

CÁC THUẬT GIẢI ĐỒ THỊ CƠ BẢN

- Các khái niệm và thuật ngữ
- Biểu diễn đồ thị
- Tìm kiếm theo chiều rộng
- Tìm kiếm theo chiều sâu

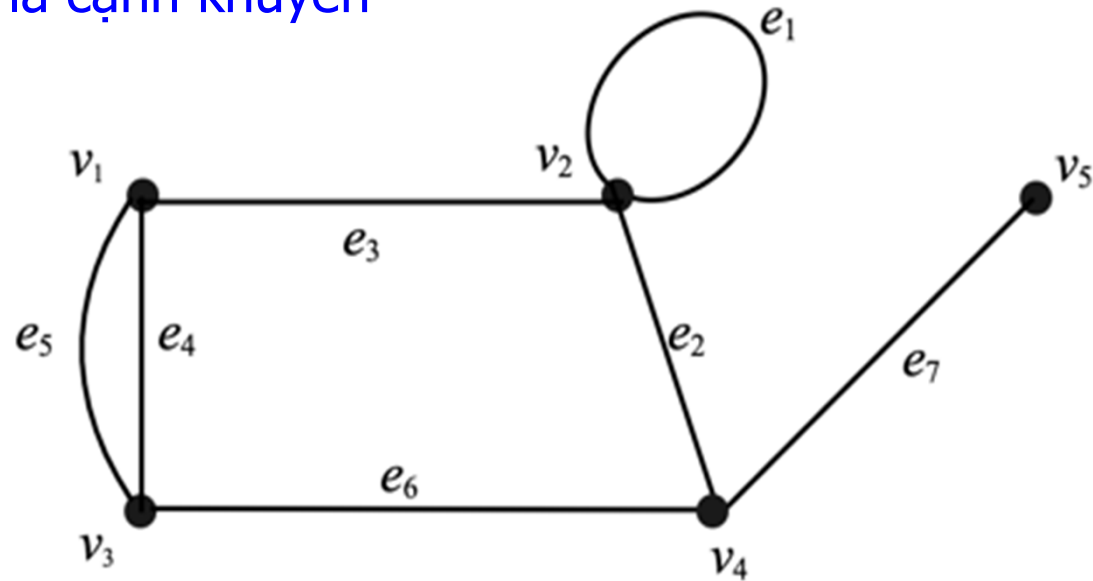
KHÁI NIỆM VÀ THUẬT NGỮ

- Đồ thị vô hướng (undirected graph) $G = (V, E)$, gồm một tập V các đỉnh (vertice) và một tập E các cạnh (edge), mỗi cạnh $e = (u, v) \in E$ ứng với một cặp không có thứ tự các đỉnh $u, v \in V$
- Đồ thị có hướng (directed graph) $G = (V, E)$, gồm một tập V các đỉnh và một tập E các cạnh, mỗi cạnh $e = (u, v) \in E$ ứng với một cặp có thứ tự các đỉnh $u, v \in V$

KHÁI NIỆM VÀ THUẬT NGỮ

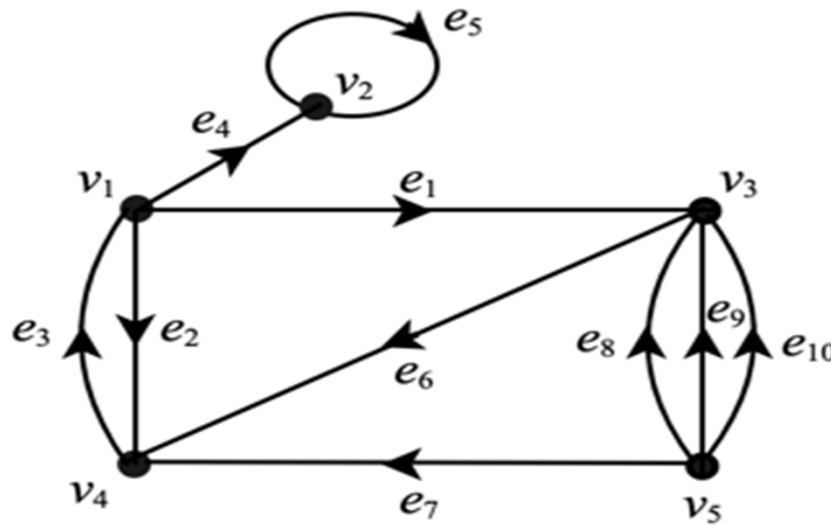
$e_4=(v_1, v_3)$ và $e_5=(v_1, v_3)$ là các cạnh song song

$e_1=(v_2, v_2)$ là cạnh khuyên



KHÁI NIỆM VÀ THUẬT NGỮ

$e_8=(v_5, v_3)$, $e_9=(v_5, v_3)$ và $e_{10}=(v_5, v_3)$ là các cạnh song song
 $e_5=(v_2, v_2)$ là cạnh khuyên



KHÁI NIỆM VÀ THUẬT NGỮ

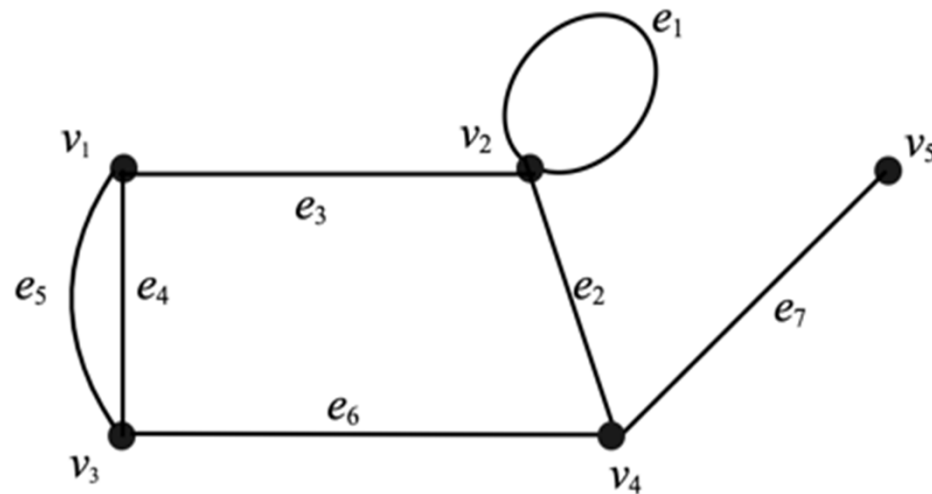
- Một đồ thị không có cạnh khuyên hoặc cạnh song song gọi là đơn đồ thị (simple graph), ngược lại gọi là đa đồ thị (multigraph)

KHÁI NIỆM VÀ THUẬT NGỮ

- Đỉnh u và v là **kề nhau** (adjacent) nếu có cạnh $e = (u, v)$, cạnh e gọi là **liên thuộc với u và v**
- Đường đi độ dài n từ đỉnh x_0 đến đỉnh x_n trong một đồ thị là dãy $P = x_0, x_1, \dots, x_n$ trong đó mỗi (x_i, x_{i+1}) là một cạnh
- Đường đi có đỉnh đầu x_0 trùng với đỉnh cuối x_n gọi là **chu trình**
- Đường đi hay chu trình gọi là **đơn** nếu không có cạnh lặp lại

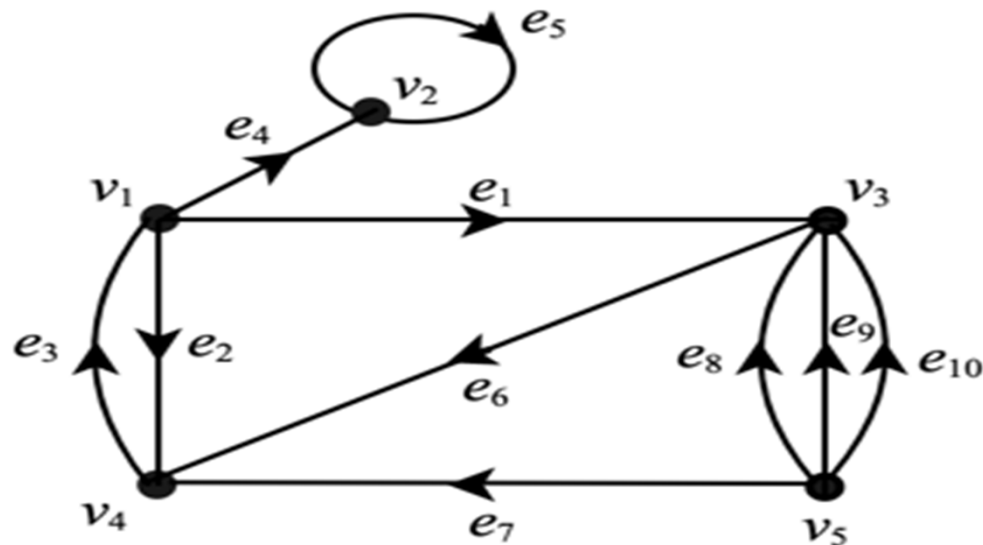
KHÁI NIỆM VÀ THUẬT NGỮ

$P = v_1, v_3, v_4, v_5$ là một đường đi và cũng là đường đi đơn,
 $C = v_1, v_3, v_4, v_2, v_1, v_3, v_1$ là chu trình không đơn vì cạnh e_4
 $= (v_1, v_3)$ hoặc $e_5 = (v_1, v_3)$ được lặp lại một lần



KHÁI NIỆM VÀ THUẬT NGỮ

$P = v_1, v_3, v_4, v_1, v_3$ là một đường đi không đơn, do cạnh $e_1 = (v_1, v_3)$ được lặp lại một lần trên đường đi này, $C = v_1, v_3, v_4, v_1$ là một chu trình đơn

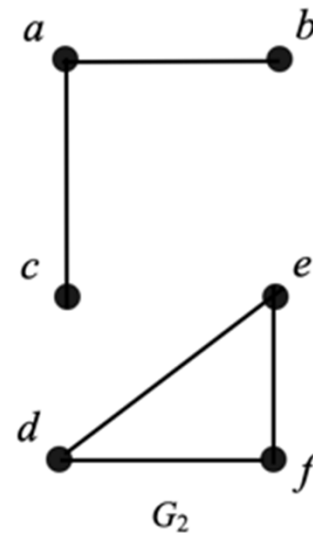
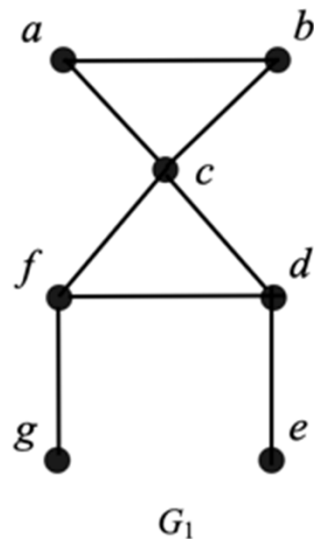


KHÁI NIỆM VÀ THUẬT NGỮ

- Một đồ thị được gọi là **liên thông** nếu luôn tìm được đường đi giữa hai đỉnh bất kỳ của nó

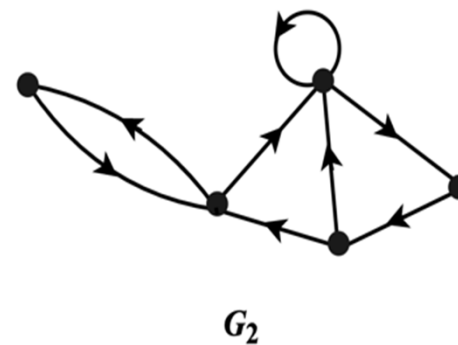
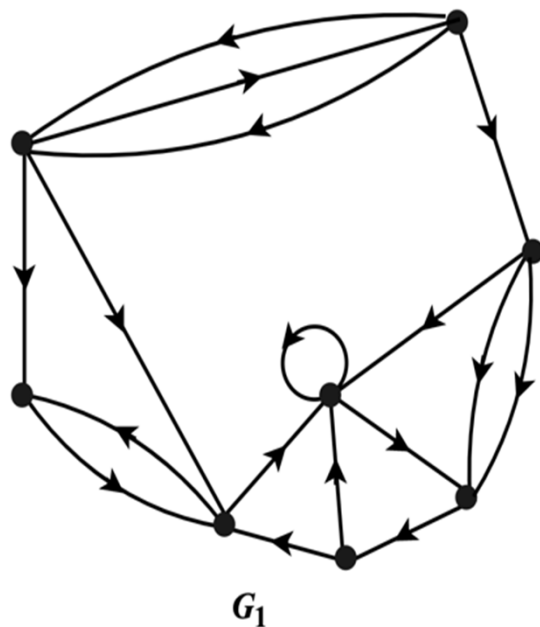
KHÁI NIỆM VÀ THUẬT NGỮ

Đồ thị G_1 liên thông, đồ thị G_2 không liên thông



KHÁI NIỆM VÀ THUẬT NGỮ

Đồ thị G_1 không liên thông, đồ thị G_2 liên thông



BIỂU DIỄN ĐỒ THỊ

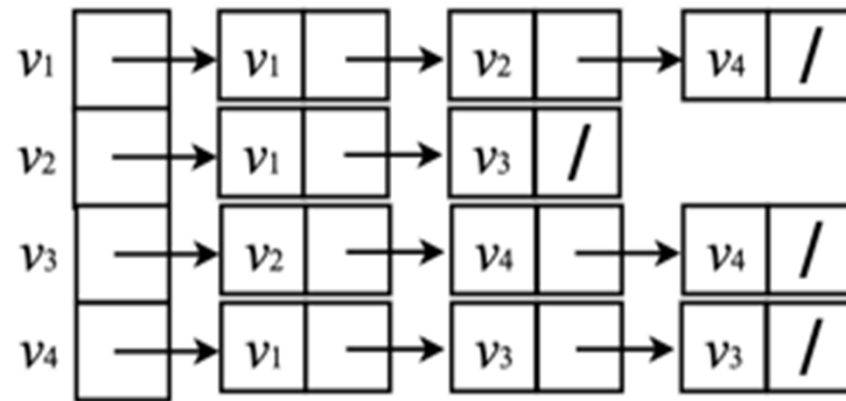
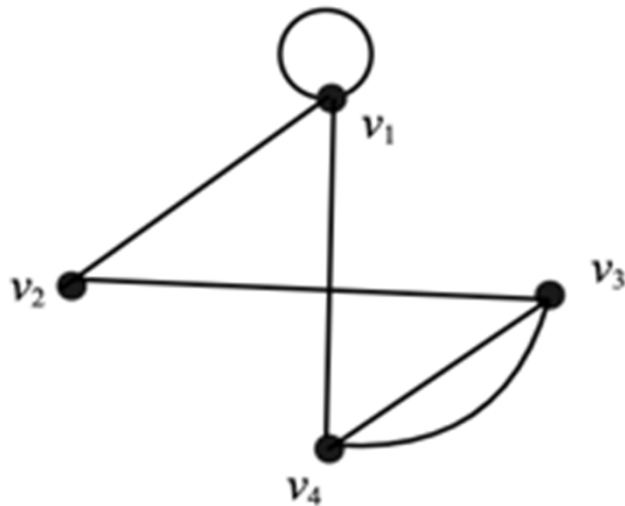
- Biểu diễn bằng danh sách kề (adjacency list)
- Biểu diễn bằng ma trận kề (adjacency matrix)
- So sánh các phương pháp biểu diễn đồ thị

DANH SÁCH KẼ

- Danh sách kề của đỉnh u : $\text{adj}(u) = \{v \in V \mid (u, v) \in E\}$
- Có thể biểu diễn đồ thị $G = (V, E)$ như một tập các danh sách kề bằng cách lưu trữ mỗi đỉnh $u \in V$ cùng với danh sách các đỉnh kề với u

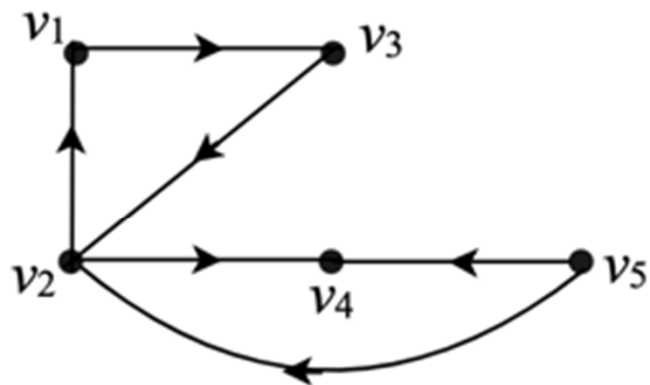
DANH SÁCH KẼ

Danh sách kề của đồ thị vô hướng

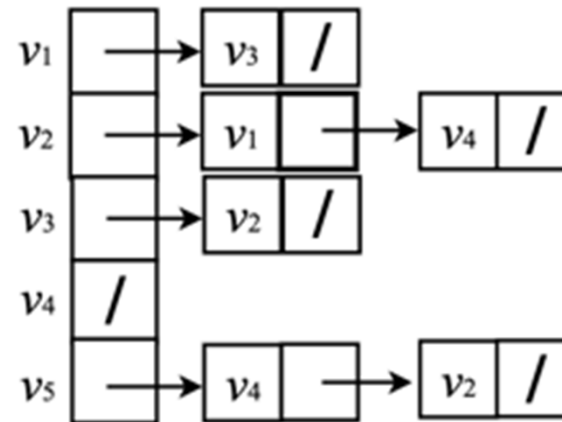


DANH SÁCH KẼ

Danh sách kề của đồ thị **có hướng**



(a)



(b)

MA TRẬN KỀ

- Cho đơn đồ thị $G = (V, E)$, với tập đỉnh $V = \{1, 2, \dots, n\}$, ma trận kề của G là

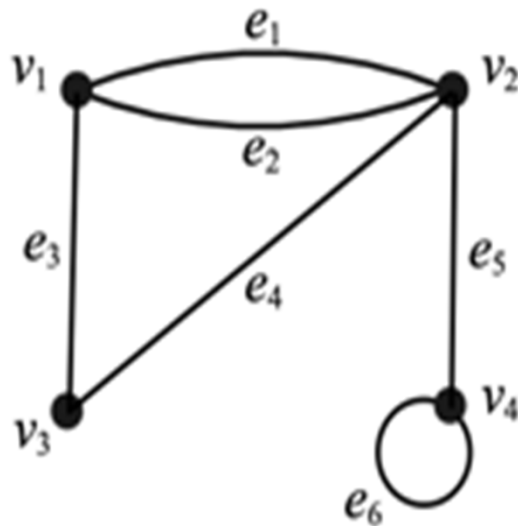
$$A = \{a_{ij} \mid i, j = 1, 2, \dots, n\}, \text{ } a_{ij} = 0 \text{ nếu } (i, j) \notin E \text{ và } a_{ij} = 1 \text{ nếu } (i, j) \in E$$

- Nếu G là đa đồ thị thì

$$a_{ij} = 0 \text{ nếu } (i, j) \notin E \text{ và } a_{ij} = k \text{ nếu có } k \text{ cạnh nối hai đỉnh } i \text{ và } j$$

MA TRẬN KỀ

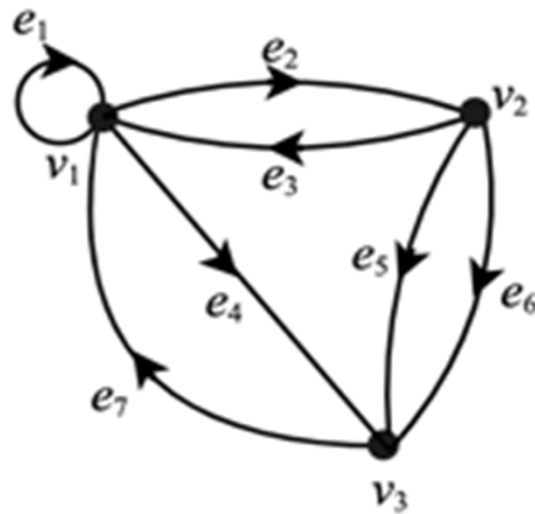
Ma trận kề của đồ thị vô hướng



$$A_G = \begin{bmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

MA TRẬN KẼ

Ma trận kề của đồ thị có hướng



$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 2 \\ 1 & 0 & 0 \end{bmatrix}$$

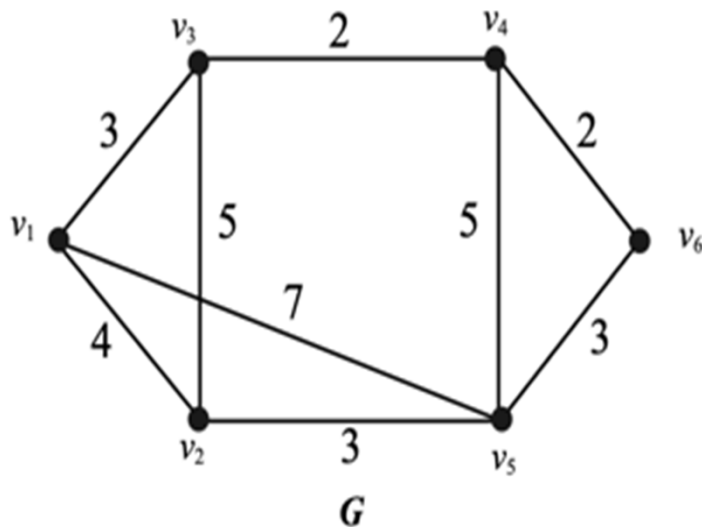
MA TRẬN KỀ

- Đồ thị **có trọng số** (weighted graph) là đồ thị mà mỗi cạnh (i, j) được **gán một số thực $w(i, j)$**
- Một đồ thị có trọng số với n đỉnh có thể được biểu diễn bởi ma trận trọng số

$C = \{c_{ij} : i, j = 1, 2, \dots, n\}$, trong đó $c_{ij} = w(i, j)$ nếu có cạnh (i, j) và $c_{ij} = 0, \infty$, hoặc $-\infty$ nếu không có cạnh (i, j)

MA TRẬN KỀ

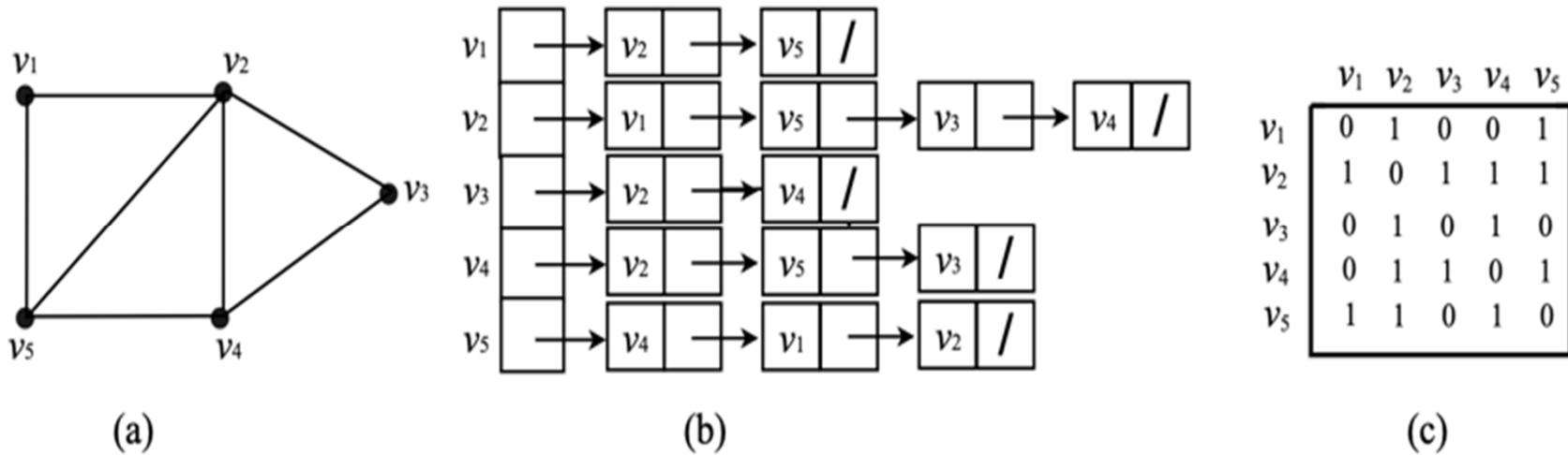
Ma trận **trọng số** của đồ thị vô hướng



$$C = \begin{bmatrix} 0 & 4 & 3 & 0 & 7 & 0 \\ 4 & 0 & 5 & 0 & 3 & 0 \\ 3 & 5 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 5 & 2 \\ 7 & 3 & 0 & 5 & 0 & 3 \\ 0 & 0 & 0 & 2 & 3 & 0 \end{bmatrix}$$

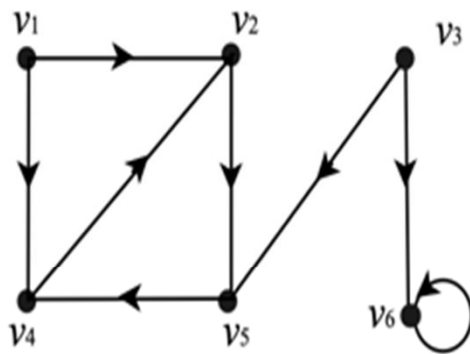
SO SÁNH CÁC CÁCH BIỂU DIỄN

Biểu diễn đồ thị vô hướng bằng danh sách và ma trận

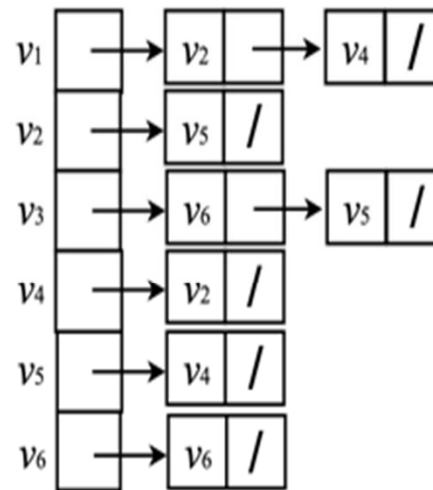


SO SÁNH CÁC CÁCH BIỂU DIỄN

Biểu diễn đồ thị **có hướng** bằng danh sách và ma trận



(a)



(b)

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	0	1	0	0
v_2	0	0	0	0	1	0
v_3	0	0	0	0	1	1
v_4	0	1	0	0	0	0
v_5	0	0	0	1	0	0
v_6	0	0	0	0	0	1

(c)

SO SÁNH CÁC CÁCH BIỂU DIỄN

- Chi phí bộ nhớ cho ma trận là $O(|V|^2)$ và cho danh sách là $O(|V| + 2|E|)$
- Chi phí xử lý khi dùng ma trận là $O(1)$ và khi dùng danh sách là $O(|V|)$

TÌM KIẾM THEO CHIỀU RỘNG (Breadth-First Search-BFS)

- Thuật giải BFS
- Phân tích BFS
- Đường đi ngắn nhất
- Cây tìm kiếm theo chiều rộng

THUẬT GIẢI BFS

Ý tưởng thuật giải

- Bắt đầu tìm kiếm từ **đỉnh s** cho trước tùy ý
- Tại thời điểm đã tìm thấy u , thuật toán tiếp tục tìm kiếm tập tất cả các **đỉnh kề với u**
- Thực hiện quá trình này cho các đỉnh còn lại

THUẬT GIẢI BFS

Ý tưởng thuật giải

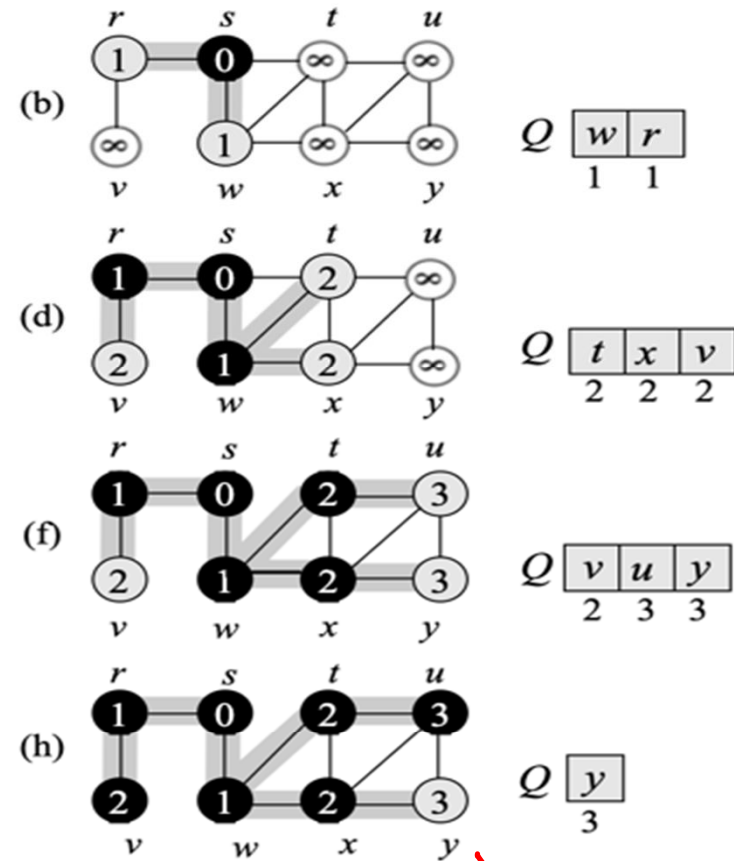
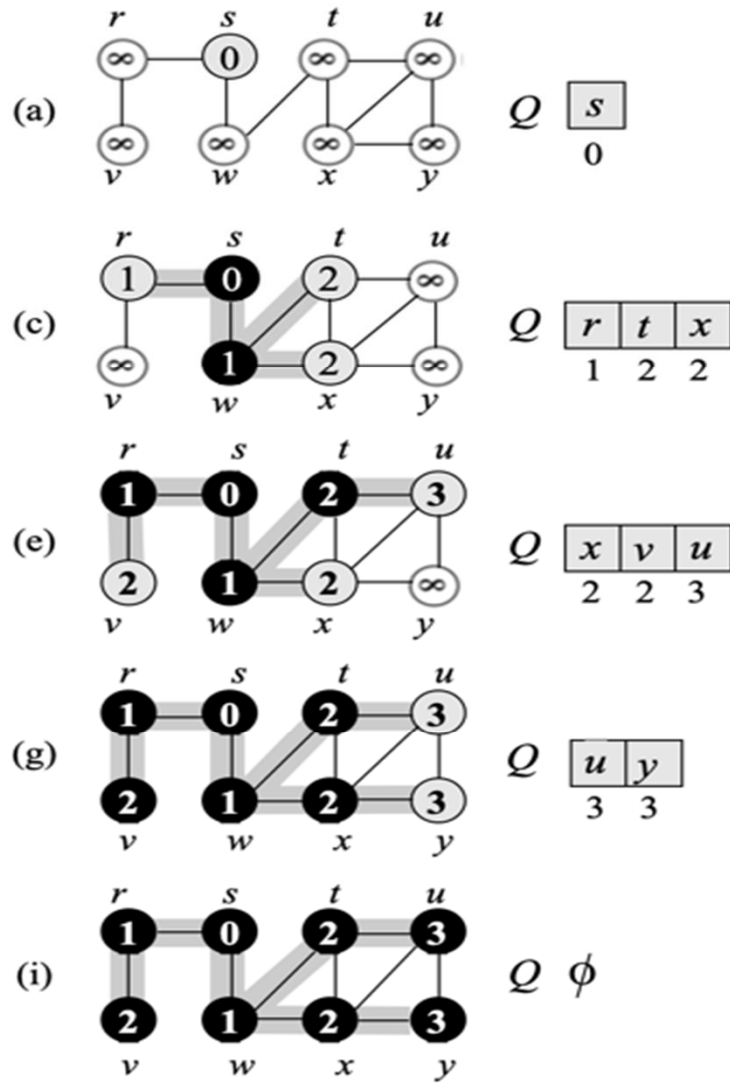
- Dùng một hàng đợi để duy trì trật tự tìm kiếm theo chiều rộng
- Dùng các màu để không lặp lại các đỉnh tìm kiếm
- Dùng một mảng để xác định đường đi ngắn nhất từ s đến các đỉnh đã được tìm kiếm
- Dùng một mảng để lưu trữ đỉnh đi trước của đỉnh được tìm kiếm

THUẬT GIẢI BFS

BFS(G, s)

```
1 for each  $u \in V[G] - \{s\}$ 
2    $\text{color}[u] = \text{WHITE}$ 
3    $d[u] = \infty$ 
4    $\pi[u] = \text{NIL}$ 
5  $\text{color}[s] = \text{GRAY}$ 
6  $d[s] = 0$ 
7  $\pi[s] = \text{NIL}$ 
8  $Q = \emptyset$ 
9 ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11    $u = \text{DEQUEUE}(Q)$ 
12   for each  $v \in \text{Adj}[u]$ 
13     if  $\text{color}[v] == \text{WHITE}$ 
14        $\text{color}[v] = \text{GRAY}$ 
15        $d[v] = d[u] + 1$ 
16        $\pi[v] = u$ 
17       ENQUEUE( $Q, v$ )
18    $\text{color}[u] = \text{BLACK}$ 
```

THUẬT GIẢI BFS



$v = s$

$d(s, s) = 0$
 $d(s, u) = \dots$

PHÂN TÍCH BFS

- Tổng chi phí khởi tạo là $O(V)$
- Mỗi thao tác trên hàng đợi là $O(1)$, vì vậy tổng thời gian cho thao tác trên hàng đợi là $O(V)$
- Tổng thời gian chi phí cho quét các danh sách kề là $O(E)$
- Tổng thời gian chạy của BFS là $O(V+E)$

ĐƯỜNG ĐI NGẮN NHẤT

- Khoảng cách đường đi ngắn nhất (shortest-path distance) từ s đến v là số cạnh ít nhất trong các đường đi từ s đến v , ký hiệu $\delta(s, v)$
- Qui ước $\delta(s, v) = \infty$ nếu không có đường đi từ s đến v
- Một đường đi độ dài bằng $\delta(s, v)$ từ s đến v được gọi là đường đi ngắn nhất từ s đến v

ĐƯỜNG ĐI NGẮN NHẤT

- **Định lý:** Cho BFS chạy trên một đồ thị từ đỉnh s , thì thuật giải tìm kiếm được mọi đỉnh v mà có thể đạt được từ s , khi kết thúc, BFS xác định các đường đi ngắn nhất từ s đến v sao cho $d[v] = \delta(s, v)$ với mọi $v \in V$

ĐƯỜNG ĐI NGẮN NHẤT

PrintPath(G, s, v)

1 **if** $v == s$

2 Print(s)

3 **elseif** $\pi[v] == \text{NIL}$

4 Print("no path from" s "to" v "exists")

5 **else** PrintPath($G, s, \pi[v]$)

6 Print(v)

ĐƯỜNG ĐI NGẮN NHẤT

- **Hệ quả:** Cho BFS chạy trên một đồ thị từ đỉnh s , thì thuật giải tìm kiếm được mọi đỉnh v mà có thể đạt được từ s , khi kết thúc, BFS xác định một cây có gốc tại đỉnh s và các đường đi từ s đến các đỉnh v trên cây là các đường đi ngắn nhất có độ dài $d[v] = \delta(s, v)$ với mọi $v \in V$

TÌM KIẾM THEO CHIỀU SÂU (Depth-First Search-DFS)

- Thuật giải DFS
- Phân tích DFS
- Tính chất của DFS

THUẬT GIẢI DFS

Ý tưởng thuật giải

- Bắt đầu tìm kiếm từ **một đỉnh u** nào đó
- Chọn **đỉnh kề v tùy ý của u** để tiếp tục quá trình tìm kiếm và lặp lại quá trình tìm kiếm này đối với v

THUẬT GIẢI DFS

Ý tưởng thuật giải

- Dùng các màu để không lặp lại các đỉnh tìm kiếm
- Dùng các biến thời gian để xác định các thời điểm phát hiện và hoàn thành tìm kiếm của một đỉnh
- Dùng một mảng để lưu trữ đỉnh đi trước của đỉnh được tìm kiếm

THUẬT GIẢI DFS

DFS(G)

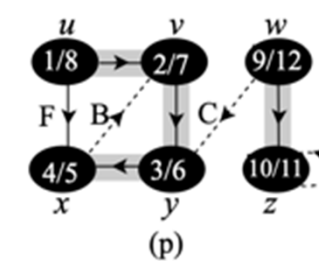
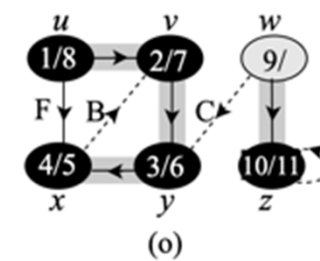
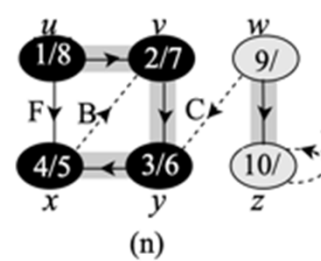
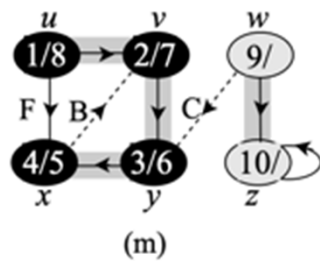
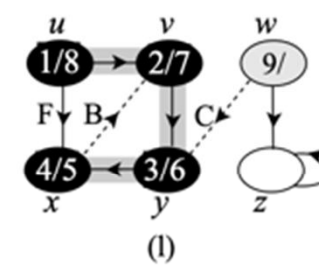
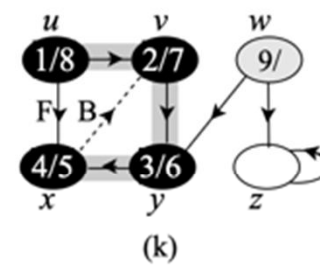
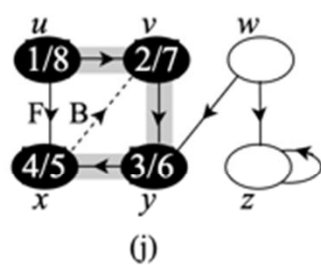
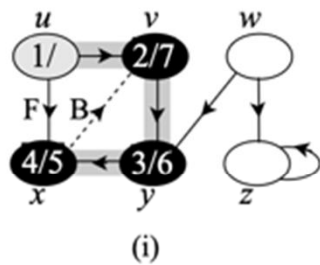
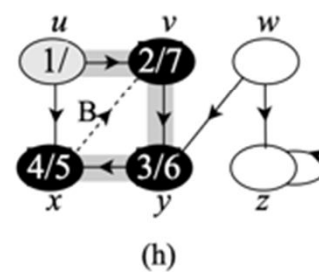
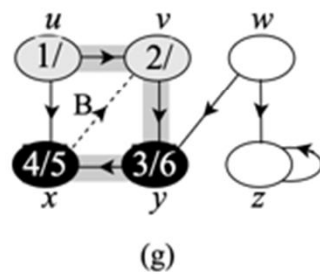
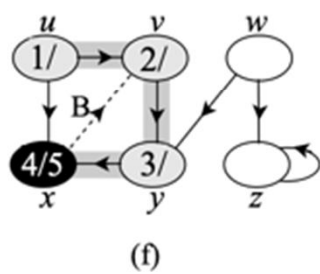
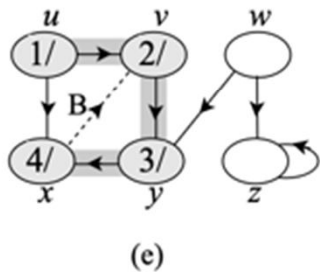
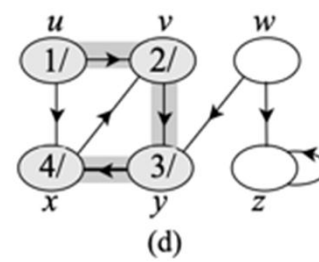
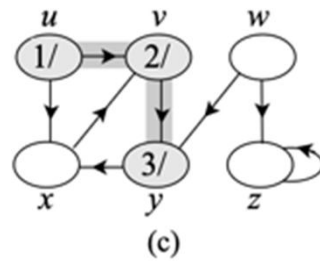
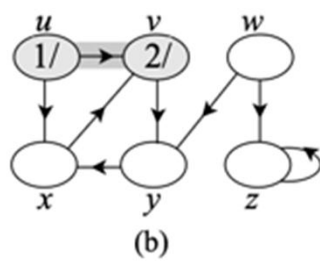
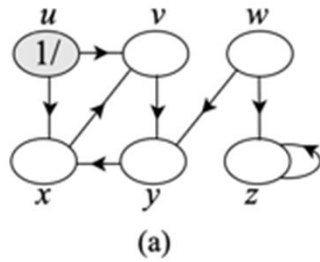
```
1   for each  $u \in V[G]$ 
2       color[u] = WHITE
3        $\pi[u] = \text{NIL}$ 
4   time = 0
5   for each  $u \in V[G]$ 
6       if color[u] == WHITE
7           DFSVisit(G, u)
```

THUẬT GIẢI DFS

DFSVisit(G, u)

```
1  time = time + 1           // đỉnh trắng u được phát hiện
2  d[u] = time
3  color[u] = GRAY
4  for each  $v \in \text{Adj}[u]$       // duyệt đỉnh v theo cách (u, v)
5      if color[v] == WHITE
6           $\pi[v] = u$ 
7          DFSVisit( $G, v$ )
8  color[u] = BLACK          // tô đen u khi nó hoàn thành tìm kiếm u
9  time = time + 1
10 f[u] = time
```

THUẬT GIẢI DFS



PHÂN TÍCH DFS

- Nếu chưa tính thời gian thực thi DFSVisit, vòng lặp 1-3 và 5-7 có chi phí là $O(V)$
- Trong một lần thực thi DFSVisit(u), vòng lặp 4-7 thực hiện $|Adj[u]|$ lần
- Vì $\sum_{u \in V} |Adj[u]| = O(E)$, nên tổng chi phí thực thi dòng 4-7 của DFSVisit là $O(E)$.
- Vậy thời gian chạy của DFS là $O(V+E)$

TÍNH CHẤT CỦA DFS

- Kết thúc thuật giải DFS trên đồ thị G , một rừng tìm kiếm theo chiều sâu được tạo ra với số cây trong rừng có gốc tại các đỉnh u tương ứng là số lần thực thi $\text{DFSVisit}(u)$ trong vòng lặp 5-7 của $\text{DFS}(G)$
- Số cây trong rừng tìm kiếm theo chiều sâu trên đồ thị G cũng là số thành phần liên thông của đồ thị G

TÍNH CHẤT CỦA DFS

CONNECTED-COMPONENT(G)

```
1 for each  $u \in V[G]$ 
2     do color[u] = white
3 d = 0
4 for each  $u \in V[G]$ 
5     do if color[u] == white
6         then DFSVisit(u)
7             d = d+1
8 return d           //d là số thành phần liên thông
```

TÍNH CHẤT CỦA DFS

- Nếu $d[u] < d[v]$ thì $f[v] < f[u]$ với mọi $u, v \in G$
- Đỉnh v là con cháu của đỉnh u trong rừng tìm kiếm theo chiều sâu trên đồ thị G nếu và chỉ nếu $d[u] < d[v] < f[v] < f[u]$

ĐỌC VÀ TÌM HIỂU Ở NHÀ

- Đọc chương 22 sách Introduction to Algorithms của Cormen và cộng sự
- Làm bài tập về nhà chương 3 đã cho trong DS bài tập