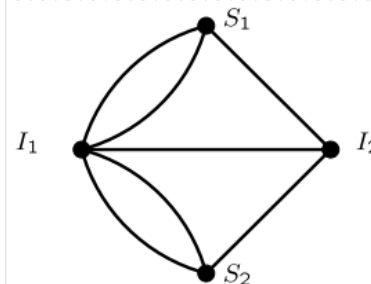
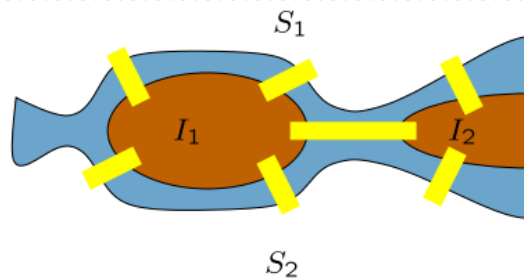


# LÝ THUYẾT ĐỒ THỊ

Phạm Nguyên Khang  
BM. Khoa học máy tính, CNTT  
[pnkhang@cit.ctu.edu.vn](mailto:pnkhang@cit.ctu.edu.vn)



Cần Thơ, 2012

# Tính liên thông của đồ thị

- **Đường đi** (walk): đường đi chiều dài  $k$  đi từ đỉnh  $u = v_0$  đến đỉnh  $v_k = v$  là danh sách các đỉnh và cung xen kẽ nhau:
  - $v_0, e_1, v_1, e_2, v_2, e_3, \dots, e_k, v_k$
  - Cung  $e_i$  có đỉnh đầu là  $v_{i-1}$  và đỉnh cuối  $v_i$
- **Đường đi đơn cung** (trail): là đường đi các cung đều khác nhau.
- **Đường đi đơn đỉnh** (path): là đường đi các đỉnh đều khác nhau.
- Đường đi đơn cung (trail) có đỉnh đầu và đỉnh cuối trùng nhau gọi là một **đường vòng** (circuit)
- Một đường vòng không có đỉnh nào lặp lại gọi là **chu trình** (cycle).
- Chiều dài của walk, trail, path, circuit hay cycle là số cung của nó.

# Tính liên thông của đồ thị

- Định nghĩa:

- Đồ thị vô hướng  $G$  được gọi là liên thông nếu và chỉ nếu với mọi cặp đỉnh  $u, v \in V$  luôn tồn tại đường đi (walk) từ  $u \rightarrow v$ . Ngược lại,  $G$  được gọi là không liên thông.
- Đỉnh  $u$  được gọi là liên thông với đỉnh  $v \Leftrightarrow$  tồn tại đường đi từ  $u \rightarrow v$ . Quan hệ liên thông trên  $G$  là tập các cặp có thứ tự  $(u, v)$  sao cho  $u$  liên thông với  $v$ .

# Tính liên thông của đồ thị

- Định lý:
  - Quan hệ liên thông là một quan hệ tương đương
  - C/M: xem như bài tập
- Bổ đề:
  - Mỗi đường đi (walk) đi từ  $u$  đến  $v$  luôn chứa 1 một đường đi đơn đỉnh (path) đi từ  $u$  đến  $v$ .
  - C/M: xem như bài tập

# Bộ phận liên thông

- Các bộ phận liên thông của đồ thị  $G$  là tập các đồ thị con liên thông lớn nhất của  $G$  (là các lớp tương đương của quan hệ liên thông).
- Đỉnh cô lập (có bậc bằng 0) cũng là một bộ phận liên thông chỉ gồm chính nó và được là bộ phận liên thông tầm thường (trivial connected component).

# Bài tập

- CMR trong một đồ thị vô hướng, đường đi (walk) ngắn nhất giữa hai đỉnh là một đường đi đơn đỉnh (path).

# Khoảng cách giữa 2 đỉnh

- **Định nghĩa:** Trên một đồ thị vô hướng, khoảng cách từ đỉnh  $u$  đến đỉnh  $v$ , ký hiệu  $d(u,v)$  được định nghĩa bằng:
  - 0, nếu  $u \equiv v$
  - Chiều dài đường đi ngắn nhất từ  $u$  đến  $v$ , nếu tồn tại đường đi từ  $u$  đến  $v$ .
  - $\infty$ , nếu không có đường đi từ  $u$  đến  $v$ .
- **Đường kính** của đồ thị vô hướng là khoảng cách lớn nhất giữa 2 đỉnh trên đồ thị.
- Bài tập:
  - Chứng minh bất đẳng thức tam giác:
    - $d(u, v) \leq d(u, x) + d(x, v)$

# Bài tập

1. Cho đơn đồ thị vô hướng  $G = \langle V, E \rangle$  có  $n$  đỉnh, gọi bậc nhỏ nhất của đồ thị là:

$$\delta(G) = \min_{v \in V} \{\deg(v)\}$$

- CMR: nếu  $\delta(G) \geq \left\lfloor \frac{n}{2} \right\rfloor$  thì  $G$  liên thông

1. CMR: Số cung ít nhất của một đơn đồ thị vô hướng liên thông có  $n$  đỉnh là  $n - 1$ .
  - Gợi ý: C/M quy nạp trên số cung.

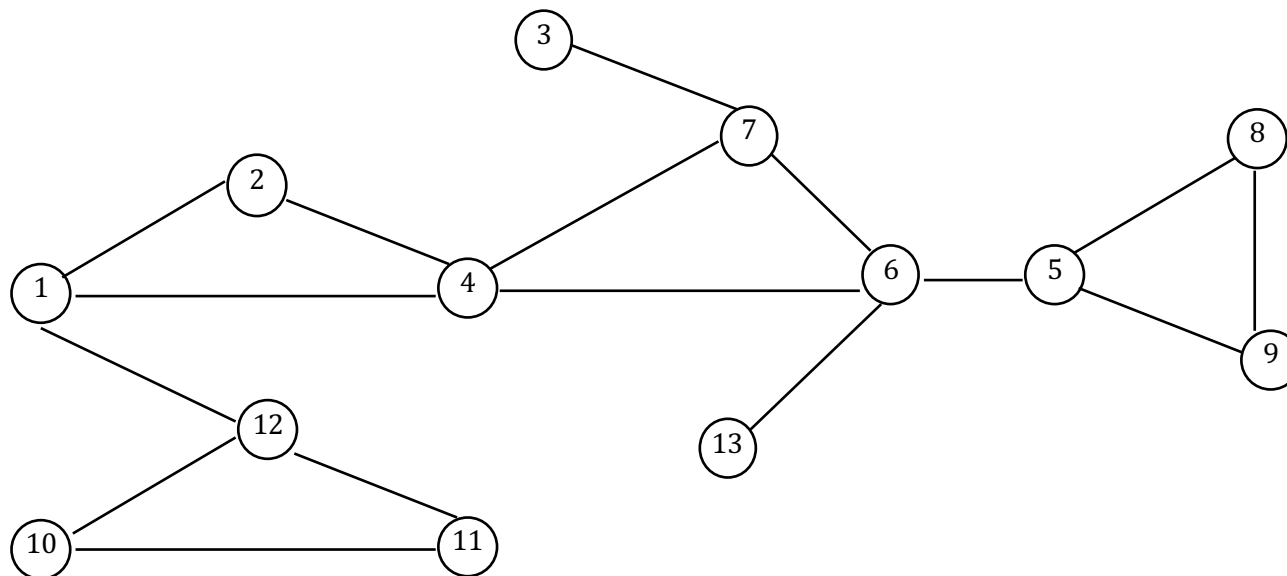


# Bài tập

1. Cho đơn đồ thị vô hướng  $G = \langle V, E \rangle$  có  $2n$  đỉnh. CMR: nếu tất cả các đỉnh đều có bậc lớn hơn hoặc bằng  $n$ , thì  $G$  liên thông.
2. Cho đồ thị vô hướng  $G$  có  $n$  đỉnh và  $m$  cung. CMR  $G$  có ít nhất  $n - m$  thành phần liên thông.

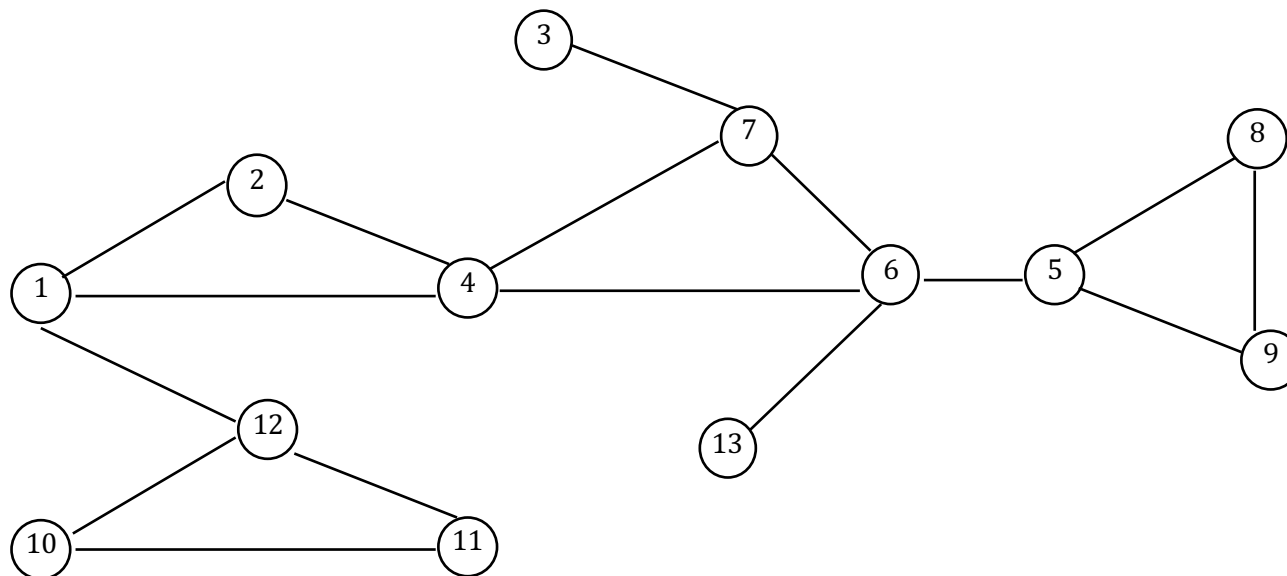
# Giải thuật kiểm tra tính liên thông của đồ thị – Thực hành (1/2)

- Áp dụng giải thuật duyệt đồ thị để đánh số/đánh dấu (gán nhãn) các đỉnh
- Nếu sau khi duyệt tất cả các đỉnh đều có nhãn => liên thông, ngược lại không liên thông.



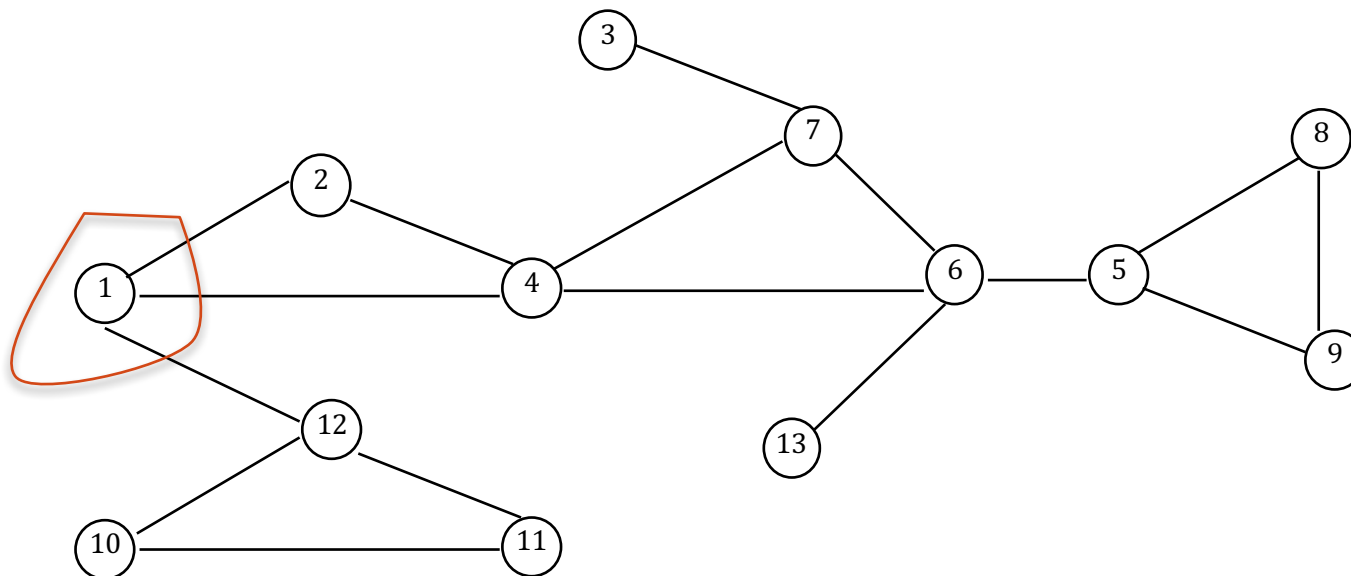
# Duyệt đồ thị

- Lần lượt xem xét từng đỉnh của đồ thị (mỗi đỉnh chỉ xét lần)
- Bắt đầu từ 1 đỉnh bất kỳ, xét đỉnh các đỉnh kề của nó, xét các đỉnh kề của các đỉnh kề, ...



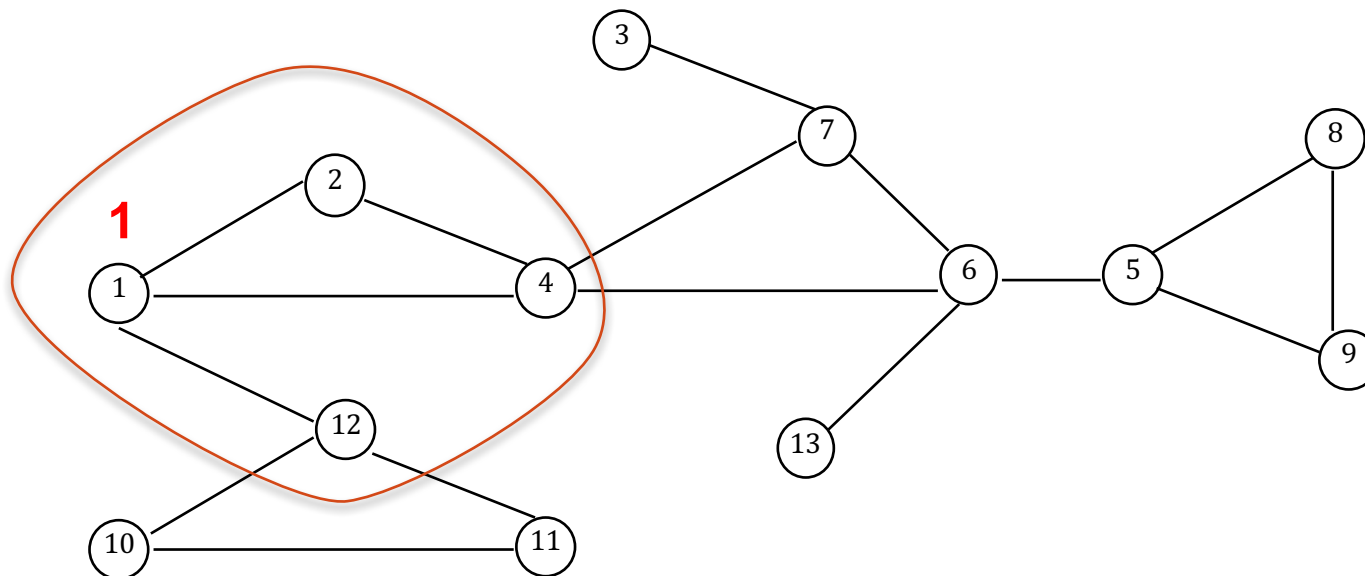
# Duyệt đồ thị

- Lần lượt xem xét từng đỉnh của đồ thị (mỗi đỉnh chỉ xét lần)
- Bắt đầu từ 1 đỉnh bất kỳ, xét đỉnh các đỉnh kề của nó, xét các đỉnh kề của các đỉnh kề, ...



# Duyệt đồ thị

- Lần lượt xem xét từng đỉnh của đồ thị (mỗi đỉnh chỉ xét lần)
- Bắt đầu từ 1 đỉnh bất kỳ, xét đỉnh các đỉnh kề của nó, xét các đỉnh kề của các đỉnh kề, ...



# Giải thuật kiểm tra tính liên thông của đồ thị – Thực hành (2/2)

- Giải thuật Trémaux (1882) tìm bộ phận liên thông chứa một đỉnh cho trước:
  - Áp dụng giải thuật duyệt đồ thị theo chiều sâu để đánh số các đỉnh (giải thuật đệ quy). Khởi tạo tất cả các đỉnh có  $\text{num}[x] = -1$  và  $k = 1$ ;

```
void Traversal(Graph* G, int x) {  
    if (num[x] > 0) return;  
    num[x] = k; k++;  
    for (các đỉnh kề y của x)  
        Traversal(G, y);  
}
```

- Khi giải thuật kết thúc, các đỉnh được đánh số ( $\text{num}[x] > 0$ ) chính là bộ phận liên thông cần tìm.

# Giải thuật tìm tất cả các bộ phận liên thông (thực hành)

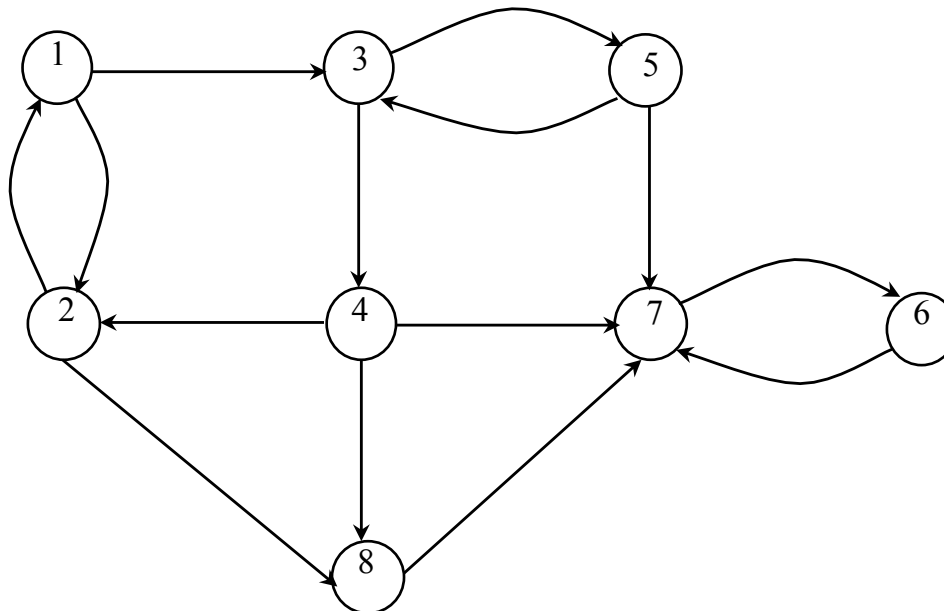
- Sử dụng giải thuật duyệt đồ thị (chiều rộng hoặc chiều sâu): Khởi tạo tất cả các đỉnh có  $\text{num}[x] = -1$  (chưa được đánh số), xét qua các đỉnh, nếu chưa được đánh dấu  $\Rightarrow$  gọi Traversal để duyệt nó.

```
for (x = 1; x <= n; x++)  
    if (num[x] < 0) {  
        Traversal(G, x);  
        //Tìm được 1 bộ phận liên thông.  
    }
```

- Mỗi lần duyệt xong 1 đỉnh x, ta sẽ tìm được 1 bộ phận liên thông.

# Tính liên thông của đồ thị có hướng

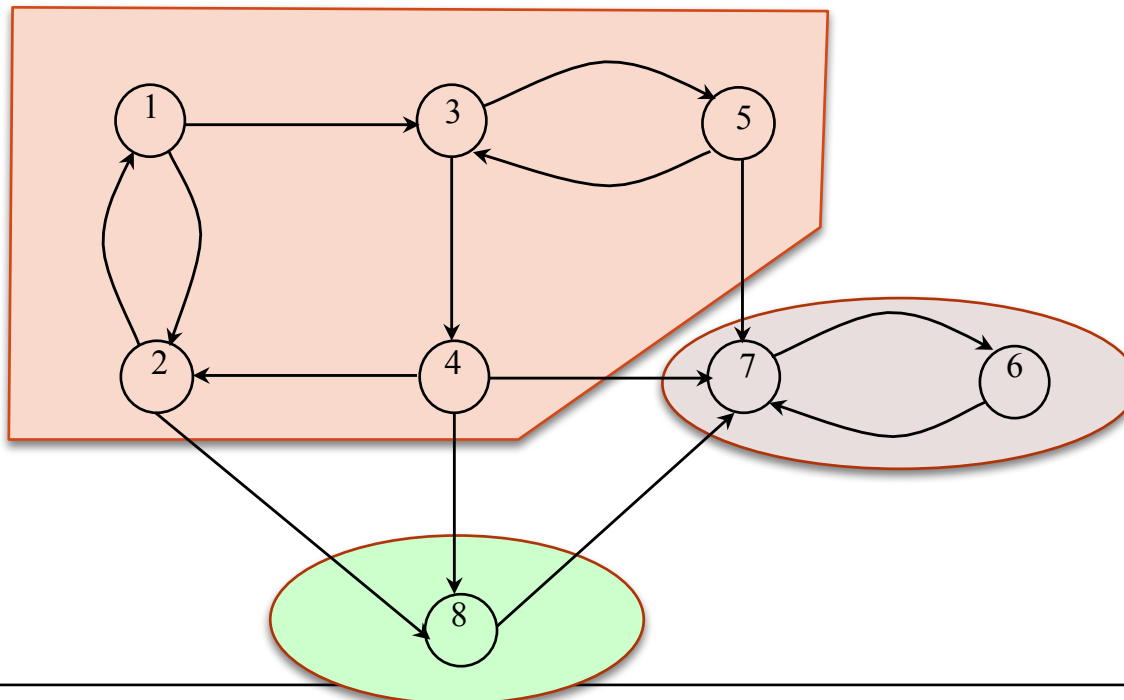
- Cho đồ thị **có hướng**  $G = \langle V, E \rangle$ 
  - $G$  được gọi là **liên thông yếu**  $\Leftrightarrow$  đồ thị vô hướng nền của nó liên thông (xem tính liên thông của đồ thị vô hướng)
  - $G$  được gọi là **liên thông mạnh**  $\Leftrightarrow$  giữa hai đỉnh  $x, y$  bất kỳ, luôn có đường đi từ  $x$  đến  $y$ .





# Tính liên thông của đồ thị có hướng

- Bộ phận liên thông mạnh
  - Đồ thị con liên thông mạnh: có đường đi giữa hai đỉnh bất kỳ.
  - Đồ thị có hướng không liên thông mạnh bao gồm nhiều bộ phận liên thông mạnh



# Tìm các bộ phận liên thông mạnh

- Nhận xét:
  - Các đỉnh trong một chu trình liên thông với nhau
- Giải thuật tìm các bộ phận liên thông mạnh
  - Tìm các chu trình (lớn nhất có thể) của một đồ thị
  - Giải thuật Tarjan (1972)
    - Áp dụng duyệt theo chiều sâu (đệ quy hoặc không đệ quy) để đánh số các đỉnh.
    - Để tìm được chu trình, với mỗi đỉnh  $v$ , ngoài  $\text{num}[v]$ , ta lưu thêm  $\text{min\_num}[v]$  (là chỉ số nhỏ nhất trong các đỉnh có thể đi đến được từ  $v$ ). Trong quá trình duyệt,  $\text{min\_num}[v]$  sẽ được cập nhật.
    - Khi duyệt xong 1 đỉnh, nếu  $\text{num}[v] = \text{min\_num}[v]$  thì  $v$  là đỉnh bắt đầu (đỉnh gốc/đỉnh khớp) của bộ phận liên thông.

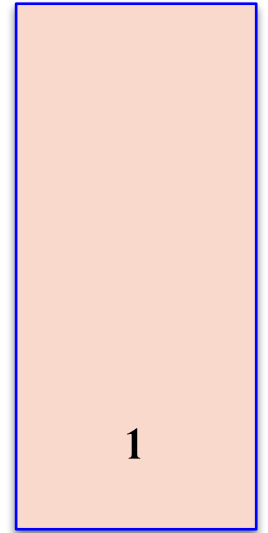
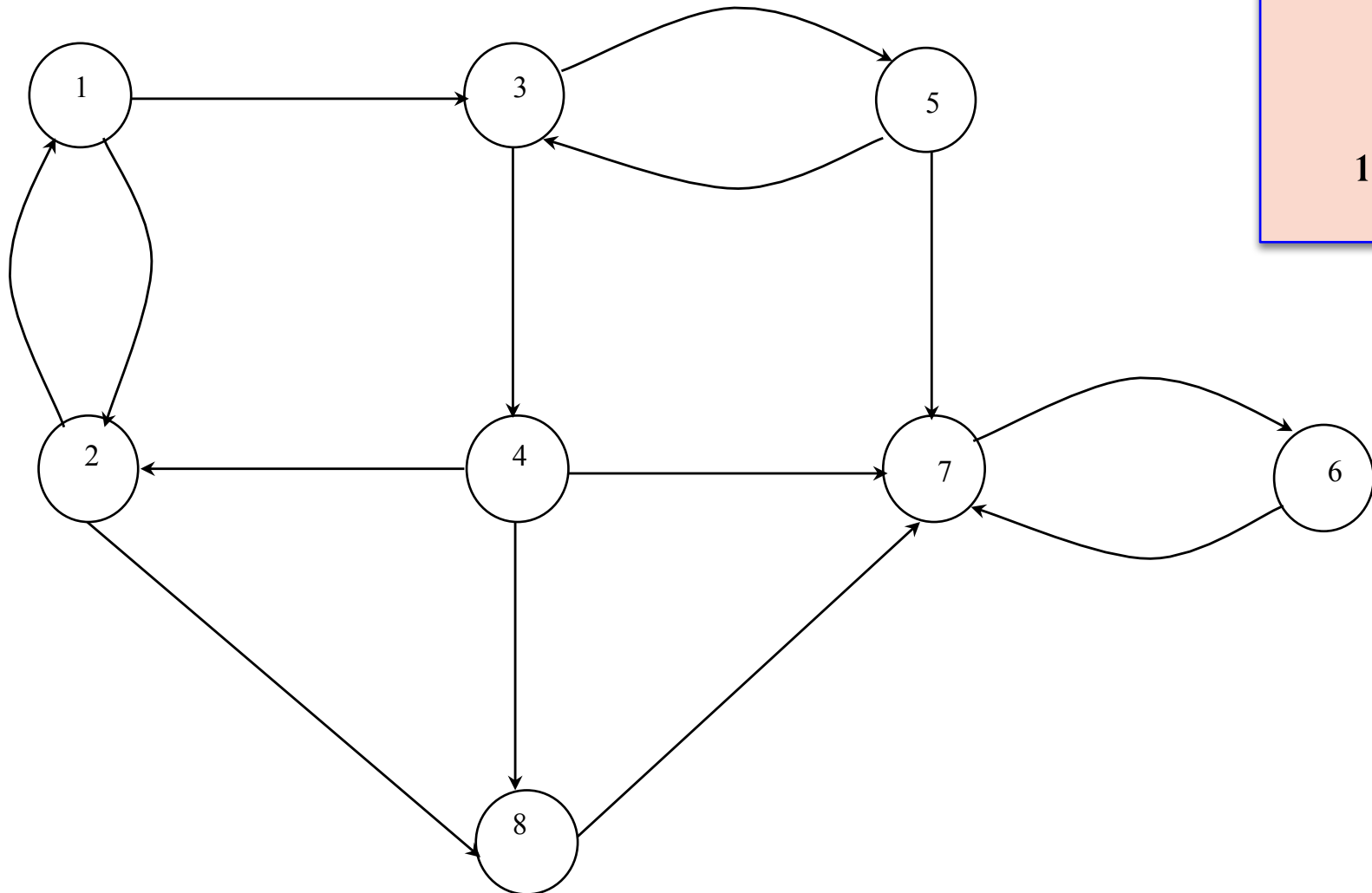
# Tìm các bộ phận liên thông mạnh

```
void strong_connect(Graph* G, int x) {
    num[x] = min_num[x] = idx; idx++;
    push(&S, x);                /* Đưa x vào stack */
    on_stack[x] = 1; /* x đang ở trên stack */
    /* Lấy các đỉnh kề và duyệt nó */
    List list = neighbors(G, x);
    for (j = 1; j <= list.size; j++) {
        int y = element_at(&list, j);
        if (num[y] == -1) {
            strong_connect(G, y);
            min_num [x] = min(min_num[x], min_num[y]);
        } else if (on_stack[y])
            min_num[x] = min(min_num[x], num[y]);
    }
    /* Kiểm tra nếu num[x] == min_num[x] */
}
```

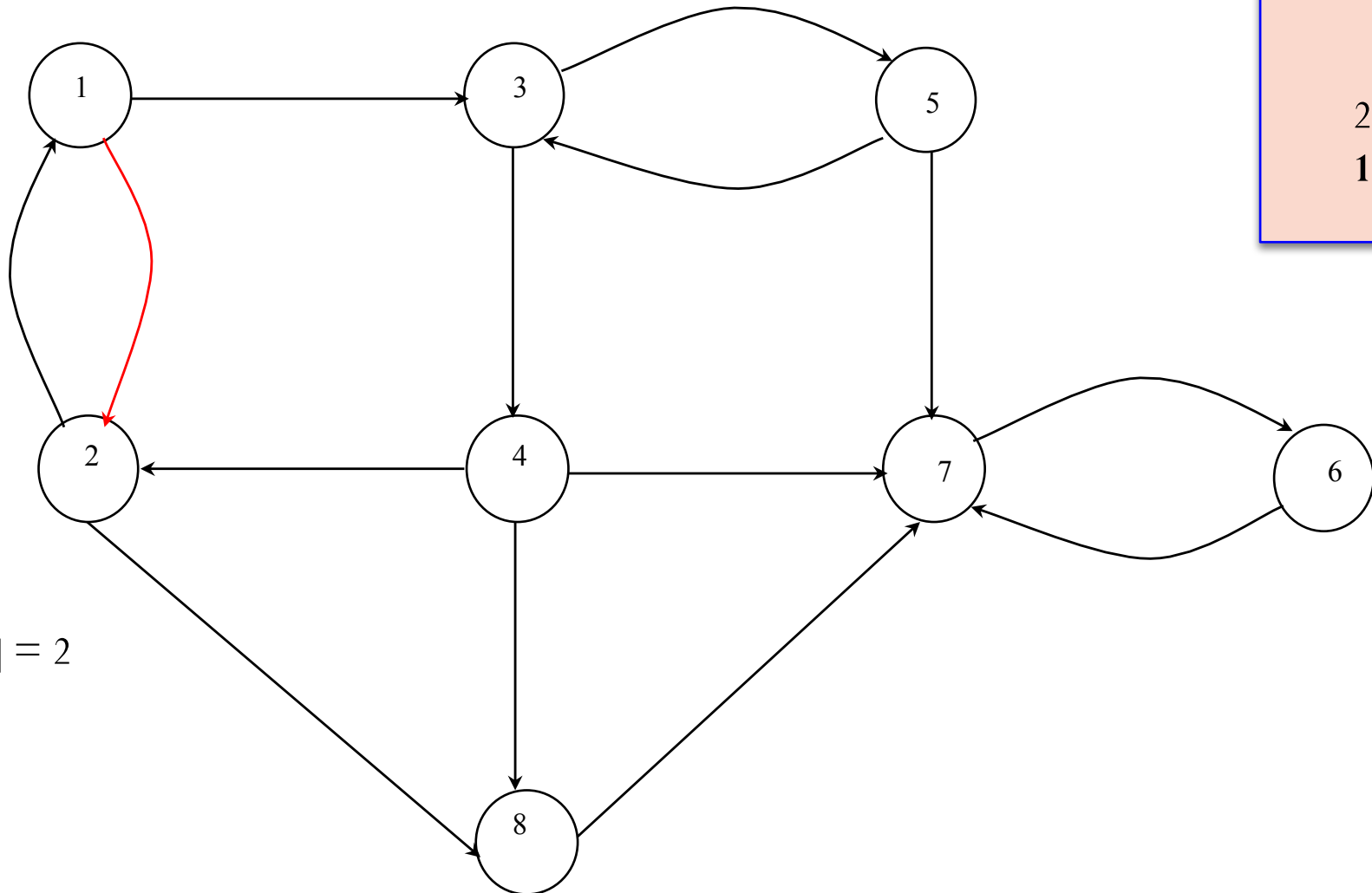
# Tìm các bộ phận liên thông mạnh

```
void strong_connect(Graph* G, int x) {  
    /* đánh dấu x */  
    /* Lấy các đỉnh kề và duyệt nó */  
    /* Kiểm tra nếu num[x] == min_num[x] */  
    if (num[x] == min_num[x]) {  
        /* Loại bỏ các đỉnh ra khỏi stack */  
        int w;  
        do {  
            w = top(&S); pop(&S);  
            on_stack[w] = 0;  
            /* làm gì đó trên w, vd: in ra màn hình  
        } while (w != x);  
    }  
}
```

$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$



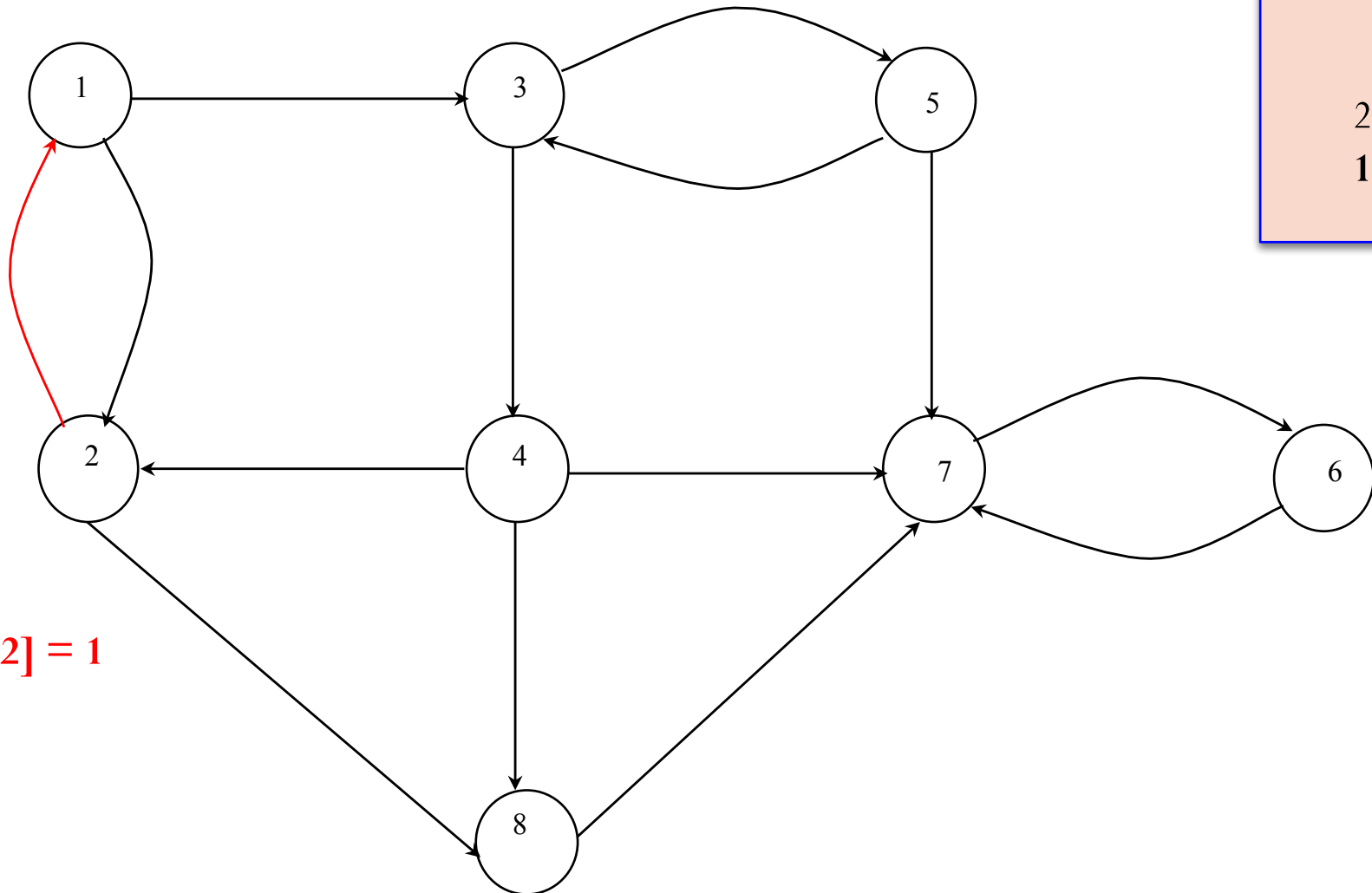
$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$



2  
1

$\text{num}[2] = 2$   
 $\text{min\_num}[2] = 2$

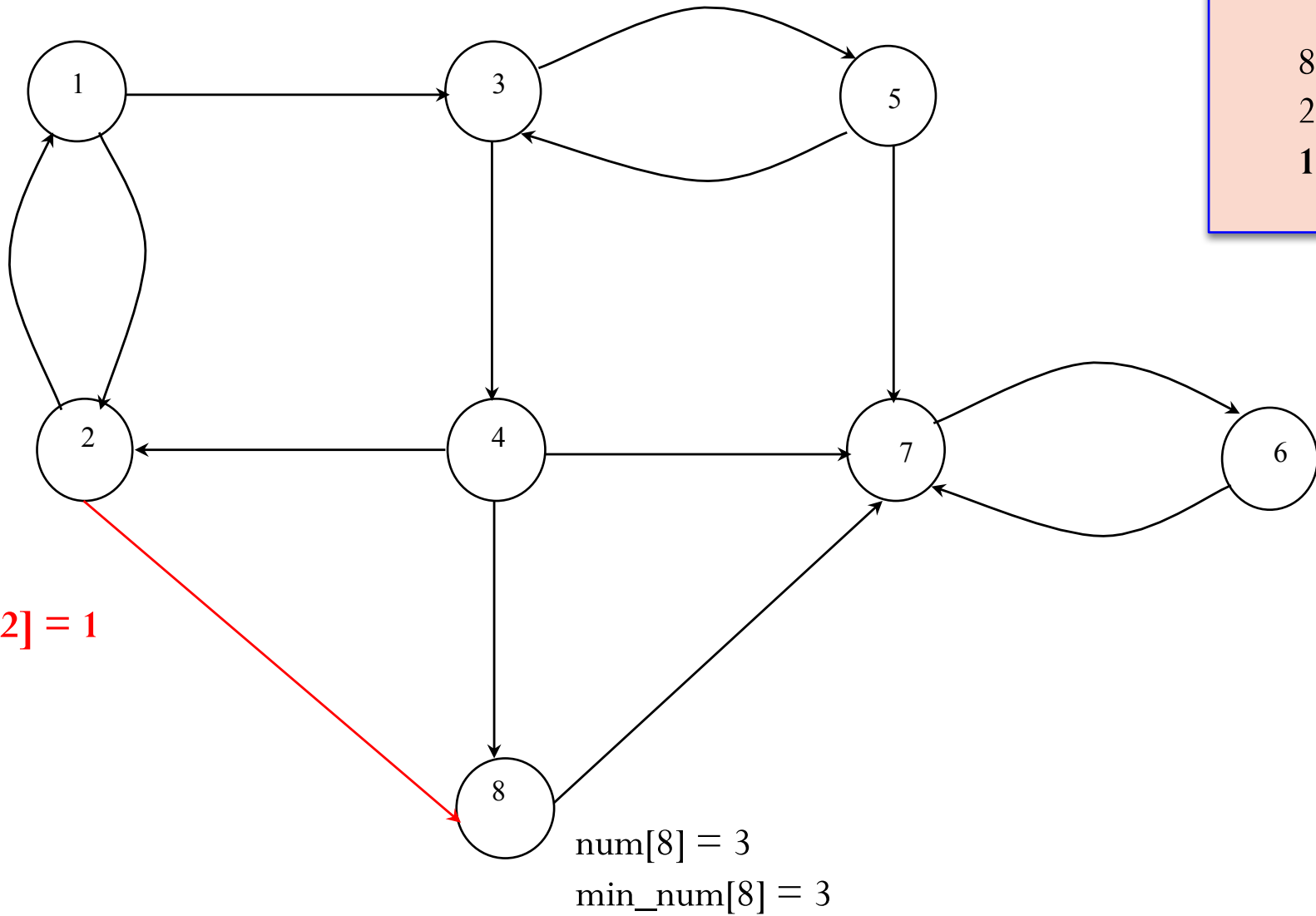
$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$



2  
1

$\text{num}[2] = 2$   
 **$\text{min\_num}[2] = 1$**

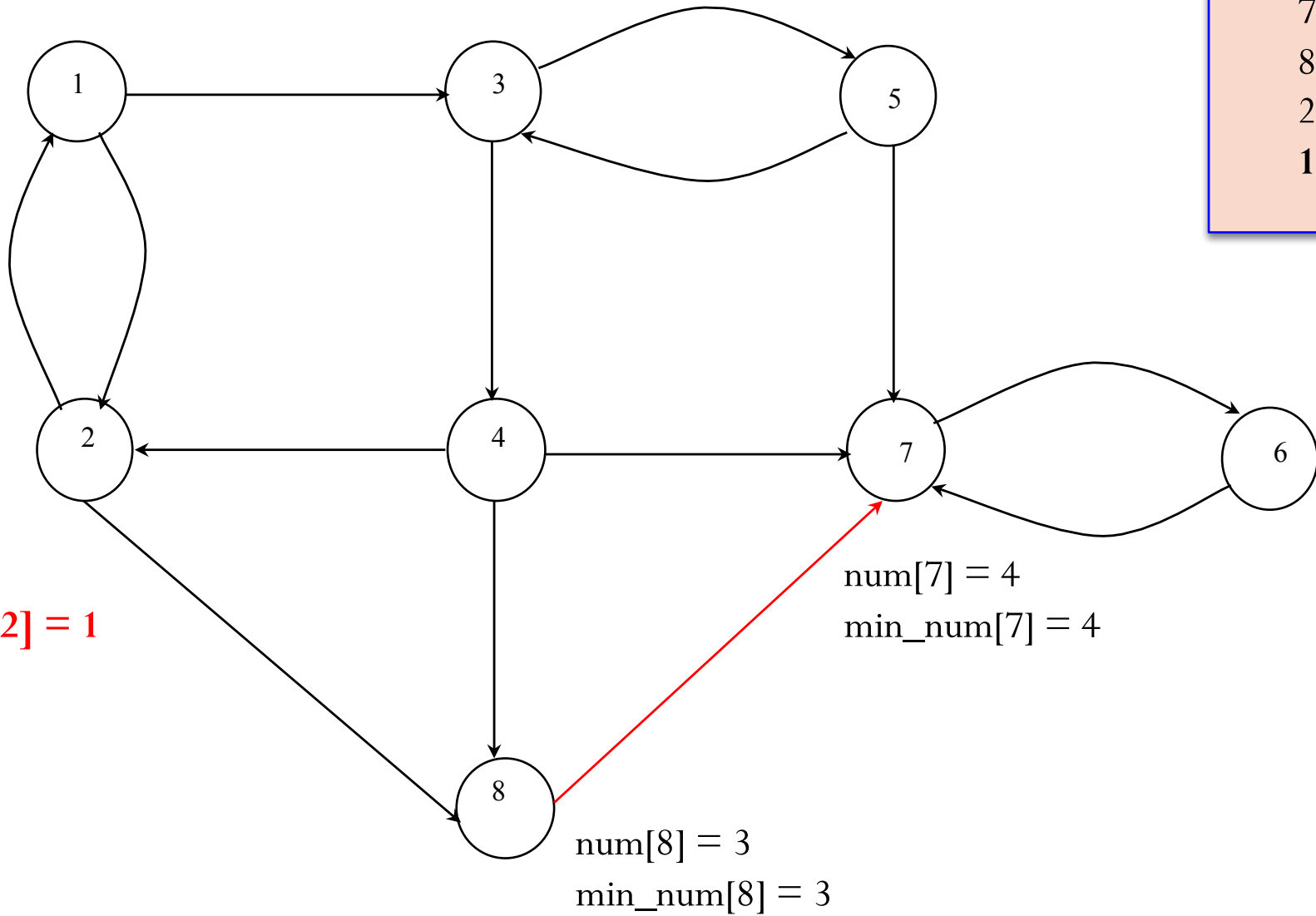
$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$



8  
2  
1

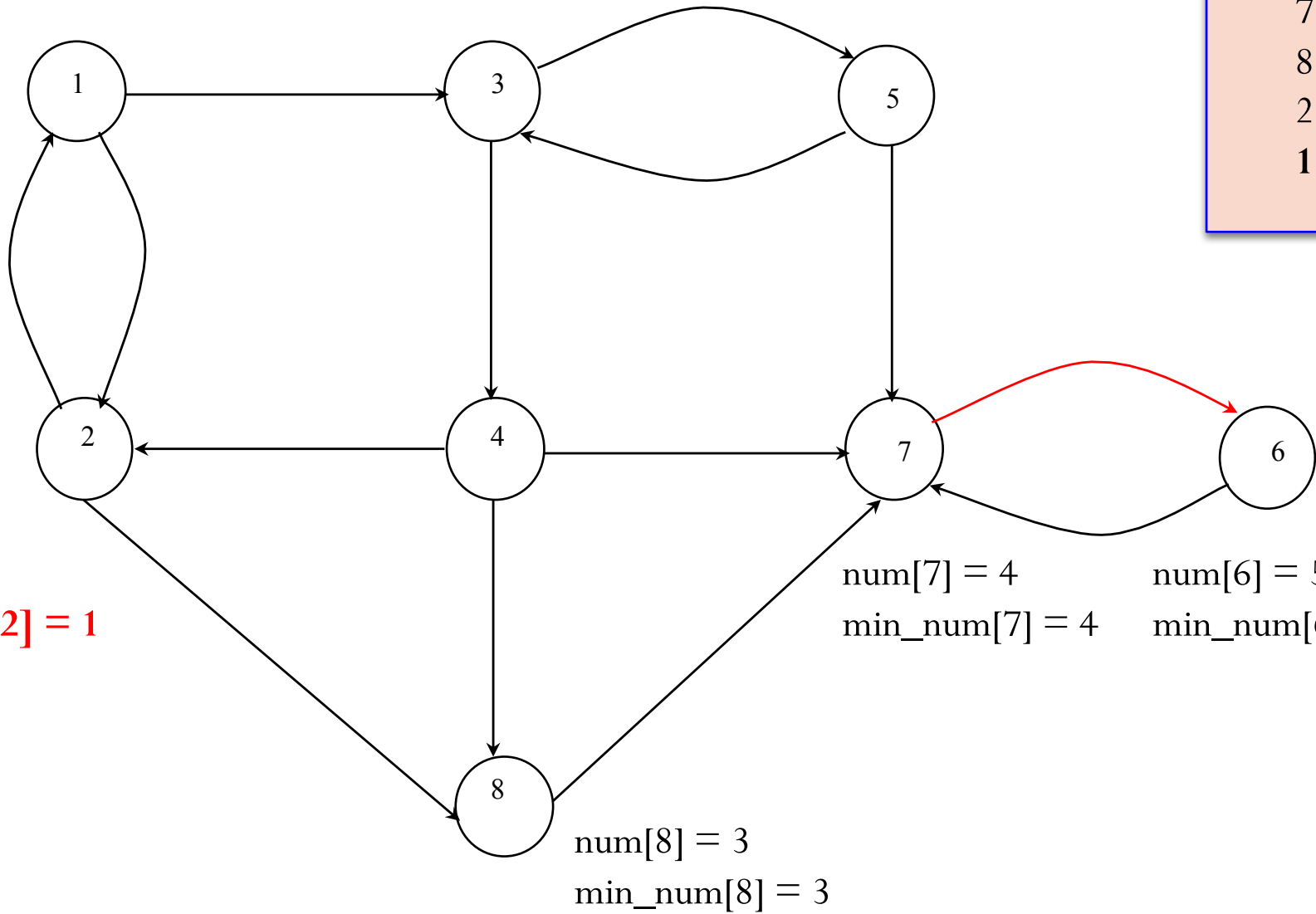


$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$

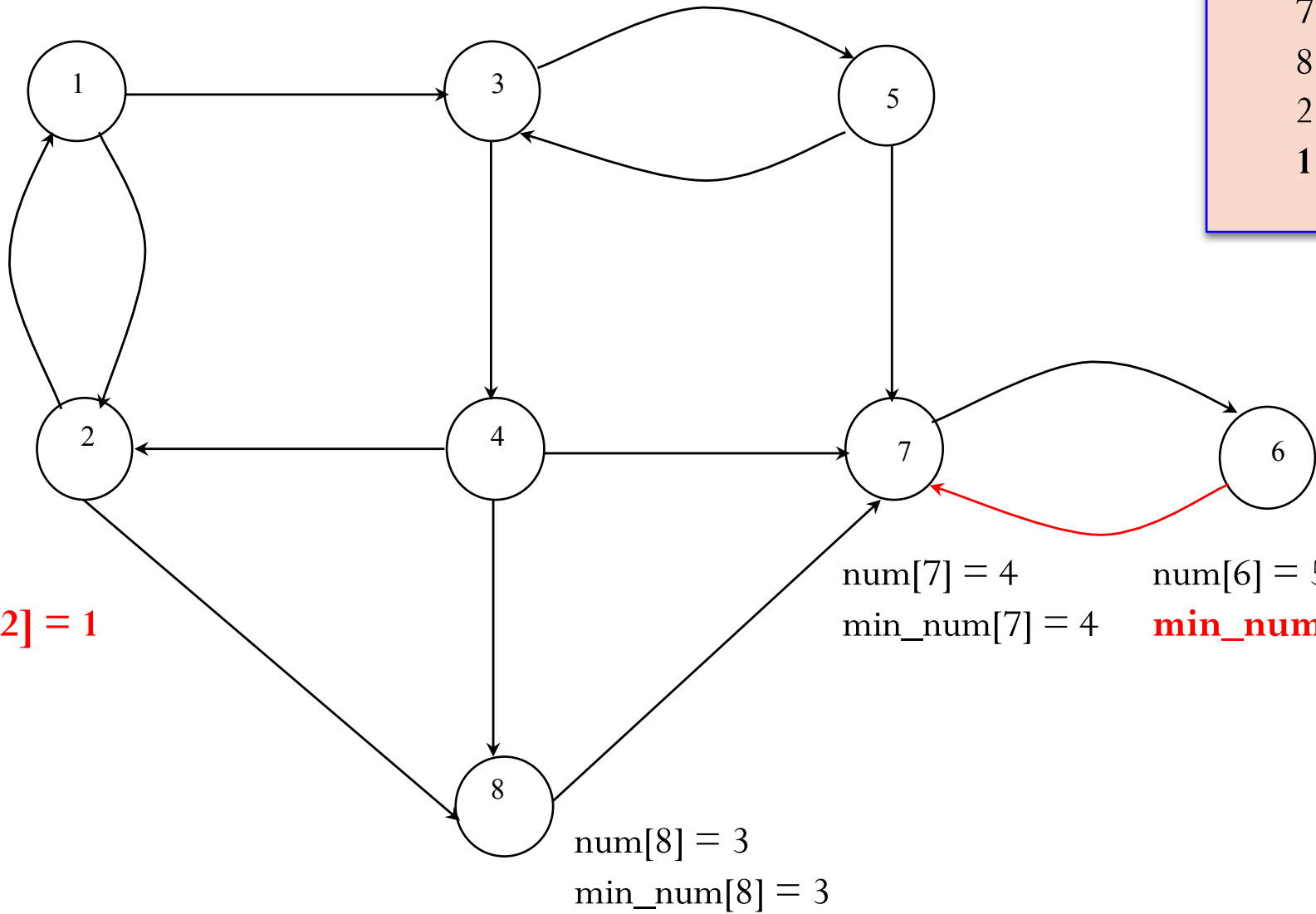


7  
8  
2  
1

$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$

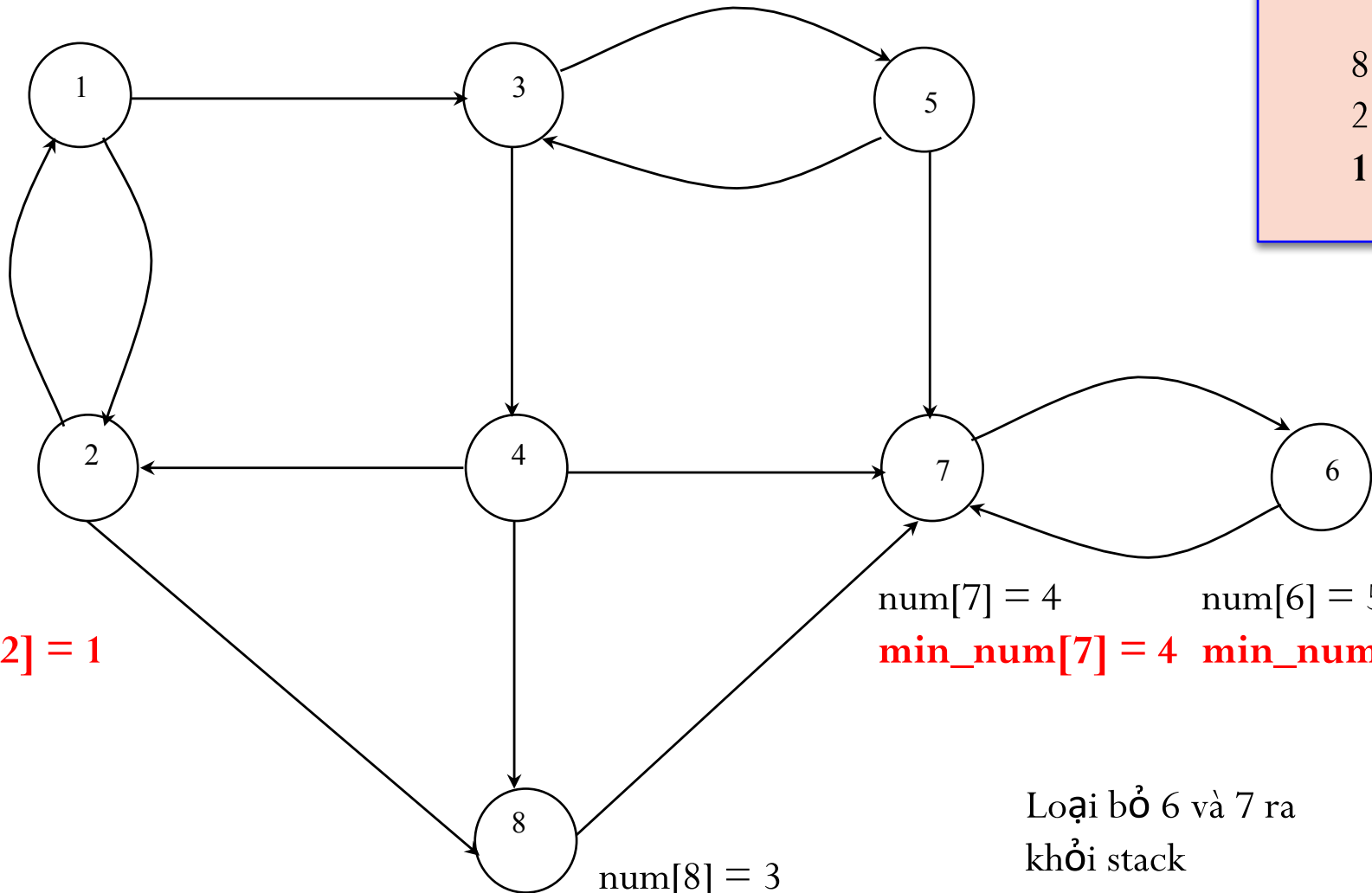


$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$



6  
7  
8  
2  
1

$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$



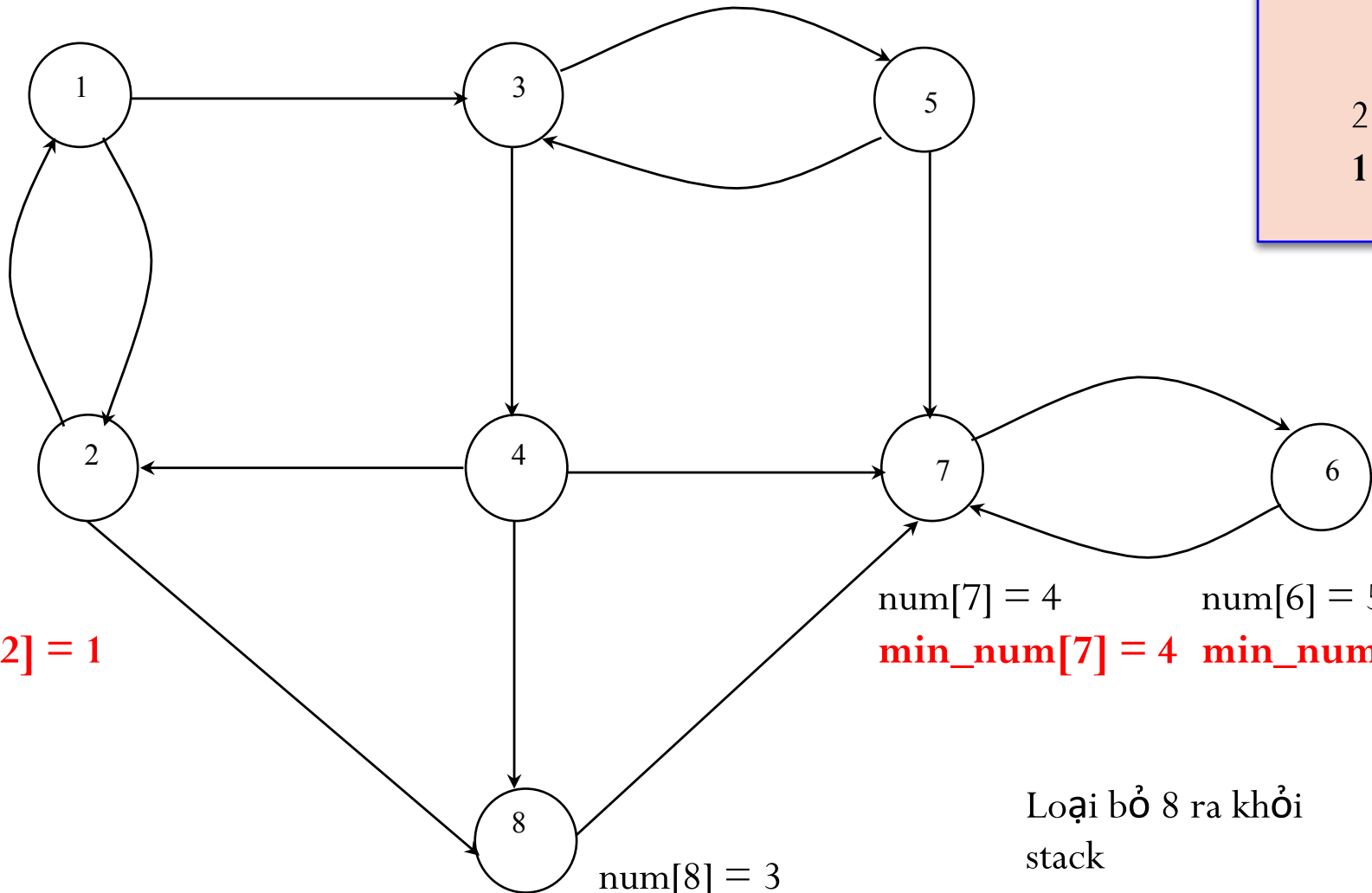
$\text{num}[2] = 2$   
 **$\text{min\_num}[2] = 1$**

$\text{num}[7] = 4$        $\text{num}[6] = 5$   
 **$\text{min\_num}[7] = 4$     $\text{min\_num}[6] = 4$**

$\text{num}[8] = 3$   
 $\text{min\_num}[8] = 3$

Loại bỏ 6 và 7 ra  
khỏi stack

$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$



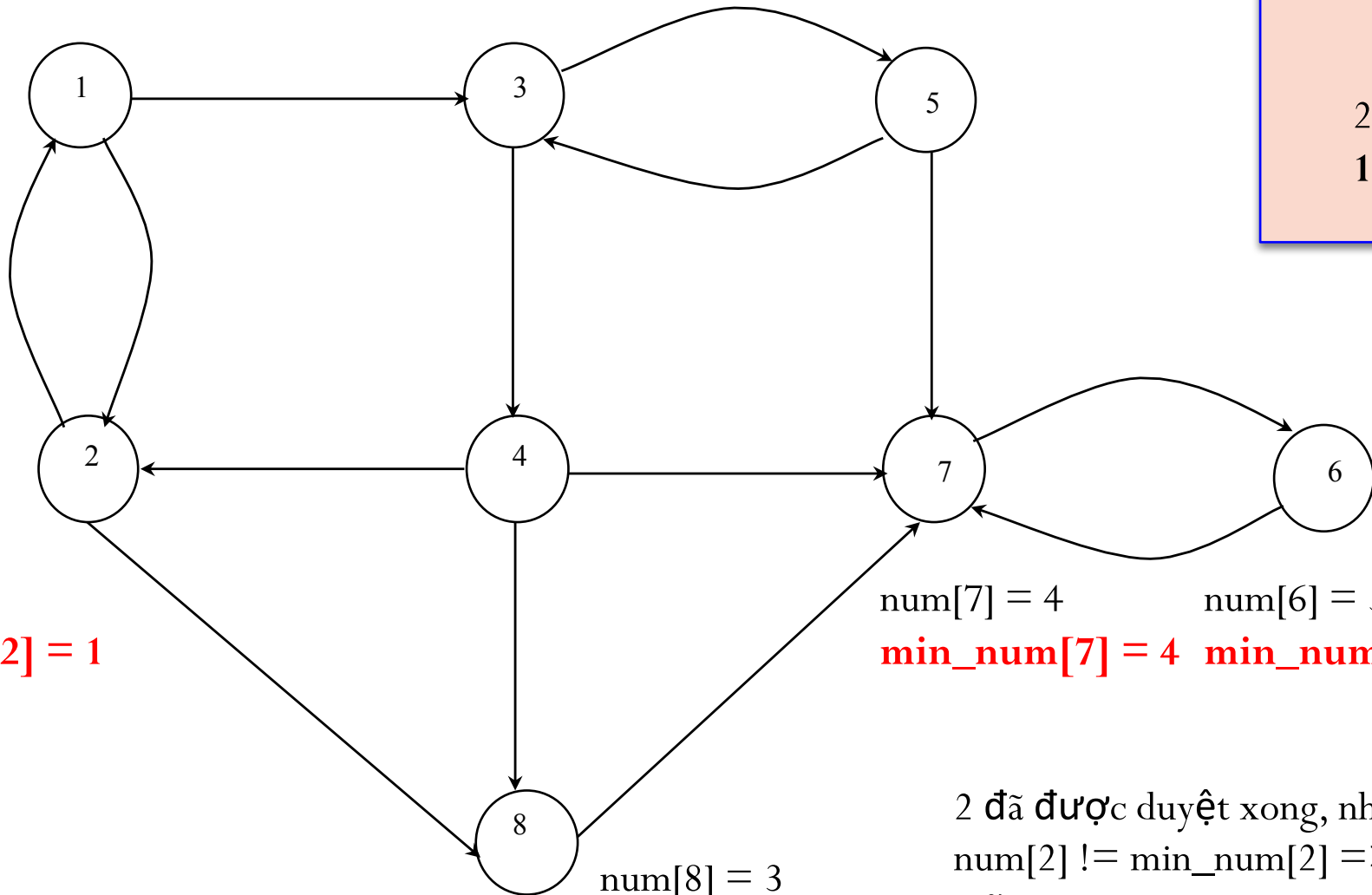
$\text{num}[2] = 2$   
 **$\text{min\_num}[2] = 1$**

$\text{num}[7] = 4$        $\text{num}[6] = 5$   
 **$\text{min\_num}[7] = 4$     $\text{min\_num}[6] = 4$**

$\text{num}[8] = 3$   
 **$\text{min\_num}[8] = 3$**

Loại bỏ 8 ra khỏi  
stack

$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$



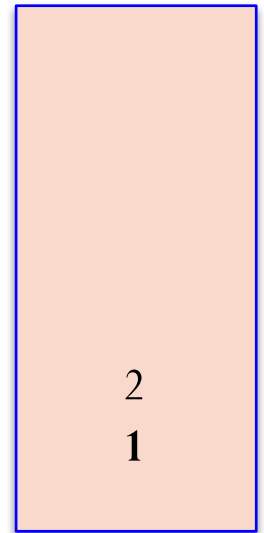
$\text{num}[2] = 2$   
 $\text{min\_num}[2] = 1$

$\text{num}[7] = 4$        $\text{num}[6] = 5$   
 $\text{min\_num}[7] = 4$     $\text{min\_num}[6] = 4$

$\text{num}[8] = 3$

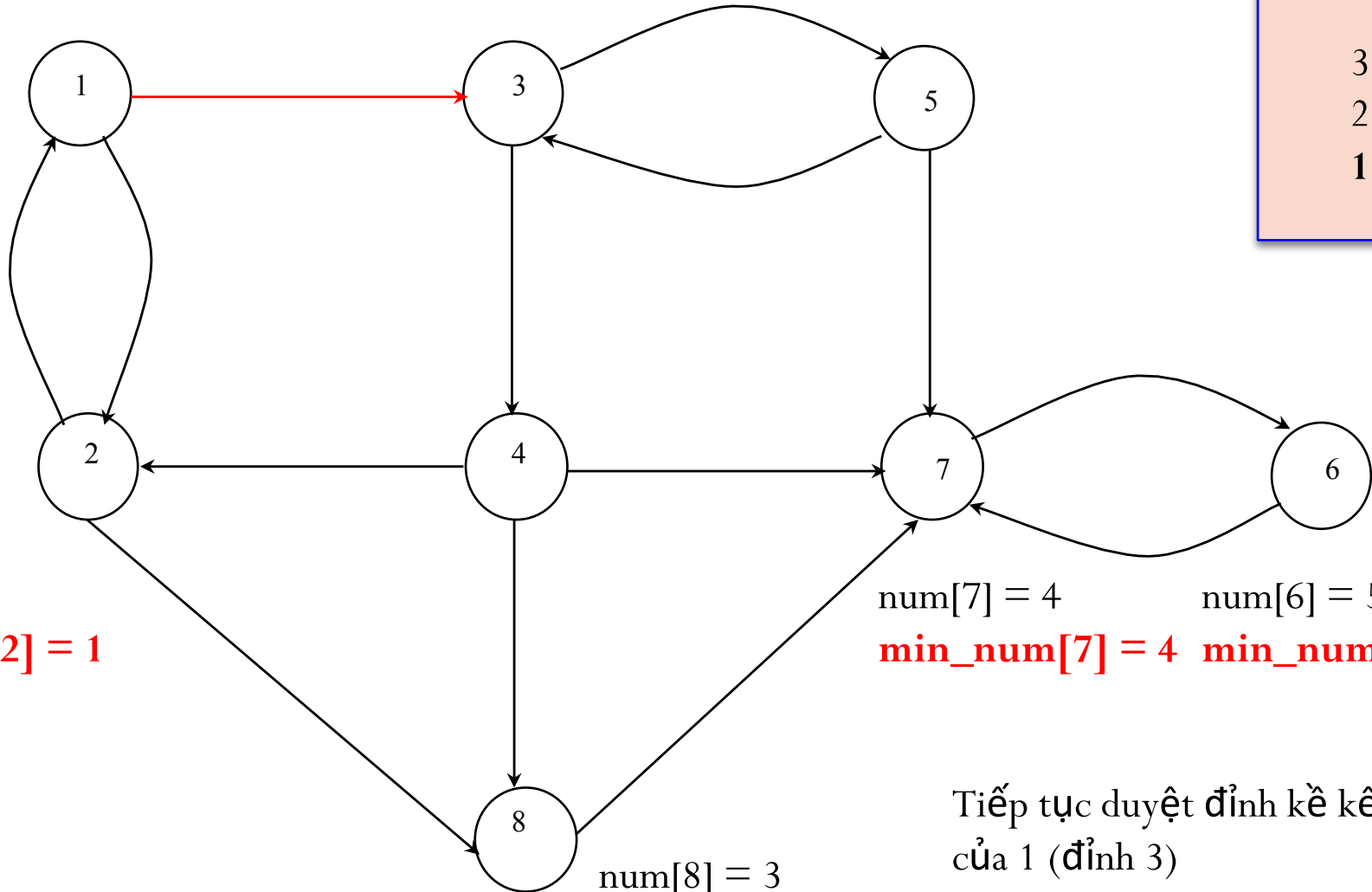
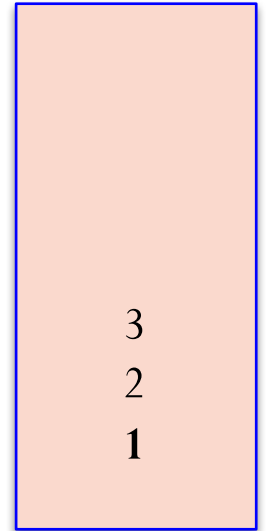
$\text{min\_num}[8] = 3$

2 đã được duyệt xong, nhưng  
 $\text{num}[2] \neq \text{min\_num}[2] \Rightarrow$   
giữ 2 lại trong stack



$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$

$\text{num}[3] = 6$   
 $\text{min\_num}[3] = 6$



Tiếp tục duyệt đỉnh kề tiếp  
của 1 (đỉnh 3)

$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$

$\text{num}[3] = 6$   
 $\text{min\_num}[3] = 6$

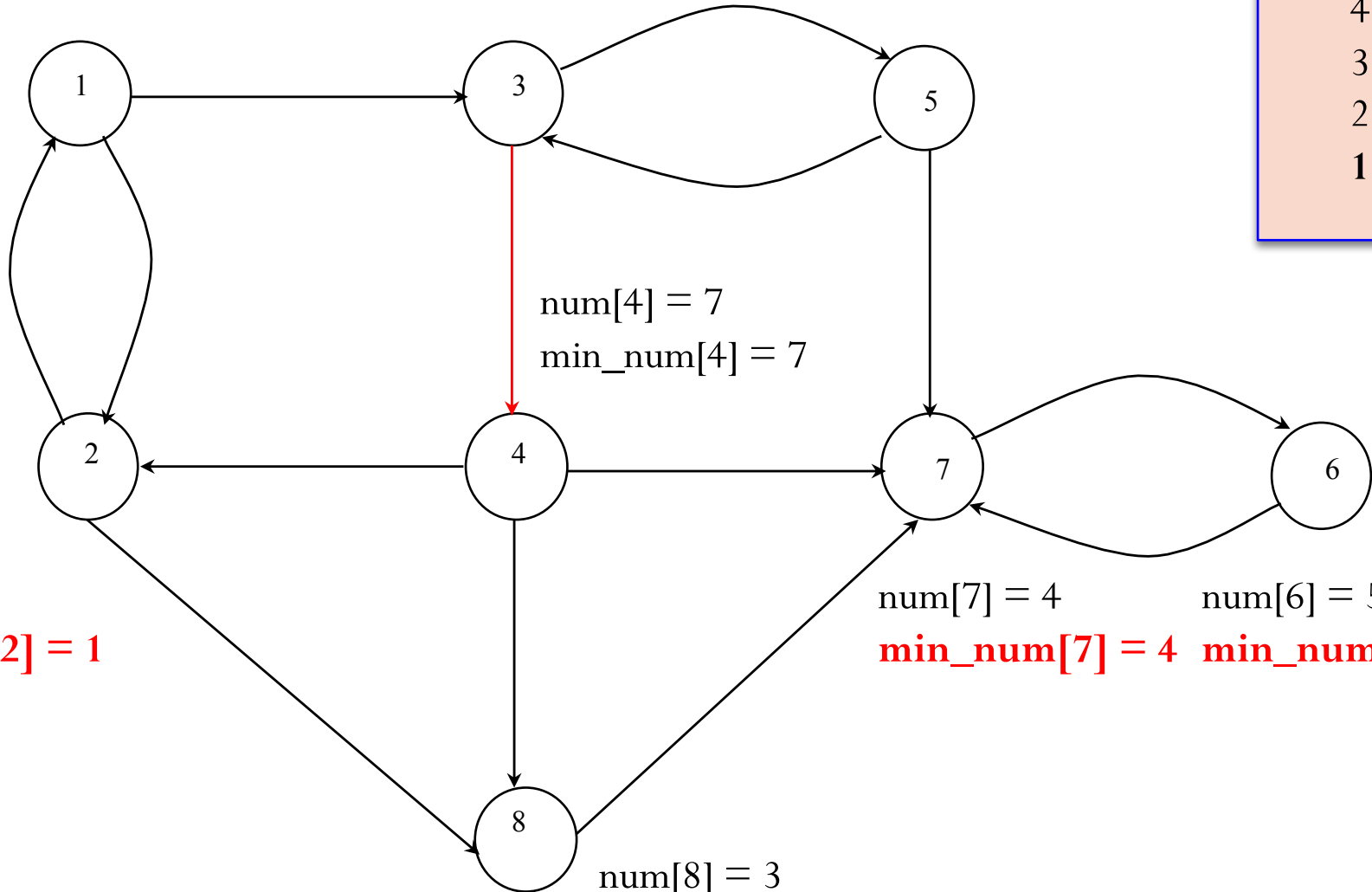
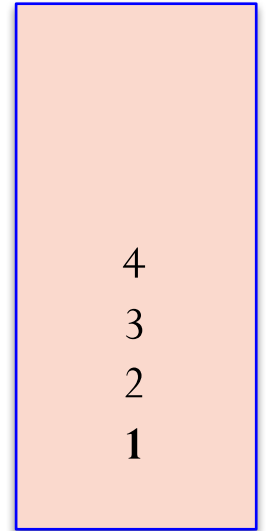
$\text{num}[4] = 7$   
 $\text{min\_num}[4] = 7$

$\text{num}[2] = 2$   
 $\text{min\_num}[2] = 1$

$\text{num}[7] = 4$   
 $\text{min\_num}[7] = 4$

$\text{num}[6] = 5$   
 $\text{min\_num}[6] = 4$

$\text{num}[8] = 3$   
 $\text{min\_num}[8] = 3$





$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$

$\text{num}[3] = 6$   
 $\text{min\_num}[3] = 6$

$\text{num}[4] = 7$   
 $\text{min\_num}[4] = 2$

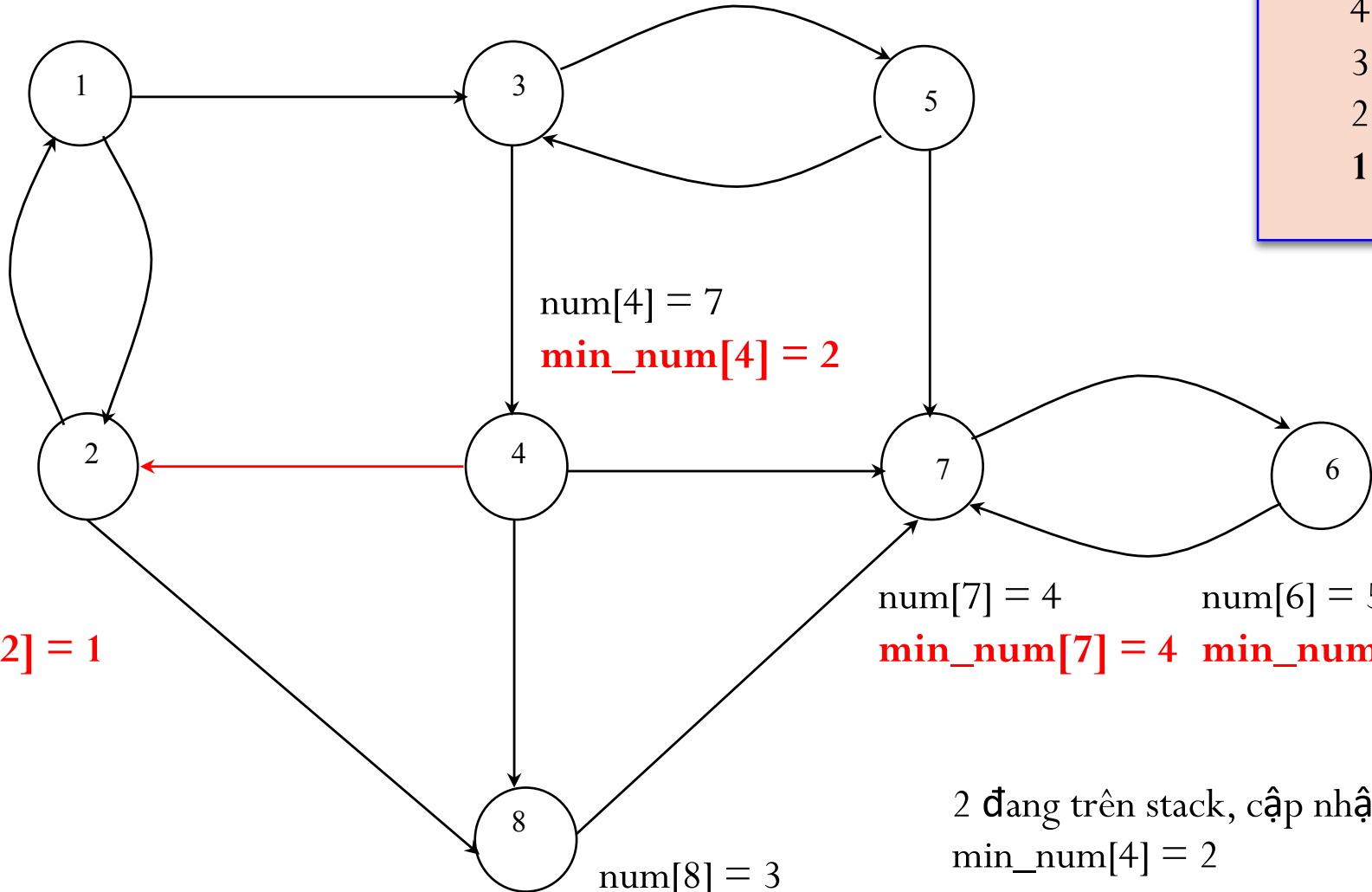
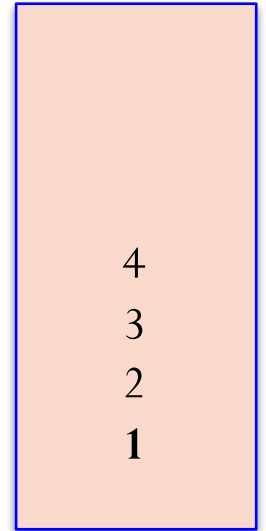
$\text{num}[2] = 2$   
 $\text{min\_num}[2] = 1$

$\text{num}[7] = 4$   
 $\text{min\_num}[7] = 4$

$\text{num}[6] = 5$   
 $\text{min\_num}[6] = 4$

$\text{num}[8] = 3$   
 $\text{min\_num}[8] = 3$

2 đang trên stack, cập nhật  
 $\text{min\_num}[4] = 2$



$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$

$\text{num}[3] = 6$   
 $\text{min\_num}[3] = 6$

$\text{num}[4] = 7$   
 $\text{min\_num}[4] = 2$

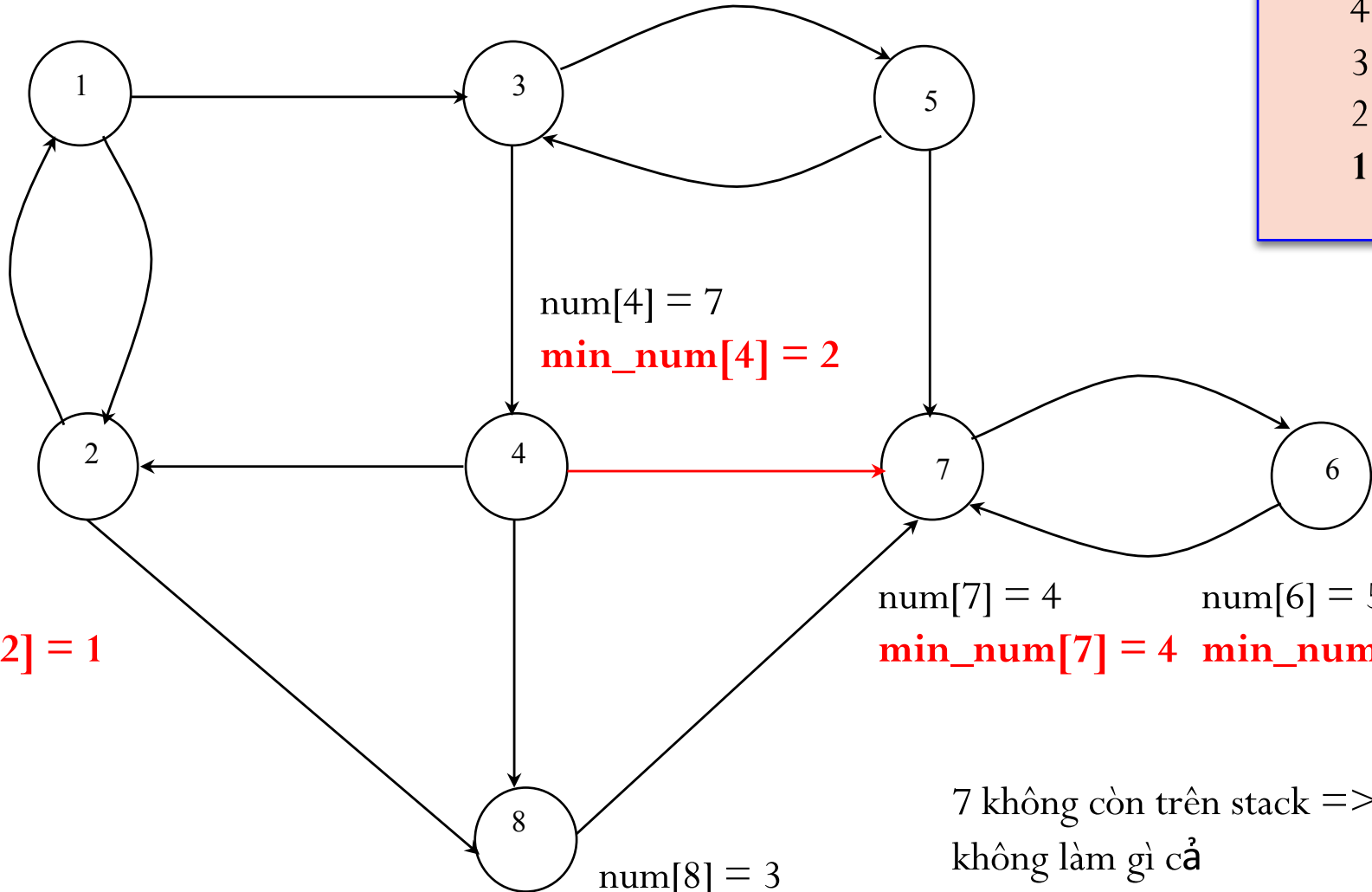
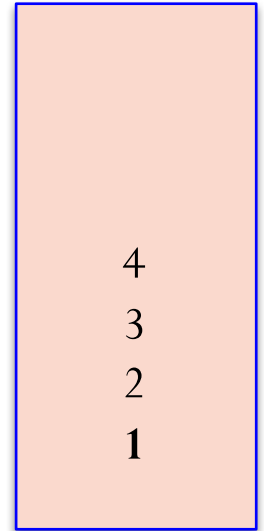
$\text{num}[2] = 2$   
 $\text{min\_num}[2] = 1$

$\text{num}[7] = 4$   
 $\text{min\_num}[7] = 4$

$\text{num}[6] = 5$   
 $\text{min\_num}[6] = 4$

$\text{num}[8] = 3$   
 $\text{min\_num}[8] = 3$

7 không còn trên stack =>  
không làm gì cả



num[1] = 1  
**min\_num[1] = 1**

num[3] = 6  
min\_num[3] = 6

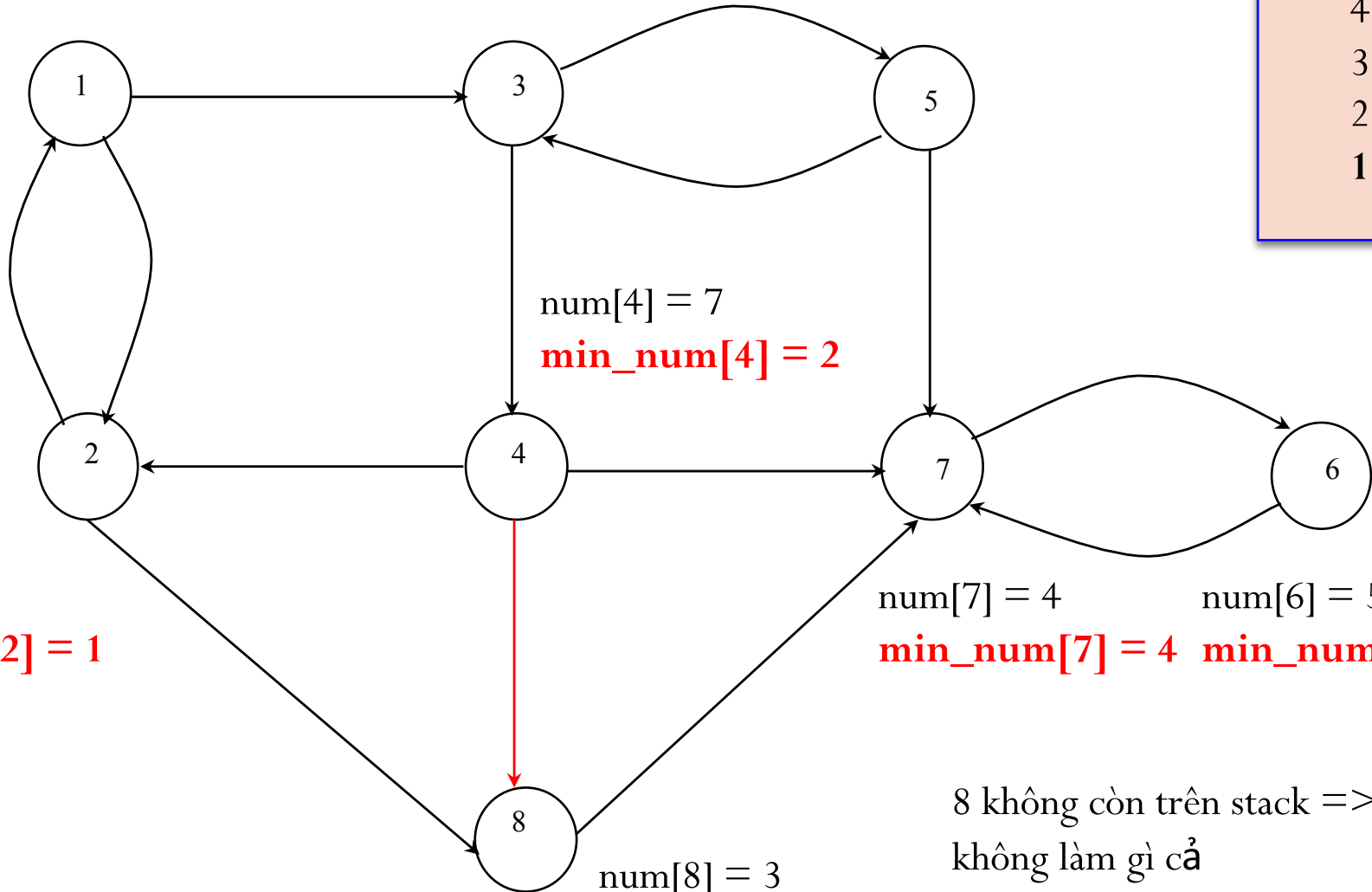
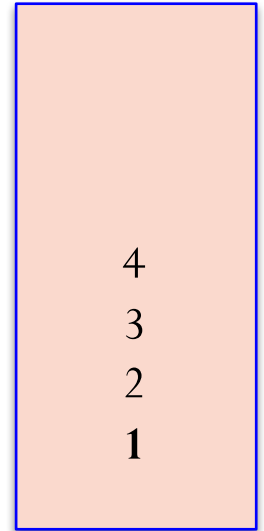
num[4] = 7  
**min\_num[4] = 2**

num[2] = 2  
**min\_num[2] = 1**

num[7] = 4  
**min\_num[7] = 4**    num[6] = 5  
**min\_num[6] = 4**

num[8] = 3  
**min\_num[8] = 3**

8 không còn trên stack =>  
không làm gì cả

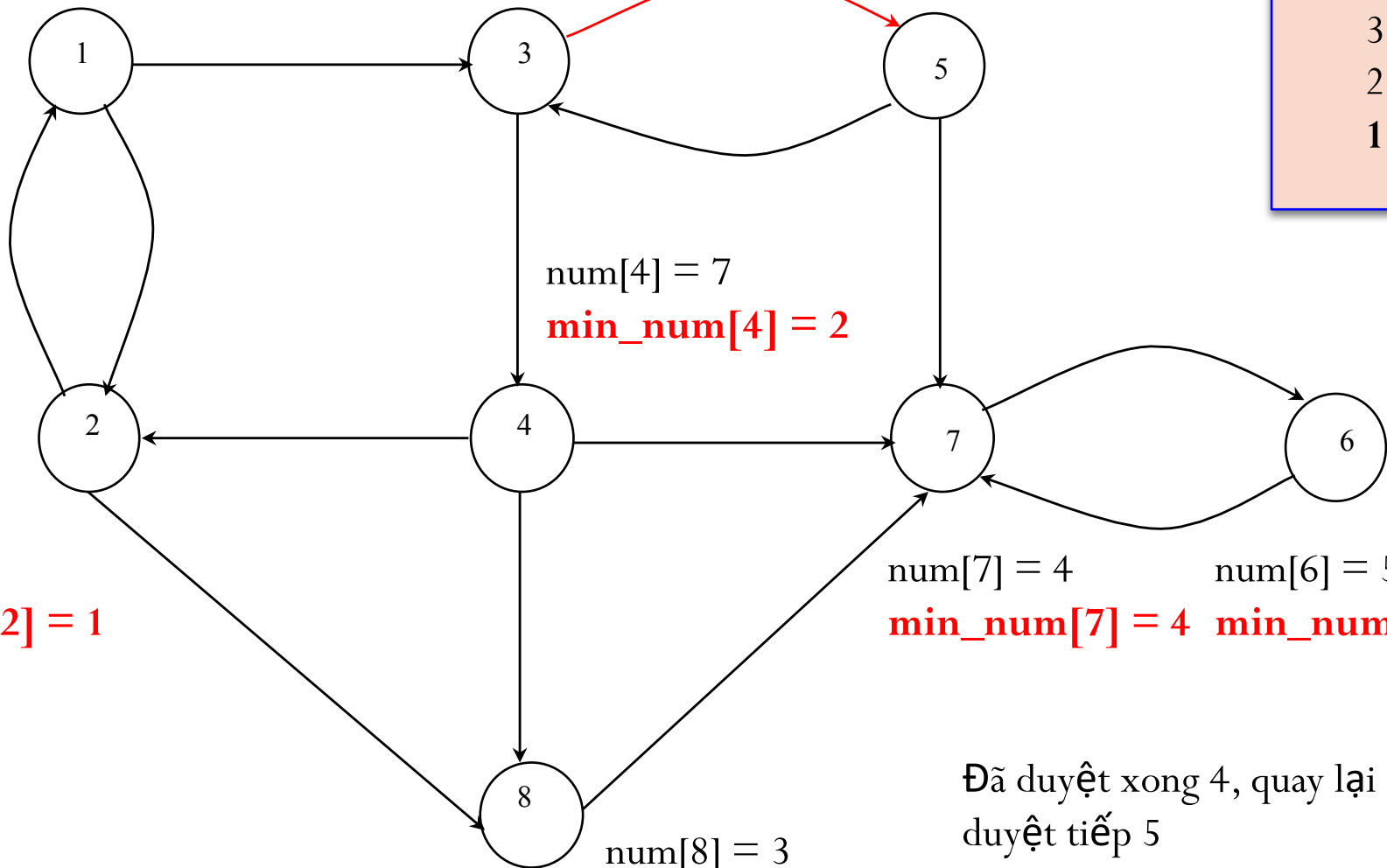


$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$

$\text{num}[3] = 6$   
 $\text{min\_num}[3] = 2$

$\text{num}[5] = 8$   
 $\text{min\_num}[5] = 8$

5  
4  
3  
2  
1



$\text{num}[4] = 7$   
 $\text{min\_num}[4] = 2$

$\text{num}[2] = 2$   
 $\text{min\_num}[2] = 1$

$\text{num}[7] = 4$   
 $\text{min\_num}[7] = 4$

$\text{num}[6] = 5$   
 $\text{min\_num}[6] = 4$

$\text{num}[8] = 3$   
 $\text{min\_num}[8] = 3$

Đã duyệt xong 4, quay lại  
duyệt tiếp 5

$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$

$\text{num}[3] = 6$   
 $\text{min\_num}[3] = 2$

$\text{num}[5] = 8$   
 $\text{min\_num}[5] = 6$

$\text{num}[4] = 7$   
 $\text{min\_num}[4] = 2$

$\text{num}[2] = 2$   
 $\text{min\_num}[2] = 1$

$\text{num}[7] = 4$   
 $\text{min\_num}[7] = 4$

$\text{num}[6] = 5$   
 $\text{min\_num}[6] = 4$

$\text{num}[8] = 3$   
 $\text{min\_num}[8] = 3$

3 trên stack, cập nhật  
 $\text{min\_num}[5] = 6$

5  
4  
3  
2  
1

$\text{num}[1] = 1$   
 $\text{min\_num}[1] = 1$

$\text{num}[3] = 6$   
 $\text{min\_num}[3] = 2$

$\text{num}[5] = 8$   
 $\text{min\_num}[5] = 6$

$\text{num}[4] = 7$   
 $\text{min\_num}[4] = 2$

$\text{num}[2] = 2$   
 $\text{min\_num}[2] = 1$

$\text{num}[7] = 4$   
 $\text{min\_num}[7] = 4$

$\text{num}[6] = 5$   
 $\text{min\_num}[6] = 4$

$\text{num}[8] = 3$   
 $\text{min\_num}[8] = 3$

7 không còn trên stack =>  
không làm gì cả

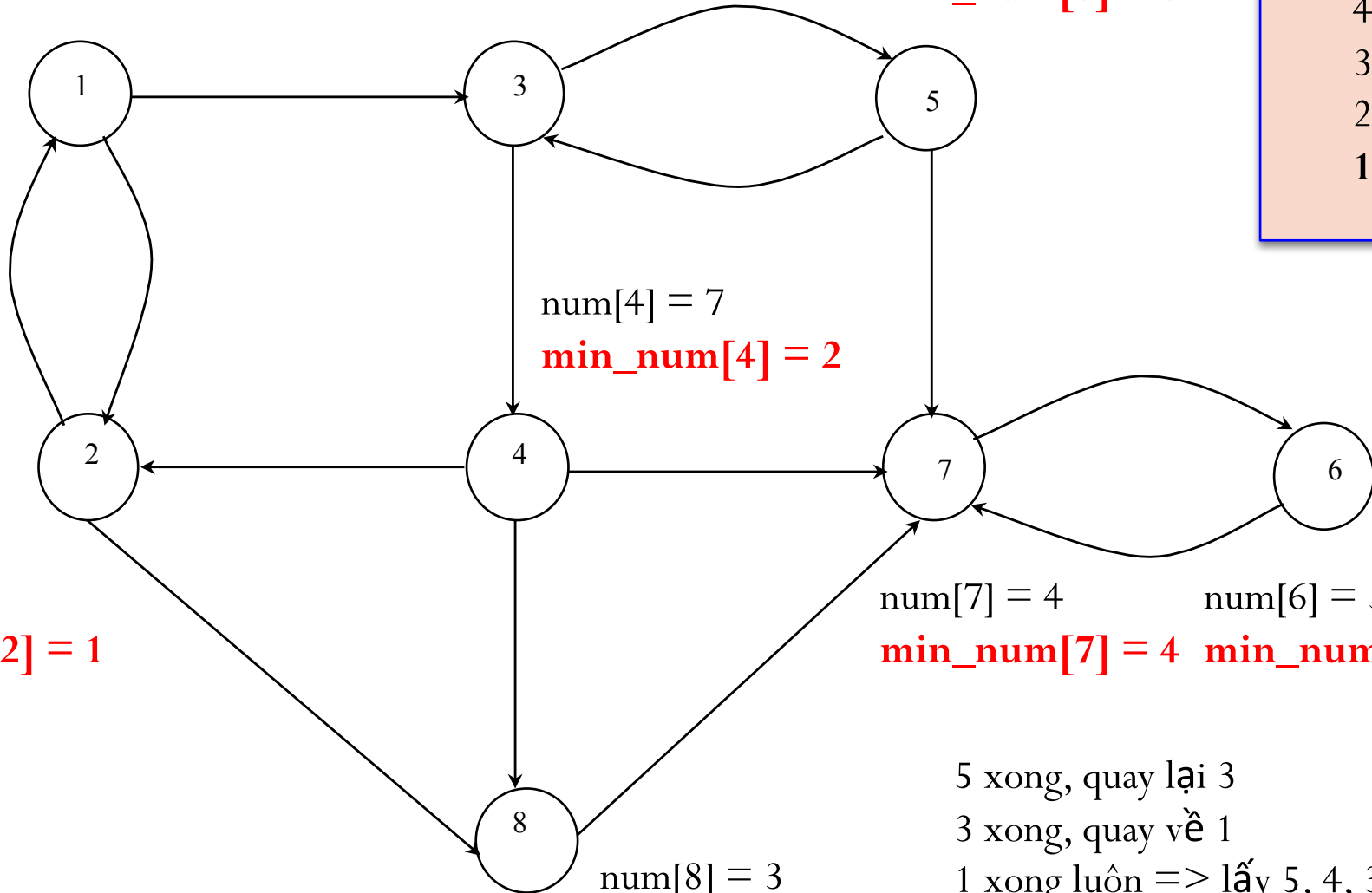
5  
4  
3  
2  
1

num[1] = 1  
**min\_num[1] = 1**

num[3] = 6  
**min\_num[3] = 2**

num[5] = 8  
**min\_num[5] = 6**

5  
4  
3  
2  
1



num[2] = 2  
**min\_num[2] = 1**

num[4] = 7  
**min\_num[4] = 2**

num[7] = 4  
**min\_num[7] = 4**

num[6] = 5  
**min\_num[6] = 4**

num[8] = 3  
**min\_num[8] = 3**

5 xong, quay lại 3  
3 xong, quay về 1  
1 xong luôn => lấy 5, 4, 3, 2,  
1 ra khỏi stack

## Lưu ý:

- Thứ tự các đỉnh kề của 1 đỉnh được sắp xếp từ bé đến lớn.
- Khi duyệt xong 1 đỉnh  $y$ , quay về đỉnh cha  $x$  (đỉnh trước), cập nhật lại  $\text{min\_num}[x]$  (so với  **$\text{min\_num}[y]$** )
- Khi xét 1 đỉnh kề  $y$  của  $x$  mà  $y$  đang có mặt trong stack  $\Rightarrow$  cập nhật lại  $\text{min\_num}[x]$  (so với  **$\text{num}[y]$** ).



