_{v \in V} \sum_{e \in E} f(e, v)$$

- Ta có: $\deg(v) = \sum_{e \in E} f(e, v) \quad \Rightarrow \quad 2|E| = \sum_{v \in V} \deg(v)$ [đpcm]

Định lý về số đỉnh bậc lẻ

- Số lượng đỉnh bậc lẻ trong một đồ thị vô hướng là số chẵn.

- **Chứng minh**
  - Giả sử V_1 và V_2 tương ứng là tập đỉnh bậc chẵn và tập đỉnh bậc lẻ của đồ thị vô hướng $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$. Khi đó:

$$2e = \sum_{v \in V} \deg(v) = \sum_{v \in V_1} \deg(v) + \sum_{v \in V_2} \deg(v)$$

  - Vì $\deg(v)$ chẵn với mọi $v \in V_1$, nên $\sum_{v \in V_1} \deg(v)$ phải là số chẵn. Do đó $\sum_{v \in V_2} \deg(v)$ cũng phải là số chẵn.
  - Vì tất cả các số hạng của tổng $\sum_{v \in V_2} \deg(v)$ là số lẻ, nên số lượng số hạng của nó phải là số chẵn. [đpcm]

Cạnh và Đỉnh trong đồ thị có hướng

- Xét (u, v) là cạnh của đồ thị có hướng G.
- u được gọi là nối tới v và v được gọi là nối từ u.
- Đỉnh u là đỉnh đầu (initial vertex), đỉnh v là đỉnh cuối (terminal hoặc end vertex) của cạnh (u, v).
- Cạnh $e = (u, v)$ được gọi là đi từ đỉnh u tới đỉnh v hoặc đi ra đỉnh u vào đỉnh v.

Bậc vào và Bậc ra trong đồ thị có hướng

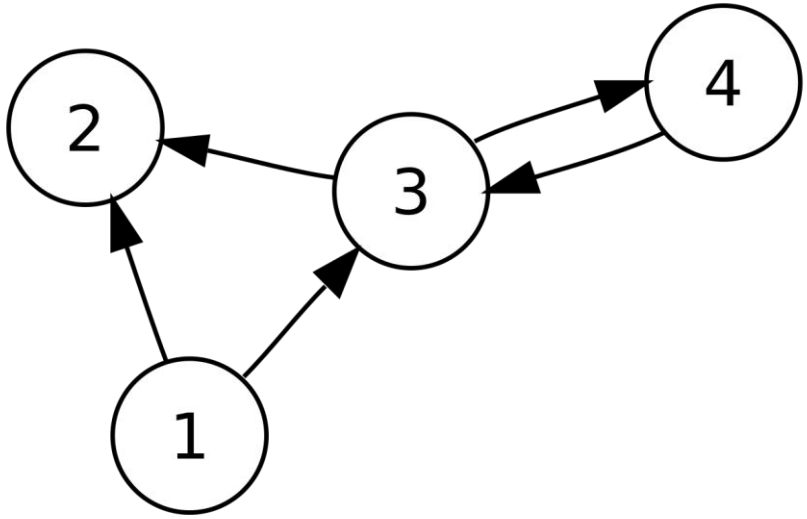
- Bậc vào (in-degree) của đỉnh v là số cạnh có đỉnh cuối là v, ký hiệu là $\deg^-(v)$.
- Bậc ra (out-degree) của đỉnh v là số cạnh có đỉnh đầu là v, ký hiệu là $\deg^+(v)$.
- Một cạnh khuyên đóng góp 1 đơn vị vào bậc vào và 1 đơn vị vào bậc ra của đỉnh chứa khuyên.

$deg^-1 = 0$ $deg^+1 = 2$

$deg^-2 = 2$ $deg^+2 = 0$

$deg^-3 = 2$ $deg^+3 = 2$

$deg^-4 = 1$ $deg^+4 = 1$



Số cạnh trong đồ thị có hướng

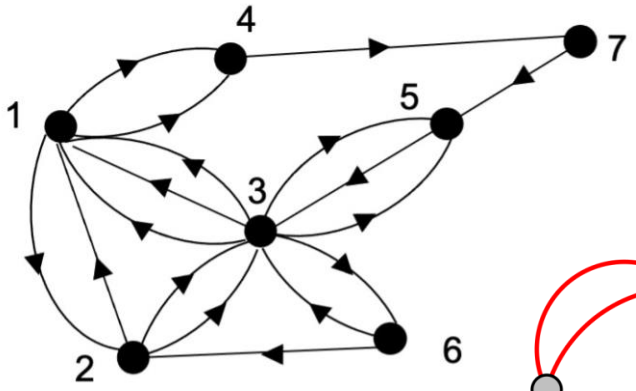
- Cho đồ thị có hướng $\boldsymbol{G = (V, E)}$.
- Số cạnh liên hệ bậc của đỉnh như sau

$$|\boldsymbol{E}| = \sum_{\boldsymbol{v \in V}} \mathbf{deg}^{-}(\boldsymbol{v}) = \sum_{\boldsymbol{v \in V}} \mathbf{deg}^{+}(\boldsymbol{v})$$

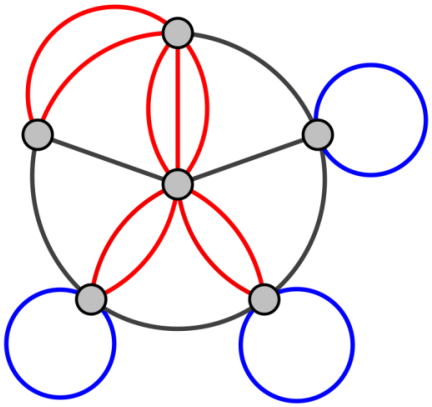
  - V và E là tập đỉnh và tập cạnh của G, $\deg^{-}(v)$ và $\deg^{+}(v)$ lần lượt là bậc của đỉnh v.

Bài tập rèn luyện

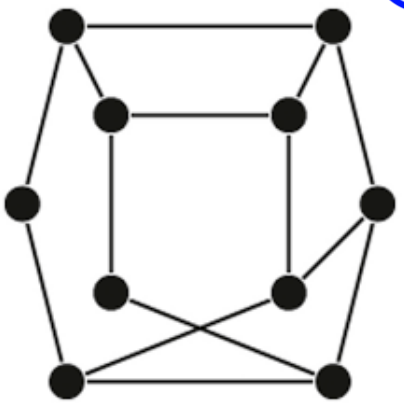
- Xác định các thông tin sau cho mỗi đồ thị được cho dưới đây.
  - Đếm bậc hoặc đếm bậc vào/bậc ra
  - Số lượng đỉnh, đỉnh cô lập, và đỉnh treo.
  - Số lượng cạnh, cạnh khuyên, và cặp đỉnh xuất hiện cạnh bội



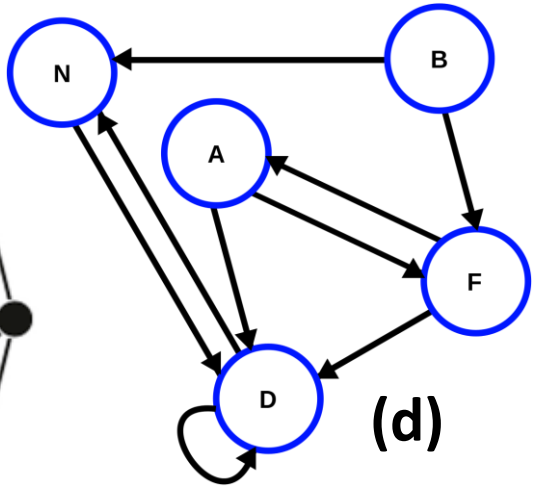
(a)



(b)



(c)



(d)

Bài tập rèn luyện

- Cho G là một đồ thị đơn, vô hướng có số đỉnh $n > 3$.
- Chứng minh G có chứa 2 đỉnh cùng bậc.

Bài tập rèn luyện

- Có thể tồn tại đồ thị đơn có 15 đỉnh, mỗi đỉnh có bậc bằng 5 hay không?

Bài tập rèn luyện

- Có thể tồn tại đồ thị đơn chứa năm đỉnh với các bậc sau đây hay không?
- Nếu có hãy vẽ đồ thị đó.

a) 3, 3, 3, 3, 2

b) 1, 2, 3, 4 ,5

c) 1, 1, 1, 1, 1

d) 1, 2, 1, 2, 1

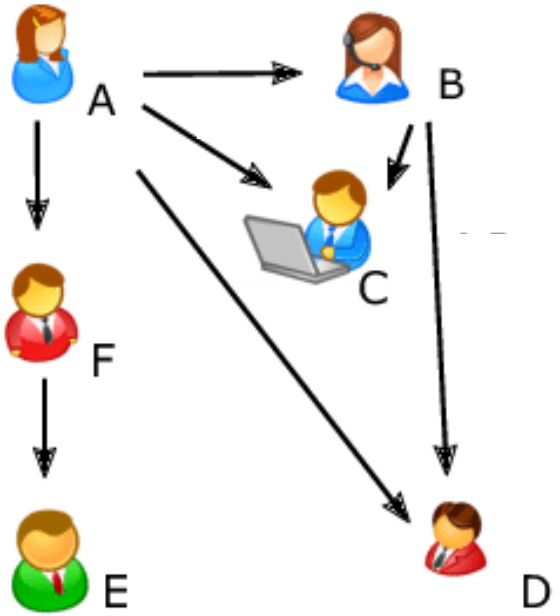
e) 2, 1, 0, 2, 1

Một số mô hình đồ thị

Biểu diễn đồ thị cho mạng xã hội

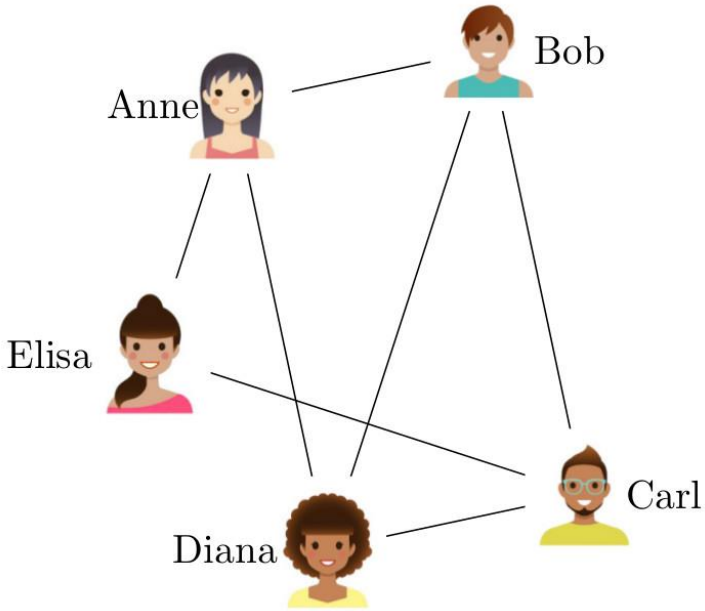
- Đồ thị có thể mô hình hóa các cấu trúc xã hội dựa trên các loại mối quan hệ khác nhau giữa các cá nhân hoặc nhóm.
- Đỉnh đại diện cho cá nhân hoặc nhóm
- Cạnh đại diện cho mối quan hệ giữa những đối tượng này.

Đồ thị ảnh hưởng (influence graph): Nếu cá nhân A có thể ảnh hưởng đến một cá nhân B khác, ta có cạnh có hướng (A, B).



Biểu diễn đồ thị cho mạng xã hội

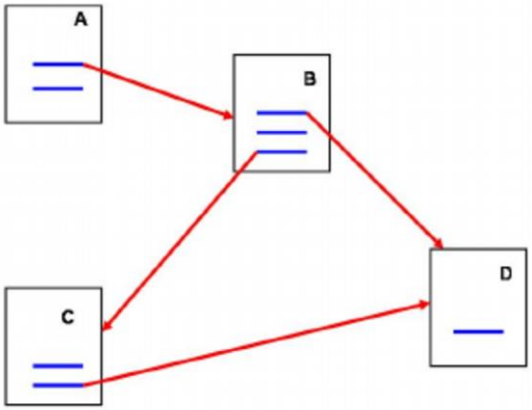
- Đồ thị tình bạn (friendship graph): Hai cá nhân được kết nối nếu họ là bạn bè của nhau (trong thế giới thực hoặc trong một thế giới ảo cụ thể).
- Đồ thị cộng tác (collaboration graph): Hai cá nhân/nhóm được kết nối nếu họ cộng tác theo một cách cụ thể nào đó.



- Đồ thị cộng tác học thuật (academic collaboration graph): Hai nhà khoa học cùng nhau viết một bài báo về một chủ đề cụ thể sẽ có kết nối.
- Đồ thị Hollywood (Holywood graph): Tồn tại một kết nối giữa hai diễn viên nếu họ xuất hiện trong cùng một bộ phim.

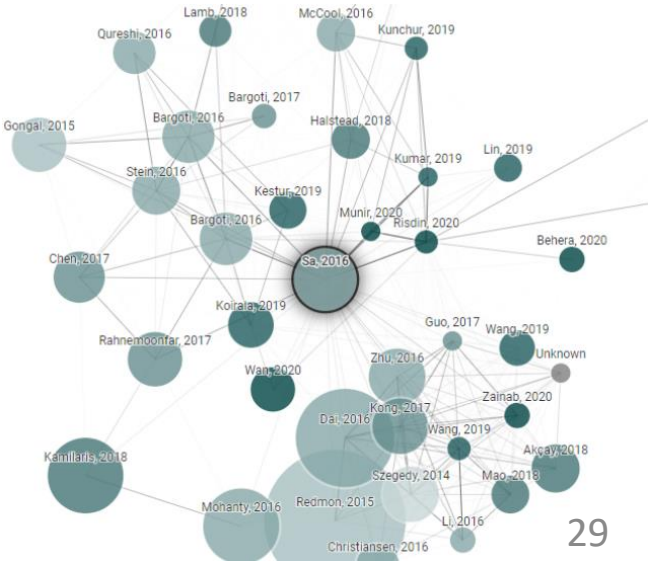
Biểu diễn đồ thị cho mạng thông tin

- Đồ thị có thể được sử dụng để mô hình hóa các loại mạng khác nhau liên kết các loại thông tin khác nhau.



Đồ thị Web (Web graph): Đỉnh là trang web và hyperlink là cạnh có hướng, biểu diễn toàn bộ WWW hoặc một phiên bản web thu nhỏ được sử dụng trong công cụ tìm kiếm.

Mạng trích dẫn (citation network): Đỉnh là bài báo nghiên cứu chuyên ngành. Khi bài báo A trích dẫn một bài báo B khác làm tài liệu tham khảo, ta sẽ có cạnh có hướng (A, B).



Biểu diễn đồ thị cho mạng lưới giao thông

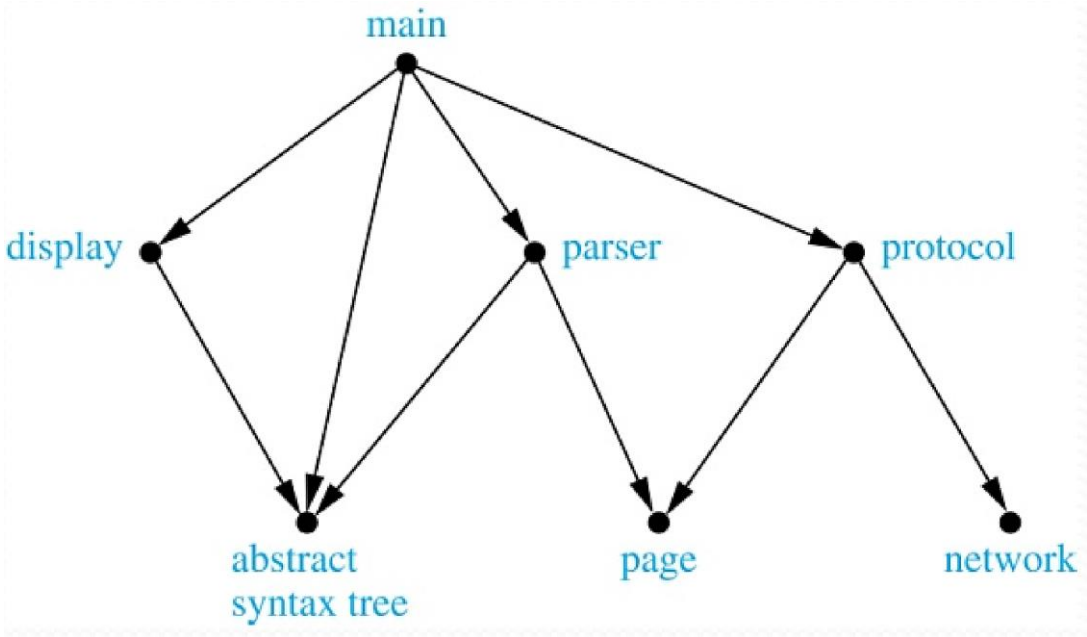
- Các mô hình đồ thị được sử dụng rộng rãi trong nghiên cứu về mạng lưới giao thông.
- Mạng hàng không (airline network) có thể được mô hình hóa bằng đa đồ thị có hướng, trong đó
  - Đỉnh là sân bay. Mỗi chuyến bay được biểu diễn bằng một cạnh có hướng từ đỉnh của sân bay khởi hành đến đỉnh của sân bay đích.
- Mạng lưới đường bộ (road network) là đồ thị trong đó
  - Đỉnh biểu diễn giao lộ và các cạnh biểu diễn đường bộ.
  - Cạnh vô hướng biểu diễn đường hai chiều và cạnh có hướng biểu diễn đường một chiều.

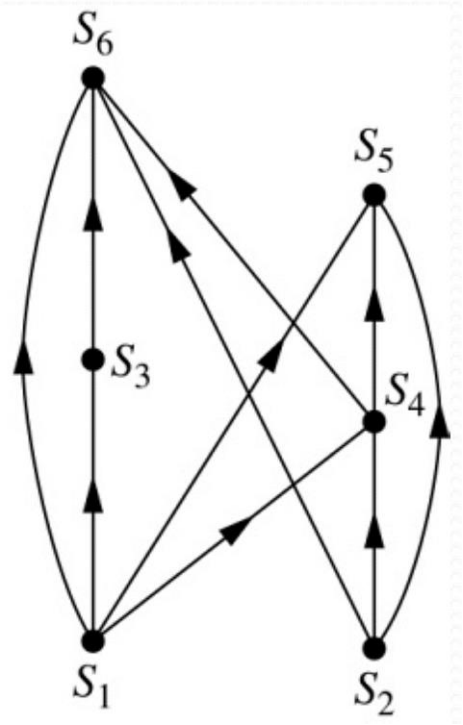
Biểu diễn đồ thị trong thiết kế phần mềm

- Mô hình đồ thị hiện diện rộng rãi trong thiết kế phần mềm.
- Đồ thị phụ thuộc mô-đun (module dependency graph) biểu diễn sự phụ thuộc giữa các mô-đun trong một hệ thống.
  - Đỉnh biểu diễn một mô-đun. Tồn tại một cạnh từ mô-đun A sang một mô-đun B khác nếu B phụ thuộc vào A.
  - Những phụ thuộc này cần được hiểu trước khi lập trình.
- Đồ thị thứ tự ưu tiên (precedence graph) biểu diễn thứ tự của các câu lệnh trước khi ta thực thi từng câu lệnh.
  - Đỉnh biểu diễn câu lệnh trong một chương trình máy tính. Tồn tại một cạnh có hướng từ đỉnh A đến một đỉnh B khác nếu B không thể được thực thi trước A.

Biểu diễn đồ thị trong thiết kế phần mềm

Đồ thị phụ thuộc mô-đun biểu diễn sự phụ thuộc giữa bảy mô-đun trong thiết kế của trình duyệt web.





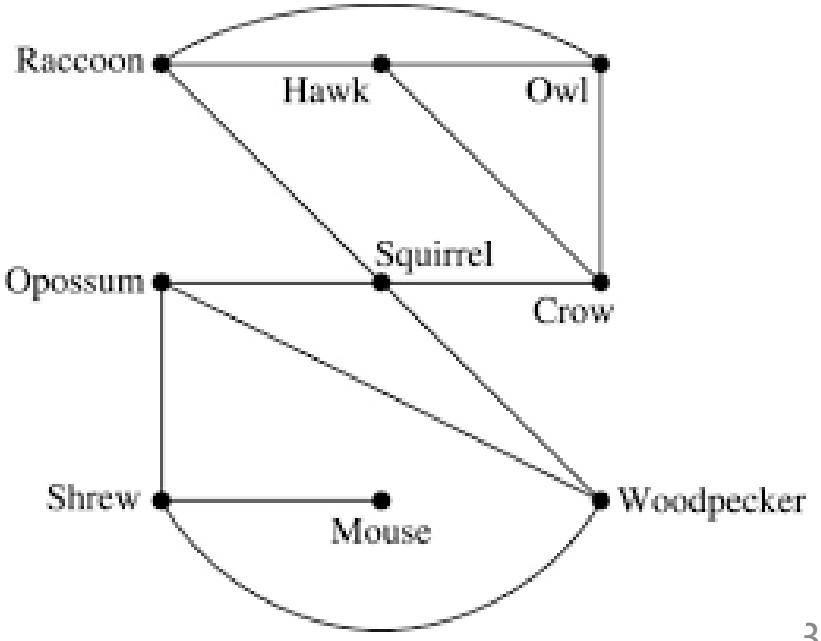
Đồ thị thứ tự ưu tiên chỉ định câu lệnh nào phải được thực thi trước trong sáu câu lệnh của chương trình.

S1.	a := 0	S4.	d := b + a
S2.	b := 1	S5.	e := d + 1
S3.	c := a + 1	S6.	e := c + d

Biểu diễn đồ thị trong sinh học

- Lĩnh vực sinh học cũng thường sử dụng các mô hình đồ thị.
- Đồ thị chồng lấn (Niche overlap graph) mô hình hóa sự cạnh tranh giữa các loài trong một hệ sinh thái.
  - Đỉnh biểu diễn loài và cạnh thể hiện sự cạnh tranh nguồn thức ăn của hai loài.

Đây là đồ thị chồng lấn cho một hệ sinh thái rừng gồm có chín loài.



Bài tập rèn luyện

- Xây dựng đồ thị ảnh hưởng cho các thành viên lãnh đạo của một công ty theo mô tả như bên dưới.
- Chủ tịch có ảnh hưởng lên giám đốc nghiên cứu & phát triển, giám đốc marketing, giám đốc điều hành.
- Cả giám đốc nghiên cứu & phát triển và giám đốc marketing đều có ảnh hưởng lên giám đốc điều hành.
- Không ai có thể ảnh hưởng lên trưởng phòng tài chính và trưởng phòng tài chính không ảnh hưởng lên bất cứ ai.

Bài tập rèn luyện

- Hãy xây dựng đồ thị ưu tiên cho chương trình sau:

S1.	x := 0	S4.	z := y
S2.	x := x + 1	S5.	x := x + 2
S3.	y := 2	S6.	y := x + z

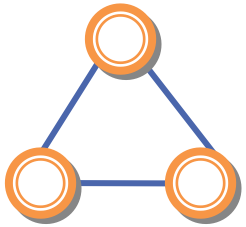
Đơn đồ thị đặc biệt

Đồ thị đầy đủ

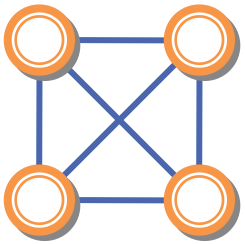
- Đồ thị đầy đủ (complete graph) là đơn đồ thị chứa đúng một cạnh nối mỗi cặp đỉnh phân biệt.
- Ký hiệu: K_n, với n là số đỉnh trong đồ thị.

K_1

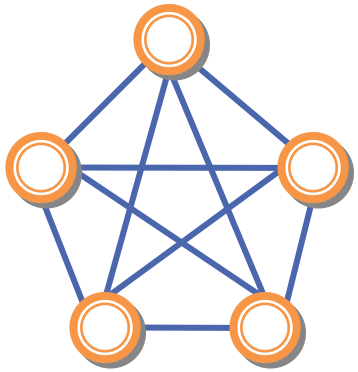
K_2



K_3



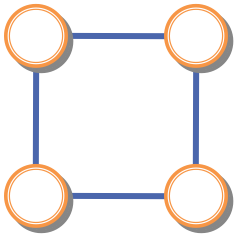
K_4



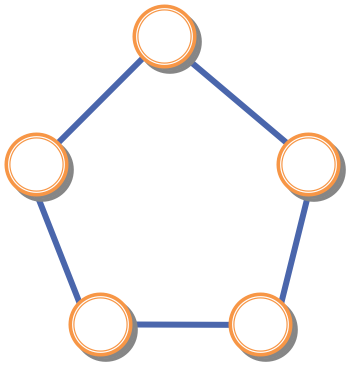
K_5

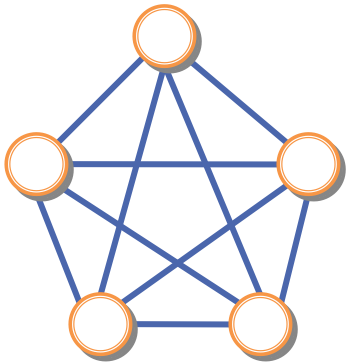
Đồ thị chính quy

- Đồ thị chính quy (regular graph) là đơn đồ thị mà bậc của mọi đỉnh đều bằng nhau.
- Ký hiệu: k-chính quy, với k là bậc của các đỉnh trong đồ thị.

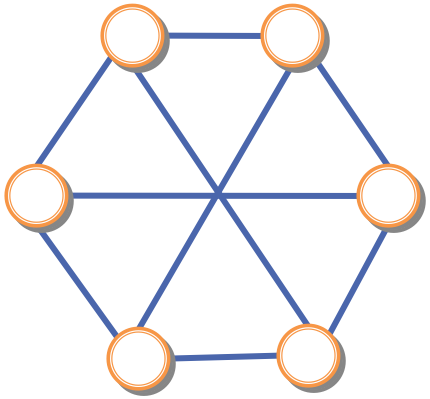


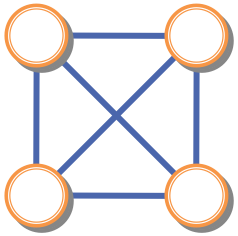
2 – chính quy





4 – chính quy

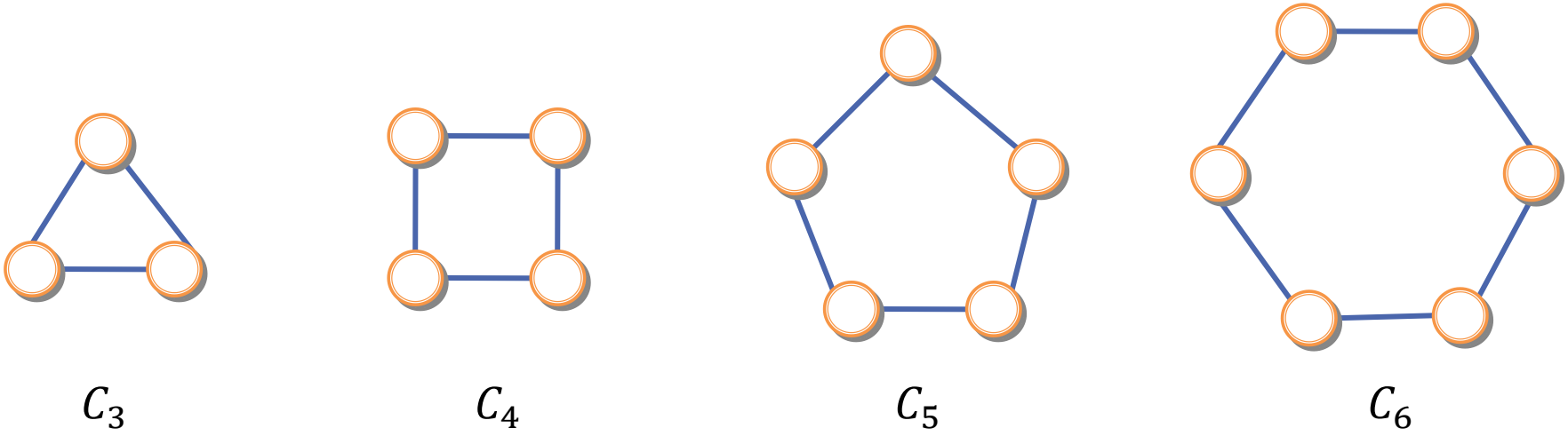




3 – chính quy

Đồ thị vòng

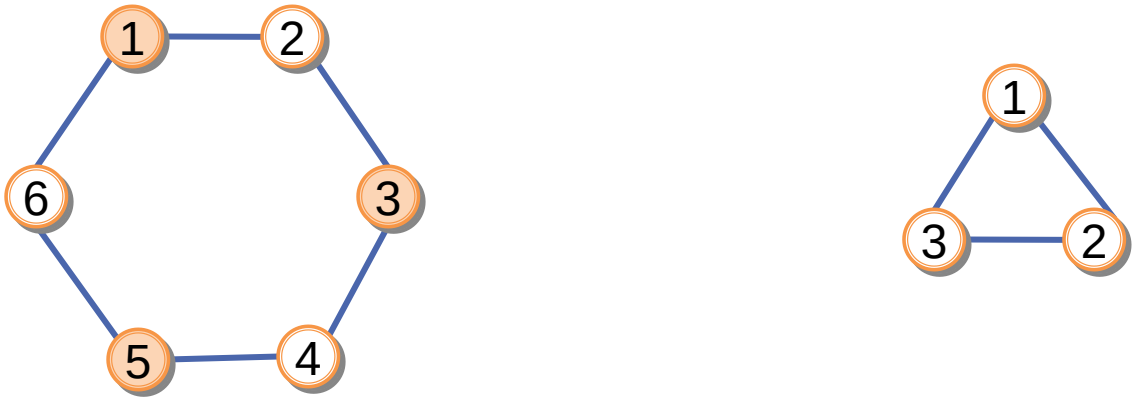
- Đồ thị vòng (cycle graph) C_n $(n \geq 3)$ là đồ thị có n đỉnh $v_1, v_2, \ldots, v_n$và các cạnh $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}$.



C_3 C_4 C_5 C_6

Đồ thị phân đôi (lưỡng phân)

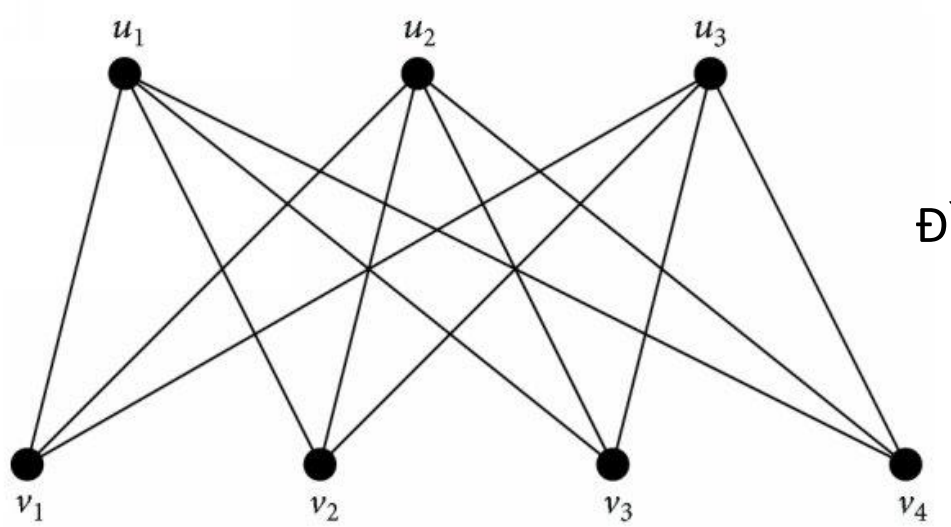
- Đồ thị phân đôi (bipartite graph) $\boldsymbol{G}$ có tập đỉnh $\boldsymbol{V}$ có thể phân làm hai tập con không rỗng rời nhau, V_1 và V_2, sao cho mỗi cạnh của đồ thị nối một đỉnh của V_1 với một đỉnh của V_2.



Hình trái: Đồ thị C_6 là đồ thị phân đôi.
Hình phải: Đồ thị K_3 không phải là đồ thị phân đôi.

Đồ thị phân đôi đầy đủ

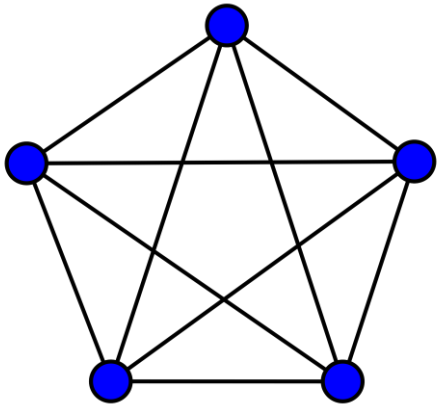
- Đồ thị phân đôi đầy đủ (complete bipartite graph) là đồ thị phân đôi trong đó một đỉnh thuộc tập con này có cạnh nối đến mọi đỉnh thuộc tập con kia.
- Ký hiệu: $K_{x,y}$, với x và y là số lượng đỉnh của hai tập con.



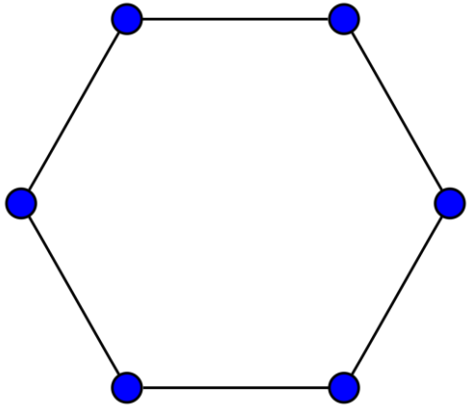
Đồ thị phân đôi đầy đủ $K_{3,4}$

Bài tập rèn luyện

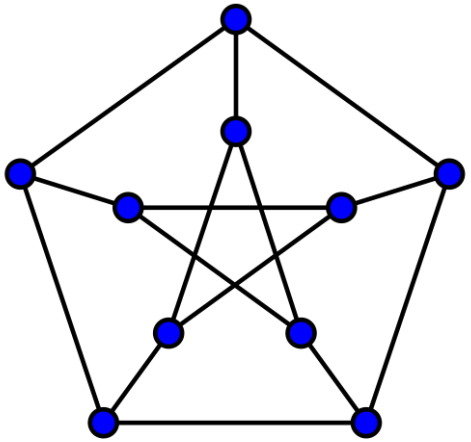
- Xác định loại đơn đồ thị đặc biệt cho các đồ thị bên dưới.
- Lưu ý: một đồ thị có thể thuộc về nhiều hơn một loại đồ thị đặc biệt.



(a)



(b)



(c)

Bài tập rèn luyện

- Hãy vẽ một đồ thị minh họa cho mỗi yêu cầu sau đây.
- Lưu ý: Đồ thị cho mỗi câu phải khác với đồ thị đã có ở câu hỏi trước.

(a)

Đồ thị chỉ là đồ thị chính quy

(b)

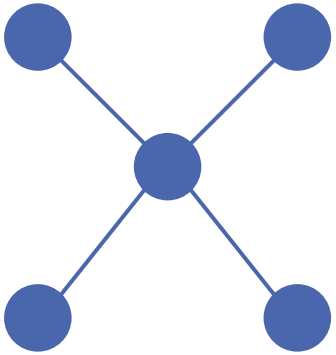
Đồ thị vừa là đồ thị vòng vừa là đồ thị lưỡng phân

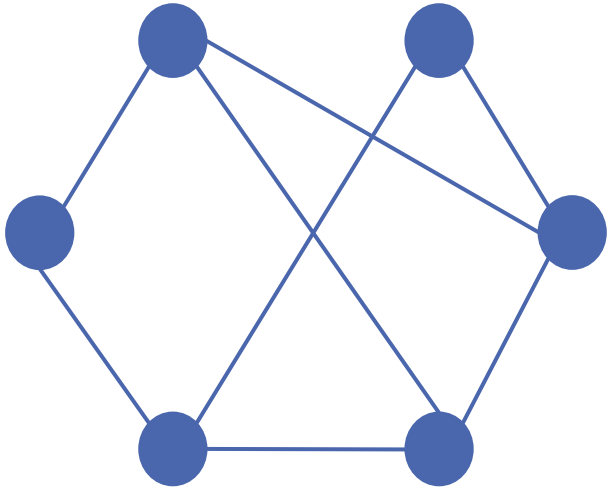
(c)

Đồ thị là đồ thị vòng nhưng không phải là đồ thị lưỡng phân

Bài tập rèn luyện

- Các đồ thị sau đây có phải là đồ thị phân đôi không?





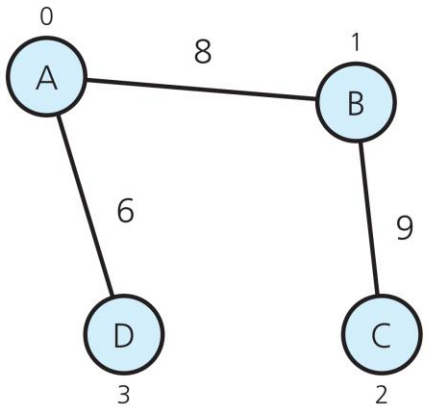
Biểu diễn đồ thị

Ma trận kề

- Xét đồ thị $\boldsymbol{G}$ có n đỉnh, được đánh chỉ mục $0, 1, \ldots, n-1$.
- Ma trận kề (adjacency matrix) là ma trận vuông kích thước n sao cho với mọi cặp đỉnh i and j thì

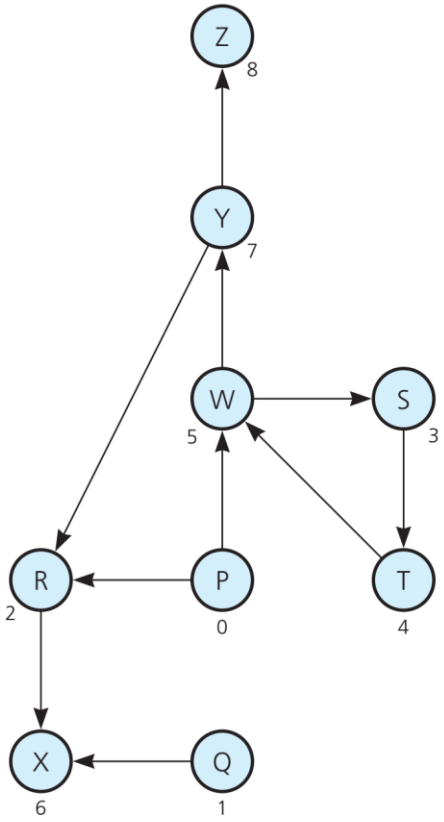
$$matrix[i][j] = \begin{cases} 1 & nếu\ có\ cạnh\ từ\ i\ đến\ j \\ 0 & ngược\ lại \end{cases}$$

- Trong đồ thị có trọng số, $matrix[i][j]$ là giá trị trọng số của cạnh đi từ đỉnh i to đến j.
  - $matrix[i][j] = \infty$ khi không có cạnh đi từ i đến j.
- Ma trận kề cho đồ thị vô hướng có tính đối xứng.



		0	1	2	3
		A	B	C	D
0	A	∞	8	∞	6
1	B	8	∞	9	∞
2	C	∞	9	∞	∞
3	D	6	∞	∞	∞

Đồ thị vô hướng có trọng số và ma trận kề tương ứng.



		0	1	2	3	4	5	6	7	8
		P	Q	R	S	T	W	X	Y	Z
0	P	0	0	1	0	0	1	0	0	0
1	Q	0	0	0	0	0	0	1	0	0
2	R	0	0	0	0	0	0	1	0	0
3	S	0	0	0	0	1	0	0	0	0
4	T	0	0	0	0	0	1	0	0	0
5	W	0	0	0	1	0	0	0	1	0
6	X	0	0	0	0	0	0	0	0	0
7	Y	0	0	1	0	0	0	0	0	1
8	Z	0	0	0	0	0	0	0	0	0

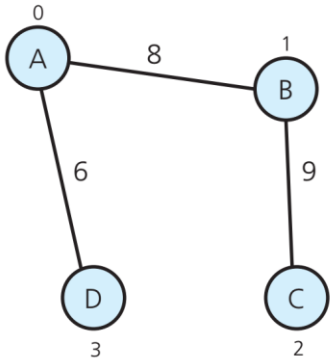
Đồ thị có hướng và ma trận kề tương ứng.

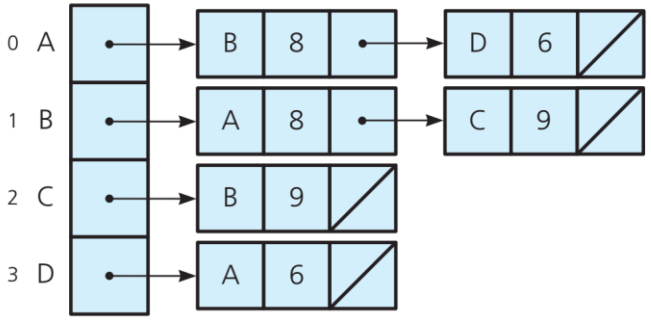
Nguồn ảnh: Data Abstraction and Problem Solving with C++: Walls and Mirrors, 6th edition

Danh sách kề

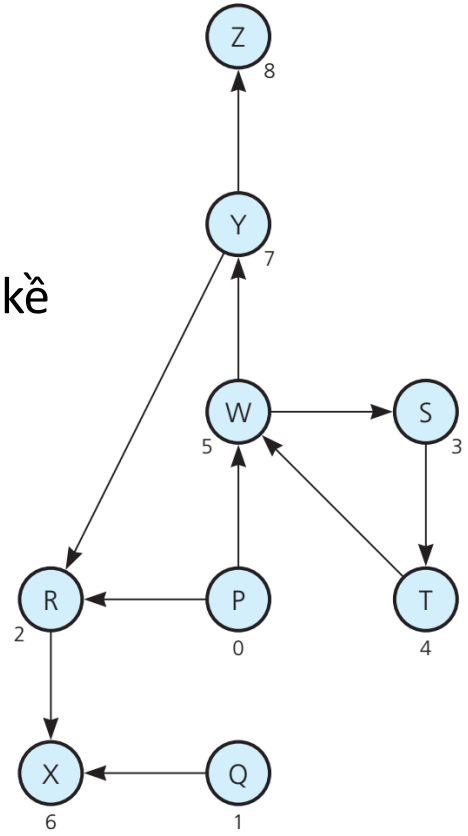
- Xét đồ thị $\boldsymbol{G}$ có n đỉnh, được đánh chỉ mục $0, 1, \ldots, n-1$.
- Danh sách kề (adjacency list) chứa n danh sách liên kết, mỗi danh sách biểu diễn cho một đỉnh.
  - Danh sách liên kết thứ i chứa phần tử j khi và chỉ khi tồn tại cạnh nối từ đỉnh i đến đỉnh j.
- Trong đồ thị vô hướng, mỗi cạnh được xem như gồm hai cạnh có hướng theo hai chiều ngược nhau.

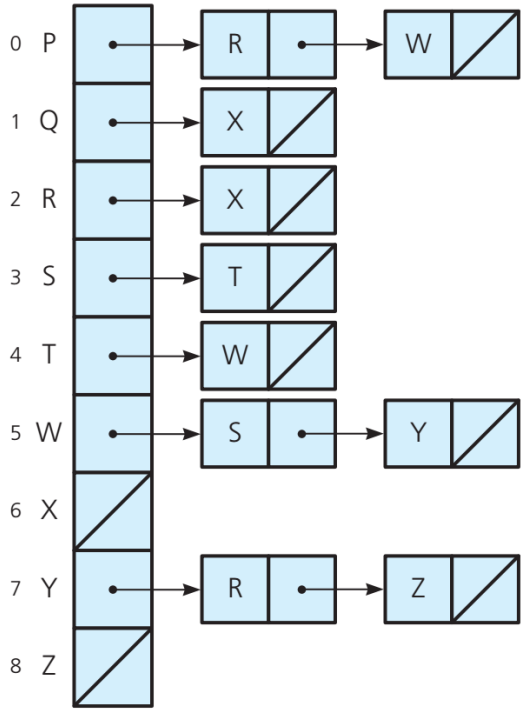
Đồ thị vô hướng có trọng số và danh sách kề tương ứng.





Đồ thị có hướng và danh sách kề tương ứng.



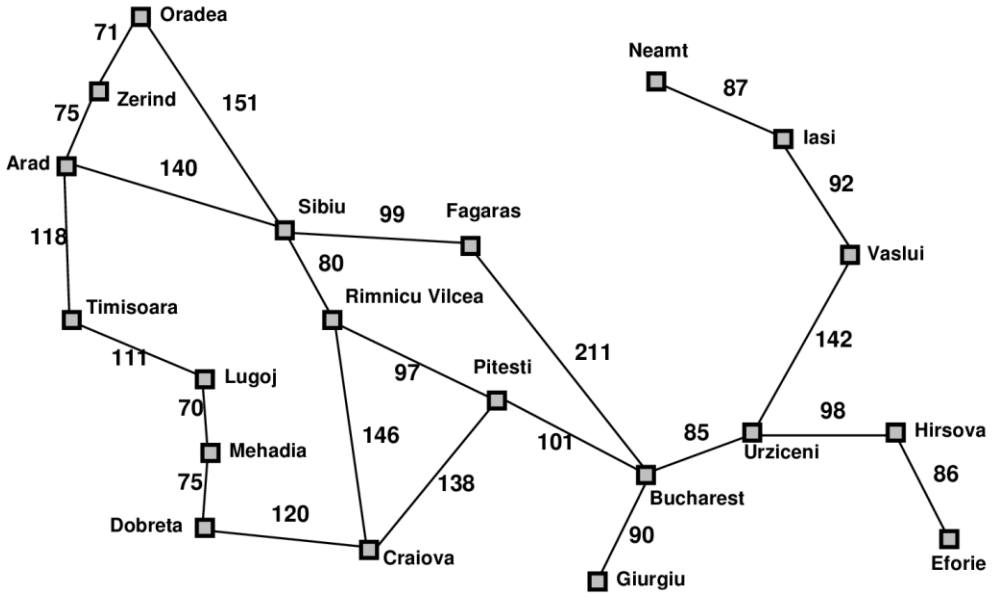


Nguồn ảnh: Data Abstraction and Problem Solving with C++: Walls and Mirrors, 6th edition

Ma trận kề so với Danh sách kề

- Việc chọn lựa phương án biểu diễn đồ thị phụ thuộc tình huống sử dụng đồ thị trong ứng dụng của bạn.

  - Thao tác nào cần thực hiện thường xuyên nhất trên đồ thị?
  - Số lượng cạnh mà đồ thị có thể chứa (trong thực tế)?



Tác vụ: Tìm mọi thành phố liền kề với một thành phố cho trước.

Ma trận kề hay Danh sách kề sẽ phù hợp hơn cho tác vụ đã nêu?

Ma trận kề so với Danh sách kề

- Hai thao tác đồ thị được thực hiện thường xuyên nhất là

Xác định sự tồn tại của cạnh nối từ đỉnh i đến đỉnh j

- **Ma trận kề:** kiểm tra giá trị của ô $matrix[i][j]$
- **Danh sách kề:** Duyệt danh sách kề thứ i để tìm phần tử tương ứng với đỉnh j

Tìm mọi đỉnh kề với đỉnh i cho trước

- **Ma trận kề:** Duyệt dòng thứ i để tìm mọi đỉnh kề với đỉnh i đã cho
- **Danh sách kề:** Duyệt danh sách kề thứ i với ít phần tử hơn

Ma trận kề so với Danh sách kề (C/C++)

- Mặc dù danh sách kề có n con trỏ đầu (head pointer), nó vẫn cần ít dung lượng lưu trữ hơn so với ma trận kề.

Ma trận kề	Danh sách kề
Luôn luôn có n^2 ô	Tối đa có $n(n-1)$ ô
Mỗi ô chỉ chứa một số nguyên	Mỗi ô có thành phần dữ liệu và con trỏ kế tiếp

Cấu trúc dữ liệu biểu diễn đồ thị (C#)

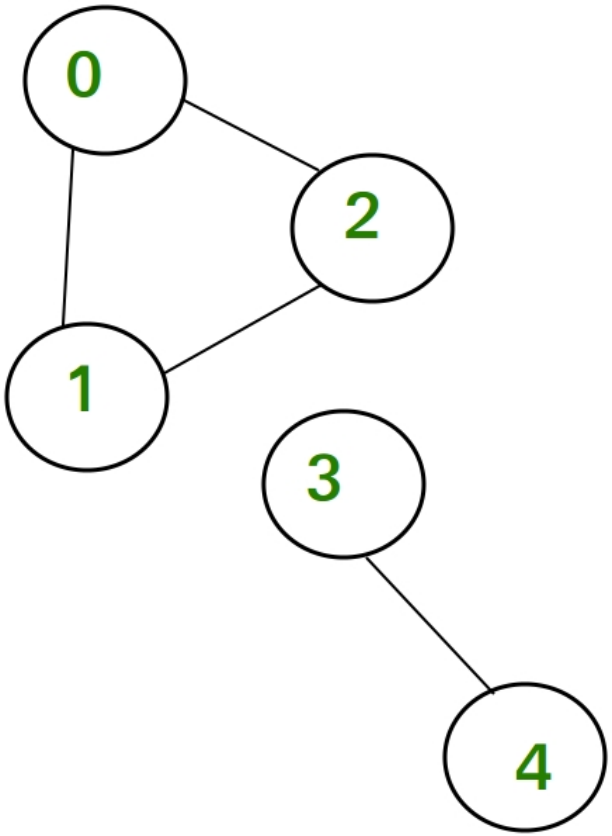
```
class AdjacencyList{                        // DANH SÁCH KỀ
    public int n;                           // số đỉnh của đồ thị
    public List<int>[] data;                // dữ liệu về đồ thị

    // Các hàm cài đặt thao tác cơ bản trên đồ thị
}
```

```
class AdjacencyMatrix{                      // MA TRẬN KỀ
    public int n;                           // số đỉnh của đồ thị
    public int[,] data;                     // dữ liệu về đồ thị

    // Các hàm cài đặt thao tác cơ bản trên đồ thị
}
```

Đọc đồ thị từ tập tin văn bản (C#)




```
5
0 1 1 0 0
1 0 1 0 0
1 1 0 0 0
0 0 0 0 1
0 0 0 1 0
```

Ma trận kề

Danh sách kề

```
5
1 2
0 2
1 2
4
3
```

Các thao tác cơ bản trên đồ thị

- **Kiểm tra** đồ thị có rỗng hay không
- **Lấy** số lượng đỉnh trong đồ thị
- **Lấy** số lượng cạnh trong đồ thị
- **Lấy** đỉnh của đồ thị chứa một giá trị cho trước
- **Kiểm tra** có tồn tại một cạnh giữa hai đỉnh cho trước
- **Thêm** một đỉnh vào đồ thị (giả sử các đỉnh đã có của đồ thị chứa giá trị phân biệt và khác với giá trị của đỉnh mới thêm vào)
- **Thêm** một cạnh giữa hai đỉnh cho trước trong đồ thị
- **Xóa** một đỉnh cụ thể ra khỏi đồ thị và xóa mọi cạnh giữa đỉnh này với các đỉnh khác
- **Xóa** một cạnh giữa hai đỉnh cho trước trong đồ thị

Đọc dữ liệu vào danh sách kề (C#)

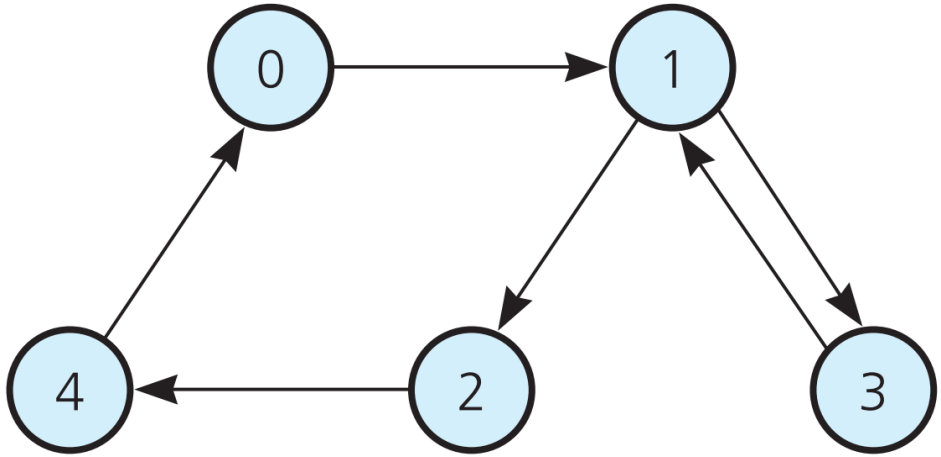
```
public void readToAdjacencyList(string filename){
    string[] lines = File.ReadAllLines(filename);
    n = Int32.Parse(lines[0]);
    data = new List<int>[n];
    for (int i = 0; i < n; ++i){
        string[] tokens = lines[i + 1].Split(new char[]
            {‘ ‘}, StringSplitOptions.RemoveEmptyEntries);
        data[i] = new List<int>();
        for (int j = 0; j < tokens.Length; ++j)
            data[i].Add(Int32.Parse(tokens[j]));
    }
}
```

Đọc dữ liệu vào ma trận kề (C#)

```
public void readToAdjacencyMatrix(string filename){
    string[] lines = File.ReadAllLines(filename);
    n = Int32.Parse(lines[0]);
    data = new int[n, n];
    for (int i = 0; i < n; ++i){
        string[] tokens = lines[i + 1].Split(new char[]
            {‘ ‘}, StringSplitOptions.RemoveEmptyEntries);
        for (int j = 0; j < n; ++j)
            data[i, j] = Int32.Parse(tokens[j]);
    }
}
```

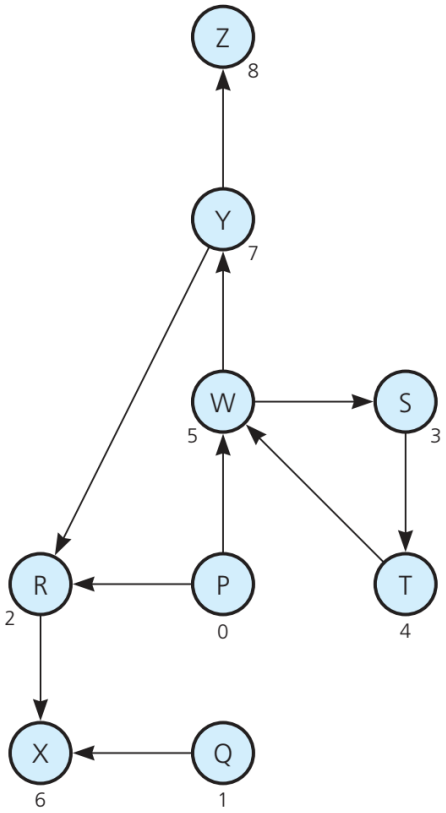
Bài tập rèn luyện

- Biểu diễn đồ thị được bằng ma trận kề và danh sách kề.

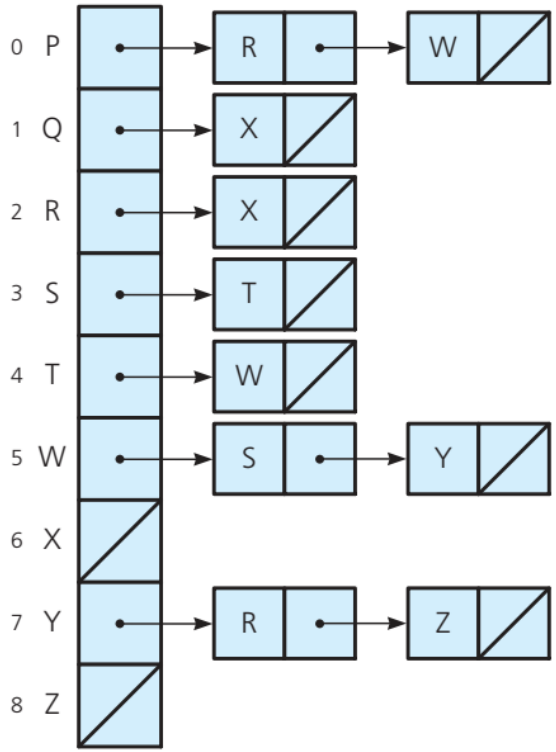


Bài tập rèn luyện

- Cho đồ thị có hướng (hình trái) cùng với ma trận kề (hình giữa) và danh sách kề (hình phải) tương ứng. Chứng minh rằng, trong C/C++, danh sách kề cần ít bộ nhớ hơn ma trận kề.



		0	1	2	3	4	5	6	7	8
		P	Q	R	S	T	W	X	Y	Z
0	P	0	0	1	0	0	1	0	0	0
1	Q	0	0	0	0	0	0	1	0	0
2	R	0	0	0	0	0	0	1	0	0
3	S	0	0	0	0	1	0	0	0	0
4	T	0	0	0	0	0	1	0	0	0
5	W	0	0	0	1	0	0	0	1	0
6	X	0	0	0	0	0	0	0	0	0
7	Y	0	0	1	0	0	0	0	0	1
8	Z	0	0	0	0	0	0	0	0	0



Bài tập rèn luyện

- Hãy vẽ các đồ thị vô hướng cho bởi ma trận kề sau:

1	3	2
3	0	4
2	4	0

1	2	0	1
2	0	3	0
0	3	1	1
1	0	1	0

0	1	3	0	4
1	2	1	3	0
3	1	1	0	1
0	3	0	0	2
4	0	1	2	3

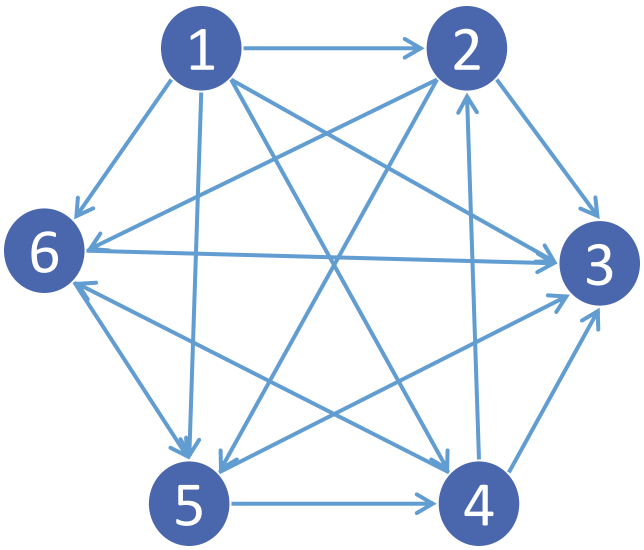
Đồ thị con

Đồ thị bộ phận

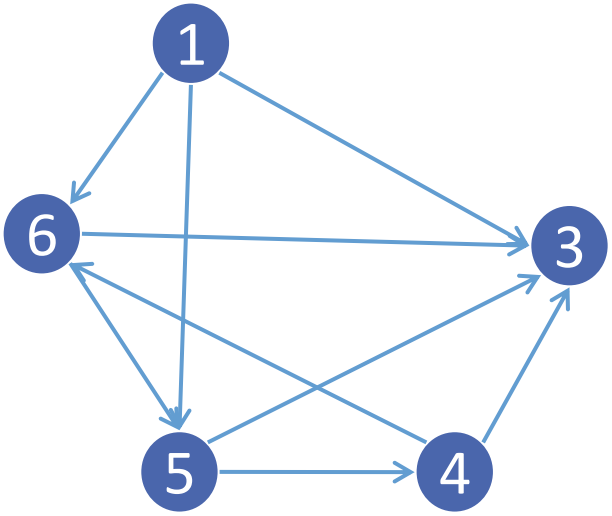
Đồ thị đẳng cấu

Đồ thị con – Đồ thị bộ phận

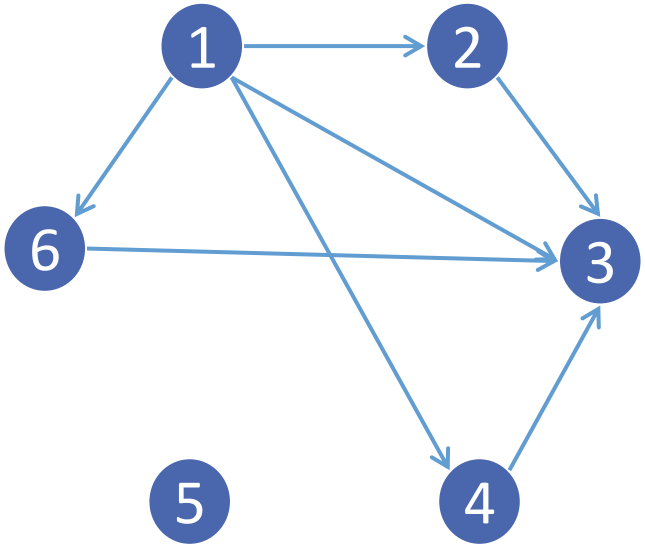
- Đồ thị con (subgraph) của đồ thị $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ là đồ thị $\boldsymbol{H} = (\boldsymbol{W}, \boldsymbol{F})$, trong đó $W \subseteq V$ và $F \subseteq E$.
- Đồ thị $\boldsymbol{H}$ còn gọi là đồ thị bộ phận (spanning subgraph) của $\boldsymbol{G}$ khi $W = V$.



Đồ thị G



Đồ thị con của G

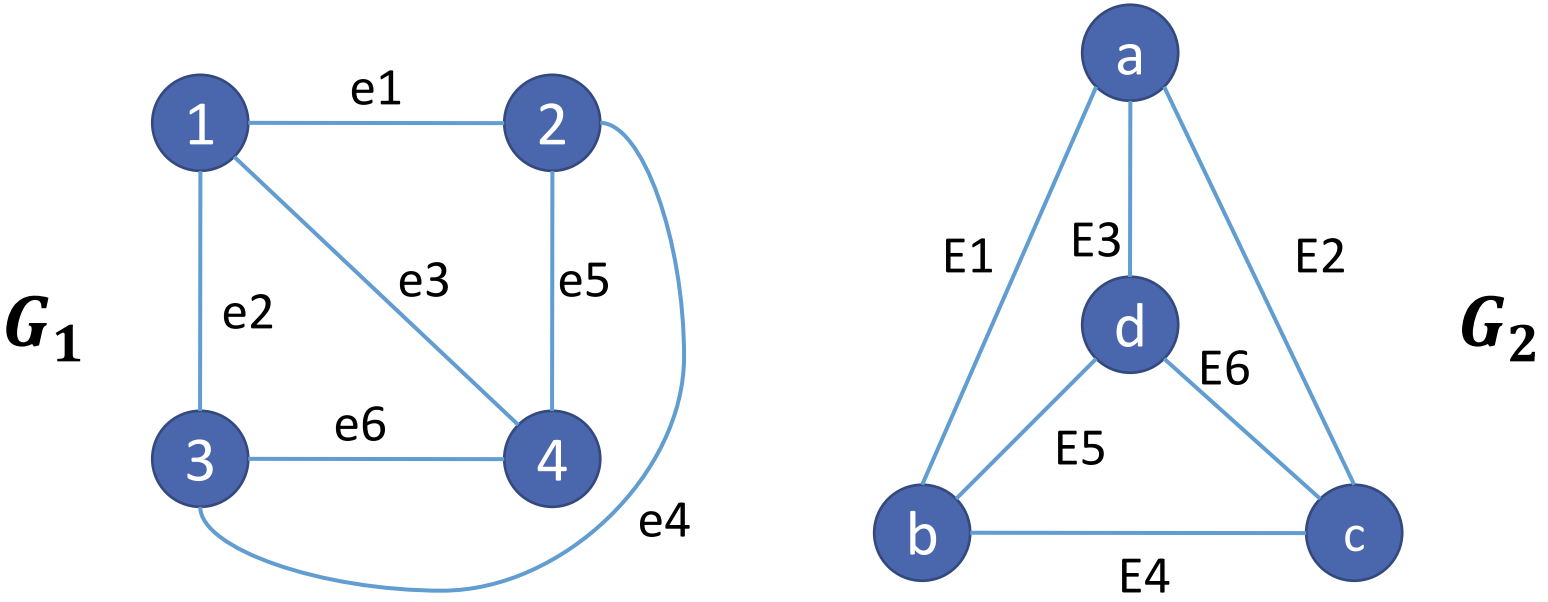


Đồ thị bộ phận của G

Quan hệ đẳng cấu

- Hai đơn đồ thị $\boldsymbol{G_1 = (V_1, E_1)}$ và $\boldsymbol{G_2 = (V_2, E_2)}$ được gọi là đẳng cấu (isomorphic) với nhau nếu tồn tại hàm song ánh $\boldsymbol{f}$ từ V_1 lên V_2 sao cho các đỉnh u và v là liền kề trong $\boldsymbol{G_1}$ nếu và chỉ nếu $f(u)$ và $f(v)$ là liền kề trong $\boldsymbol{G_2}$ với mọi $u, v \in V_1$.
- Hàm f như vậy được gọi là một đẳng cấu.

Ví dụ về đồ thị đẳng cấu



Đồ thị G1 và G2 là đẳng cấu với nhau vì tồn tại ánh xạ như sau

- f(1) = a, f(2) = b, f(3) = c, f(4) = d
- e1 – E1, e2 – E2, e3 – E3,…

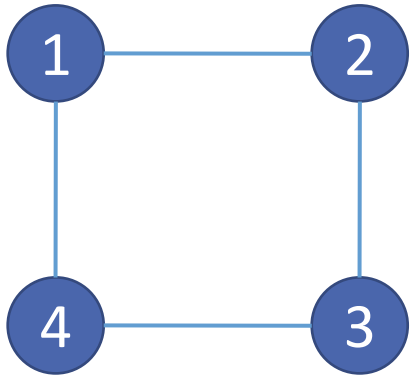
Chứng minh đồ thị đẳng cấu

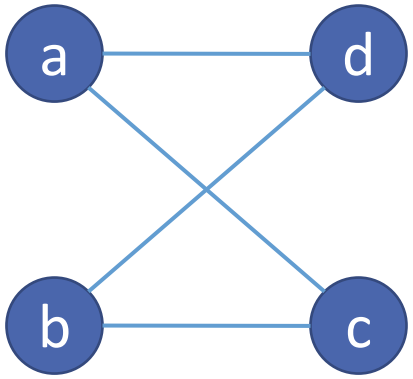
- Hai định lý sau đây giúp xác định sự đẳng cấu của hai đồ thị (đã gán nhãn).
- Định lý 1: Hai đồ thị là đẳng cấu với nhau khi và chỉ khi các đỉnh của chúng có thể được gán nhãn sao cho ma trận kề tương ứng là giống nhau.
- Định lý 2: Hai đồ thị $\boldsymbol{G_1}$ và $\boldsymbol{G_2}$, đã được gán nhãn và có hai ma trận kề tương ứng là $\boldsymbol{A_1}$ và $\boldsymbol{A_2}$, đẳng cấu với nhau khi và chỉ khi tồn tại ma trận hoán vị $\boldsymbol{P}$ sao cho $\boldsymbol{PA_1P^T = A_2}$.

Ghi chú: Ma trận hoán vị là một ma trận có được bằng cách hoán vị các hàng và/hoặc cột của một ma trận đơn vị n x n. Như vậy ma trận hoán vị là một ma trận vuông mà mỗi hàng và cột chỉ có 1 phần tử có giá trị '1', các phần tử còn lại có giá trị '0'.

Chứng minh đồ thị đẳng cấu

- Xét hai đồ thị G_1 và G_2 được gán nhãn như bên dưới và ma trận kề tương ứng.





	1	2	3	4
1	0	1	0	1
2	1	0	1	0
3	0	1	0	1
4	1	0	1	0

	a	b	c	d
a	0	0	1	1
b	0	0	1	1
c	1	1	0	0
d	1	1	0	0

Chứng minh đồ thị đẳng cấu

- Xét ma trận hoán vị $P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- Ta có:

$$PA_1P^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} = A_2$$

- Vậy $\boldsymbol{G_1}$ và $\boldsymbol{G_2}$ đẳng cấu với nhau theo ánh xạ: 1 $\rightarrow$ a, 2 $\rightarrow$ c, 3 $\rightarrow$ b và 4 $\rightarrow$ d.

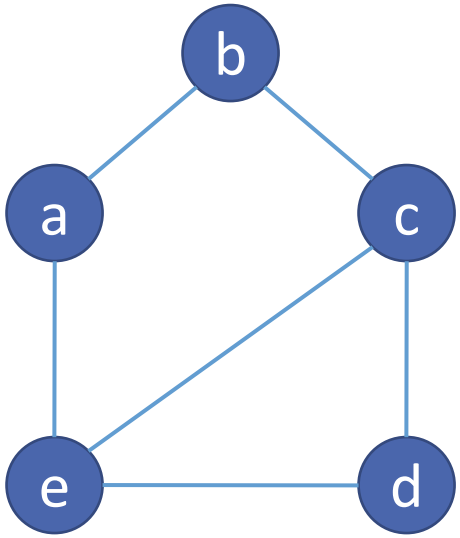
Xác định đồ thị đẳng cấu

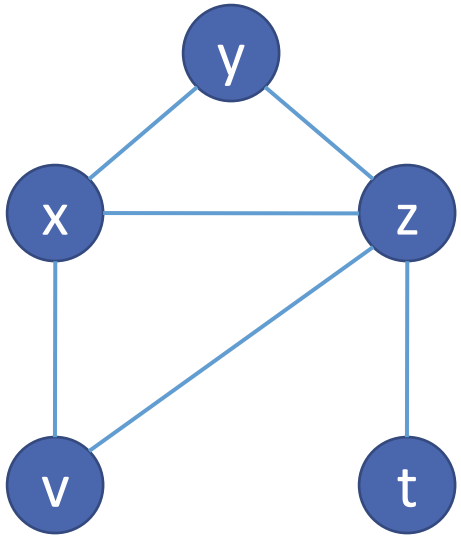
- Ta cần đưa ra một quan hệ tương đương (đẳng cấu) giữa hai đồ thị để chứng minh chúng đẳng cấu với nhau.
  - Cần phải chỉ ra ma trận hoán vị $P \rightarrow$ có đến $n!$ hoán vị khác nhau với đồ thị n đỉnh.
  - Do vậy, bài toán xác định đồ thị đẳng cấu có độ phức tạp rất lớn!

- Để chứng minh hai đồ thị không đẳng cấu với nhau, ta cần chỉ ra chúng không có chung một tính chất mà các đồ thị đẳng cấu phải có $\rightarrow$ đưa ra ví dụ phản chứng.

Ví dụ về đồ thị đẳng cấu

G_1

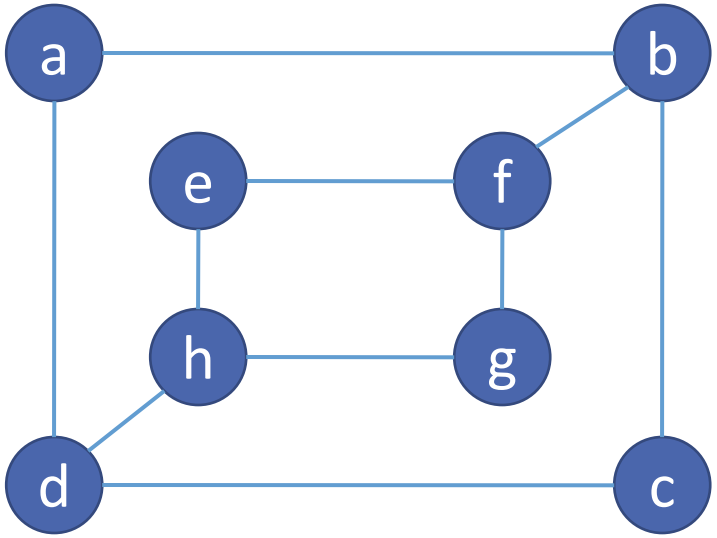




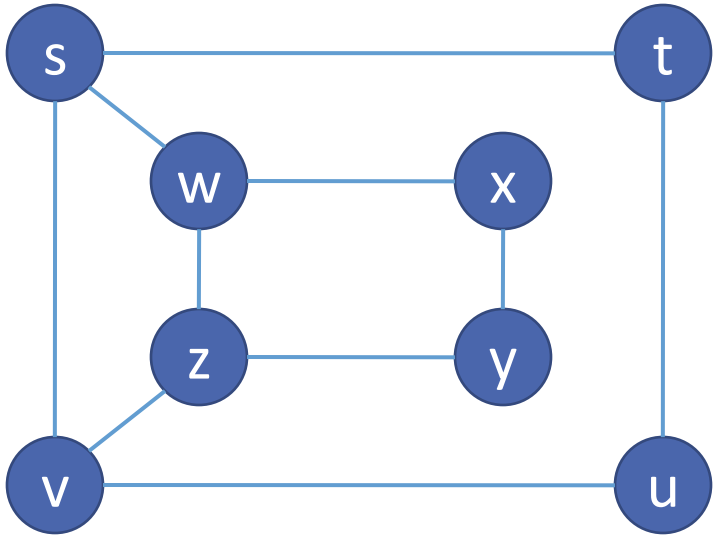
G_2

Hai đồ thị G_1 và G_2 không là đồ thị đẳng cấu vì tồn tại đỉnh trong G_2 có bậc 4 nhưng không tồn tại đỉnh bậc 4 trong G_1.

Ví dụ về đồ thị đẳng cấu



G_1



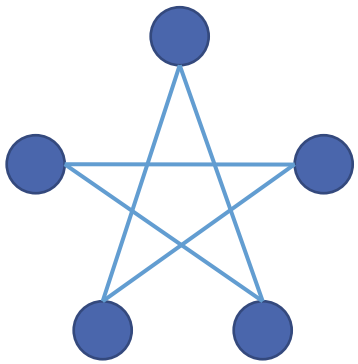
G_2

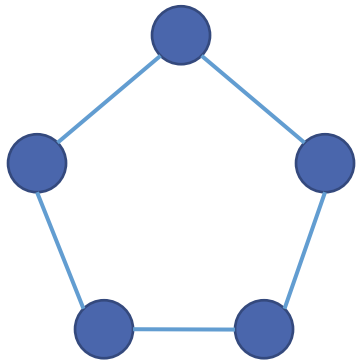
Hai đồ thị G_1 và G_2 không là đồ thị đẳng cấu vì đỉnh a có bậc 2, không kề với đỉnh bậc 2 nào, nhưng các đỉnh x, y, u, t – bậc 2 – đều có kề với đỉnh bậc 2.

Bài tập rèn luyện

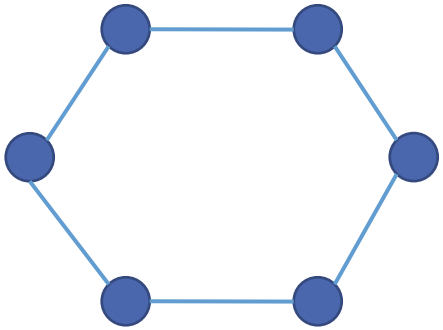
- Các cặp đồ thị sau đây có đẳng cấu hay không?

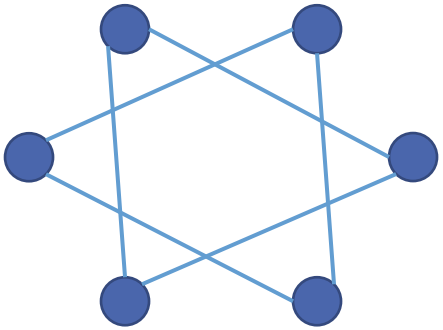
(a)





(b)

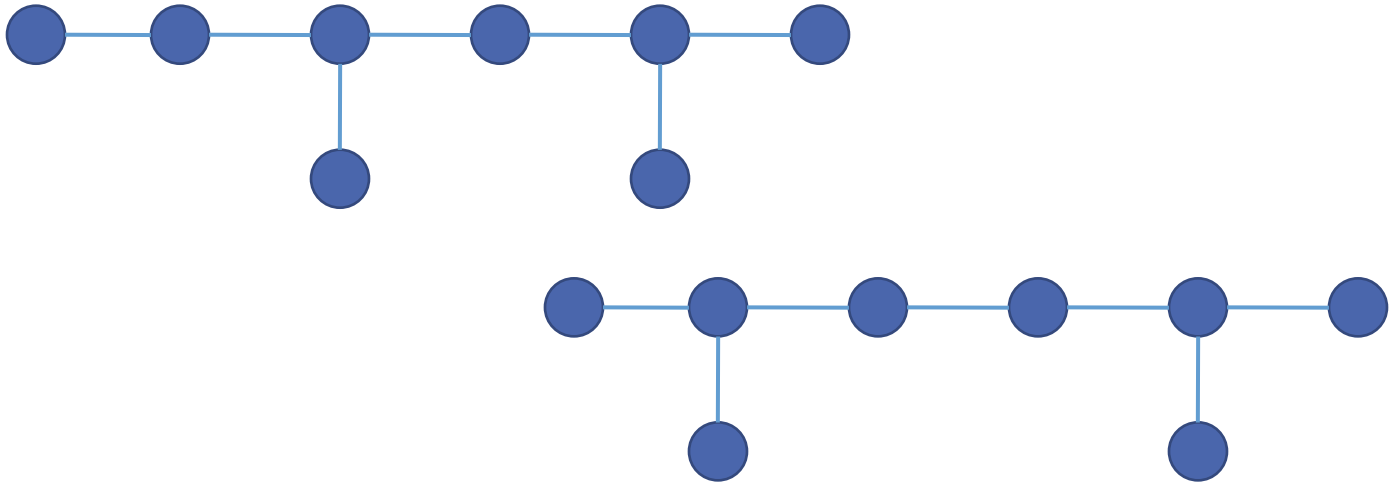




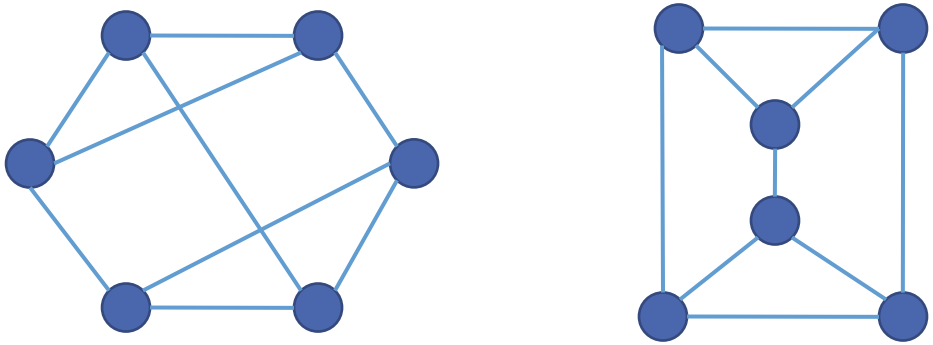
Bài tập rèn luyện

- Các cặp đồ thị sau đây có đẳng cấu hay không?

(c)



(d)

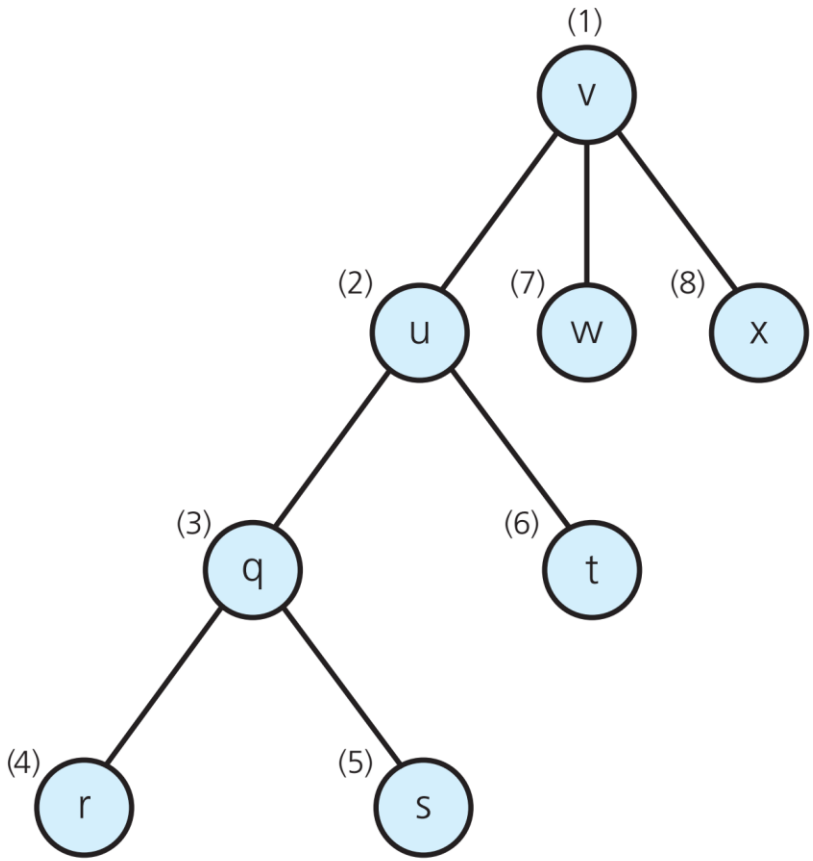


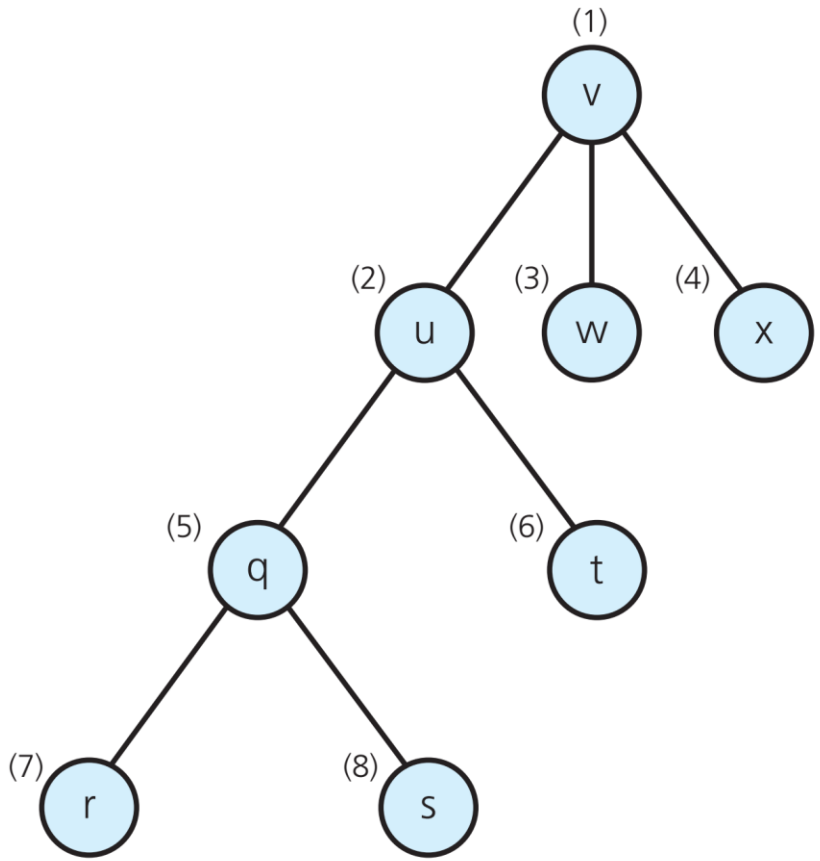
Các chiến lược duyệt đồ thị

Duyệt đồ thị

- Giải thuật duyệt đồ thị (graph traversal) đề ra một lộ trình để viếng thăm mọi đỉnh trong đồ thị hay trong cùng một thành phần liên thông.
  - Xuất phát tại một đỉnh v bất kỳ, giải thuật viếng thăm mọi đỉnh w mà có tồn tại đường đi từ v đến w.
- Gồm có: duyệt theo chiều sâu (depth-first search, DFS) and duyệt theo chiều rộng (breadth-first search, BFS)
  - Các giải thuật này áp dụng cho cả đồ thị có hướng và vô hướng.
  - Nếu xuất phát từ cùng một đỉnh, DFS và BFS sẽ viếng thăm cùng một tập hợp đỉnh nhưng theo những thứ tự khác nhau.

Duyệt đồ thị theo DFS và BFS





Thứ tự viếng thăm các đỉnh được thực hiện bởi DFS (trái) và BFS (phải).

Chiến lược duyệt đồ thị

- Nếu đồ thị liên thông, giải thuật duyệt đồ thị có thể viếng thăm mọi đỉnh trong đồ thị, bất kể đỉnh bắt đầu
- Ngược lại, giải thuật bắt đầu tại đỉnh v chỉ có thể viếng thăm các đỉnh cùng thành phần liên thông với v.
- Do đó, các chiến lược duyệt đồ thị được dùng để tìm thành phần liên thông trong đồ thị.
  - Bắt đầu từ đỉnh v bất kỳ, DFS/BFS thăm các đỉnh cho đến khi không thể đi tiếp được nữa. Khi đó, ta thu được một thành phần liên thông.
  - Lặp lại thao tác trên cho một thành phần liên thông mới.
- Đỉnh đã viếng thăm cần được ghi nhận lại để tránh đi trùng.

Duyệt theo chiều sâu (DFS)

- Từ một đỉnh v đã cho, DFS đi theo đường đi từ v sâu nhất có thể trước khi quay trở về.
  - Tức là, sau khi thăm một đỉnh, DFS sẽ viếng thăm một đỉnh khác chưa được viếng thăm, nếu có thể.
- Ý tưởng của DFS: viếng thăm sau cùng – khảo sát đầu tiên

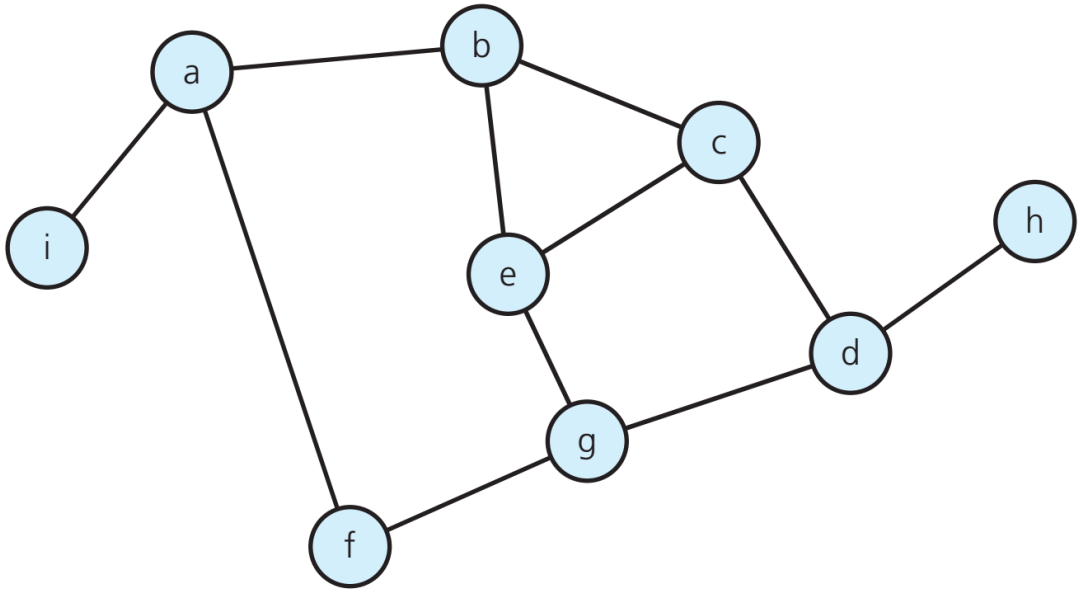
Giả sử ta bắt đầu duyệt tại đỉnh v.

```
dfs(v: Đỉnh)
    Đánh dấu v đã viếng thăm
    for (mỗi đỉnh chưa viếng thăm u kề với v)
        dfs(u)
```

```
dfs(v: Đỉnh)
    s = Ngăn xếp mới khởi tạo rỗng
    s.push(v)              // Đẩy v vào ngăn xếp và đánh dấu đã viếng thăm
     Đánh dấu v đã viếng thăm
    // Tồn tại đường đi từ v tại đáy ngăn xếp s đến đỉnh ở đầu ngăn xếp s
    while (!s.isEmpty()){
        if (không có đỉnh chưa viếng thăm nào kề với đỉnh ở đầu ngăn xếp)
            s.pop()        // Quay lui
        else{
            Chọn một đỉnh chưa viếng thăm u kề với đỉnh ở đầu ngăn xếp
            s.push(u)
            Đánh dấu u đã viếng thăm
        }
    }
```

Ví dụ duyệt theo chiều sâu

Đỉnh đã viếng thăm	Ngăn xếp (đáy → đỉnh)
a	a
b	a b
c	a b c
d	a b c d
g	a b c d g
e	a b c d g e
(backtrack)	a b c d g
f	a b c d g f
(backtrack)	a b c d g
(backtrack)	a b c d
h	a b c d h
(backtrack)	a b c d
(backtrack)	a b c
(backtrack)	a b
(backtrack)	a
i	a i
(backtrack)	a
(backtrack)	*(empty)*



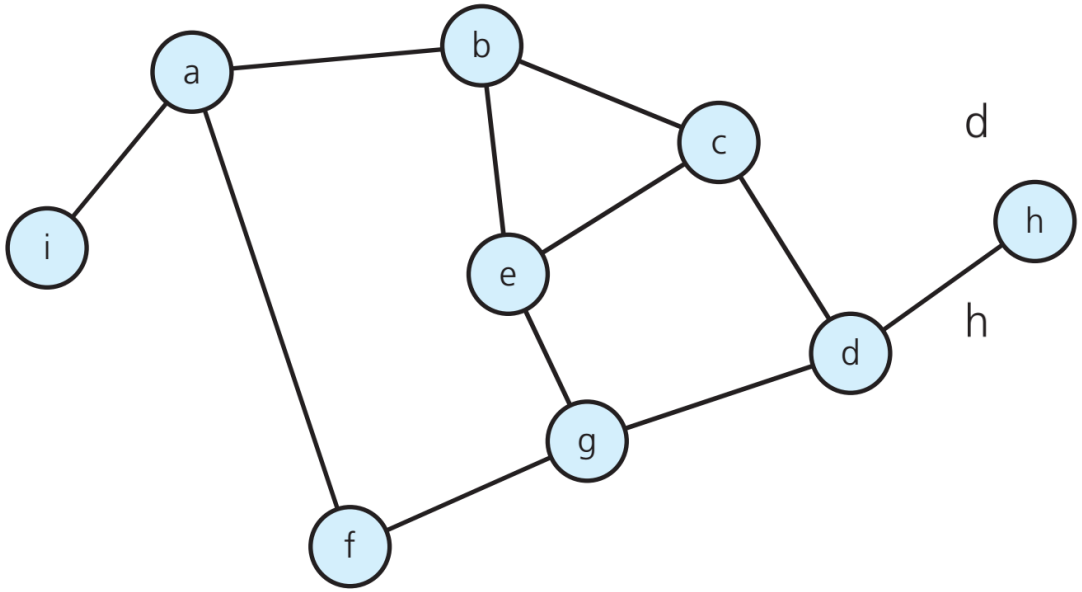
Duyệt theo chiều rộng (BFS)

- Sau khi viếng thăm một đỉnh v đã cho, BFS viếng thăm mọi đỉnh kề với v trước khi thăm viếng những đỉnh khác.
- Ý tưởng của BFS: viếng thăm đầu tiên – khảo sát đầu tiên

```
bfs(v: Đỉnh)
    q = Hàng đợi mới khởi tạo rỗng
    q.enqueue(v)        // Thêm v vào hàng đợi và đánh dấu
    Đánh dấu v đã viếng thăm
    while (!q.isEmpty()) {
        q.dequeue(w)
        for (mỗi đỉnh chưa viếng thăm u kề với w){
            Đánh dấu u đã viếng thăm
            q.enqueue(u)
        }
    }
```

Giả sử ta bắt đầu duyệt tại đỉnh v.

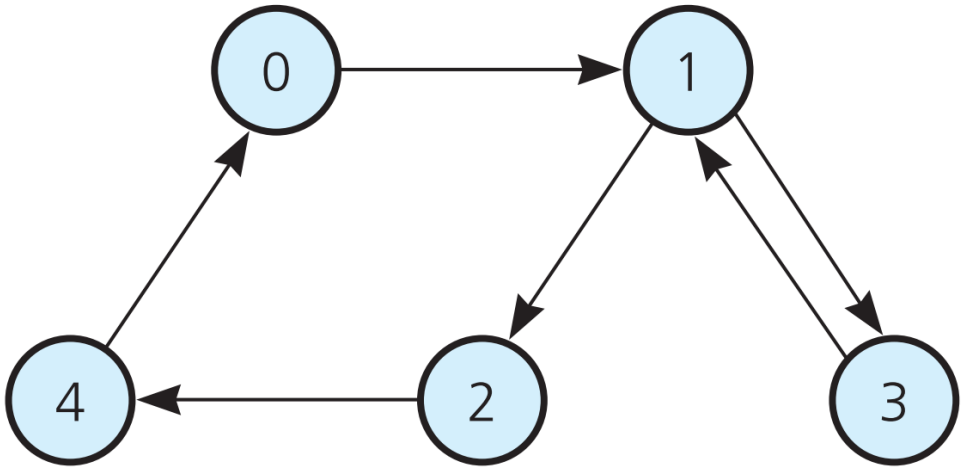
Ví dụ về duyệt theo chiều rộng



Đỉnh đã viếng thăm	Hàng đợi (đầu → cuối)
a	a
	(empty)
b	b
f	b f
i	b f i
	f i
c	f i c
e	f i c e
	i c e
g	i c e g
	c e g
	e g
d	e g d
	g d
	d
	(empty)
h	h
	(empty)

Bài tập rèn luyện

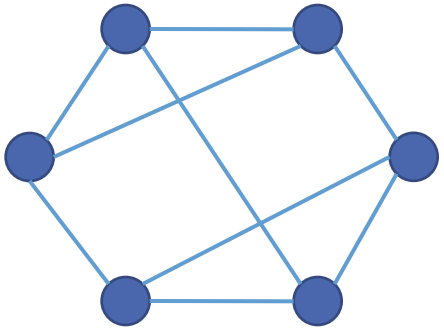
- Duyệt đồ thị lần lượt bằng chiến lược DFS và BFS, bắt đầu từ đỉnh 0.
- Liệt kê các đỉnh theo thứ tự mà phép duyệt ghé thăm chúng

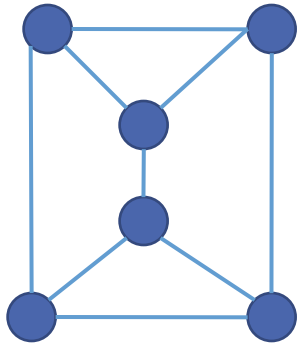


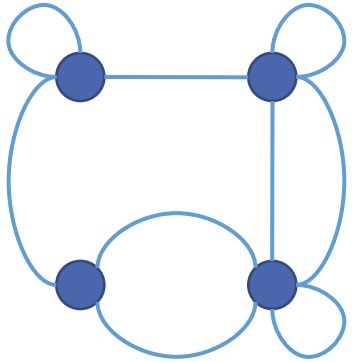
Tính liên thông của đồ thị

Đồ thị liên thông vô hướng

- Đồ thị liên thông vô hướng (connected graph) có đường đi giữa mọi cặp đỉnh phân biệt của đồ thị.
- Ngược lại, đồ thị này gọi là không liên thông (disconnected).

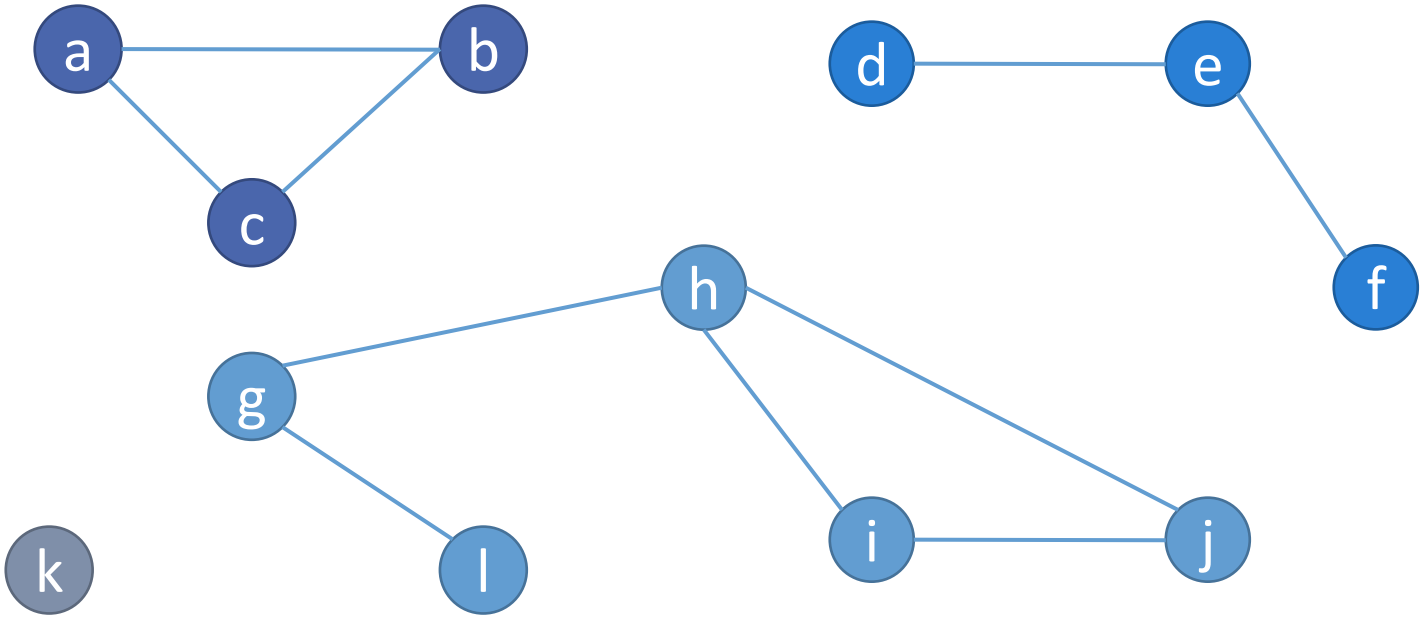






Thành phần liên thông

- Đồ thị không liên thông sẽ chứa nhiều đồ thị con liên thông.
- Các đồ thị con này gọi là thành phần liên thông (connected component).



Định lý về đường đi giữa hai đỉnh bậc lẻ

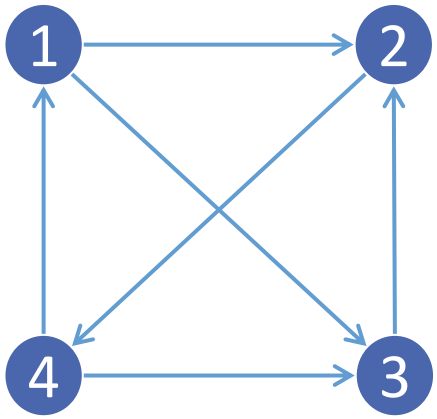
- Nếu đồ thị $\boldsymbol{G}$ (liên thông hoặc không liên thông) có đúng hai đỉnh bậc lẻ, chắc chắn sẽ có một đường đi nối hai đỉnh này.

- **Chứng minh**
  - TH1: G liên thông: rõ ràng có đường nối hai đỉnh bậc lẻ này (do định nghĩa của đồ thị liên thông).
  - TH2: G không liên thông: các thành phần liên thông của G là một đồ thị. Do đó, theo định lý về số đỉnh bậc lẻ, hai đỉnh này phải thuộc về cùng một thành phần liên thông. Do vậy, phải có đường nối.

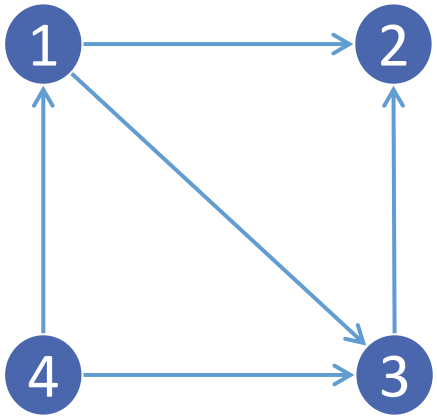
Đồ thị liên thông có hướng

- Đồ thị có hướng gọi là liên thông mạnh (strongly connected) nếu có đường đi từ a tới b và từ b tới a với mọi cặp đỉnh a và b của đồ thị.
- Đồ thị có hướng là liên thông yếu (weakly connected) nếu có đường đi giữa hai đỉnh bất kỳ của đồ thị vô hướng nền.
- Đồ thị có hướng gọi là liên thông một phần (unilaterally connected) nếu với mọi cặp đỉnh a và b bất kỳ, có ít nhất một đỉnh đến được đỉnh còn lại.

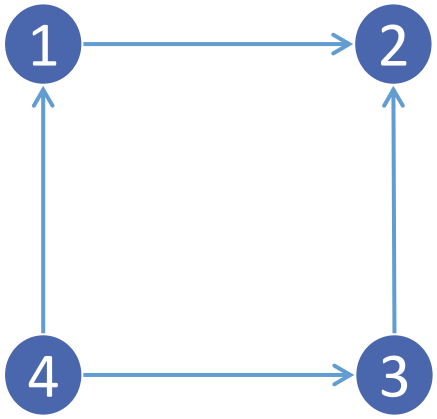
Đồ thị liên thông có hướng



Đồ thị liên thông mạnh



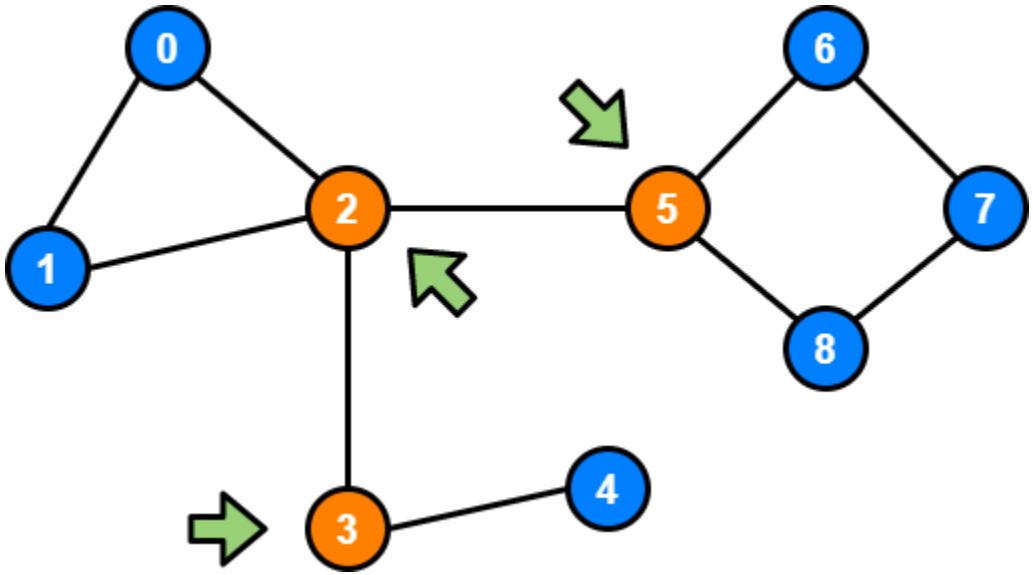
Đồ thị liên thông một phần



Đồ thị liên thông yếu

Đỉnh khớp

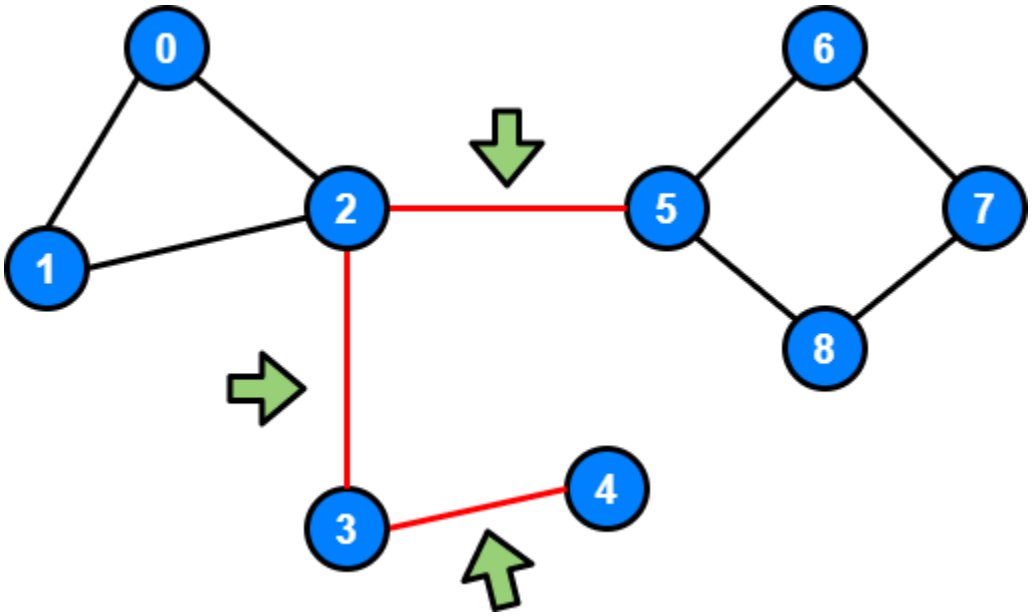
- Đỉnh khớp (cut vertex, articulation point) của một đồ thị vô hướng là đỉnh mà nếu xóa nó và các cạnh nối đến nó khỏi đồ thị thì số thành phần liên thông sẽ tăng thêm.



Nguồn ảnh: emre.me

Cạnh cầu

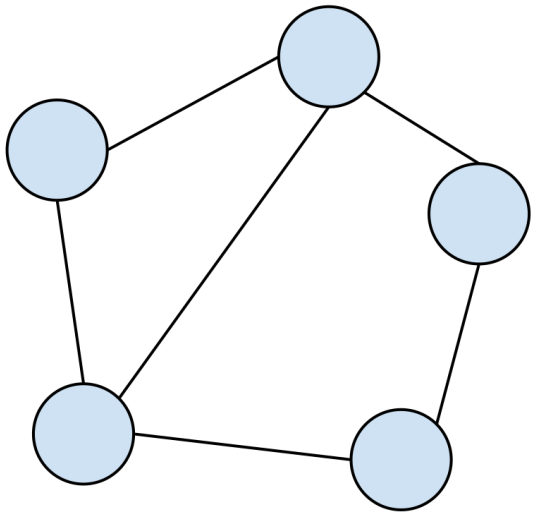
- Cạnh cầu (bridge) của một đồ thị vô hướng là cạnh mà nếu xóa đi khỏi đồ thị thì số thành phần liên thông sẽ tăng thêm.



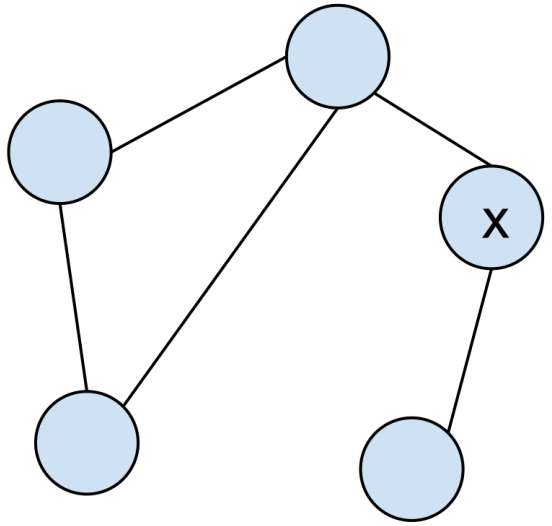
Nguồn ảnh: emre.me

Đồ thị song liên thông

- Đồ thị song liên thông (biconnectivity) là đồ thị **không** chứa đỉnh khớp.



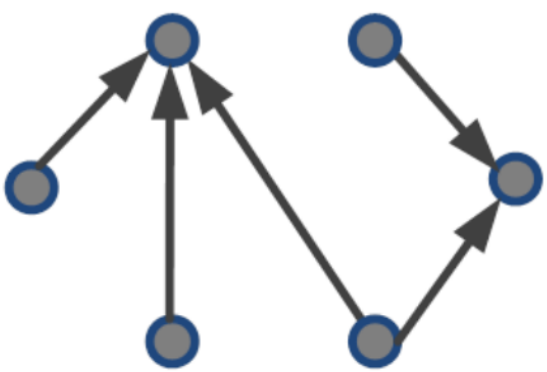
Đồ thị song liên thông có năm đỉnh và sáu cạnh



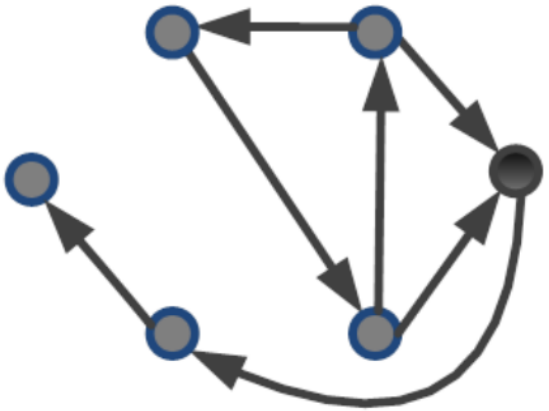
Đồ thị không song liên thông. Việc loại bỏ đỉnh x sẽ làm đồ thị mất kết nối.

Bài tập rèn luyện

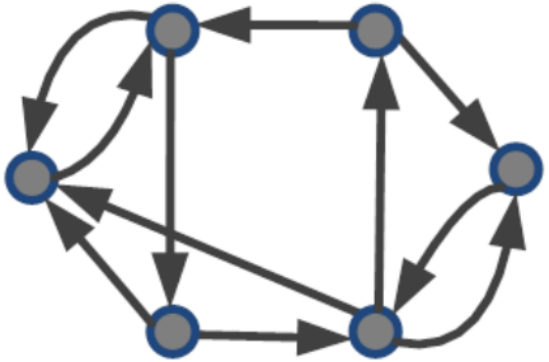
- Xác định loại liên thông cho những đồ thị có hướng sau.



(a)



(b)



(c)


...the end.