

TOÁN RỜI RẠC 2

CHƯƠNG 2

Giảng viên: Vũ Văn Thỏa

CHƯƠNG 2. CÁC THUẬT TOÁN TÌM KIẾM TRÊN ĐỒ THỊ

- Bài toán tìm kiếm
- Tìm kiếm theo chiều sâu
- Tìm kiếm theo chiều rộng
- Ứng dụng

■ Đặt bài toán:

Input: Đồ thị $G = (V, E)$ gồm n đỉnh, m cạnh;

Một đỉnh $u \in G$;

Output: Thứ tự tìm kiếm các đỉnh $v \in G$ bắt đầu từ đỉnh u và các cạnh sử dụng trong quá trình tìm kiếm;

■ Yêu cầu:

Mỗi đỉnh $v \in G$ chỉ được tìm kiếm đúng 1 lần.

Phương pháp giải bài toán tìm kiếm

- Thuật toán DFS
- Thuật toán BFS



2.1 Thuật toán tìm kiếm theo chiều sâu (Depth-First Search)

- Giới thiệu thuật toán
- Mô tả thuật toán
- Cài đặt và kiểm nghiệm thuật toán

2.1.1 Giới thiệu thuật toán

Bước khởi tạo: Tất cả các đỉnh $v \in G$ chưa được xét ($vs[v] = 0$) và chưa sử dụng cạnh nào liên thuộc v ($e[v] = 0$);

Bước 1: Tìm kiếm theo chiều sâu bắt đầu từ u bằng cách thăm u và đánh dấu u được xét ($vs[u] = 1$);

Bước 2: Chọn một đỉnh v kề với u và chưa được xét;

Bước 3: Nếu chọn được v thì cập nhật $e[v] = u$ (cạnh từ u đến v được sử dụng) và quay lại bước 1 với v đóng vai trò u ;

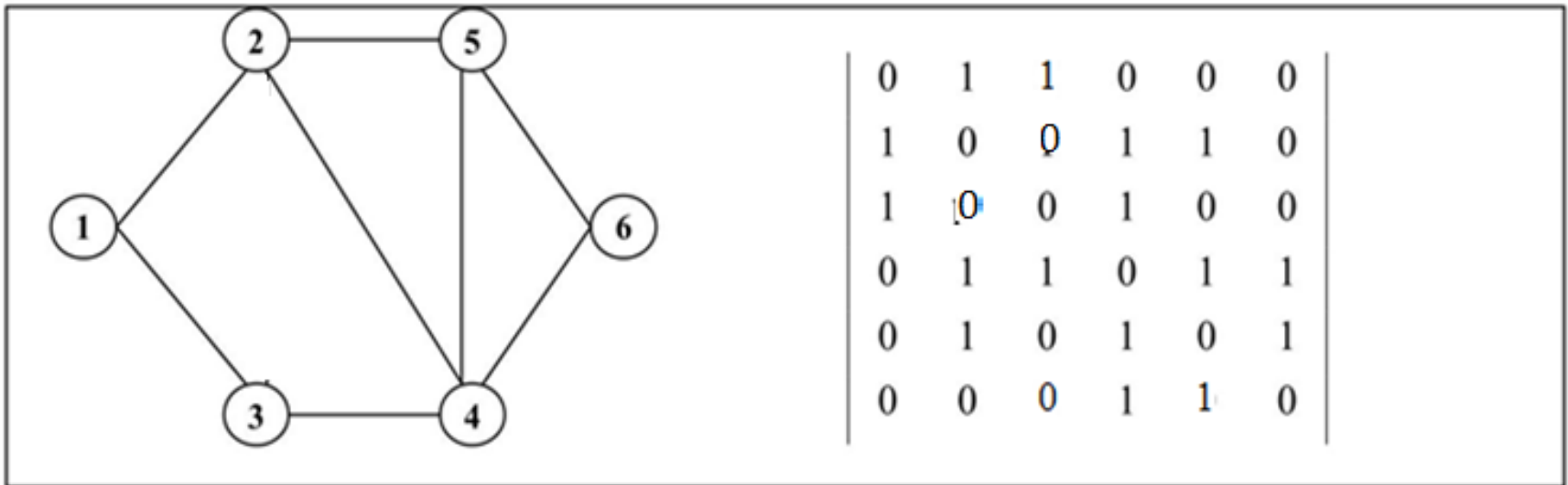
Bước 4: Nếu không chọn được v thì quay lại bước 2 và đỉnh đóng vai trò u là đỉnh i có thứ tự duyệt ngay trước đỉnh đóng vai trò u hiện thời;

Bước 5: Nếu tất cả các đỉnh kề của u đều đã được xét thì dừng;

2.1.2 Mô tả thuật toán

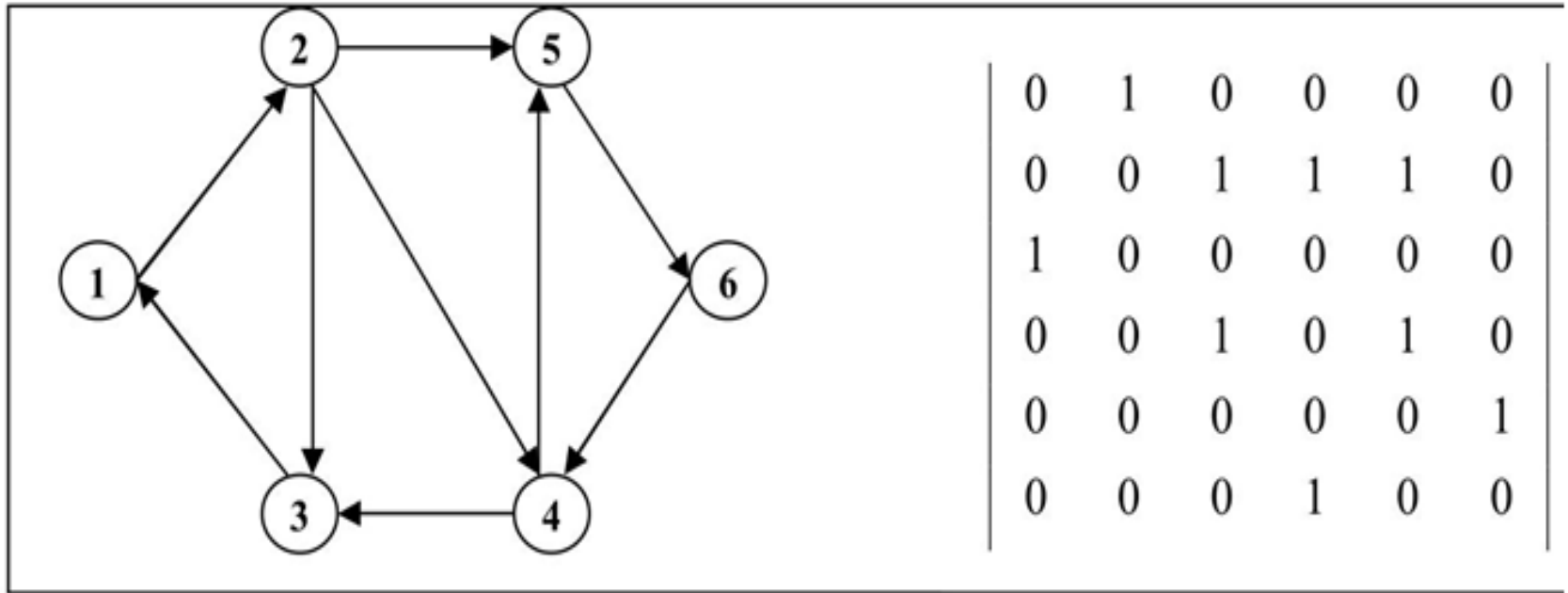
Thuật toán: Dfs(u){
 Thăm(u);
 vs[u] = 1;
 for v \in ke(u) do
 if (vs[v] = 0) {
 e[v] = u; Dfs(v);
 }
}

Ví dụ 1: Dfs(u) với $u = 1$, G vô hướng



$Dfs(1) = \{1(0); 2(1); 4(2); 3(4); 5(4); 6(5)\}$

Ví dụ 2: Dfs(u) với $u = 2$, G có hướng



$Dfs(2) = \{2(0); 3(2); 1(3); 4(2); 5(4); 6(5)\}$

2.1.3 Cài đặt và kiểm nghiệm thuật toán

Cài đặt 1: (Đệ qui)

```
// G cho bởi ma trận kề a[i][j]
int a[100][100], n, u, vs[100], e[100];
void DfsDequy(int u) { int v;
    cout << u << " ";
    vs[u]= 1;
    for (v= 1; v<=n; v++)
        if (vs[v]==0 && a[u][v]==1){ e[v]= u;
            DfsDequy(v);
        }
}
```

Cài đặt 2: (Sử dụng ngăn xếp)

```
// G cho bởi ma trận kề a[i][j]
int a[100][100], n, u, vs[100], e[100], s[100];
void DfsNx(int u){int top= 1; s[top]= u; vs[u] = 1;
    while (top > 0){ int v = s[top]; int ok = 1;
        for (int i= 1; i<=n; i++)
            if (vs[i]==0 && a[v][i]==1) { top++; s[top] = i;
                vs[i] = 1; e[i]= v; ok = 0; break;
            }
        if (ok==1) top--;
    }
}
```

Ví dụ 3: Dfs(u), $u = 1$, G vô hướng gồm 13 đỉnh biểu diễn bởi ma trận kề

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	1	1	0	0	0	0	0	0	0	0	0
2	1	0	1	1	0	1	0	0	0	0	0	0	0
3	1	1	0	1	1	0	0	0	0	0	0	0	0
4	1	1	1	0	0	0	1	0	0	0	0	0	0
5	0	0	1	0	0	1	1	1	0	0	0	1	0
6	0	1	0	0	1	0	1	0	0	0	0	1	0
7	0	0	0	1	1	1	0	1	0	0	0	0	0
8	0	0	0	0	1	0	1	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1	1	0	1
10	0	0	0	0	0	0	0	0	1	0	1	1	1
11	0	0	0	0	0	0	0	0	1	1	0	0	1
12	0	0	0	0	1	1	0	1	0	1	0	0	0
13	0	0	0	0	0	0	0	0	1	1	1	0	0

$Dfs(1) = \{1(0); 2(1); 3(2); 4(3); 7(4); 5(7); 6(5); 12(6); 8(12); 10(12); 9(10); 11(9); 13(11)\}$

Ví dụ 4: Dfs(u), $u = 1$, G có hướng gồm 13 đỉnh biểu diễn bởi ma trận kề

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0	1
4	1	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	1	0	1	0
7	0	0	0	0	0	0	0	0	0	0	1	0	1
8	0	0	0	1	0	0	0	0	0	0	0	1	0
9	0	0	0	0	1	0	1	0	0	0	0	0	0
10	0	1	1	0	0	0	0	0	0	0	0	0	0
11	0	1	0	0	0	0	0	1	0	0	0	0	0
12	0	0	0	1	0	0	0	0	0	1	0	0	0
13	0	0	0	0	0	0	0	0	1	0	1	0	0

$Dfs(1) = \{1(0); 6(1); 10(6); 2(10); 3(2), 9(3); 5(9); 7(5); 11(7); 8(11); 4(8); 12(8); 13(7)\}$

Độ phức tạp của thuật toán DFS

Độ phức tạp của thuật toán DFS phụ thuộc vào phương pháp biểu diễn đồ thị:

- Biểu diễn đồ thị bằng ma trận kề

Độ phức tạp: $O(n^2)$, với n là số đỉnh;

- Biểu diễn đồ thị bằng danh sách cạnh

Độ phức tạp: $O(nxm)$, với n là số đỉnh, m là số cạnh;

- Biểu diễn đồ thị bằng danh sách kề

Độ phức tạp: $O(n)$, với n là số đỉnh;

2.2 Thuật toán tìm kiếm theo chiều rộng (Breadth - first Search)

- Giới thiệu thuật toán
- Mô tả thuật toán
- Cài đặt và kiểm nghiệm thuật toán

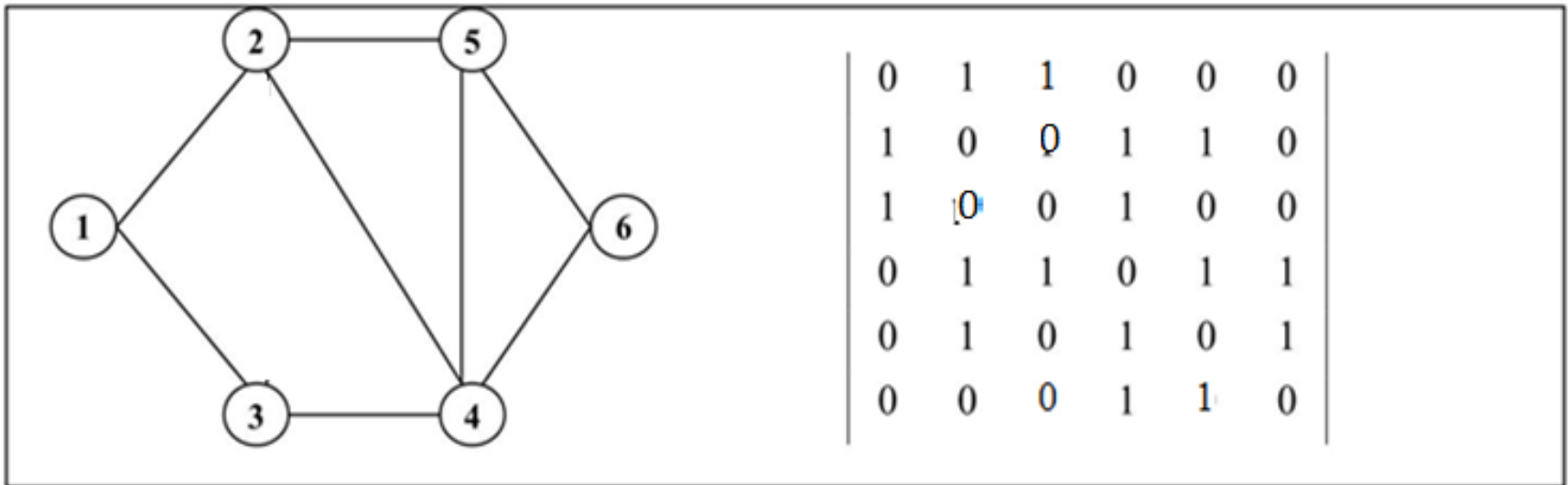
2.2.1 Giới thiệu thuật toán

- **Bước khởi tạo:** Tất cả các đỉnh $v \in G$ chưa được xét ($vs[v] = 0$) và chưa sử dụng cạnh liên thuộc v ($e[v] = 0$);
- **Bước 1:** Xây dựng hàng đợi q bắt đầu từ u và đánh dấu u đã xét ($vs[u] = 1$);
- **Bước 2:** Nếu q rỗng thì kết thúc. Ngược lại, lấy v ra khỏi hàng đợi và thăm v ;
- **Bước 3:** Đưa vào hàng đợi tất cả các đỉnh i kề với v và chưa được xét, đánh dấu i đã xét ($vs[i] = 1$), cập nhật $e[i] = v$ (cạnh từ v đến i được sử dụng);
- **Bước 4:** Quay lại bước 2.

2.2.2 Mô tả thuật toán

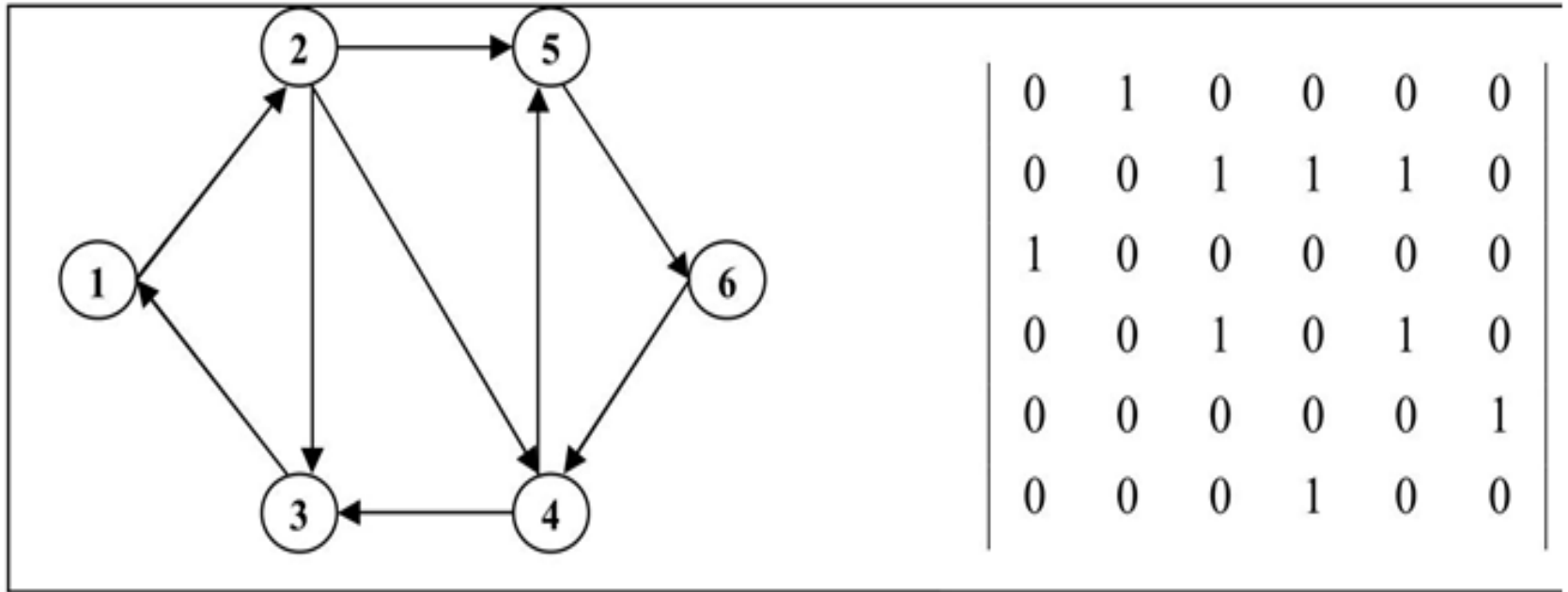
Thuật toán: Bfs(u) { $q = \emptyset$;
 <Đưa u vào q>;
 $vs[u] = 1$;
 while $q \neq \emptyset$ { <Lấy v ra khỏi q>; Thăm(v);
 for $i \in ke(v)$ do
 if ($vs[i] = 0$) { <Đưa i vào q>;
 $vs[i] = 1$; $e[i] = v$;
 }
 }
 }

Ví dụ 5: Bfs(u), $u = 1$, G vô hướng



$Bfs(1) = \{1(0); 2(1), 3(1); 4(2), 5(2); 6(4)\}$

Ví dụ 6: Bfs(u), $u = 2$, G có hướng



$\text{Bfs}(2) = \{2(0); 3(2), 4(2), 5(2); 1(3); 6(5)\}$

2.2.3 Cài đặt và kiểm nghiệm thuật toán

```
// G cho bởi ma trận kề a[i][j]
int a[100][100], n, u, vs[100], e[100], q[100];
void Bfs(int u){ int v, dq = 1, cq = 0;
    cq++; q[cq] = u; vs[u] = 1;
    while (dq <= cq){ v = q[dq]; dq++;
        cout << v << " ";
        for (int i= 1; i<=n; i++)
            if (vs[i]==0 && a[v][i]==1) {
                cq++; q[cq] = i; vs[i] = 1; e[i]= v; }
    }
}
```

Ví dụ 7: Bfs(u), u = 1, G vô hướng gồm 13 đỉnh biểu diễn bởi ma trận kề

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	1	1	0	0	0	0	0	0	0	0	0
2	1	0	1	1	0	1	0	0	0	0	0	0	0
3	1	1	0	1	1	0	0	0	0	0	0	0	0
4	1	1	1	0	0	0	1	0	0	0	0	0	0
5	0	0	1	0	0	1	1	1	0	0	0	1	0
6	0	1	0	0	1	0	1	0	0	0	0	1	0
7	0	0	0	1	1	1	0	1	0	0	0	0	0
8	0	0	0	0	1	0	1	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1	1	0	1
10	0	0	0	0	0	0	0	0	1	0	1	1	1
11	0	0	0	0	0	0	0	0	1	1	0	0	1
12	0	0	0	0	1	1	0	1	0	1	0	0	0
13	0	0	0	0	0	0	0	0	1	1	1	0	0

$Bfs(1) = \{1(0); 2(1), 3(1), 4(1); 6(2); 5(3); 7(4); 12(6); 8(5); 10(12); 9(10), 11(10), 13(10)\}$

Ví dụ 8: Bfs(u), $u = 1$, G có hướng gồm 13 đỉnh biểu diễn bởi ma trận kề

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0	1
4	1	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	1	0	1	0
7	0	0	0	0	0	0	0	0	0	0	1	0	1
8	0	0	0	1	0	0	0	0	0	0	0	1	0
9	0	0	0	0	1	0	1	0	0	0	0	0	0
10	0	1	1	0	0	0	0	0	0	0	0	0	0
11	0	1	0	0	0	0	0	1	0	0	0	0	0
12	0	0	0	1	0	0	0	0	0	1	0	0	0
13	0	0	0	0	0	0	0	0	1	0	1	0	0

Bfs(1) = {1(0); 6(1); 10(6), 12(6); 2(10), 3(10); 4(12); 8(2); 9(3), 13(3); 5(9), 7(9); 11(13)}

Độ phức tạp thuật toán BFS

Độ phức tạp của thuật toán BFS phụ thuộc vào phương pháp biểu diễn đồ thị:

- Biểu diễn đồ thị bằng ma trận kề

Độ phức tạp: $O(n^2)$, với n là số đỉnh;

- Biểu diễn đồ thị bằng danh sách cạnh

Độ phức tạp: $O(nxm)$, với n là số đỉnh, m là số cạnh;

- Biểu diễn đồ thị bằng danh sách kề

Độ phức tạp: $O(n)$, với n là số đỉnh;

Ghi chú

- Khi thực hiện Dfs(u)/Bfs(u) các đỉnh được duyệt đến là các đỉnh có đường đi tới xuất phát từ u.
- Đối với đồ thị vô hướng G, tập hợp các đỉnh thuộc Dfs(u)/Bfs(u) là một thành phần liên thông của G chứa đỉnh u.

2.3 Ứng dụng các thuật toán tìm kiếm trên đồ thị

- Duyệt tất cả các đỉnh của đồ thị
- Tìm đường đi giữa các đỉnh trên đồ thị
- Tính liên thông trong đồ thị vô hướng
- Đỉnh trụ và cạnh cầu trong đồ thị vô hướng
- Tính liên thông trong đồ thị có hướng

2.3.1 Duyệt tất cả các đỉnh của đồ thị

- **Input:** Đồ thị $G = (V, E)$ gồm n đỉnh, m cạnh;
- **Output:** Thứ tự thăm tất cả các đỉnh $v \in G$ bắt đầu từ đỉnh 1;

Thuật toán: *Duyệt tất cả các đỉnh của đồ thị;*

- **Bước khởi tạo:** Tất cả các đỉnh $v \in G$ chưa được thăm ($vs[v] = 0$);
- **Bước 1:** Nếu tất cả các đỉnh đều được thăm thì kết thúc;
- **Bước 2:** Chọn $v \in G$ chưa được thăm (bắt đầu từ $v = 1$);
- **Bước 3:** Thực hiện DFS(v)/BFS(v);
- **Bước 4:** Quay lại bước 1;

Cài đặt 1 (Sử dụng DFS):

```
void DuyệtDfs(){int v;  
    for (v = 1; v <= n; v++) {  
        vs[v] = 0; e[v]= 0;}  
    for (v = 1; v <= n; v++)  
        if (vs[v] == 0) DfsDequy(v);  
}
```

Cài đặt 2 (Sử dụng BFS):

```
void DuyệtBfs(){int v;  
    for (v = 1; v <= n; v++) {  
        vs[v] = 0; e[v]= 0;}  
    for (v = 1; v <= n; v++)  
        if (vs[v] == 0) Bfs(v);  
}
```



Ví dụ 9: Duyệt các đỉnh của đồ thị vô hướng cho bởi danh sách kề

$$\begin{aligned} Ke(1) &= \{6, 7\} & Ke(2) &= \{8, 9, 10\} & Ke(3) &= \{4\} & Ke(4) &= \{3\} & Ke(5) &= \{\} \\ Ke(6) &= \{1, 7\} & Ke(7) &= \{1, 6\} & Ke(8) &= \{2, 10\} & Ke(9) &= \{2, 10\} & Ke(10) &= \{2, 8, 9\} \end{aligned}$$

■ Sử dụng DFS:

$$Dfs(1) = \{1(0); 6(1); 7(6)\}; Dfs(2) = \{2(0); 8(2); 10(8); 9(10)\}$$

$$Dfs(3) = \{3(0); 4(3)\}; Dfs(5) = \{5(0)\}$$

⇒ Thứ tự duyệt: 1, 6, 7, 2, 8, 10, 9, 3, 4, 5

■ Sử dụng BFS:

$$Bfs(1) = \{1(0); 6(1), 7(1)\}; Bfs(2) = \{2(0); 8(2), 9(2), 10(2)\}$$

$$Bfs(3) = \{3(0); 4(3)\}; Bfs(5) = \{5(0)\}$$

⇒ Thứ tự duyệt: 1, 6, 7, 2, 8, 9, 10, 3, 4, 5

Ví dụ 10: Duyệt các đỉnh của đồ thị có hướng cho bởi danh sách kề

$$\begin{aligned} \text{Ke}(1) &= \{6, 7\} & \text{Ke}(2) &= \{8, 9\} & \text{Ke}(3) &= \{4\} & \text{Ke}(4) &= \{5\} & \text{Ke}(5) &= \{3\} \\ \text{Ke}(6) &= \{7\} & \text{Ke}(7) &= \{1\} & \text{Ke}(8) &= \{10\} & \text{Ke}(9) &= \{10\} & \text{Ke}(10) &= \{2\} \end{aligned}$$

■ Sử dụng DFS:

$$\text{Dfs}(1) = \{1(0); 6(1); 7(6)\}; \text{Dfs}(2) = \{2(0); 8(2); 10(8); 9(2)\}$$

$$\text{Dfs}(3) = \{3(0); 4(3); 5(4)\}$$

⇒ Thứ tự duyệt: 1, 6, 7, 2, 8, 10, 9, 3, 4, 5

■ Sử dụng BFS:

$$\text{Bfs}(1) = \{1(0); 6(1), 7(1)\}; \text{Bfs}(2) = \{2(0); 8(2), 9(2); 10(8)\}$$

$$\text{Bfs}(3) = \{3(0); 4(3); 5(4)\}$$

⇒ Thứ tự duyệt: 1, 6, 7, 2, 8, 9, 10, 3, 4, 5

2.3.2 Tìm đường đi giữa các đỉnh trên đồ thị

Khái niệm

- Đường đi độ dài k từ u tới $v \in G$ là dãy các đỉnh x_0, x_1, \dots, x_k , trong đó $x_0 = u$, $x_k = v$ và $(x_{i-1}, x_i) \in E$, $1 \leq i \leq k$.
- Đường đi là đơn nếu không chứa một cạnh quá một lần.

Định lý (Đếm số lượng đường đi giữa cặp đỉnh)

Cho G là đồ thị với ma trận kề A gồm n đỉnh đánh số $1, 2, \dots, n$. Số các đường đi khác nhau độ dài r từ i đến j (r nguyên dương) bằng giá trị của phần tử (i, j) của ma trận A^r .

Ví dụ 11: Tìm số lượng đường đi độ dài 4 từ đỉnh 1 đến 4 trên đồ thị vô hướng

Có ma trận kề của G:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \Rightarrow A^4 = \begin{bmatrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{bmatrix} \Rightarrow \text{có 8 đường đi độ dài 4 từ 1 đến 4.}$$



Tìm đường đi trên đồ thị từ đỉnh u đến đỉnh v

Input: Đồ thị $G = (V, E)$ gồm n đỉnh, m cạnh;

Hai đỉnh $u, v \in G$;

Output: Đường đi từ đỉnh u đến đỉnh v ;

Thuật toán: Tìm đường đi từ đỉnh u đến đỉnh v ;

Bước khởi tạo: Tất cả $x \in G$ chưa được thăm ($vs[x] = 0$) và chưa xác định cạnh ($e[x] = 0$);

Bước 1: Thực hiện DFS(u)/BFS(u);

Bước 2: (Trả lại kết quả)

Nếu $vs[v] = 0 \Rightarrow$ không có đường đi từ u đến v ;

Nếu $vs[v] = 1 \Rightarrow$ xuất đường đi từ u đến v bằng cách sử dụng $e[]$;

Cài đặt 1 (Sử dụng DFS):

```
int a[100][100], n, u, v, vs[100], e[100];  
void DuongDiDfs(int u, int v) {  
    for (int x= 1; x<=n; x++){ vs[x]= 0; e[x]= 0;}  
    DfsDequy(u);  
    if (vs[v] == 1){ int t= v;  
        while(t> 0){ cout << t << " <- ";  
            t= e[t]; }  
        }; else  
        cout << " NO ";  
}
```

Cài đặt 2 (Sử dụng BFS):

```
int a[100][100], n, u, v, vs[100], e[100];  
void DuongDiBfs(int u, int v) {  
    for (int x= 1; x<=n; x++){ vs[x]= 0; e[x]= 0;}  
    Bfs(u);  
    if (vs[v] == 1){ int t= v;  
        while(t> 0){ cout << t << " <- ";  
            t= e[t]; }  
        }; else  
        cout << " NO ";  
}
```

Ví dụ 12: Tìm đường đi từ $u=2$ đến $v=13$ trên G vô hướng gồm 13 đỉnh cho bởi ma trận kề

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	1	1	0	0	0	0	0	0	0	0	0
2	1	0	1	1	0	1	0	0	0	0	0	0	0
3	1	1	0	1	1	0	0	0	0	0	0	0	0
4	1	1	1	0	0	0	1	0	0	0	0	0	0
5	0	0	1	0	0	1	1	1	0	0	0	1	0
6	0	1	0	0	1	0	1	0	0	0	0	1	0
7	0	0	0	1	1	1	0	1	0	0	0	0	0
8	0	0	0	0	1	0	1	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1	1	0	1
10	0	0	0	0	0	0	0	0	1	0	1	1	1
11	0	0	0	0	0	0	0	0	1	1	0	0	1
12	0	0	0	0	1	1	0	1	0	1	0	0	0
13	0	0	0	0	0	0	0	0	1	1	1	0	0

$Dfs(2) = \{2(0); 1(2); 3(1); 4(3); 7(4); 5(7); 6(5); 12(6); 8(12); 10(12); 9(10); 11(9); 13(11)\}$

\Rightarrow Đường đi từ 2 đến 13: $13 \leftarrow 11 \leftarrow 9 \leftarrow 10 \leftarrow 12 \leftarrow 6 \leftarrow 5 \leftarrow 7 \leftarrow 4 \leftarrow 3 \leftarrow 1 \leftarrow 2$

$Bfs(2) = \{2(0); 1(2), 3(2), 4(2), 6(2); 5(3); 7(4); 12(6); 8(5); 10(12); 9(10), 11((10), 13(10))\}$

\Rightarrow Đường đi từ 2 đến 13: $13 \leftarrow 10 \leftarrow 12 \leftarrow 6 \leftarrow 2$

Ví dụ 13: Tìm đường đi từ $u=13$ đến $v=1$ trên G có hướng gồm 13 đỉnh cho bởi ma trận kề

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0	1
4	1	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	1	0	1	0
7	0	0	0	0	0	0	0	0	0	0	1	0	1
8	0	0	0	1	0	0	0	0	0	0	0	1	0
9	0	0	0	0	1	0	1	0	0	0	0	0	0
10	0	1	1	0	0	0	0	0	0	0	0	0	0
11	0	1	0	0	0	0	0	1	0	0	0	0	0
12	0	0	0	1	0	0	0	0	0	1	0	0	0
13	0	0	0	0	0	0	0	0	1	0	1	0	0

$Dfs(13) = \{13(0); 9(13); 5(9); 7(5); 11(7); 2(11); 3(2); 8(2); 4(8), 1(4); 6(1), 10(6); 12(6)\}$

\Rightarrow Đường đi từ 13 đến 1: $1 \leftarrow 4 \leftarrow 8 \leftarrow 2 \leftarrow 11 \leftarrow 7 \leftarrow 5 \leftarrow 9 \leftarrow 13$

$Bfs(13) = \{13(0); 9(13), 11(13); 5(9), 7(9); 2(11), 8(11); 3(2); 4(8), 12(8); 1(4), 6(4); 10(12)\}$

\Rightarrow Đường đi từ 13 đến 1: $1 \leftarrow 4 \leftarrow 8 \leftarrow 11 \leftarrow 13$

Ghi chú:

- Độ phức tạp tính toán của thuật toán tìm đường đi từ đỉnh u đến đỉnh v thuộc G bằng độ phức tạp của DFS(u)/BFS(u) sử dụng cài đặt.
- Đường đi từ u đến v sử dụng BFS là đường đi có ít cạnh nhất.

2.3.3 Tính liên thông trong đồ thị vô hướng

- **Định nghĩa** : Một đồ thị vô hướng *liên thông* \Leftrightarrow có đường đi giữa hai đỉnh bất kỳ.
- Đồ thị vô hướng G không liên thông là hợp các đồ thị con liên thông, không có đỉnh chung gọi là các *thành phần liên thông* của G .
- Đồ thị vô hướng G *liên thông* \Leftrightarrow số *thành phần liên thông* của G là 1.



Thuật toán kiểm tra tính liên thông của đồ thị vô hướng

- **Input:** Đồ thị vô hướng $G = (V, E)$ gồm n đỉnh và m cạnh;
- **Output:** Giá trị 1 nếu G liên thông, giá trị 0 nếu G không liên thông;

Thuật toán: Kiểm tra tính liên thông của đồ thị vô hướng;

Bước 1: Thực hiện Dfs(1)/Bfs(1);.

Bước 2: Tính số lượng k các đỉnh được duyệt;

Bước 3: Nếu $k = n$ xuất 1; nếu $k < n$ xuất 0.

Cài đặt 1: Sử dụng DFS

// G cho bởi ma trận kề $a[i][j]$

```
int a[100][100], vs[100], n;
```

```
int ItDfs() {int v;
```

```
    for (v= 1; v<= n; v++) vs[v]= 0;
```

```
    DfsDequy(1);
```

```
        for (v= 1; v<= n; v++) if (vs[v]== 0) return(0);
```

```
        return(1);
```

```
}
```

Cài đặt 2: Sử dụng BFS

// G cho bởi ma trận kề $a[i][j]$

```
int a[100][100], vs[100], n;
```

```
int ItBfs() {int v;
```

```
    for (v= 1; v<= n; v++) vs[v]= 0;
```

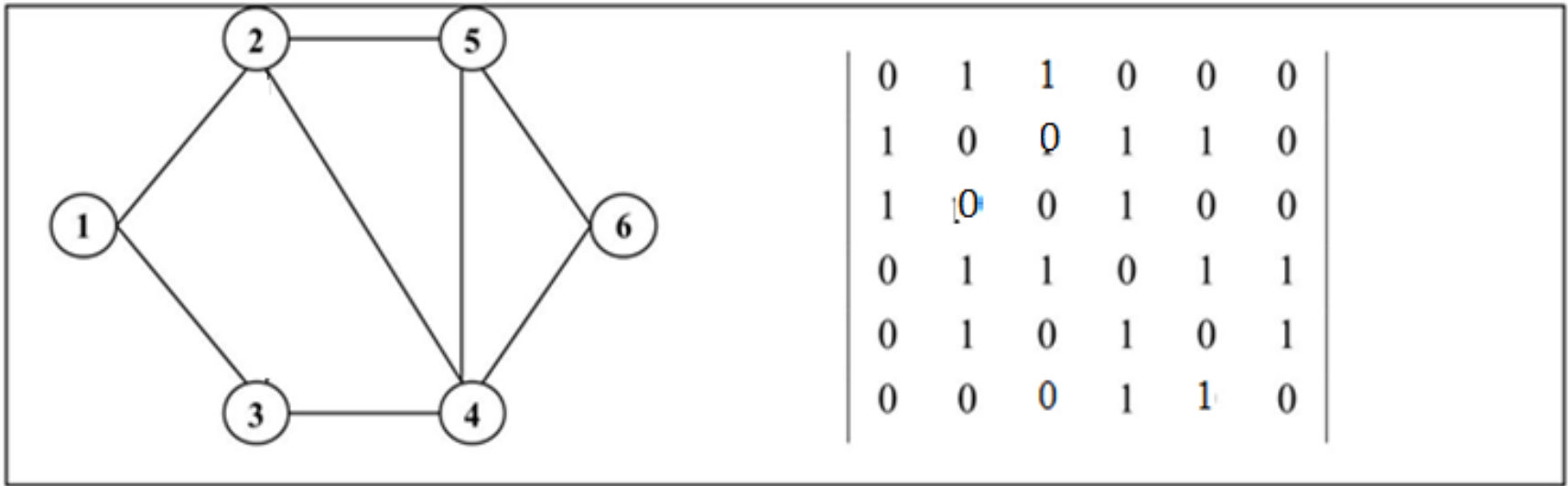
```
    Bfs(1);
```

```
        for (v= 1; v<= n; v++) if (vs[v]== 0) return(0);
```

```
    return(1);
```

```
}
```

Ví dụ 14: Tính liên thông của G vô hướng



- **Cách 1:** Dfs(1) = {1(0); 2(1); 4(2); 3(4); 5(4); 6(5)} = V
 \Rightarrow G liên thông
- **Cách 2:** Bfs(1) = {1(0); 2(1), 3(1); 4(2), 5(2); 6(4)} = V
 \Rightarrow G liên thông

Thành phần liên thông của đồ thị vô hướng

- **Input:** Đồ thị vô hướng $G = (V, E)$ gồm n đỉnh và m cạnh;
- **Output:**
 - Số k các thành phần liên thông của G ;
 - Số thứ tự của thành phần liên thông chứa u của mọi đỉnh u ;

Thuật toán: Tìm các thành phần liên thông của đồ thị vô hướng;

Bước khởi tạo: $k = 0$; $lt[u] = 0$ với mọi đỉnh u ;

Bước 1: Nếu mọi đỉnh u đều có $lt[u] > 0$ thì chuyển bước 3, ngược lại chọn đỉnh u có $lt[u] = 0$;

Bước 2: $k = k + 1$, thực hiện Dfs(u)/Bfs(u) và gán cho các đỉnh i được duyệt tới $lt[i] = k$ (thay $vs[]$ bởi $lt[]$); quay lại bước 1;

Bước 3: Xuất k và $lt[u]$ với mọi đỉnh u ;

Cài đặt 1: Sử dụng DFS

// G cho bởi ma trận kề $a[i][j]$

```
int a[100][100], lt[100], n;
```

```
int tpltDfs(){int u, int k= 0;
```

```
    for (u= 1; u<= n; u++) lt[u]= 0;
```

```
    for (u= 1; u<= n; u++) if (lt[u] == 0) {  
        k++; DfsDequy(u); }
```

```
    return(k);
```

```
}
```

Cài đặt 2: Sử dụng BFS

// G cho bởi ma trận kề $a[i][j]$

```
int a[100][100], lt[100], n, dq, cq, q[100];
```

```
int tpltBfs(){int u, int k= 0;
```

```
    for (u= 1; u<= n; u++) lt[u]= 0;
```

```
    for (u= 1; u<= n; u++) if (lt[u] == 0) {  
        k++; Bfs(u); }
```

```
    return(k);
```

```
}
```

Ví dụ 15: Tìm số thành phần liên thông của G vô hướng cho bởi danh sách kề

$$Ke(1) = \{6, 7\} \quad Ke(2) = \{8, 9, 10\} \quad Ke(3) = \{4\} \quad Ke(4) = \{3\} \quad Ke(5) = \{\}$$

$$Ke(6) = \{1, 7\} \quad Ke(7) = \{1, 6\} \quad Ke(8) = \{2, 10\} \quad Ke(9) = \{2, 10\} \quad Ke(10) = \{2, 8, 9\}$$

■ Sử dụng DFS:

$$Dfs(1) = \{1(0); 6(1); 7(6)\}; \quad Dfs(2) = \{2(0); 8(2); 10(8); 9(10)\}$$

$$Dfs(3) = \{3(0); 4(3)\}; \quad Dfs(5) = \{5(0)\}$$

⇒ Số thành phần liên thông $k = 4$;

Thành phần liên thông 1 = $\{1, 6, 7\}$;

Thành phần liên thông 2 = $\{2, 8, 9, 10\}$;

Thành phần liên thông 3 = $\{3, 4\}$;

Thành phần liên thông 4 = $\{5\}$;

Ví dụ 15: Sử dụng BFS

$$\begin{aligned} Ke(1) &= \{6, 7\} & Ke(2) &= \{8, 9, 10\} & Ke(3) &= \{4\} & Ke(4) &= \{3\} & Ke(5) &= \{\} \\ Ke(6) &= \{1, 7\} & Ke(7) &= \{1, 6\} & Ke(8) &= \{2, 10\} & Ke(9) &= \{2, 10\} & Ke(10) &= \{2, 8, 9\} \end{aligned}$$

■ Sử dụng BFS:

$$Bfs(1) = \{1(0); 6(1), 7(1)\}; Bfs(2) = \{2(0); 8(2), 9(2), 10(2)\}$$

$$Bfs(3) = \{3(0); 4(3)\}; Bfs(5) = \{5(0)\}$$

⇒ Số thành phần liên thông $k = 4$;

Thành phần liên thông 1 = $\{1, 6, 7\}$;

Thành phần liên thông 2 = $\{2, 8, 9, 10\}$;

Thành phần liên thông 3 = $\{3, 4\}$;

Thành phần liên thông 4 = $\{5\}$;

2.3.4 Đỉnh trụ và cạnh cầu trong đồ thị vô hướng

- Đỉnh $u \in G$ vô hướng là **đỉnh trụ** (đỉnh cắt hay đỉnh khớp) \Leftrightarrow xóa u và các cạnh liên thuộc sẽ tạo ra một đồ thị mới $G \setminus \{u\}$ có nhiều thành phần liên thông hơn G .
- Cạnh $e = (u, v) \in G$ là **cạnh cầu** (hay cạnh cắt) \Leftrightarrow xóa e sẽ được đồ thị mới $G \setminus \{e\}$ có nhiều thành phần liên thông hơn G .

Tìm các đỉnh trụ của đồ thị vô hướng

- **Input:** Đồ thị vô hướng $G = (V, E)$ gồm n đỉnh và m cạnh;
- **Output:** Các đỉnh trụ của G ;

Thuật toán: *Tìm đỉnh trụ của đồ thị vô hướng G ;*

Bước 1: Tìm số k thành phần liên thông của G ;

Bước 2: Xét mọi đỉnh $u \in G$:

2.1: Bỏ u và các cạnh liên thuộc u và tính số thành phần liên thông l của đồ thị $G \setminus \{u\}$;

2.2: Nếu $l > k$ thì ghi nhận u là đỉnh trụ;

2.3: Trả lại u và các cạnh liên thuộc u ;

Bước 3: Xuất danh sách các đỉnh trụ;

- Sinh viên tự cài đặt xem như bài tập

Ví dụ 16: Tìm đỉnh trụ của G vô hướng cho bởi danh sách kề

$$Ke(1) = \{2, 3, 4\} \quad Ke(2) = \{1, 3\} \quad Ke(3) = \{1, 2\} \quad Ke(4) = \{1, 5, 6\} \quad Ke(5) = \{4, 6\} \quad Ke(6) = \{4, 5\}$$

■ Sử dụng DFS:

Số đỉnh $n = 6$

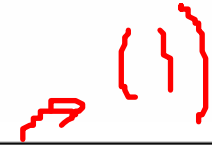
$$Dfs(1) = \{1(0); 2(1); 3(2); 4(1); 5(4); 6(5)\}$$

\Rightarrow Số thành phần liên thông của G: $k = 1$.

Ví dụ 16: Tìm đỉnh trụ với DFS

$$Ke(1) = \{2, 3, 4\} \quad Ke(2) = \{1, 3\} \quad Ke(3) = \{1, 2\} \quad Ke(4) = \{1, 5, 6\} \quad Ke(5) = \{4, 6\} \quad Ke(6) = \{4, 5\}$$

Lập bảng:



Đỉnh u	Số thành phần liên thông của $G \setminus \{u\}$	$l > k?$	Đỉnh trụ
1	$Dfs(2) = \{2(0); 3(2)\}, Dfs(4) = \{4(0); 5(4); 6(5)\} \Rightarrow l = 2$	Yes	1
2	$Dfs(1) = \{1(0); 3(1); 4(1); 5(4); 6(5)\} \Rightarrow l = 1$	No	-
3	$Dfs(1) = \{1(0); 2(1); 4(1); 5(4); 6(5)\} \Rightarrow l = 1$	No	-
4	$Dfs(1) = \{1(0); 2(1); 3(2)\}, Dfs(5) = \{5(0); 6(5)\} \Rightarrow l = 2$	Yes	4
5	$Dfs(1) = \{1(0); 2(1); 3(2); 4(1); 6(4)\} \Rightarrow l = 1$	No	-
6	$Dfs(1) = \{1(0); 2(1); 3(2); 4(1); 5(4)\} \Rightarrow l = 1$	No	-

Kết luận: G có 2 đỉnh trụ là 1 và 4.

Tìm các cạnh cầu của đồ thị vô hướng

- **Input:** Đồ thị vô hướng $G = (V, E)$ gồm n đỉnh và m cạnh;
- **Output:** Các cạnh cầu của G ;

Thuật toán: *Tìm cạnh cầu của đồ thị vô hướng G ;*

Bước 1: Tìm số k thành phần liên thông của G ;

Bước 2: Xét mọi cạnh $e \in G$:

2.1: Bỏ cạnh $e = (u,v)$ (các đỉnh u và v vẫn giữ lại) và tính số thành phần liên thông l của đồ thị $G \setminus \{e\}$;

2.2: Nếu $l > k$ thì ghi nhận e là cạnh cầu;

2.3: Trả lại e ;

Bước 3: Xuất danh sách các cạnh cầu;



Cài đặt thuật toán

- Sinh viên tự cài đặt xem như bài tập

Ví dụ 17: Tìm cạnh cầu của G vô hướng cho bởi danh sách kề

$$Ke(1) = \{2, 3, 4\} \quad Ke(2) = \{1, 3\} \quad Ke(3) = \{1, 2\} \quad Ke(4) = \{1, 5, 6\} \quad Ke(5) = \{4, 6\} \quad Ke(6) = \{4, 5\}$$

■ Sử dụng BFS:

Số đỉnh $n = 6$

$Bfs(1) = \{1(0); 2(1), 3(1), 4(1); 5(4), 6(4)\}$

\Rightarrow Số thành phần liên thông của G: $k = 1$.

Ví dụ 17: Tìm cạnh cầu với BFS

$$Ke(1) = \{2, 3, 4\} \quad Ke(2) = \{1, 3\} \quad Ke(3) = \{1, 2\} \quad Ke(4) = \{1, 5, 6\} \quad Ke(5) = \{4, 6\} \quad Ke(6) = \{4, 5\}$$

Lập bảng:

cạnh e	Số thành phần liên thông của $G \setminus \{e\}$	$l > k$?	Cạnh cầu
(1,2)	$Bfs(1) = \{1(0); 3(1), 4(1); 2(3); 5(4), 6(4)\} \Rightarrow l = 1$	No	
(1,3)	$Bfs(1) = \{1(0); 2(1), 4(1); 3(2); 5(4), 6(4)\} \Rightarrow l = 1$	No	
(1,4)	$Bfs(1) = \{1(0); 2(1), 3(1)\}, Bfs(4) = \{4(0); 5(4), 6(4)\} \Rightarrow l = 2$	Yes	(1,4)
(2,3)	$Bfs(1) = \{1(0); 2(1), 3(1), 4(1); 5(4), 6(4)\} \Rightarrow l = 1$	No	
(4,5)	$Bfs(1) = \{1(0); 2(1), 3(1), 4(1); 6(4); 5(6)\} \Rightarrow l = 1$	No	
(4,6)	$Bfs(1) = \{1(0); 2(1), 3(1), 4(1); 5(4); 6(5)\} \Rightarrow l = 1$	No	
(5,6)	$Bfs(1) = \{1(0); 2(1), 3(1), 4(1); 5(4), 6(4)\} \Rightarrow l = 1$	No	

Kết luận: G có 1 cạnh cầu là (1,4).

- Khi kiểm tra đỉnh u có phải là đỉnh trụ cần xét đồ thị $G \setminus \{u\}$ không chứa đỉnh u .
- Khi kiểm tra cạnh $e = (u, v)$ có phải là cạnh cầu cần xét đồ thị $G \setminus \{e\}$ không chứa cạnh e những vẫn bao gồm cả hai đỉnh u và v .

2.3.5 Tính liên thông trong đồ thị có hướng

Định nghĩa.

- Đồ thị có hướng G là *liên thông mạnh* \Leftrightarrow có đường đi giữa hai đỉnh bất kỳ $u, v \in G$.
 - Đồ thị có hướng G là *liên thông yếu* \Leftrightarrow đồ thị vô hướng nền là liên thông
- \Rightarrow đồ thị liên thông mạnh thì cũng liên thông yếu.
- Đồ thị có hướng G không liên thông mạnh là hợp các đồ thị con có hướng liên thông mạnh, không có đỉnh chung gọi là các *thành phần liên thông mạnh* của G .

Kiểm tra tính liên thông của đồ thị có hướng

- **Input:** Đồ thị có hướng $G = (V, E)$ gồm n đỉnh và m cạnh;
- **Output:** Giá trị 1 nếu G liên thông mạnh, giá trị 2 nếu G không liên thông mạnh nhưng liên thông yếu, giá trị 0 trong trường hợp còn lại;

- Thuật toán: ***Kiểm tra tính liên thông của đồ thị có hướng***

Bước khởi tạo: $i = 1$;

Bước 1: Thực hiện Dfs(i)/Bfs(i) và Tính số k các đỉnh đã duyệt;

Bước 2: Nếu $k < n$ chuyển bước 4; nếu $k = n$ thì chuyển bước 3;

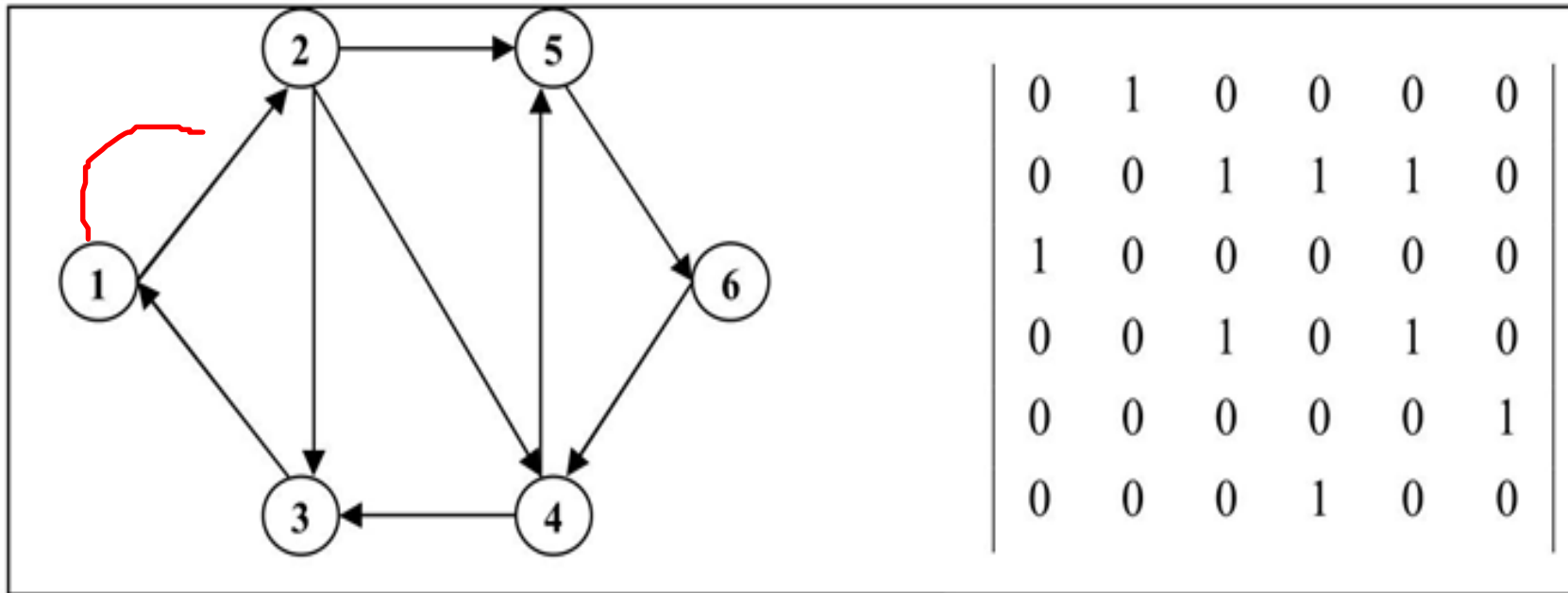
Bước 3: Nếu $i = n$ xuất 1, nếu $i < n$ thì $i = i + 1$ và quay lại bước 1;

Bước 4: Thực hiện Dfs(1)/Bfs(1) khi coi các cạnh của đồ thị là vô hướng và Tính số k các đỉnh đã duyệt;

Bước 5: Nếu $k < n$ xuất 0; nếu $k = n$ thì xuất 2;

- Sinh viên tự cài đặt xem như bài tập

Ví dụ 18: Kiểm tra tính liên thông của đồ thị có hướng dạng ma trận kề



Số đỉnh của G là $n = 6$.

Ví dụ 18: Sử dụng DFS

0	1	0	0	0	0
0	0	1	1	1	0
1	0	0	0	0	0
0	0	1	0	1	0
0	0	0	0	0	1
0	0	0	1	0	0

Dfs(1) = {1(0); 2(1); 3(2); 4(2); 5(4); 6(5)} = V

Dfs(2) = {2(0); 3(2); 1(3); 4(2); 5(4); 6(5)} = V

Dfs(3) = {3(0); 1(3); 2(1); 4(2); 5(4); 6(5)} = V

Dfs(4) = {4(0); 3(4); 1(3); 2(1); 5(2); 6(5)} = V

Dfs(5) = {5(0); 6(5); 4(6); 3(4); 1(3); 2(1)} = V

Dfs(6) = {6(0); 4(6); 3(4); 1(3); 2(1); 5(2)} = V

Kết luận: G là liên thông mạnh

Ví dụ 18: Sử dụng BFS

0	1	0	0	0	0
0	0	1	1	1	0
1	0	0	0	0	0
0	0	1	0	1	0
0	0	0	0	0	1
0	0	0	1	0	0

$Bfs(1) = \{1(0); 2(1); 3(2), 4(2), 5(2); 6(5)\} = V$

$Bfs(2) = \{2(0); 3(2), 4(2), 5(2); 1(3); 6(5)\} = V$

$Bfs(3) = \{3(0); 1(3); 2(1); 4(2), 5(2); 6(5)\} = V$

$Bfs(4) = \{4(0); 3(4), 5(4); 1(3); 2(1); 6(5)\} = V$

$Bfs(5) = \{5(0); 6(5); 4(6); 3(4); 1(3); 2(1)\} = V$

$Bfs(6) = \{6(0); 4(6); 3(4), 5(4); 1(3); 2(1)\} = V$

Kết luận: G là liên thông mạnh

Ví dụ 19: Kiểm tra tính liên thông của đồ thị có hướng cho bởi danh sách kề

$$\begin{aligned} \text{Ke}(1) &= \{6, 7\} & \text{Ke}(2) &= \{8, 9\} & \text{Ke}(3) &= \{4\} & \text{Ke}(4) &= \{5\} & \text{Ke}(5) &= \{3\} \\ \text{Ke}(6) &= \{7\} & \text{Ke}(7) &= \{1\} & \text{Ke}(8) &= \{10\} & \text{Ke}(9) &= \{10\} & \text{Ke}(10) &= \{2\} \end{aligned}$$

■ Sử dụng DFS:

$$\text{Dfs}(1) = \{1(0); 6(1); 7(6)\} \neq V$$

Xét đồ thị vô hướng nền của G: $\text{Dfs}(1) = \{1(0); 6(1); 7(6)\} \neq V$

Kết luận: G không liên thông mạnh, không liên thông yếu.

■ Sử dụng BFS:

$$\text{Bfs}(1) = \{1(0); 6(1), 7(1)\} \neq V$$

Xét đồ thị vô hướng nền của G: $\text{Bfs}(1) = \{1(0); 6(1), 7(1)\} \neq V$

Kết luận: G không liên thông mạnh, không liên thông yếu.

Tìm các thành phần liên thông mạnh của đồ thị có hướng

- **Input:** Đồ thị có hướng $G = (V, E)$ gồm n đỉnh và m cạnh;
- **Output:**
 - Số k các thành phần liên thông mạnh của G ;
 - Số thứ tự của thành phần liên thông chứa u của mọi đỉnh u ;

Thuật toán: *Tìm các thành phần liên thông mạnh của đồ thị có hướng*

Bước khởi tạo: $k = 0$; $lt[u] = 0$ với mọi đỉnh u ;

Bước 1: Thực hiện DFS(u)/BFS(u) với mọi $u \in G$ và đánh dấu $vs[u][v] = 1$ với mỗi v được duyệt đến;

Bước 2: Nếu mọi đỉnh u đều có $lt[u] > 0$ thì chuyển bước 4, ngược lại chọn đỉnh u có $lt[u] = 0$;

Bước 3: $k = k + 1$, $lt[u] = k$ và xét tất cả các đỉnh v có $vs[u][v] = 1$, $vs[v][u] = 1$ và $lt[v] = 0$ thì $lt[v] = k$; quay lại bước 2;

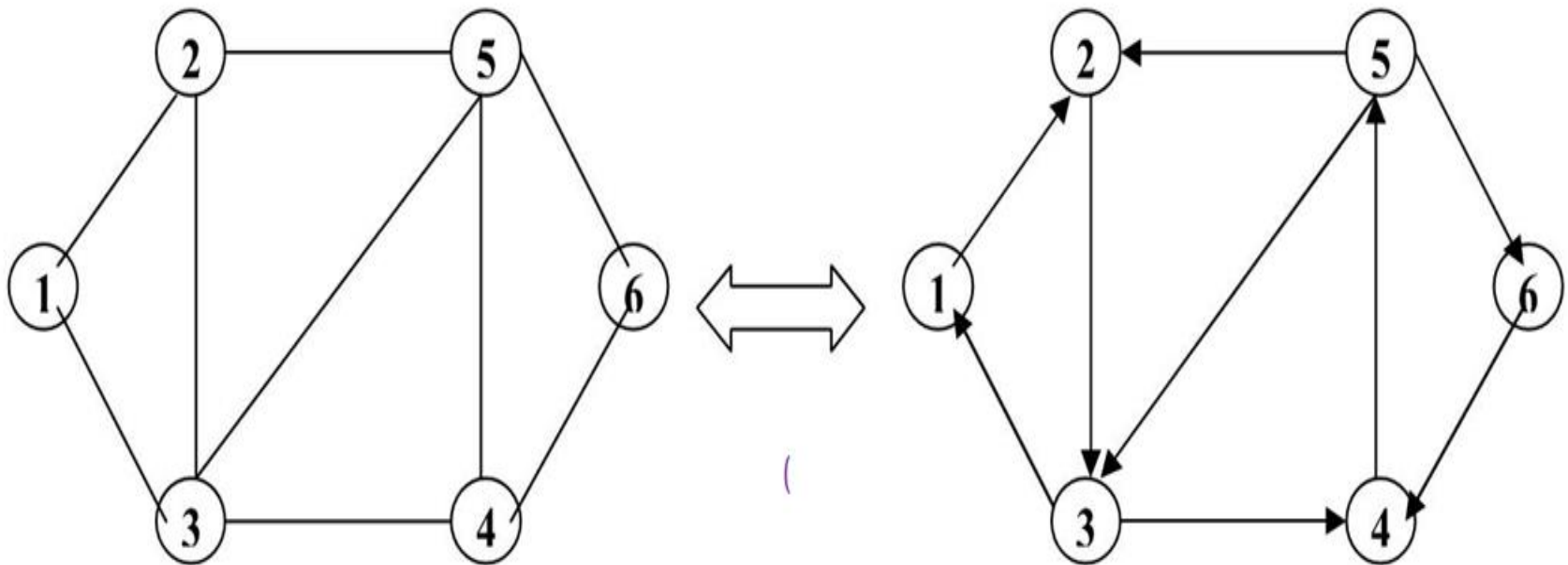
Bước 4: Xuất k và $lt[u]$ với mọi đỉnh u ;

■ Định nghĩa

Phép định chiều đồ thị vô hướng liên thông G là phép biến đổi G thành đồ thị có hướng liên thông mạnh bằng cách định chiều mỗi cạnh vô hướng thành một cung có hướng.

■ Đồ thị vô hướng $G = (V, E)$ được gọi là đồ thị **định chiều được** nếu có thể biến đổi thành đồ thị có hướng liên thông mạnh bằng một phép định chiều.

Ví dụ 20: Định chiều đồ thị vô hướng





Điều kiện để đồ thị vô hướng định chiều được

■ Định lý

Đồ thị vô hướng G **định chiều được** $\Leftrightarrow G$ liên thông và không chứa cạnh cầu.

Ghi chú: Một số vấn đề nâng cao

Sinh viên tự tìm hiểu một số vấn đề nâng cao:

- Viết chương trình tìm thành phần liên thông mạnh của đồ thị có hướng G
- Chứng minh một đồ thị vô hướng G là định chiều được
- Viết chương trình kiểm tra định chiều được một đồ thị vô hướng
- Chỉ ra một phép định chiều trên một đồ thị vô hướng

Tổng kết chương 2

■ Về lý thuyết:

- Hai thuật toán DFS và BFS
- Ứng dụng DFS/BFS giải quyết các bài toán cụ thể

■ Về các dạng bài tập

- Cài đặt được các hàm mô tả các thuật toán
- Sử dụng DFS/BFS giải các bài toán theo mẫu

