CHƯƠNG 1
CÁC KHÁI NIỆM CƠ BẢN

Bùi Tiến Lên

Đại học Khoa học Tự nhiên TPHCM

1/1/2018

Nội dung

KHÁI NIỆM ĐỒ THỊ

Đồ thị có hướng

Định nghĩa 1.1

Đồ thị G được gọi là **đồ thị có hướng** (directed graph) bao gồm

- Tập hợp các **đỉnh** (vertex) $V = \{v_0, v_1, ..., v_{n-1}\} \neq \emptyset$
- Tập hợp các **cạnh** (edge) $E = \{e_0, e_1, ..., e_{m-1}\}$
- Mỗi cạnh $e_k \in E$ được liên kết với một cặp đỉnh (v_i, v_j), v_i được gọi là *đỉnh đầu* và v_j được gọi là *đỉnh cuối*

Đồ thị được ký hiệu $G = (V, E)$

- Số lượng các đỉnh của V được gọi là **bậc** (order) của đồ thị G
- Số lượng các cạnh của E được gọi là **kích thước** (size) của đồ thị G

Đồ thị có hướng (cont.)

Lưu ý

Các đỉnh hoặc các cạnh của đồ thị có thể được

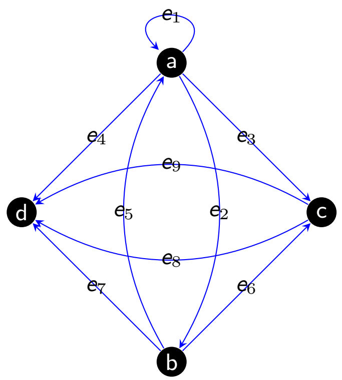
- Gán nhãn
- Gán màu
- Gán trọng số

Mục đích biểu diễn thông tin dữ liệu để *giải quyết một bài toán nào đó*

Đồ thị có hướng (cont.)

- **Cạnh khuyên** (loop edge) là cạnh có đỉnh cuối trùng với đỉnh đầu
- **Hai cạnh song song** (multiple edges) là hai cạnh có đỉnh đầu trùng nhau và đỉnh cuối trùng nhau

Đồ thị có hướng (cont.)



Hình 1.1: Đồ thị có hướng 4 đỉnh và 9 cạnh. Hãy xác định cạnh khuyên và song song.

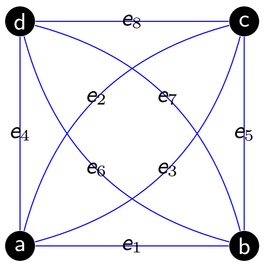
Đồ thị vô hướng

Định nghĩa 1.2

Đồ thị G được gọi là **đồ thị vô hướng** (undirected graph) được định nghĩa bởi

- Tập hợp các đỉnh $V \neq \emptyset$
- Tập hợp các cạnh E
- Mỗi cạnh $e \in E$ được liên kết với một cặp đỉnh $\{v_i, v_j\}$ không phân biệt thứ tự

Đồ thị vô hướng (cont.)



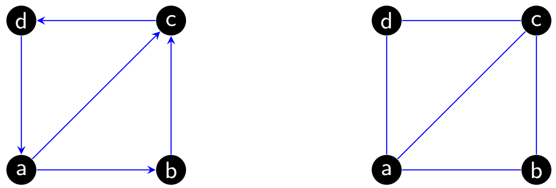
Hình 1.2: Đồ thị vô hướng 4 đỉnh và 8 cạnh. Hãy xác định các cạnh khuyên và song song

Đồ thị vô hướng (cont.)

Định nghĩa 1.3

Đồ thị vô hướng có được bằng cách loại bỏ hướng của các cạnh của đồ thị có hướng được gọi là **đồ thị nền** (**underlying undirected graph**)

Đồ thị vô hướng (cont.)



(a) đồ thị có hướng G **(b)** đồ thị vô hướng G

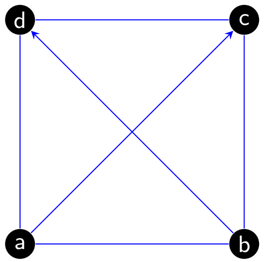
Hình 1.3: Đồ thị có hướng G và đồ thị nền vô hướng G'

Đồ thị hỗn hợp

Định nghĩa 1.4

Đồ thị hỗn hợp (**mixed graph**) là đồ thị có cả cạnh có hướng và cạnh vô hướng

Đồ thị hỗn hợp (cont.)



Hình 1.4: Đồ thị hỗn hợp 4 đỉnh và 6 cạnh

Quan hệ kề

Định nghĩa 1.5

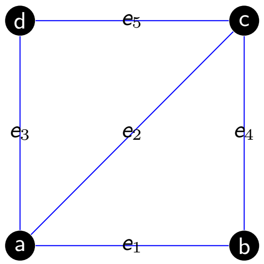
Cho một đồ thị $G = (V, E)$

Quan hệ giữa đỉnh và đỉnh Nếu hai đỉnh v_i và v_j được liên kết bằng một cạnh e thì hai đỉnh này được gọi là **kề** (**adjacent**) nhau

Quan hệ giữa cạnh và cạnh Nếu hai cạnh e_i và e_j có một đỉnh chung v thì hai cạnh này được gọi là **kề** (**adjacent**) nhau

Quan hệ giữa cạnh và đỉnh Khi một cạnh e là liên kết của cặp đỉnh (v_i, v_j) thì đỉnh v_i và v_j **kề, liên thuộc** (**incident**) với cạnh e, cạnh e **kề** với đỉnh v_i và đỉnh v_j

Quan hệ kề (cont.)



Hình 1.5: Đồ thị vô hướng 4 đỉnh và 5 cạnh. Hãy xác định mối quan hệ kề giữa đỉnh-đỉnh, cạnh-cạnh và liên thuộc giữa đỉnh-cạnh

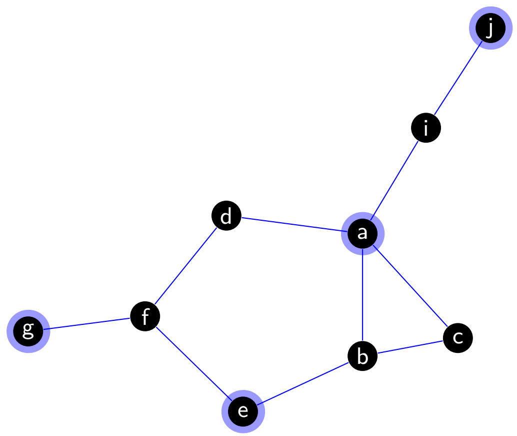
Quan hệ kề (cont.)

Định nghĩa 1.6

Cho một đồ thị $G = (V, E)$

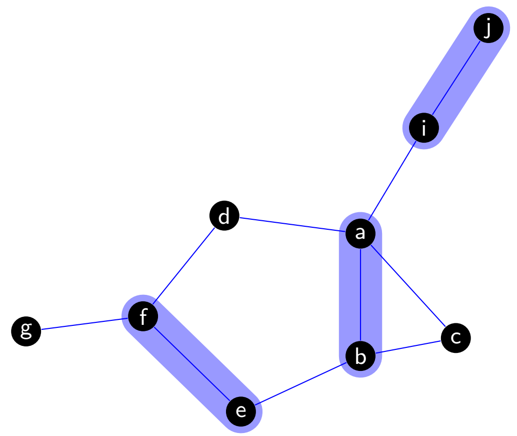
- Tập hợp $V' \subset V$ sao cho các đỉnh không kề nhau được gọi là **tập đỉnh độc lập** (**independent vertex set** or **stable set**)
- Tập hợp $E' \subset E$ sao cho các cạnh không kề nhau được gọi là **tập cạnh độc lập** (**independent edge set** or **matching set**)
- Tập hợp $V' \subset V$ sao cho các đỉnh đôi một kề nhau được gọi là **nhóm** (**clique**)

Quan hệ kề (cont.)



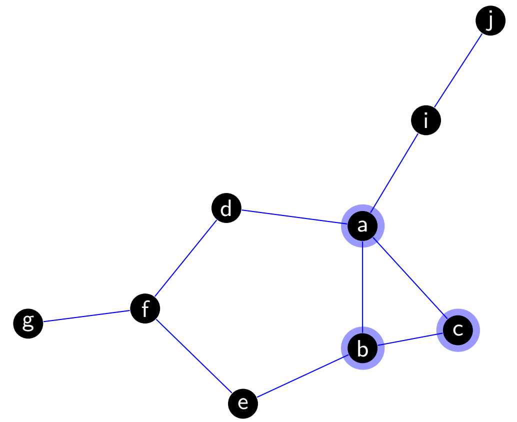
Hình 1.6: Các đỉnh {a, e, g, j} độc lập nhau

Quan hệ kề (cont.)



Hình 1.7: Các cạnh {ab, ef, ij} độc lập nhau

Quan hệ kề (cont.)



Hình 1.8: Các đỉnh {a, b, c} tạo thành một nhóm

KHÁI NIỆM BẬC CỦA ĐỈNH

Bậc của đỉnh

Định nghĩa 1.7

Cho một đồ thị G **bậc** (**degree**) của một đỉnh v của đồ thị là tổng số các cạnh kề với đỉnh v (qui ước mỗi cạnh khuyên được tính 2 lần). Bậc của đồ thị ký hiệu $deg(v)$ hoặc $d(v)$

- **Bậc cực đại** (**maximum degree**) của đồ thị G (ký hiệu là $\Delta(G)$) là giá trị lớn nhất của bậc của các đỉnh của đồ thị G.
- **Bậc cực tiểu** (**maximum degree**) của đồ thị G (ký hiệu là $\delta(G)$) là giá trị nhỏ nhất của bậc của các đỉnh của đồ thị G

Bậc của đỉnh (cont.)

Định nghĩa 1.8

Cho một đồ thị có hướng G và đỉnh v của đồ thị

- **Nửa bậc ngoài** (**out-degree**) của đỉnh v là số cạnh đi ra khỏi đỉnh v và ký hiệu là $d^+(v)$
- **Nửa bậc trong** (**in-degree**) của đỉnh v là số cạnh đi vào đỉnh v và ký hiệu là $d^-(v)$

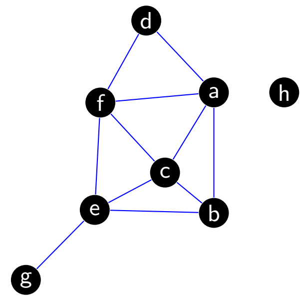
Bậc của đỉnh (cont.)

Định nghĩa 1.9

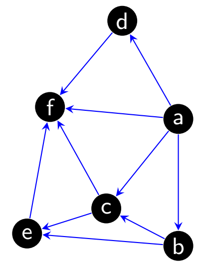
Cho một đồ thị có hướng G

- **Đỉnh cô lập** (**isolated vertex**) là đỉnh có bậc bằng 0
- **Đỉnh treo** (**pendant vertex**) là đỉnh có bậc bằng 1
- **Cạnh treo** (**pendant edge**) là cạnh kề với đỉnh treo

Bậc của đỉnh (cont.)



(a) Đồ thị vô hướng



(b) Đồ thị có hướng

Hình 1.9: Hãy xác định bậc của các đỉnh của các đồ thị

Những định lý về bậc của đỉnh

Định lý 1.1 (Định lý thứ nhất - định lý bắt tay)

Trong một đồ thị $G = (V, E)$, tổng của các bậc của các đỉnh bằng hai lần tổng số cạnh

$$\sum_{v \in V} d(v) = 2|E| \tag{1.1}$$

Chứng minh

- Nhận thấy mỗi cạnh $e = (u, v)$ được tính một lần trong $d(u)$ và một lần trong $d(v)$.
- Từ đó suy ra tổng tất cả các bậc của các đỉnh bằng hai lần số cạnh

■

Những định lý về bậc của đỉnh (cont.)

Hệ quả 1.1

Số đỉnh bậc lẻ của một đồ thị là một số chẵn

Chứng minh

- Gọi V_{even} và V_{odd} tương ứng là tập các đỉnh bậc chẵn và tập các đỉnh bậc lẻ của đồ thị $G = (V, E)$. Khi đó

$$2|E| = \sum_{v \in V_{even}} d(v) + \sum_{v \in V_{odd}} d(v) \tag{1.2}$$

- Vế trái là một số chẵn và tổng thứ nhất cũng là một số chẵn nên tổng thứ hai là một số chẵn. Vì $d(v)$ là lẻ với $v \in V_{odd}$ cho nên số phần tử của V_{odd} phải là một số chẵn

■

Những định lý về bậc của đỉnh (cont.)

Định lý 1.2

Cho một đồ thị có hướng $G = (V, E)$ ta có công thức sau

$$\sum_{v \in V} d^+(v) = \sum_{v \in V} d^-(v) = |E| \tag{1.3}$$

Chứng minh

Sinh viên tự chứng minh ■

Những định lý về bậc của đỉnh (cont.)

Định lý 1.3

Cho một đồ thị đơn có số đỉnh $n \geq 2$ luôn tồn tại ít nhất hai đỉnh có cùng bậc

Chứng minh

Sinh viên tự chứng minh ■

Những định lý về bậc của đỉnh (cont.)

Định lý 1.4

Cho một đồ thị đơn có số đỉnh $n \geq 3$ có đúng hai đỉnh cùng bậc thì hai đỉnh này không thể đồng thời có bậc 0 hoặc bậc $n-1$

Chứng minh

Sinh viên tự chứng minh ■

Những định lý về bậc của đỉnh (cont.)

Định nghĩa 1.10

Một dãy số nguyên không âm $\{d_1, d_2, ..., d_n\}$ được gọi là **khả đồ thị** (**graphical**) nếu tồn tại một đồ thị G sao cho dãy số này là bậc của các đỉnh của đồ thị

Những định lý về bậc của đỉnh (cont.)

Định lý 1.5 (Định lý Havel & Hakimi)

Một dãy n số nguyên không âm và không tăng

$$\{d_1, d_2, ..., d_n\}$$

với $d_1 \geq d_2 \geq ... \geq d_n$ 0, $n \geq 2$, $d_1 \geq 1$ và $d_{d_1+1} \geq 1$ là khả đồ thị nếu và chỉ nếu dãy $n-1$ số nguyên sau

$$\{d_2 - 1, ..., d_{d_1+1} - 1, d_{d_1+2}, ..., d_n\}$$

là khả đồ thị

Chứng minh

Sinh viên tự chứng minh ■

Những định lý về bậc của đỉnh (cont.)

Định lý 1.6 (Định lý Erdos & Gallai)

Một dãy n số nguyên dương và không tăng

$$\{d_1, d_2, ..., d_n\}$$

là dãy bậc của một đồ thị đơn nếu và chỉ nếu dãy nó thỏa

$$\sum_{i=1}^{k} d_i \leq k(k+1) + \sum_{j=k+1}^{n} \min\{k, d_j\}$$

với mọi $k = 1, ..., n-1$

Chứng minh

Sinh viên tự chứng minh ■

MỘT SỐ DẠNG ĐỒ THỊ

Giới thiệu

Đồ thị có thể là **đồ thị vô hạn** (**infinite graph**) hoặc **đồ thị hữu hạn** (**finite graph**). Trong môn học này chúng ta chỉ xem xét đồ thị hữu hạn. Có nhiều dạng đồ thị hữu hạn

- Đồ thị đơn
- Đồ thị rỗng
- Đồ thị đều
- Đồ thị đầy đủ
- Đồ thị phân đôi
- Đồ thị phân đôi, đủ
- Đồ thị vòng
- Đồ thị sao
- Đồ thị bánh xe
- Đồ thị n-lập phương

Đồ thị đơn

Định nghĩa 1.11

- **Đồ thị đơn** (**simple graph**) là đồ thị không có cạnh khuyên và không có cạnh song song
- **Đa đồ thị** (**multigraph**) là đồ thị có thể có cạnh khuyên và cạnh song song

Đồ thị rỗng

Định nghĩa 1.12

Đồ thị rỗng (**null graph**) là đồ thị có tập cạnh là tập rỗng

(a) đồ thị rỗng 3 đỉnh **(b)** đồ thị rỗng 5 đỉnh

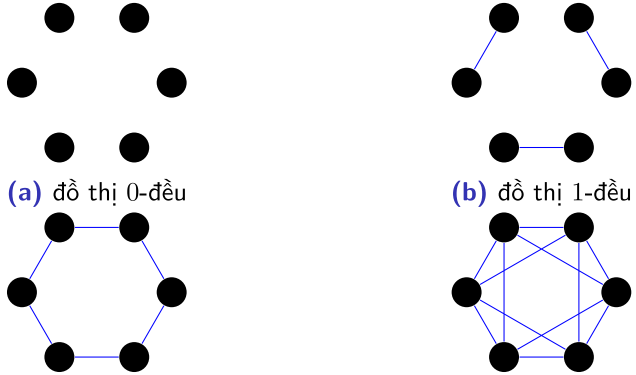
Hình 1.10: Các đồ thị rỗng

Đồ thị đều

Định nghĩa 1.13

Đồ thị đều (regular graph) là đồ thị đơn có các đỉnh cùng bậc. Gọi k là bậc của các đỉnh thì đồ thị được gọi là k-đều

Đồ thị đều (cont.)



(a) đồ thị 0-đều
(b) đồ thị 1-đều
(c) đồ thị 2-đều
(d) đồ thị 4-đều

Hình 1.11: Các kiểu đồ thị k-đều

Đồ thị đều (cont.)

Tính chất 1.1

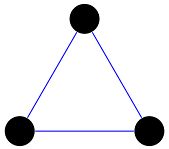
Đồ thị k-đều có n đỉnh thì sẽ có $\frac{n.k}{2}$ *cạnh*

Đồ thị đầy đủ

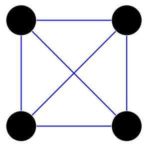
Định nghĩa 1.14

Đồ thị đầy đủ (complete graph) là đồ thị đơn mà giữa hai đỉnh bất kỳ đều có một cạnh nối chúng. Đồ thị đủ có n đỉnh được ký hiệu là K_n.

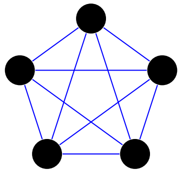
Đồ thị đầy đủ (cont.)



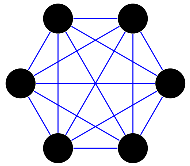
(a) đồ thị K_3



(b) đồ thị K_4



(c) đồ thị K_5



(d) đồ thị K_6

Hình 1.12: Các kiểu đồ thị K_n

Đồ thị đầy đủ (cont.)

Tính chất 1.2

- *Đồ thị đầy đủ K_n sẽ có $\frac{n(n-1)}{2}$ cạnh*
- *Đồ thị đơn với n đỉnh thì sẽ có tối đa là $\frac{n(n-1)}{2}$ cạnh*

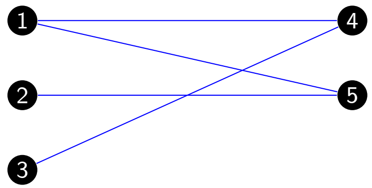
Đồ thị phân đôi

Định nghĩa 1.15

Cho một đồ thị $G = (V, E)$ là một đồ thị đơn, đồ thị G được gọi là **đồ thị phân đôi** (**bipartite graph**) nếu tập V được chia thành hai tập con V_1 và V_2 sao cho

- Hai tập con V_1 và V_2 là phân hoạch của V nghĩa là $V_1 \neq \emptyset, V_2 \neq \emptyset, V_1 \cup V_2 = V, V_1 \cap V_2 = \emptyset$
- Hai đỉnh bất kỳ của V_1 không kề nhau, và hai đỉnh bất kỳ của V_2 không kề nhau

Đồ thị phân đôi (cont.)



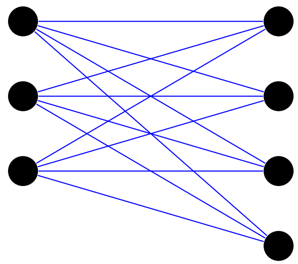
Hình 1.13: Đồ thị phân đôi

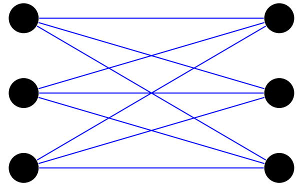
Đồ thị phân đôi, đủ

Định nghĩa 1.16

Cho $G = (V, E)$ là một đồ thị phân đôi với hai tập con V_1 và V_2, đồ thị G được gọi là **đồ thị phân đôi, đủ** nếu với mọi cặp đỉnh $x \in V_1$ và $y \in V_2$ thì có đúng một cạnh nối chúng. Đồ thị phân đôi, đủ được ký hiệu là $K_{m,n}$ với $|V_1| = m$ và $|V_2| = n$

Đồ thị phân đôi, đủ (cont.)


(a) đồ thị $K_{3,4}$


(b) đồ thị $K_{3,3}$

Hình 1.14: Các kiểu đồ thị $K_{m,n}$

Đồ thị phân đôi, đủ (cont.)

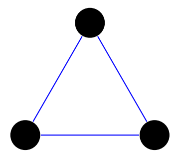
Tính chất 1.3

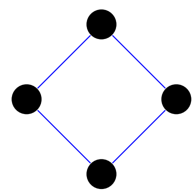
Đồ thị phân đôi, đủ $K_{m,n}$ sẽ có $m.n$ cạnh

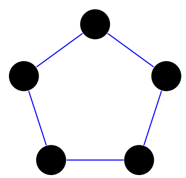
Đồ thị vòng

Định nghĩa 1.17

Đồ thị vòng (circular graph) là đồ thị đơn có n đỉnh $\{v_1, ..., v_n\}$ và n cạnh $\{(v_1, v_2), (v_2, v_3), ..., (v_{n-1}, v_n), (v_n, v_1)\}$. Đồ thị vòng với n đỉnh được ký hiệu là C_n


(a) Đồ thị C_3


(b) Đồ thị C_4

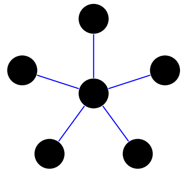

(c) Đồ thị C_5

Hình 1.15: Các kiểu đồ thị vòng

Đồ thị sao

Định nghĩa 1.18

Đồ thị sao (star graph) là đồ thị đơn có $n+1$ đỉnh. Trong đó có duy nhất một đỉnh có bậc n và các đỉnh còn lại có bậc là 1

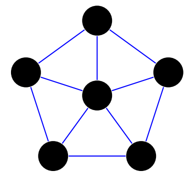


Hình 1.16: Đồ thị sao

Đồ thị bánh xe

Định nghĩa 1.19

Đồ thị bánh xe (wheel graph) là đồ thị đơn có $n+1$ đỉnh được tạo từ đồ thị C_n và một đỉnh nối với tất cả các đỉnh của C_n. Đồ thị bánh xe được ký hiệu là W_n



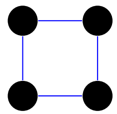
Hình 1.17: Đồ thị bánh xe W_5

Đồ thị n-lập phương

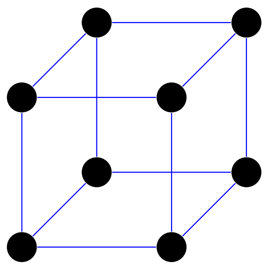
Định nghĩa 1.20

Đồ thị n-lập phương (n-cube graph) là đồ thị đơn có 2^n đỉnh. Mỗi đỉnh được biểu diễn bằng một dãy n số nhị phân. Hai đỉnh là kề nhau nếu dãy nhị phân của chúng khác nhau đúng 1 bit.
Đồ thị n-lập phương ký hiệu là Q_n

Đồ thị n-lập phương (cont.)

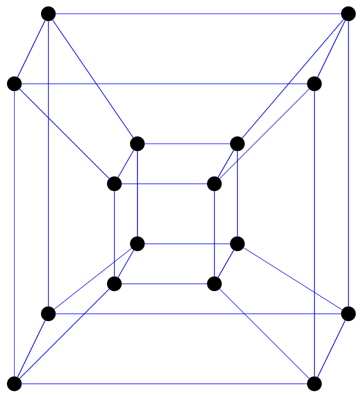


Hình 1.18: Đồ thị Q_2



Hình 1.19: Đồ thị Q_3

Đồ thị n-lập phương (cont.)



Hình 1.20: Đồ thị Q_4

Đồ thị n-lập phương (cont.)

Tính chất 1.4

- *Bậc của các đỉnh của đồ thị Q_n là n*
- *Số các cạnh của đồ thị Q_n là $n * 2^{n-1}$*

Các thao tác trên đồ thị

Cho đồ thị $G = (V, E)$ với V là tập đỉnh và E là tập cạnh

- Thêm đỉnh v vào đồ thị G
$$V = V + \{v\}$$
- Xóa đỉnh v khỏi đồ thị G
$$V = V - \{v\}$$
sau đó xóa các cạnh kề với v khỏi G
- Thêm cạnh $e = (x, y)$ vào đồ thị G
$$E = E + \{e\}$$
sau đó thêm hai đỉnh x, y vào đồ thị G
- Xóa cạnh e khỏi đồ thị G
$$E = E - \{e\}$$

Các thao tác trên đồ thị (cont.)

Định nghĩa 1.21

Cho hai đồ thị $G_1 = (V_1, E_1)$ và $G_2 = (V_2, E_2)$

- Phép hợp
$$G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$$
- Phép giao
$$G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2)$$

Các thao tác trên đồ thị (cont.)

Định nghĩa 1.22

Cho hai đồ thị $G = (V, E)$ và $G' = (V', E')$. Tích Cartesian $G \times G'$ được xác định như sau

- Tập đỉnh $V \times V'$
- Tập cạnh: hai đỉnh $(u, u') \in V \times V'$ và $(v, v') \in V \times V'$ được xem là kề nhau nếu
  - $u = v$ và u' và v' kề nhau
  - $u' = v'$ và u và v kề nhau

CÁC QUAN HỆ GIỮA ĐỒ THỊ

Giới thiệu

- Quan hệ con
- Quan hệ sinh
- Quan hệ bộ phận
- Quan hệ đẳng cấu
- Quan hệ bù

Quan hệ con

Định nghĩa 1.23

Cho hai đồ thị $G = (V, E)$ và $G' = (V', E')$. Ta nói đồ thị G' là **đồ thị con** (**subgraph**) của đồ thị G nếu và chỉ nếu $V' \subseteq V$ và $E' \subseteq E$

Ký hiệu $G' \subseteq G$

Quan hệ bộ phận

Định nghĩa 1.24

Cho hai đồ thị $G = (V, E)$ và $G' = (V', E')$. Đồ thị G' là đồ thị con của đồ thị G. Đồ thị G' được gọi là **đồ thị bộ phận** (**spanning subgraph**) của đồ thị G nếu và chỉ nếu $V' = V$

Quan hệ sinh

Định nghĩa 1.25

Cho hai đồ thị $G = (V, E)$ và $G' = (V', E')$. Đồ thị G' là đồ thị con của đồ thị G. Đồ thị G' được gọi là **đồ thị sinh** (**induced subgraph**) của đồ thị G nếu và chỉ nếu $V' \subseteq V$ và $e = (x, y) \in E, x, y \in V' \Rightarrow e \in E'$

Quan hệ đẳng cấu

Định nghĩa 1.26

Cho hai đồ thị $G = (V, E)$ và $G' = (V', E')$. Hai đồ thị được gọi là **đẳng cấu** (**isomorphic**) nếu và chỉ nếu tồn tại song ánh

$$f : \quad V \quad \rightarrow \quad V' \tag{1.4}$$

bảo toàn liên kết cạnh giữa E và E'

Quan hệ đẳng cấu (cont.)

Định lý 1.7

Điều kiện cần để hai đồ thị G và G' đẳng cấu là chúng phải có

- *Số đỉnh bằng nhau*
- *Số cạnh bằng nhau*
- *Bậc của các đỉnh tương ứng bằng nhau*

Chứng minh

Sinh viên tự chứng minh ∎

Quan hệ bù

Định nghĩa 1.27

Cho hai đồ thị $G = (V, E)$ và $G' = (V', E')$. Hai đồ thị được gọi là **bù nhau** (**complement**) nếu và chỉ nếu
$V = V', E \cap E' = \emptyset, G \cup G = K_n$ với n là số đỉnh của hai đồ thị
Ký hiệu $G' = \bar{G}$

Định nghĩa 1.28

Cho đồ thị $G = (V, E)$, đồ thị G được gọi là **tự bù** (**self complement**) nếu và chỉ nếu G đẳng cấu với $\bar{G}$

BIỂU DIỄN ĐỒ THỊ

Biểu diễn đồ thị bằng hình học

Biểu diễn đồ thị $G = (V, E)$ bằng hình học như sau

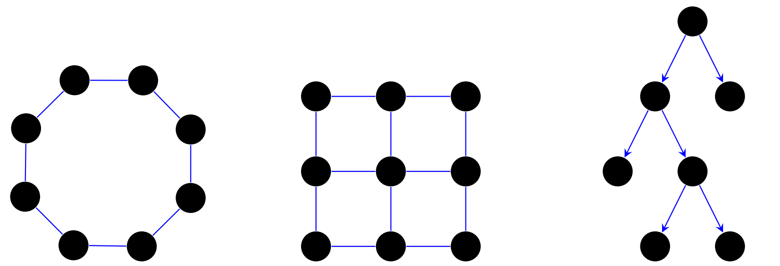
- Mỗi đỉnh $v \in V$ của đồ thị được biểu diễn bằng điểm hoặc hình tròn hoặc hình chữ nhật
- Mỗi cạnh $e \in E$ của đồ thị được biểu diễn bằng đoạn thẳng hoặc cung
- Nếu đồ thị có hướng thì mỗi đoạn thẳng hoặc cung sẽ có thêm dấu mũi tên
- Bước quan trọng nhất là **vẽ đồ thị** (**graph drawing**)

Biểu diễn đồ thị bằng hình học (cont.)

Có nhiều cách vẽ đồ thị. Trong đó có những cách vẽ phổ biến là

- Vẽ ngẫu nhiên (**random layout**)
- Vẽ dạng hình tròn (**circular layout**)
- Vẽ dạng lưới (**grid layout**)
- Vẽ dạng hình cây (**tree layout**)
- Vẽ dạng phẳng (**planar layout**)

Biểu diễn đồ thị bằng hình học (cont.)



(a) dạng hình tròn (b) dạng hình lưới (c) dạng hình cây

Hình 1.21: Một số kiểu vẽ đồ thị

Biểu diễn đồ thị bằng ma trận

Biểu diễn 1

Cho một đồ thị không có cạnh song song $G = (V, E)$ có n đỉnh $\{v_1, v_2, ..., v_n\}$ ta có thể biểu diễn đồ thị bằng **ma trận kề** (**adjacency matrix**) là một ma trận vuông A cấp n

$$A = (a_{ij})$$
$$a_{ij} = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & (v_i, v_j) \notin E \end{cases} \tag{1.5}$$

Biểu diễn đồ thị bằng ma trận (cont.)

Nhận xét

- Ma trận kề của đồ thị phụ thuộc vào thứ tự liệt kê các đỉnh. Ta có $n!$ cách liệt kê các đỉnh do đó sẽ có $n!$ ma trận kề khác nhau cho một đồ thị
- Ma trận kề của đồ thị vô hướng là một ma trận đối xứng

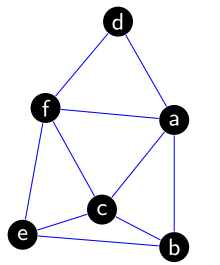
Biểu diễn đồ thị bằng ma trận (cont.)

Biểu diễn 2

Cho một đồ thị $G = (V, E)$ có n đỉnh $\{v_1, v_2, ..., v_n\}$ và có m cạnh $\{e_1, e_2, ..., e_m\}$ ta có thể biểu diễn đồ thị bằng **ma trận liên thuộc** (**incidence matrix**) là một ma trận $n \times m$

$$A = (a_{ij})$$
$$a_{ij} = \begin{cases} 1 & v_i \text{ kề } e_j \\ 0 & v_i \text{ không kề } e_j \end{cases} \tag{1.6}$$

Biểu diễn đồ thị bằng ma trận (cont.)



(a) đồ thị

$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

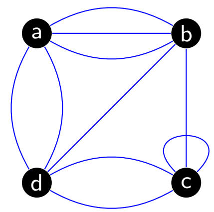
(b) ma trận kề

Hình 1.22: Biểu diễn đồ thị bằng ma trận kề

Biểu diễn đồ thị bằng ma trận (cont.)

Nhận xét

Ma trận kề cũng có thể dùng để biểu diễn đồ thị vô hướng có cạnh khuyên và cạnh song song



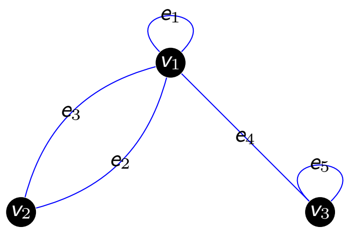
(a) đa đồ thị

$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{pmatrix} \end{array}$$

(b) ma trận kề

Hình 1.23: Biểu diễn đa đồ thị bằng ma trận kề

Biểu diễn đồ thị bằng ma trận (cont.)



(a) đồ thị

$$A = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

(b) ma trận liên thuộc

Hình 1.24: Biểu diễn đồ thị bằng ma trận liên thuộc

Đồ thị đẳng cấu và Ma trận kề

Định lý 1.8

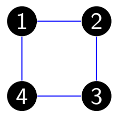
Hai đồ thị G_1 và G_2 với hai ma trận kề tương ứng A_1 và A_2 tương ứng đẳng cấu với nhau khi và chỉ khi tồn tại một ***ma trận hoán vị*** *(**permuation matrix**) P sao cho*

$$P.A_1.P^T = A_2 \tag{1.7}$$

Lưu ý

Ma trận hoán vị là ma trận mà mọi dòng, mọi cột chỉ có một phần tử "1", còn lại là phần tử "0"

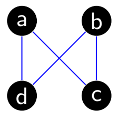
Đồ thị đẳng cấu và Ma trận kề (cont.)



(a) đồ thị G_1

$$A_1 = \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

(b) ma trận kề A_1



(c) đồ thị G_2

$$A_2 = \begin{array}{c c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

(d) ma trận kề A_2

Hình 1.25: Đồ thị và ma trận kề

Đồ thị đẳng cấu và Ma trận kề (cont.)

Xét ma trận hoán vị

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ta có

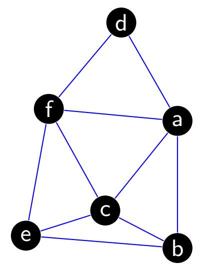
$$P.A_1.P^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = A_2$$

Vậy đồ thị G_1 và G_2 đẳng cấu với nhau

Biểu diễn đồ thị bằng danh sách

- Đồ thị $G = (V, E)$ có thể được biểu diễn bằng **danh sách cạnh** và **danh sách đỉnh**
- Đồ thị $G = (V, E)$ có thể được biểu diễn bằng **danh sách kề**

Biểu diễn đồ thị bằng danh sách (cont.)



(a) đồ thị

a
b
c
d
e
f

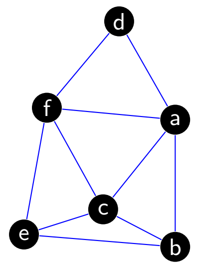
(b) danh sách đỉnh

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

(c) danh sách cạnh

Hình 1.26: Biểu diễn đồ thị bằng danh sách cạnh & danh sách đỉnh

Biểu diễn đồ thị bằng danh sách (cont.)



(a) đồ thị

đỉnh	các đỉnh kề
a	b c d f
b	a c e
c	a b e f
d	a f
e	b c f
f	a c d e

(b) danh sách kề

Hình 1.27: Biểu diễn đồ thị bằng danh sách kề

Mật độ đồ thị

Định nghĩa 1.29

Mật độ đồ thị (graph density) (hay **mật độ cạnh**) của đồ thị đơn $G = (V, E)$ được định nghĩa là tỉ lệ số cạnh và bình phương số đỉnh

$$D = \frac{2|E|}{|V|(|V|-1)} \tag{1.8}$$

- **Đồ thị thưa** (sparse graph) là đồ thị có mật độ cạnh thấp (so với số đỉnh)
- **Đồ thị dày** (dense graph) là đồ thị có mật độ cạnh cao (so với đỉnh)

Biểu diễn đồ thị đơn bằng ma trận kề hay danh sách kề

Việc lựa chọn biểu diễn bằng ma trận kề hay danh sách kề cho một đồ thị ảnh hưởng đến thời gian và bộ nhớ sử dụng của các thuật toán đồ thị

- Nếu đồ thị thưa, chọn biểu diễn nào cho đồ thị? Tại sao
- Nếu đồ thị dày, chọn biểu diễn nào cho đồ thị? Tại sao

CÁC KHÁI NIỆM VỀ ĐƯỜNG ĐI

Dây chuyền

Định nghĩa 1.30

Cho đồ thị $G = (V, E)$, **dây chuyền** (path) P trong G là một dãy luân phiên các "đỉnh - cạnh"

$$P = v_1 e_1 v_2 e_2 v_3 ... v_m$$

sao cho $e_i = (v_i, v_{i+1})$ hoặc $e_i = (v_{i+1}, v_i)$

- Đỉnh v_1 được gọi là **đỉnh đầu** và v_m được gọi là **đỉnh cuối** của dây chuyền P
- **Chiều dài** (length) của dây chuyền là "số cạnh" hay "số đỉnh - 1" trong dây chuyền

Dây chuyền (cont.)

Định nghĩa 1.30

- Đối với đồ thị đơn chúng ta có thể viết dây chuyền bằng một dãy "đỉnh"

$$P = v_1 v_2 ... v_m$$

hoặc

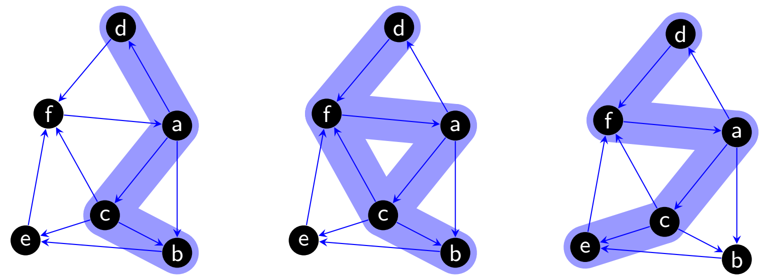
$$P = v_1 - v_m$$

Dây chuyền (cont.)

Định nghĩa 1.31

- Dây chuyền **đơn** (simple) là dây chuyền không có cạnh lặp lại
- Dây chuyền **sơ cấp** (simple) là dây chuyền không có đỉnh lặp lại

Dây chuyền (cont.)



(a) "d a c b" **(b)** "d f a c f a c b" **(c)** "d f a c e"

Hình 1.28: Một số dây chuyền

Một số nhận xét về dây chuyền

Nhận xét

1. Hai dây chuyền $P_1 = v_1 ... v_k$ và $P_2 = v_k ... v_m$ có đỉnh cuối của P_1 là đỉnh đầu của P_2 thì dãy P

$$P = P_1 \oplus P_2 = v_1 ... v_k ... v_m$$

   cũng là một dây chuyền
2. Mọi dãy con của một dây chuyền cũng là một dây chuyền
3. Dãy ngược của dây chuyền cũng là một dây chuyền

Đường đi

- Cho đồ thị $G = (V, E)$, **đường đi** (path) P trong G là một dãy luân phiên các "đỉnh - cạnh"

$$P = v_1 e_1 v_2 e_2 v_3 ... v_m$$

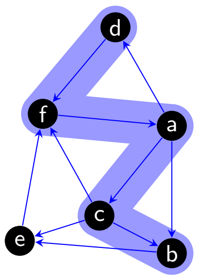
  sao cho $e_i = (v_i, v_{i+1})$
  - Đối với đồ thị đơn chúng ta có thể viết đường đi bằng một dãy "đỉnh"

$$P = v_1 v_2 ... v_m$$

    hoặc

$$P = v_1 - v_m$$

Đường đi (cont.)



Hình 1.29: Đường đi "d f a c b"

Chu trình và mạch

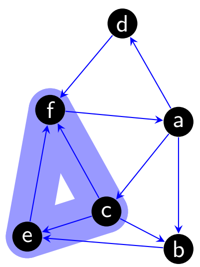
Định nghĩa 1.32

Chu trình (cycle) C của một đồ thị $G = (V, E)$ là một dây chuyền khép kín có đỉnh đầu trùng với đỉnh cuối

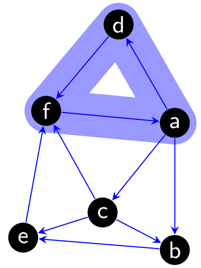
Định nghĩa 1.33

Mạch (cycle) C của một đồ thị $G = (V, E)$ là một đường đi khép kín có đỉnh đầu trùng với đỉnh cuối

Chu trình và mạch (cont.)



(a) chu trình "c f e c"



(b) mạch "d f a d"

Hình 1.30: Chu trình và mạch

TÌM KIẾM, DUYỆT TRÊN ĐỒ THỊ

Giới thiệu

- **Tìm kiếm** hay **duyệt** trên đồ thị là phương pháp **liệt kê** tất cả các đỉnh của đồ thị có thể **đến** được từ một đỉnh xuất phát *s* dựa trên thông tin **kề** của đồ thị.
  - Một trong những yêu cầu là không được bỏ sót hay lặp lại bất kỳ một đỉnh nào.
  - Hai chiến lược tìm kiếm tổng quát là **tìm kiếm theo chiều sâu** (Depth First Search - DFS) và **tìm kiếm theo chiều rộng** (Breadth First Search - BFS).

Ý tưởng DFS

Ý tưởng này được [Tarjan, 1972] tổng kết để giải quyết các bài toán cơ bản trong lý thuyết đồ thị như tìm điểm cắt, tìm thành phần liên thông, tìm thành phần song liên thông ...

1. Bắt đầu từ đỉnh được cho
2. Tại mỗi đỉnh bất kỳ *v*
   - Duyệt đỉnh *v*
   - Sau đó lần lượt đi tới những đỉnh kề với *v* và *chưa được duyệt* và lặp lại các thao tác trên đối với những đỉnh này
   - Quay lại *đỉnh trước* của *v*

Lưu ý

Để không một đỉnh nào liệt kê hai lần ta sử dụng kỹ thuật "đánh dấu"

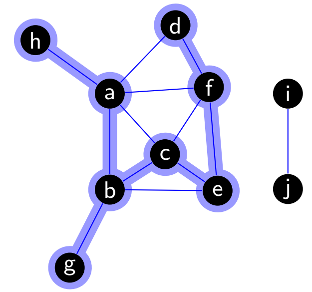
Thuật toán DFS

Cho một đồ thị $G = (V, E)$ có n đỉnh $V = \{v_1, ..., v_n\}$

Algorithm 1 Tìm kiếm theo chiều sâu

procedure $\mathrm{DFS}(v)$
    Duyệt đỉnh v
    for mỗi đỉnh u kề với v **do**
        if đỉnh u chưa được duyệt **then**
            $\mathrm{DFS}(u)$

Minh họa DFS



Hình 1.31: Thứ tự duyệt các đỉnh đồ thị bắt đầu từ đỉnh a

Ý tưởng BFS

Ý tưởng này được [Moore, 1959] đưa ra để tìm đường đi trong một mê cung

- Bắt đầu từ một đỉnh được cho
- Tại mỗi đỉnh bất kỳ v
  - Duyệt đỉnh v
  - Sau đó đi đến và duyệt các đỉnh kề với nó.
  - Tiếp tục lặp lại chiến lược cho các đỉnh kề của nó.

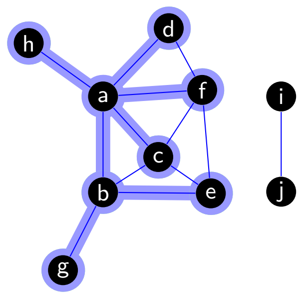
Thuật toán BFS

Cho một đồ thị $G = (V, E)$ có n đỉnh $V = \{v_1, ..., v_n\}$

Algorithm 2 Tìm kiếm theo chiều rộng

procedure $\mathrm{BFS}(v)$
    $queue \leftarrow v$
    while $queue \neq \emptyset$ **do**
        $x \leftarrow queue$
        Duyệt đỉnh x
        for mỗi đỉnh u kề với đỉnh x **do**
            if đỉnh u chưa duyệt và không có trong $queue$ **then**
                $queue \leftarrow u$

Minh họa BFS



Hình 1.32: Thứ tự duyệt các đỉnh đồ thị bắt đầu từ đỉnh a

Độ phức tạp của DFS và BFS

- Trong trường hợp đồ thị được biểu diễn bằng danh sách kề, độ phức tạp của DFS và BFS là

$$O(|V| + |E|)$$

- Trong trường hợp đồ thị được biểu diễn bằng ma trận kề thì độ phức tạp của hai thuật toán trên là

$$O\left(|V| + |V|^2\right)$$

Các bài toán đường đi và chu trình

Bài toán 1.1

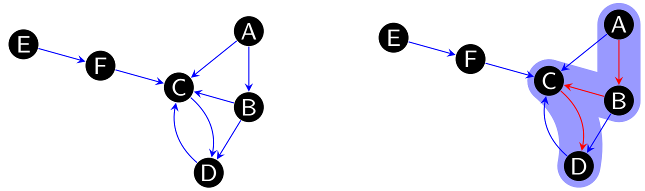
- **Tìm một đường đi sơ cấp**: Cho đồ thị $G = (V, E)$ và hai đỉnh s và t. Hãy tìm **đường đi sơ cấp** đi từ s cho đến t
- **Tìm tất cả đường đi sơ cấp**: Cho đồ thị $G = (V, E)$ và hai đỉnh s và t. Hãy tìm tất cả **đường đi sơ cấp** từ s cho đến t
- **Tìm tất cả đường đi đơn**: Cho đồ thị $G = (V, E)$ và hai đỉnh s và t. Hãy tìm **đường đi đơn** từ s cho đến t

Các bài toán đường đi và chu trình (cont.)

Bài toán 1.1

- **Tìm một đường đi có chiều dài cho trước**: Cho đồ thị $G = (V, E)$ và hai đỉnh s và t và một số dương k. Hãy tìm **đường đi** có chiều dài k đi từ s cho đến t
- **Tìm chu trình sơ cấp**: Cho đồ thị $G = (V, E)$ và hai s. Hãy tìm **chu trình sơ cấp** đi qua s.
- **Tìm tất cả chu trình sơ cấp**: Cho đồ thị $G = (V, E)$ và hai s. Hãy tìm tất cả **chu trình sơ cấp** đi qua s.

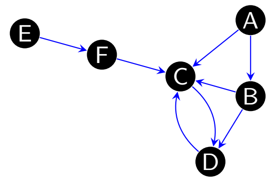
Các bài toán đường đi và chu trình (cont.)

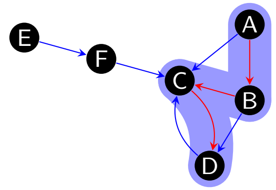


(a) đồ thị G **(b)** đường đi từ A đến D

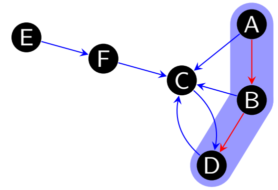
Hình 1.33: Đồ thị và một đường đi từ A đến D

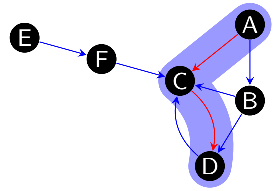
Các bài toán đường đi và chu trình (cont.)





(a) đồ thị G **(b)** đường đi 1 từ A đến D





(c) đường đi 2 từ A đến D **(d)** đường đi 3 từ A đến D

Hình 1.34: Đồ thị và tất cả đường đi từ A đến D

Áp dụng DFS để tìm đường đi

Algorithm 3 Tìm đường đi P từ đỉnh v_s đến v_e

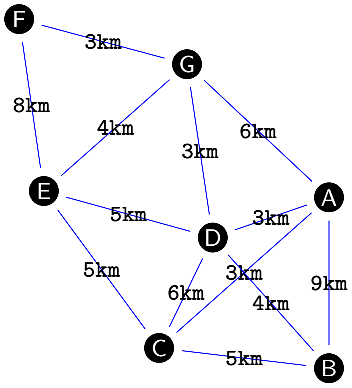
```
function DFS_FIND_PATH(v_s, v_e)
    Duyệt v_s
    if v_s = v_e then
        return true
    for mỗi đỉnh v kề với đỉnh v_s do
        if v chưa được duyệt then
            pre[v] ← v_s
            if DFS_FIND_PATH(v, v_e) then
                return true
    return false
```


Áp dụng DFS để tìm đường đi (cont.)

- Trong hàm trên đã sử dụng kỹ thuật lưu vết của đường đi thông qua việc lưu lại đỉnh trước của đỉnh v bằng phép gán $pre[v] = v_s$
- Để xác định đường đi ta sử dụng cách lần ngược từ đỉnh v_e cho đến v_s

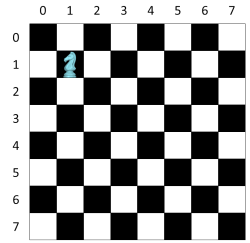
$$v_s \xleftarrow{\texttt{pre}} v_1 \xleftarrow{\texttt{pre}} \cdots \xleftarrow{\texttt{pre}} v_k \xleftarrow{\texttt{pre}} v_e$$

Áp dụng DFS để tìm đường đi (cont.)



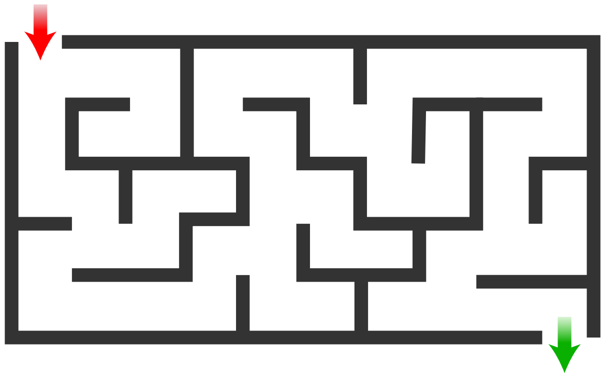
Hình 1.35: Tìm đường đi trên bản đồ các thị trấn

Áp dụng DFS để tìm đường đi (cont.)



Hình 1.36: Tìm đường đi cho con mã từ ô (1,1) đến ô (7,7)

Áp dụng DFS để tìm đường đi (cont.)



Hình 1.37: Tìm đường đi qua một mê cung

Những định lý về đường đi và chu trình

Định lý 1.9

Cho $G = (V, E)$ là một đồ thị đơn vô hướng có $n \geq 3$ đỉnh mọi đỉnh v đều có $d(v) \geq 2$ thì luôn tồn tại một chu trình sơ cấp trong G

Chứng minh

- Vì G là một đồ thị hữu hạn nên số đường đi sơ cấp trong G là hữu hạn.
- Giả sử $P = v_1 v_2 ... v_k$ là một trong các đường đi sơ cấp có độ dài cực đại. Do bậc của mỗi đỉnh không nhỏ hơn 2 nên đỉnh v_1 phải kề với một đỉnh u nào đó. Hai tình huống xảy ra
  1. Nếu đỉnh $u = v_i \in \{v_3, ..., v_k\}$ thì đồ thị G sẽ có chu trình sơ cấp $Q = v_1 ... v_i v_1$

Những định lý về đường đi và chu trình (cont.)

2. Ngược lại, nếu đỉnh $u = v_i \in \{v_3, ..., v_k\}$ khi đó trong G tồn tại đường đi sơ cấp $Q = uv_1v_2...v_k$ có độ dài lớn hơn đường sơ cấp P có độ dài lớn nhất đã chọn (mâu thuẫn). Vậy tình huống này không thể xảy ra

- Vậy trong G tồn tại một chu trình sơ cấp

■

Những định lý về đường đi và chu trình (cont.)

Định lý 1.10

Cho $G = (V, E)$ là một đồ thị đơn vô hướng có số đỉnh $n \geq 4$ và mọi đỉnh đều có bậc lớn hơn 3 thì trong G luôn tồn tại một chu trình sơ cấp có độ dài chẵn.

Chứng minh

- Vì G là một đồ thị hữu hạn nên số đường đi sơ cấp trong G là hữu hạn.
- Giả sử $P = v_1v_2...v_k$ là một trong các đường đi sơ cấp có độ dài cực đại. Do bậc của mỗi đỉnh không nhỏ hơn 3 nên đỉnh v_1 phải kề với hai đỉnh $v_i, v_j \in \{v_3, ..., v_k\}, i < j$

Những định lý về đường đi và chu trình (cont.)

- Khi đó trong G có 2 chu trình sơ cấp

$$Q_1 = v_1...v_iv_1$$

$$Q_2 = v_1...v_jv_1$$

- Nếu một trong hai Q_1, Q_2 có độ dài chẵn thì ta có điều phải chứng minh

Những định lý về đường đi và chu trình (cont.)

- Nếu cả hai Q_1, Q_2 có độ dài lẻ thì

$$R_1 = v_1...v_i$$

đường đi sơ cấp có độ dài chẵn

$$R_2 = v_i...v_jv_1$$

đường đi sơ cấp có độ dài lẻ

$$R_2 = v_1v_i...v_jv_1$$

chu trình sơ cấp có độ dài chẵn (điều phải chứng minh)

■

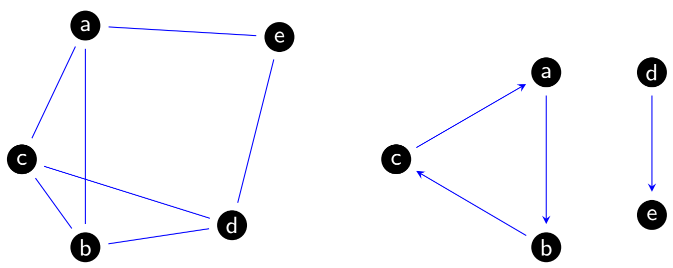
MỘT SỐ KHÁI NIỆM KHÁC TRÊN ĐỒ THỊ

Đồ thị liên thông

Định nghĩa 1.34

Cho đồ thị $G = (V, E)$, Ta nói G là **đồ thị liên thông** (**connected graph**) khi và chỉ khi với mọi đỉnh $x, y \in V$ luôn tồn tại dây chuyền từ x đến y.

Đồ thị liên thông (cont.)



(a) liên thông **(b)** không liên thông

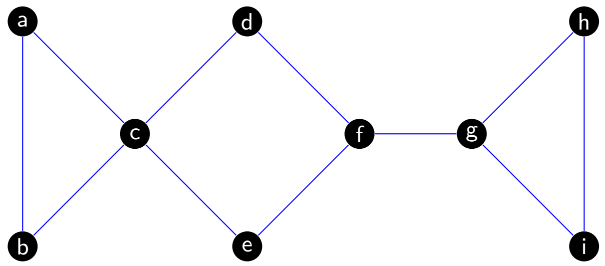
Hình 1.38: Đồ thị liên thông và không liên thông

Đồ thị liên thông (cont.)

Định nghĩa 1.35

- Cho một đồ thị liên thông G, v là một đỉnh của đồ thị, v được gọi là **đỉnh cắt** (**cut vertex**) nếu $G - \{v\}$ không liên thông
- Cho một đồ thị liên thông G, e là một cạnh của đồ thị, e được gọi là **cầu** (**bridge**) nếu $G - \{e\}$ không liên thông

Đồ thị liên thông (cont.)



Hình 1.39: Hãy xác định điểm cắt và cạnh cầu của đồ thị

Đồ thị liên thông (cont.)

Định nghĩa 1.36

- **Bậc liên thông đỉnh** (vertex connectivity) của một đồ thị G là số đỉnh ít nhất bỏ đi làm cho đồ thị mất tính liên thông. Ký hiệu là $\kappa(G)$
- **Bậc liên thông cạnh** (edge connectivity) của một đồ thị G là số cạnh ít nhất bỏ đi làm cho đồ thị mất tính liên thông. Ký hiệu là $\lambda(G)$

Quan hệ liên thông

Định nghĩa 1.37

Cho đồ thị $G = (V, E)$, Ta định nghĩa một **quan hệ liên thông** $\sim$ trên tập đỉnh V như sau
$\forall x, y \in V, x \sim y$ hoặc $x = y$ hoặc có dây chuyền từ x đến y

Các thành phần liên thông

Định nghĩa 1.38

Một **thành phần liên thông** (connected component) của một đồ thị là một lớp tương đương được xác định bởi quan hệ liên thông

Định nghĩa 1.39

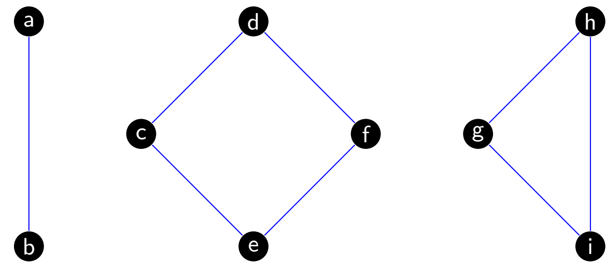
Một thành phần liên thông của đồ thị G là **đồ thị con liên thông tối đại** (maximal connected subgraph) của G

Các thành phần liên thông (cont.)

Nhận xét

- Số thành phần liên thông của một đồ thị là số lượng các lớp tương đương
- Các thành phần liên thông là các **đồ thị con**
- Một đồ thị được gọi là đồ thị liên thông nếu nó chỉ có một thành phần liên thông

Các thành phần liên thông (cont.)



Hình 1.40: Đồ thị có 3 thành phần liên thông

Áp dụng DFS tìm các thành phần liên thông

Cho một đồ thị $G = (V, E)$ có n đỉnh $V = \{v_1, ..., v_n\}$

Algorithm 4 Đánh nhãn các thành phần liên thông

```
Khởi tạo biến label = 0 và gán nhãn 0 cho tất cả các đỉnh
for i ← {1...n} do
    if nhãn đỉnh v_i là 0 then
        label ← label + 1
        DFS_ASSIGN(v_i, label)
procedure DFS_ASSIGN(v, label)
    Gán nhãn label cho đỉnh v
    for mỗi đỉnh u kề với v do
        if nhãn đỉnh u là 0 then
            DFS_ASSIGN(u, label)
```


Những định lý về liên thông

Định lý 1.11

Nếu trong đồ thị $G = (V, E)$ có đúng hai đỉnh bậc lẻ thì hai đỉnh này phải liên thông với nhau

Chứng minh

- Giả sử đồ thị $G = (V, E)$ có đúng hai đỉnh bậc lẻ x và y nhưng hai đỉnh này lại không liên thông với nhau. Do đó, x và y phải thuộc vào 2 thành phần liên thông G_1, G_2 khác nhau của G
- Theo giả thuyết do G chỉ có đúng 2 đỉnh bậc lẻ nên trong mỗi đồ thị con G_1 và G_2 chỉ có đúng một đỉnh bậc lẻ. Mâu thuẫn với tính chất số đỉnh bậc lẻ trong một đồ thị là một số chẵn.
- Vậy x và y phải liên thông với nhau.

■

Những định lý về liên thông (cont.)

Định lý 1.12

Cho đồ thị đơn $G = (V, E)$ có số đỉnh $n \geq 2$, nếu $\forall v_1, v_2 \in V$ và $d(v_1) + d(v_2) \geq n$ thì đồ thị G liên thông

Chứng minh

Sinh viên tự chứng minh ∎

Những định lý về liên thông (cont.)

Hệ quả 1.2

Cho đồ thị đơn $G = (V, E)$ có số đỉnh n, nếu $\forall v \in V, d(v) \geq \frac{n}{2}$ thì đồ thị G liên thông

Chứng minh

Sinh viên tự chứng minh ∎

Đồ thị liên thông yếu

Định nghĩa 1.40

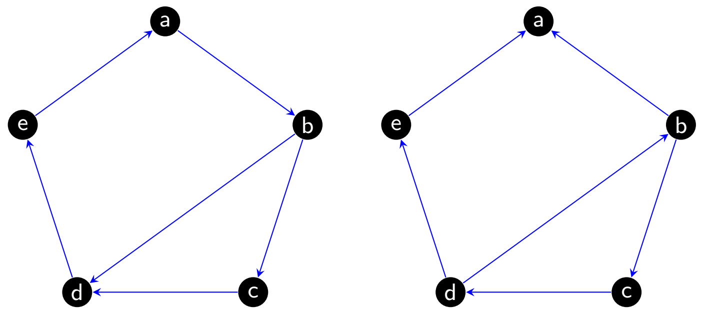
Cho đồ thị có hướng $G = (V, E)$, ta nói G là **đồ thị liên thông yếu** (**weakly connected graph**) khi và chỉ khi đồ thị nền của nó là liên thông

Đồ thị liên thông mạnh

Định nghĩa 1.41

Cho đồ thị có hướng $G = (V, E)$, ta nói G là **đồ thị liên thông mạnh** (**strongly connected graph**) khi và chỉ khi $\forall x, y \in V$ luôn tồn tại đường đi từ x đến y và ngược lại.

Đồ thị liên thông mạnh (cont.)



(a) liên thông mạnh **(b)** không liên thông mạnh

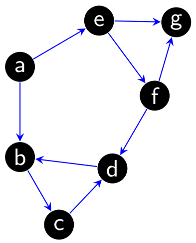
Hình 1.41: Đồ thị liên thông mạnh và không liên thông mạnh

Các thành phần liên thông mạnh

Định nghĩa 1.42

Một **thành phần liên thông mạnh** (**strongly connected component**) của đồ thị G là **đồ thị con liên thông mạnh tối đại** (**maximal strongly connected subgraph**) của G

Các thành phần liên thông mạnh (cont.)



Hình 1.42: Đồ thị có các thành phần liên thông mạnh {a},{e},{f},{g} và {b,c,d}

Đồ thị song liên thông

Định nghĩa 1.43

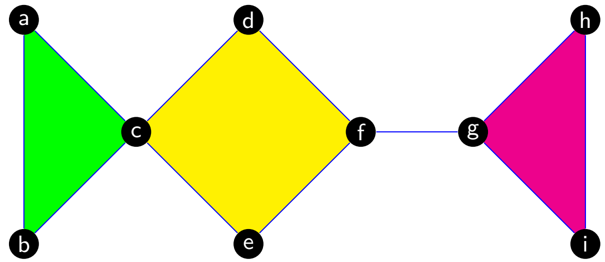
Đồ thị **song liên thông** (**biconnectivity**) là đồ thị không chứa đỉnh cắt

Các thành phần song liên thông

Định nghĩa 1.44

Một **thành phần song liên thông** (**biconnected component**) của đồ thị G là **đồ thị con song liên thông tối đại** (**maximal biconnected subgraph**) của G

Các thành phần song liên thông (cont.)



Hình 1.43: Đồ thị có 4 thành phần song liên thông

Đồ thị có gốc

Định nghĩa 1.45

Cho đồ thị có hướng $G = (V, E)$, ta nói G là **đồ thị có gốc** (**rooted graph**) nếu tồn tại một đỉnh $r \in V$ sao cho từ r có đường đi đến các đỉnh còn lại của đồ thị.

ĐỒ THỊ CÓ TRỌNG SỐ

Phân loại

Đồ thị có trọng số (weighted graph) có thể phân thành ba loại

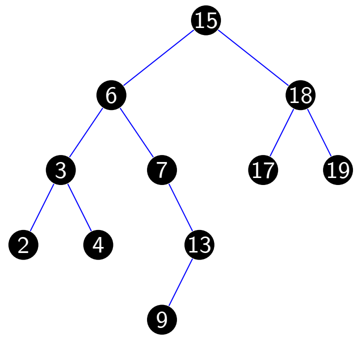
- Đồ thị có trọng số đỉnh
- Đồ thị có trọng số cạnh
- Đồ thị có trọng số hỗn hợp

Đồ thị có trọng số đỉnh

Định nghĩa 1.46

Đồ thị $G = (V, E, L)$ là **đồ thị có trọng số** (weighted graph) nếu mỗi đỉnh của nó có một giá trị số thực thông qua một hàm trọng số K

$$\begin{aligned} K: \quad V &\rightarrow R \\ v &\mapsto K(v) \end{aligned} \tag{1.9}$$



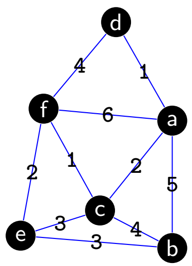
Hình 1.44: Đồ thị vô hướng có trọng số đỉnh

Đồ thị có trọng số cạnh

Định nghĩa 1.47

Đồ thị $G = (V, E, L)$ là **đồ thị có trọng số** (weighted graph) nếu mỗi cạnh có của nó có một giá trị số thực thông qua một hàm trọng số L

$$\begin{aligned} L: \quad E &\rightarrow R \\ e &\mapsto L(e) \end{aligned} \tag{1.10}$$



Hình 1.45: Đồ thị vô hướng có trọng số cạnh

Đồ thị có trọng số cạnh (cont.)

Định nghĩa 1.48

Cho đồ thị có trọng số $G = (V, E, L)$, trọng số của đồ thị là

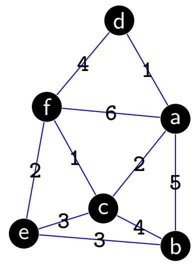
$$L(G) = \sum_{e \in E} L(e) \tag{1.11}$$

Biểu diễn đồ thị có trọng số cạnh bằng ma trận trọng số

- Cho một đồ thị có trọng số $G = (V, E, L)$ có n đỉnh $\{v_1, v_2, ..., v_n\}$ ta có thể biểu diễn đồ thị bằng **ma trận trọng số** (weighted matrix) là một ma trận vuông A cấp n

$$A = (a_{ij})$$
$$a_{ij} = \begin{cases} 0 & i = j \\ L(v_i, v_j) & i \neq j, (v_i, v_j) \in E \\ \infty & i \neq j, (v_i, v_j) \notin E \end{cases} \tag{1.12}$$

Biểu diễn đồ thị có trọng số cạnh bằng ma trận trọng số (cont.)



(a) đồ thị có trọng số

$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 0 & 5 & 2 & 1 & \infty & 6 \\ 5 & 0 & 4 & \infty & 3 & \infty \\ 2 & 4 & 0 & \infty & 1 & 3 \\ 1 & \infty & \infty & 0 & \infty & 4 \\ \infty & 3 & 1 & \infty & 0 & 2 \\ 6 & \infty & 3 & 4 & 2 & 0 \end{pmatrix} \end{array}$$

(b) ma trận trọng số

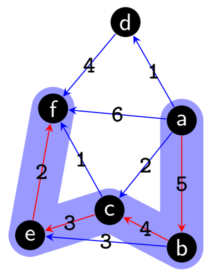
Hình 1.46: Biểu diễn đồ thị bằng ma trận trọng số

Trọng số đường đi & chu trình

Định nghĩa 1.49

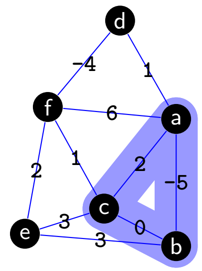
Cho $G = (V, E, L)$ là đồ thị có trọng số với V là tập đỉnh, E là tập cạnh và L là hàm trọng số trên miền tập cạnh. Trọng số của một đường đi P được định nghĩa là

$$L(P) = \sum_{e \in P} L(e) \tag{1.13}$$

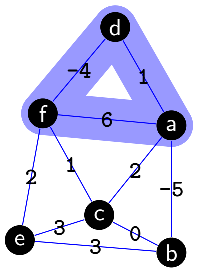


Hình 1.47: Trọng số đường đi "a b c e f" là 14

Trọng số đường đi & chu trình (cont.)



(a) trọng số chu trình "a b c a" là -3

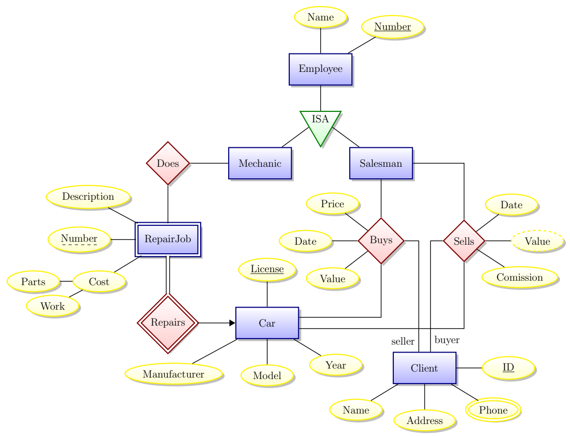


(b) trọng số chu trình "a d f a" là 3

Hình 1.48: Trọng số các chu trình

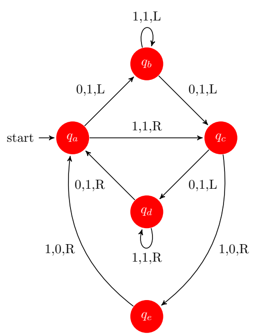
MỘT SỐ LOẠI ĐỒ THỊ ỨNG DỤNG

Một số loại đồ thị ứng dụng



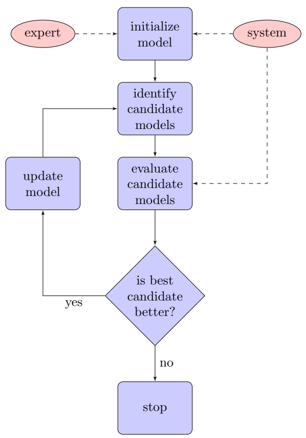
Hình 1.49: Lược đồ thực thể kết hợp trong cơ sở dữ liệu

Một số loại đồ thị ứng dụng (cont.)



Hình 1.50: Máy trạng thái

Một số loại đồ thị ứng dụng (cont.)



Hình 1.51: Lưu đồ thuật toán

Tài liệu tham khảo

Moore, E. F. (1959).
The shortest path through a maze.
Bell Telephone System.

Tarjan, R. (1972).
Depth-first search and linear graph algorithms.
SIAM journal on computing, 1(2):146–160.