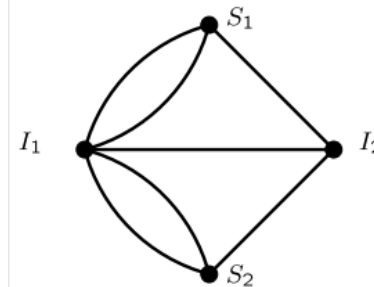
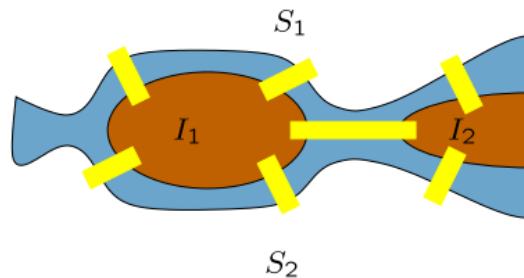


LÝ THUYẾT ĐỒ THỊ

Tính Liên thông

Phạm Nguyên Khang
BM. Khoa học máy tính, CNTT
pnkhang@cit.ctu.edu.vn



Cần Thơ, 8/2021

Trong tuần 2

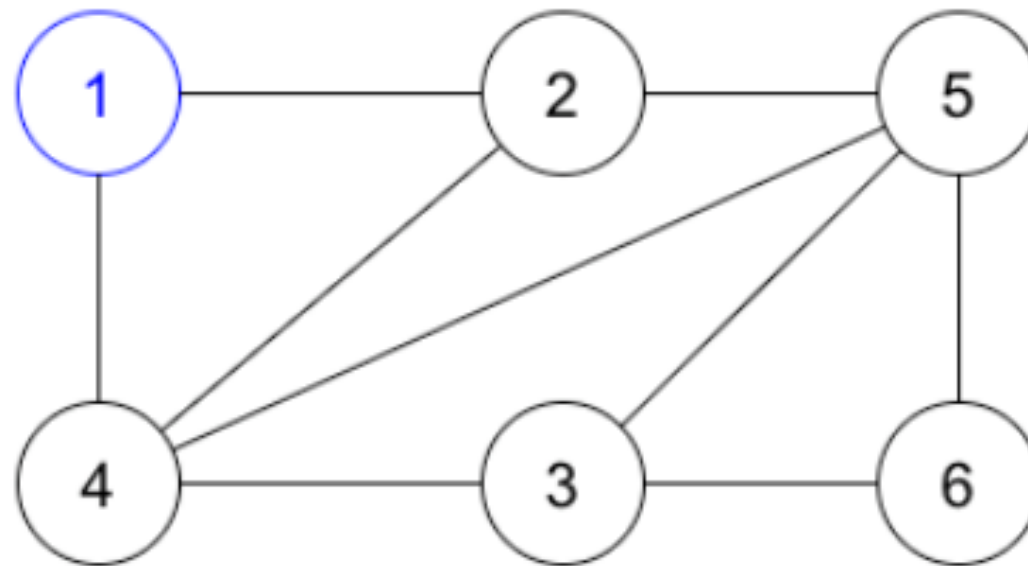
- Biểu diễn đồ thị
- Duyệt đồ thị
- Các phương pháp duyệt đồ thị

Duyệt đồ thị

- Đi qua tất cả đỉnh của một đồ thị
 - Cập nhật và/hoặc kiểm tra giá trị của chúng trên đường đi
- Ý tưởng:
 - Bắt đầu từ 1 đỉnh bất kỳ đánh dấu nó đã duyệt
 - Duyệt các đỉnh kề của nó
 - Duyệt các đỉnh kề của đỉnh kề của nó
 - ...
- Phương pháp duyệt
 - Duyệt theo chiều rộng
 - Duyệt theo chiều sâu

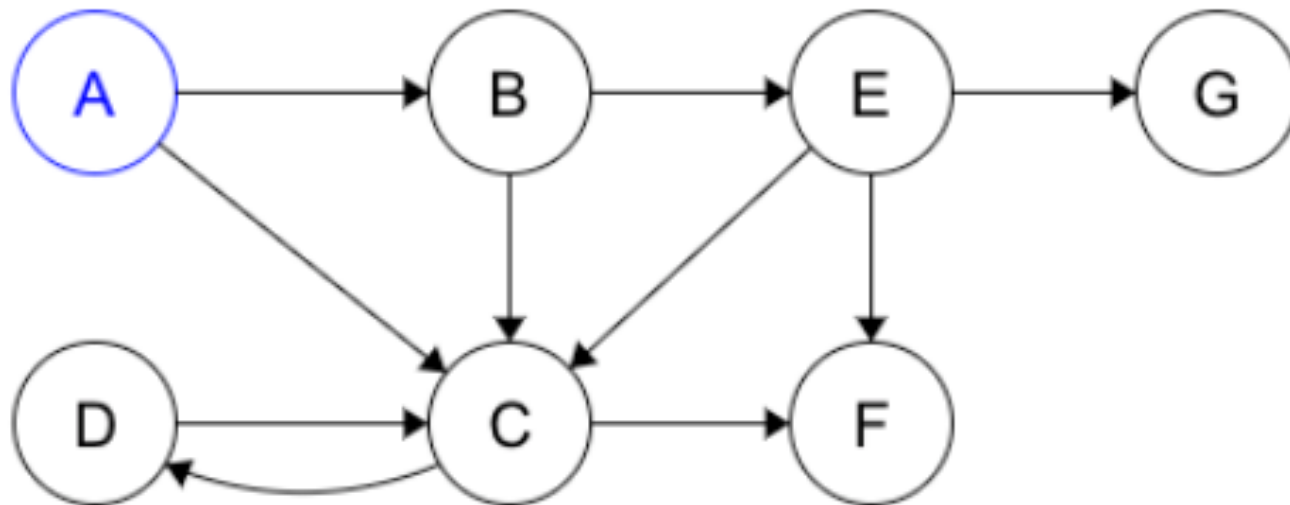
Duyệt đồ thị

- Cho biết thứ tự duyệt theo chiều rộng và chiều sâu của đồ thị sau (bắt đầu từ đỉnh 1)



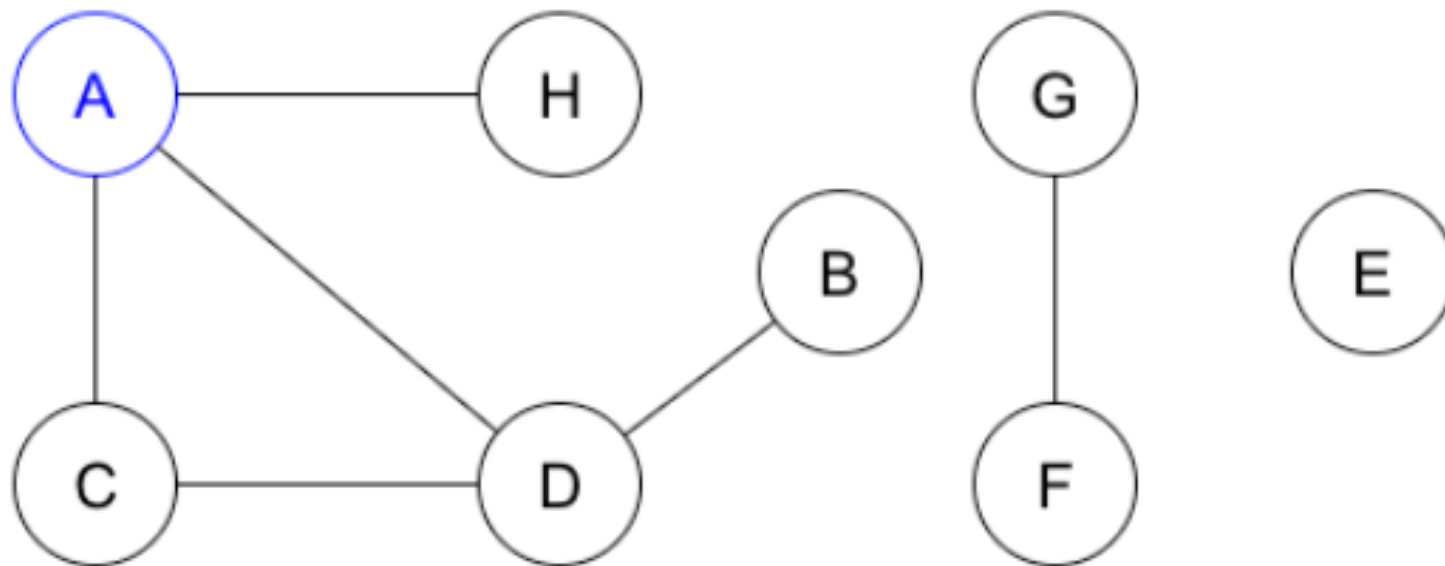
Duyệt đồ thị

- Cho biết thứ tự duyệt theo chiều rộng và chiều sâu của đồ thị sau (bắt đầu từ đỉnh A)



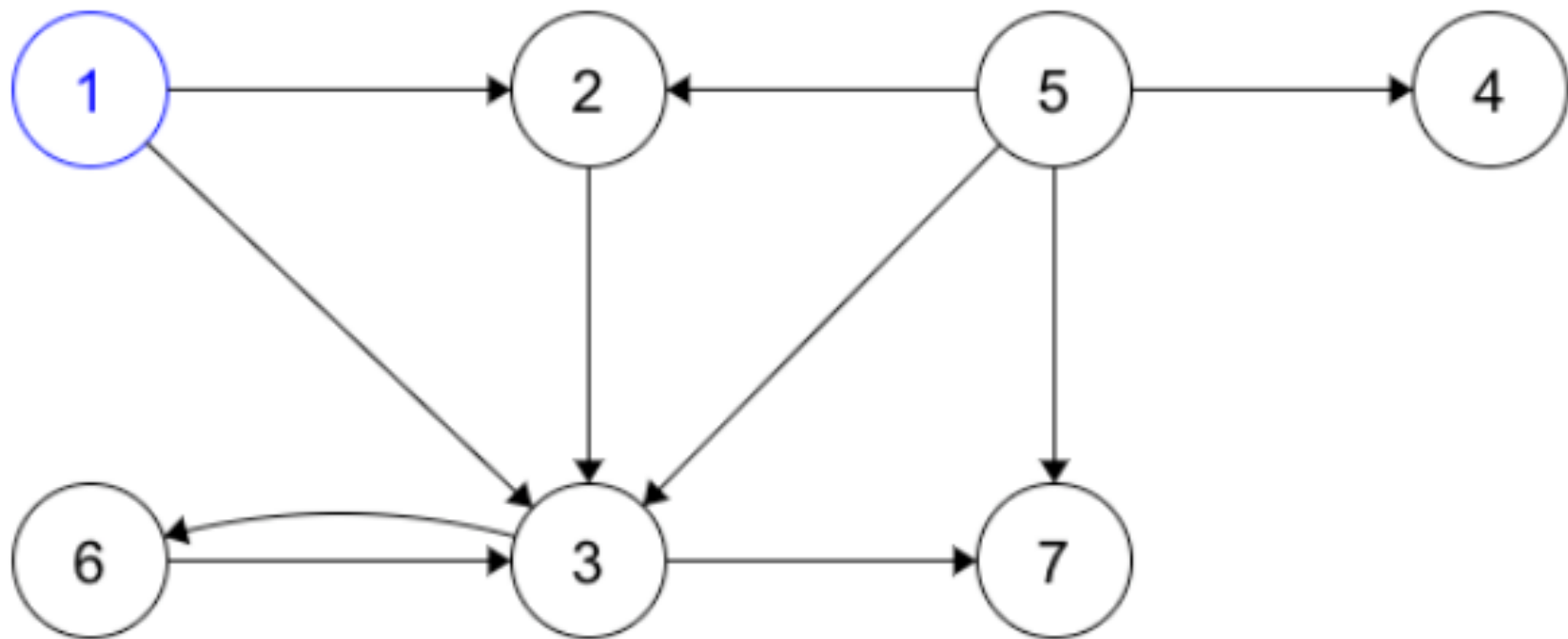
Duyệt đồ thị

- Cho biết thứ tự duyệt theo chiều rộng và chiều sâu của đồ thị sau (bắt đầu từ đỉnh A)



Duyệt đồ thị

- Cho biết thứ tự duyệt theo chiều rộng và chiều sâu của đồ thị sau (bắt đầu từ đỉnh 1)



Duyệt đồ thị

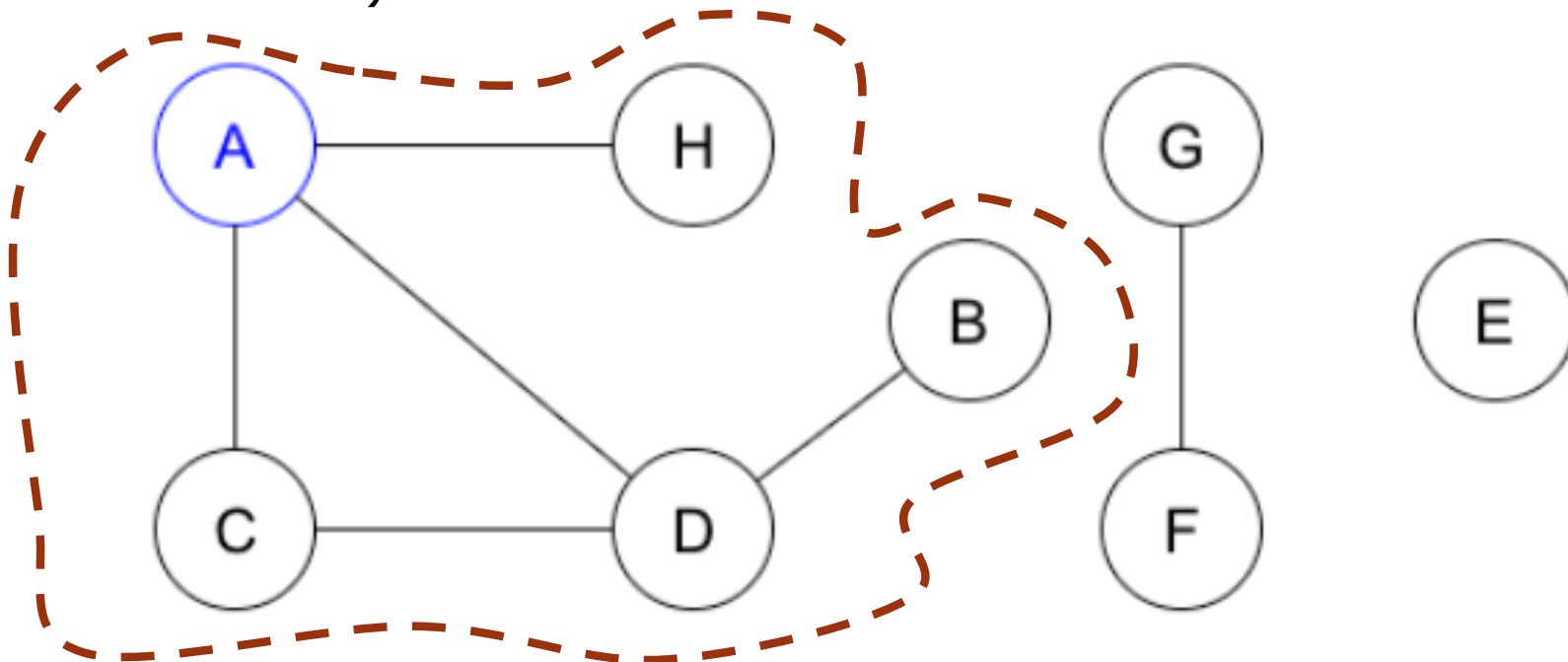
- Nhận xét
 - Khi duyệt đồ thị từ 1 đỉnh s , ta có chắc chắn là sẽ duyệt qua hết tất cả các đỉnh của đồ thị không?
 - Đồ thị vô hướng
 - Đồ thị có hướng

Duyệt đồ thị

- Nhận xét
 - Khi duyệt đồ thị từ 1 đỉnh s , ta có chắc chắn là sẽ duyệt qua hết tất cả các đỉnh của đồ thị không?
 - KHÔNG
 - Ta chỉ duyệt qua những đỉnh “đến được từ s ” (reachable from s) mà thôi.

Duyệt đồ thị

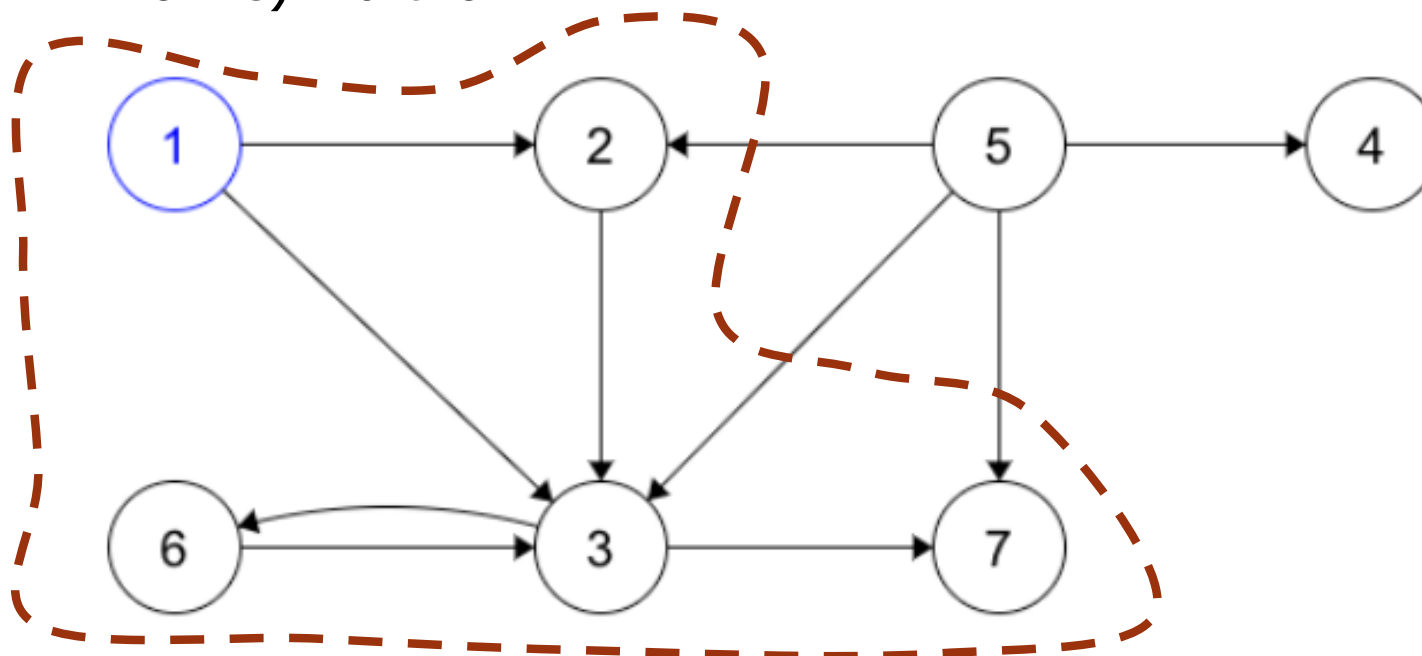
- Nhận xét
 - Khi duyệt đồ thị từ 1 đỉnh s , ta có chắc chắn là sẽ duyệt qua hết tất cả các đỉnh của đồ thị không?
 - KHÔNG
 - Ta chỉ duyệt qua những đỉnh "đến được từ s " (reachable from s) mà thôi.



Duyệt đồ thị

- Nhận xét

- Khi duyệt đồ thị từ 1 đỉnh s , ta có chắc chắn là sẽ duyệt qua hết tất cả các đỉnh của đồ thị không?
 - KHÔNG
 - Ta chỉ duyệt qua những đỉnh "đến được từ s " (reachable from s) mà thôi.



Tuần này

- Liên thông (connected)
- Tính liên thông (connectivity)



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).

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 .
 - ∞ , 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

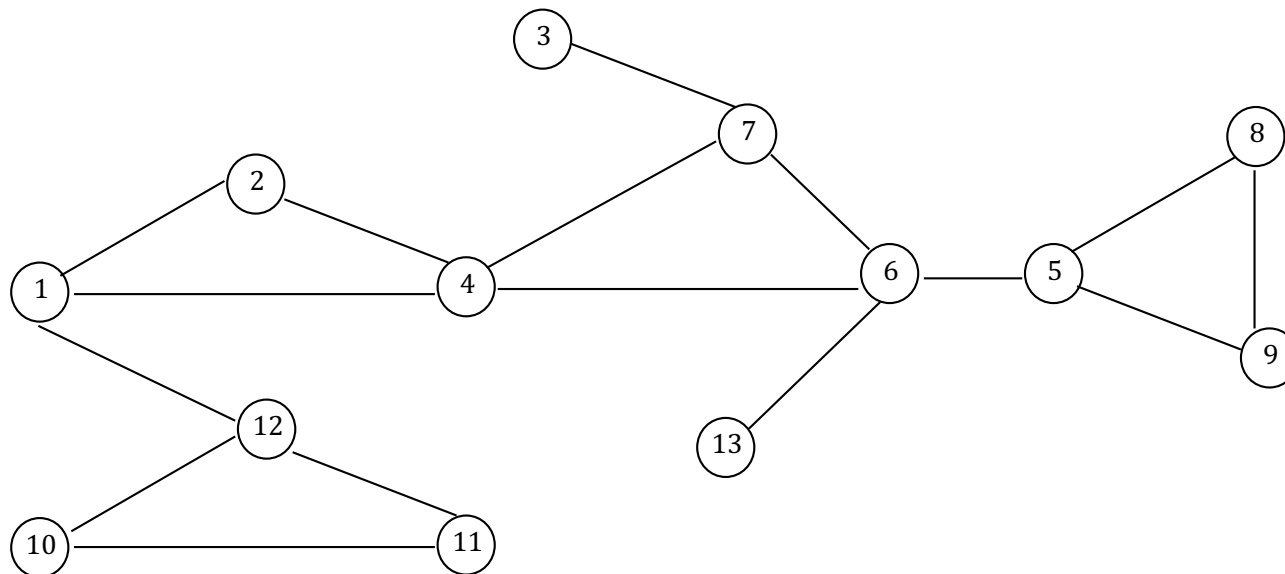
2. CMR: Số cung ít nhất của một đồ thị vô hướng liên thông có n đỉnh là $n - 1$.
 - Gợi ý: C/M quy nạp trên số cung.

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.

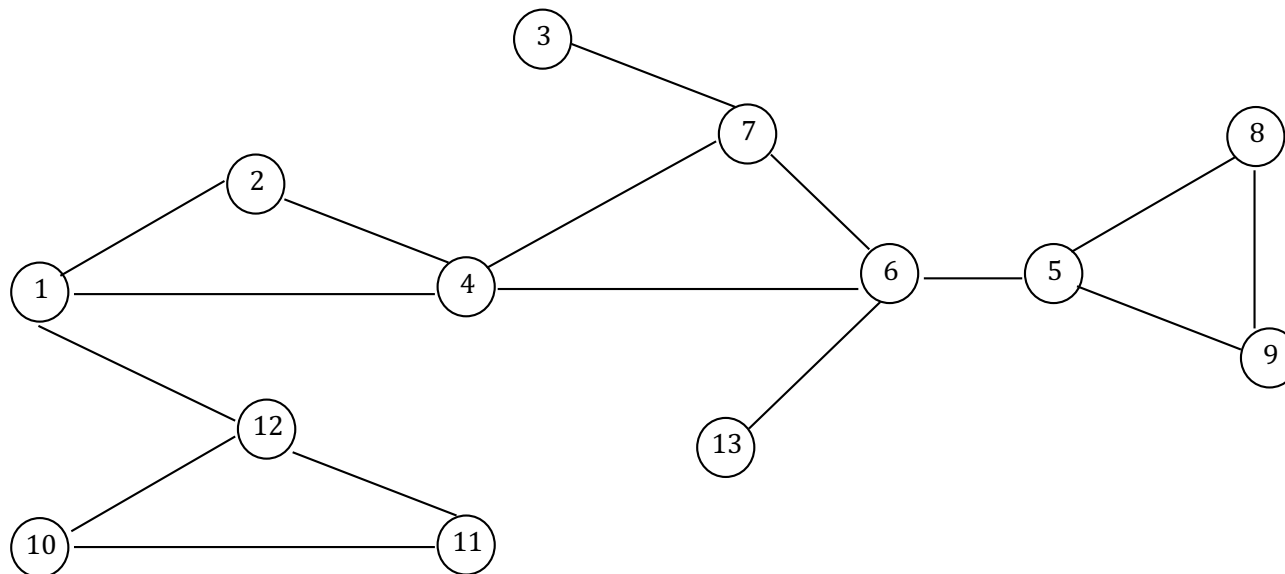
Thuật toán kiểm tra tính liên thông của đồ thị

- Áp dụng thuật toán 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.



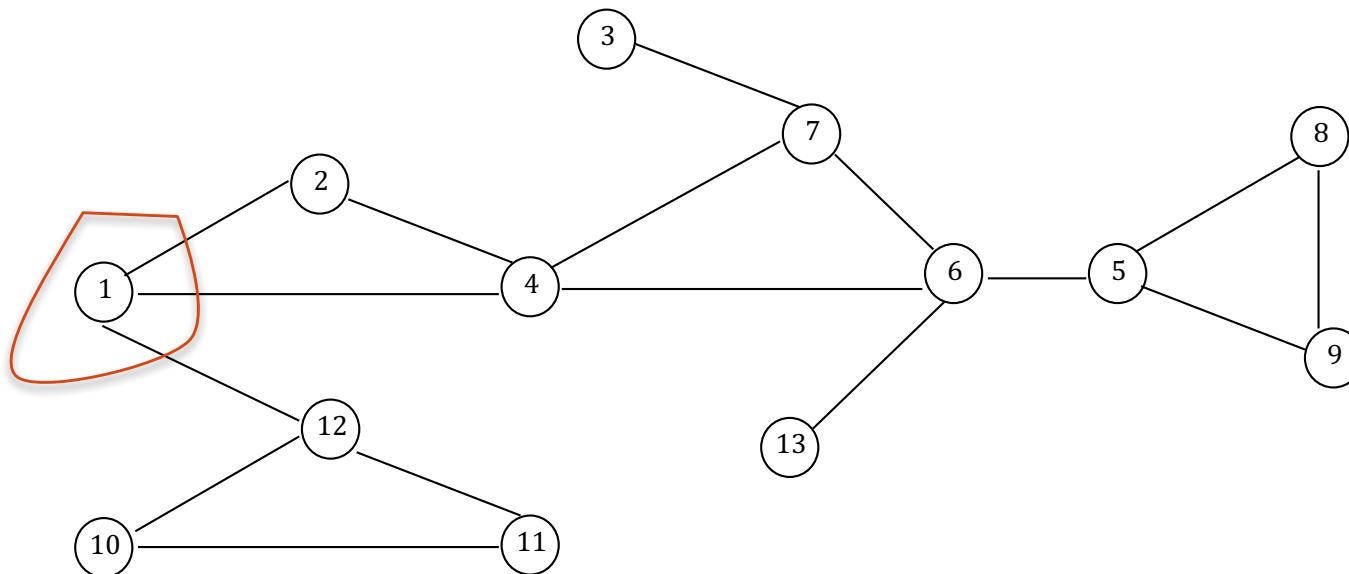
Duyệt đồ thị (nhắc lại)

- 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, ...



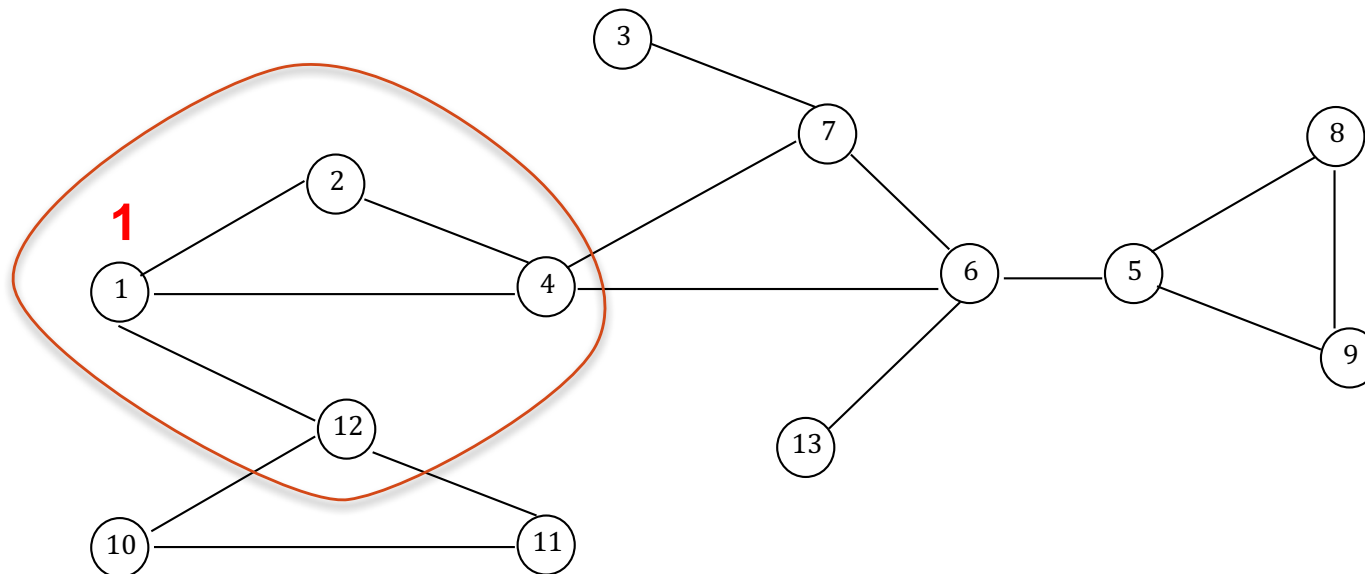
Duyệt đồ thị (nhắc lại)

- 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, ...



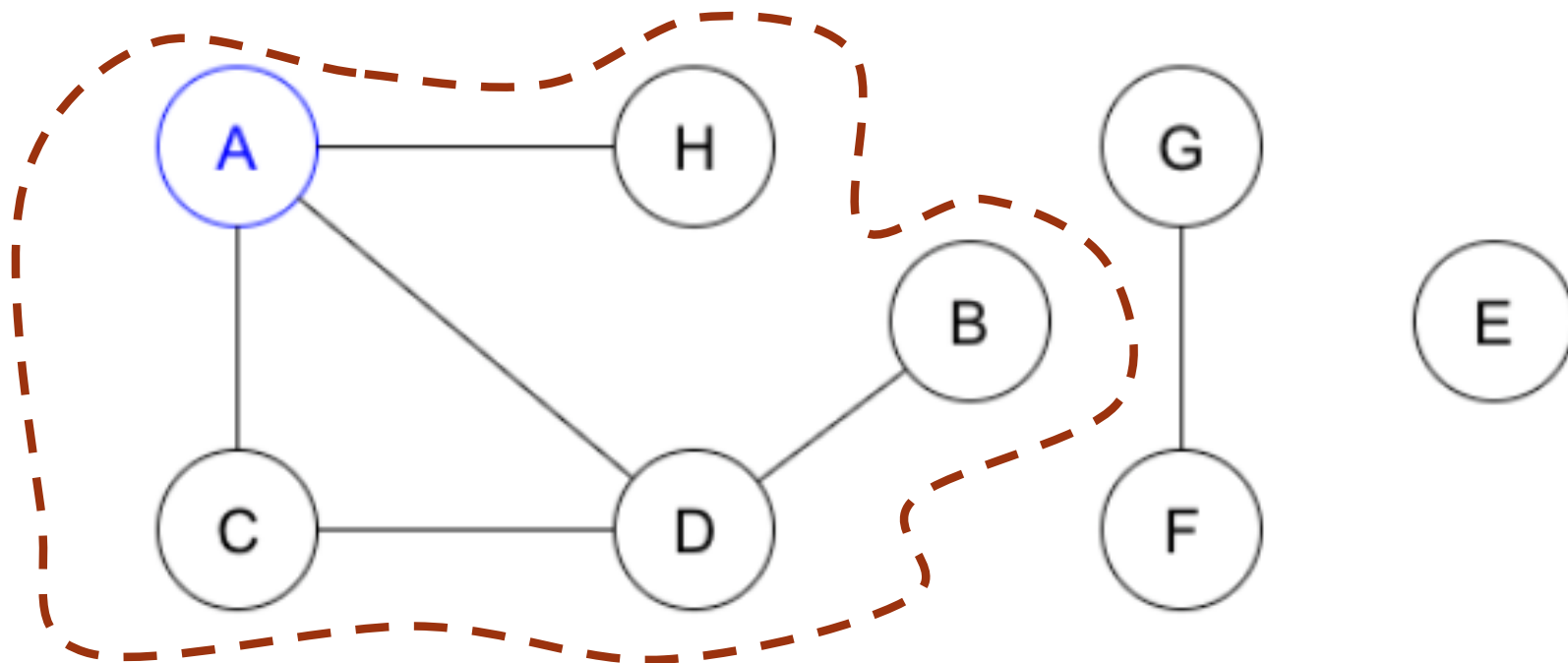
Duyệt đồ thị (nhắc lại)

- 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, ...



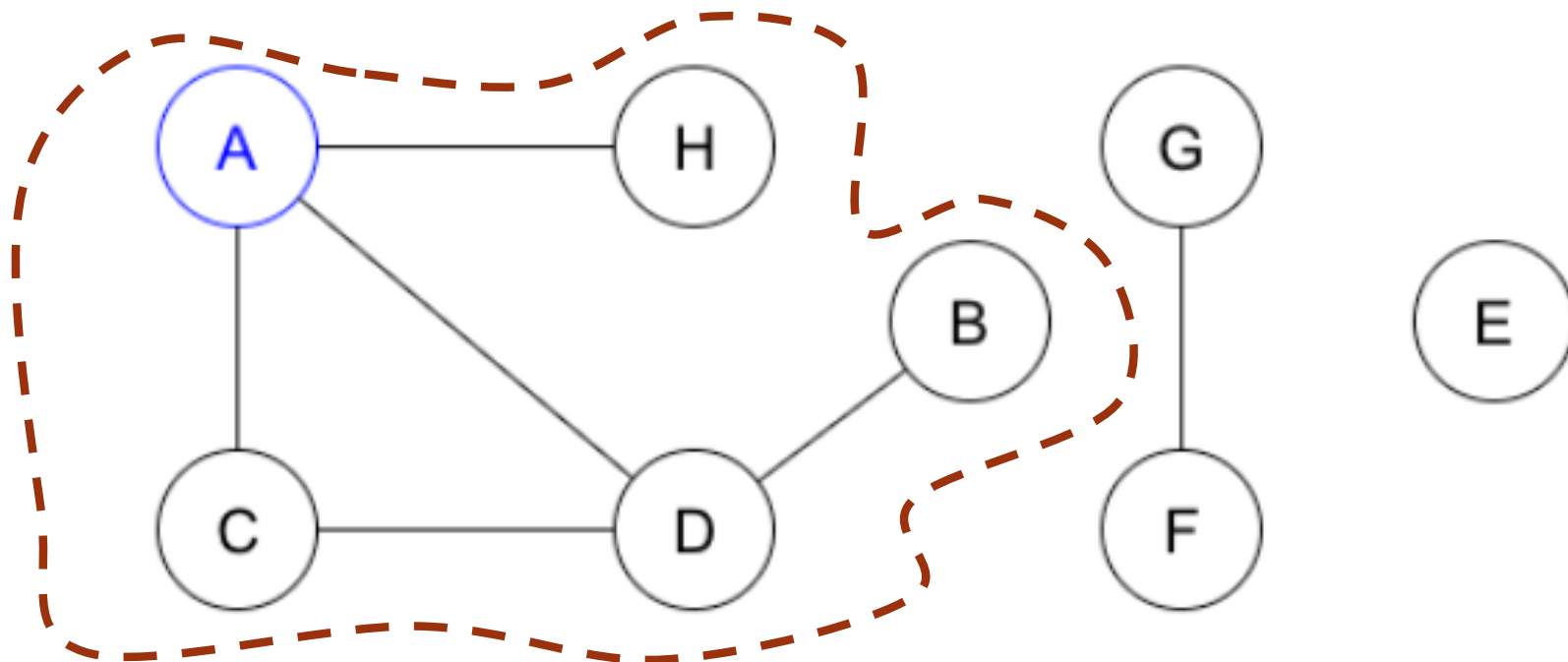
Duyệt đồ thị (nhắc lại)

- Khi duyệt đồ thị từ đỉnh s ta sẽ duyệt qua được các đỉnh "đến được từ s " (reachable from s)



Duyệt đồ thị (nhắc lại)

- Khi duyệt đồ thị từ đỉnh s ta sẽ duyệt qua được các đỉnh "đến được từ s " (reachable from s)
- Các đỉnh đến được từ s (bao gồm cả s) là **bộ phận liên thông chứa s** .



Thuật toán kiểm tra tính liên thông của đồ thị

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

```
void DFS(Graph* G, int u) {  
    if (num[u] > 0) return; //u đã duyệt rồi  
    num[u] = k; k++;  
    for (các đỉnh kề v của u)  
        DFS(G, u);  
}
```

- Ví dụ: gọi **DFS(1)**; Khi thuật toán kết thúc, các đỉnh được đánh số ($\text{num}[u] > 0$) chính là bộ phận liên thông chứa 1.

Thuật toán tìm tất cả các bộ phận liên thông (thực hành)

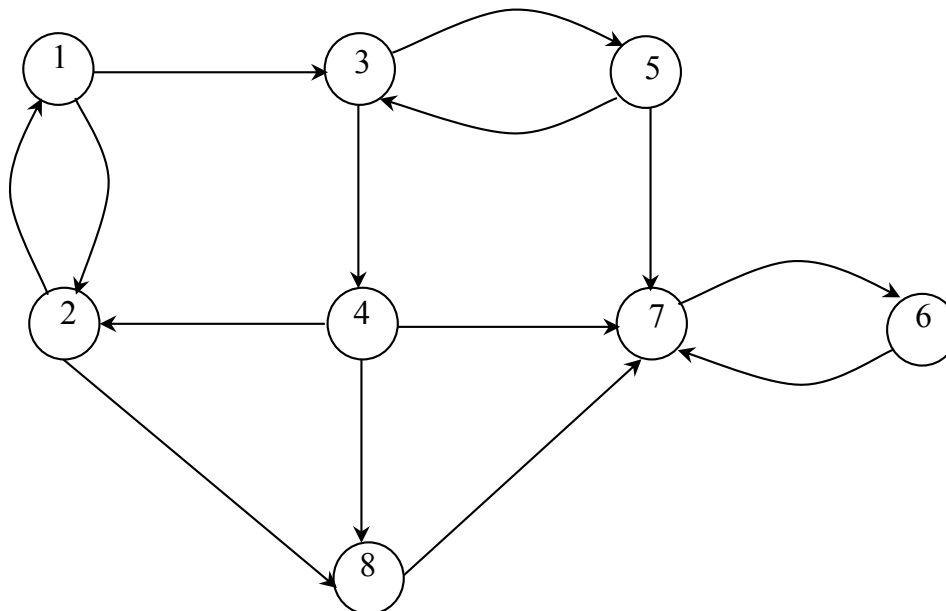
- Sử dụng giải thuật duyệt đồ thị
- Khởi tạo tất cả các đỉnh có $\text{num}[u] = -1$ (chưa duyệt)
- Xét qua các đỉnh, nếu u chưa được đánh dấu \Rightarrow gọi **DFS(u)** để duyệt nó.

```
for (u = 1; u <= n; u++)  
    if (num[u] < 0) { //u chưa duyệt  
        DFS(G, u);  
        //Tìm được 1 bộ phận liên thông chứa u.  
    }
```

- Mỗi lần duyệt xong 1 đỉnh u , ta sẽ tìm được 1 bộ phận liên thông chứa u .
- Có thể thay DFS bằng BFS.

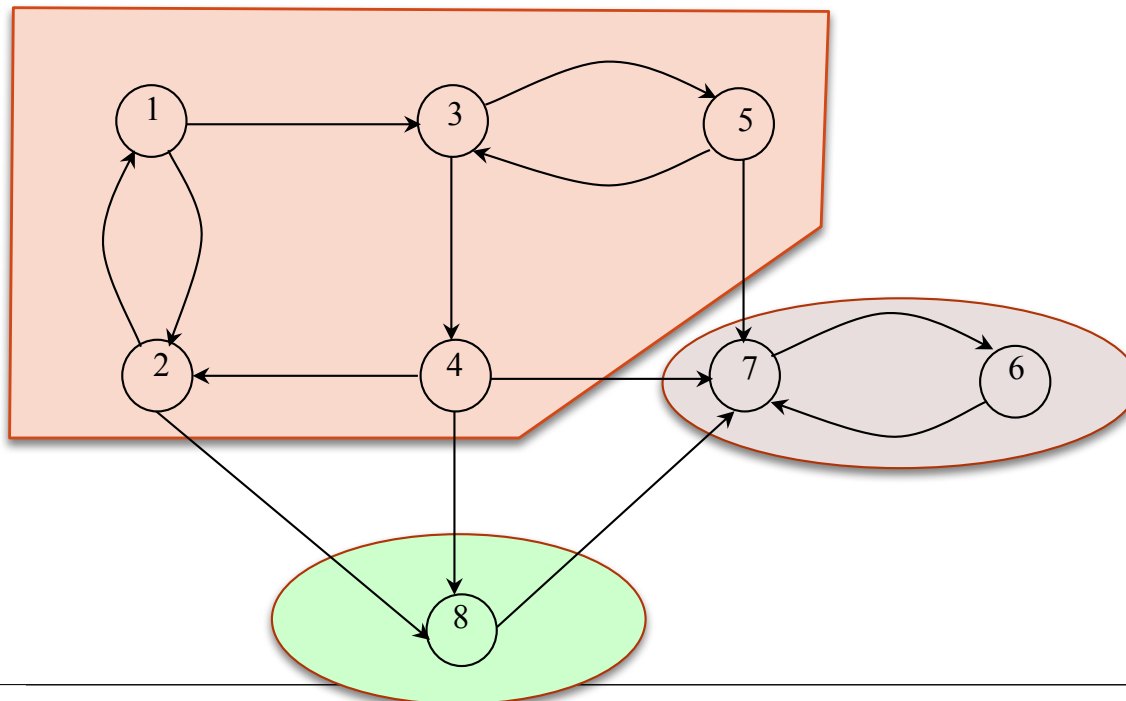
Tính liên thông của đồ thị có hướng

- 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

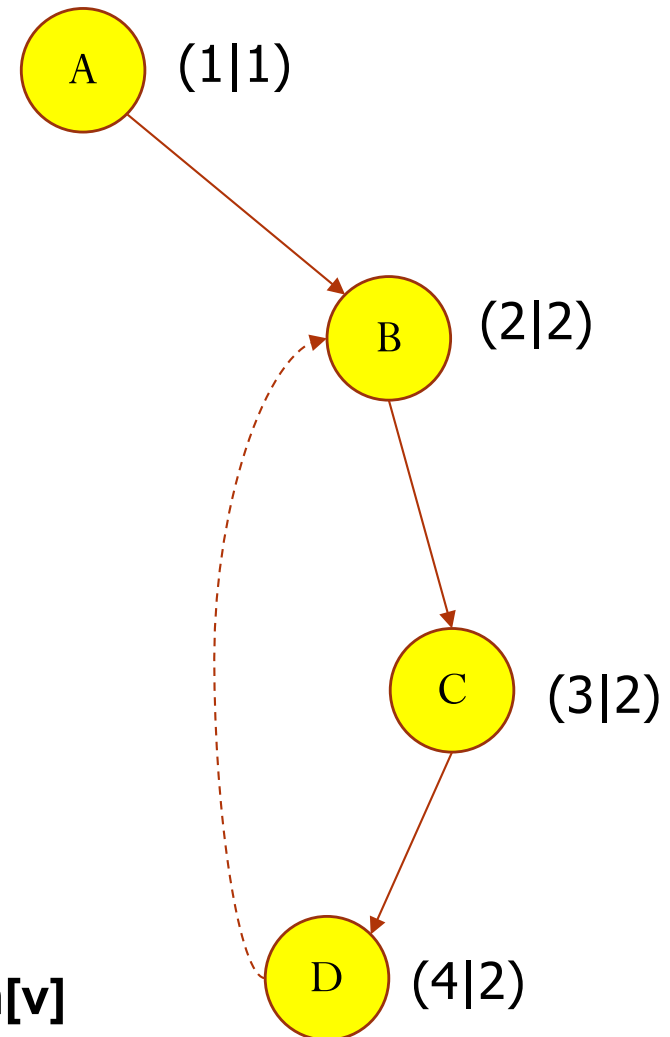
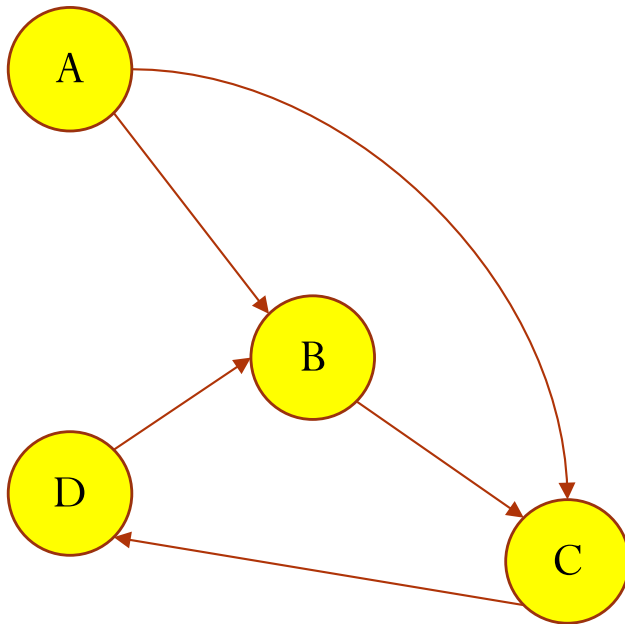
Tìm các bộ phận liên thông mạnh

- Thuật toán 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 $num[v]$, ta lưu thêm $min_num[v]$ (là chỉ số nhỏ nhất trong các đỉnh có thể đi đến được từ v). Trong quá trình duyệt, $min_num[v]$ sẽ được cập nhật.
 - Khi duyệt xong 1 đỉnh, nếu $num[v] = min_num[v]$ thì v là đỉnh bắt đầu (đỉnh gốc/đỉnh khớp) của bộ phận liên thông mạnh.

Tìm các bộ phận liên thông mạnh

- Các biến hỗ trợ:
 - **S**: stack lưu các đỉnh chưa tìm được BPLT mạnh
 - **on_stack[v]**: kiểm tra v còn trên stack không
 - **num[v]**: chỉ số của đỉnh v trong quá trình duyệt
 - **min_num[v]**: chỉ số nhỏ nhất trong các chỉ số của các đỉnh trong stack S mà v đi đến nó được.
 - **k**: chỉ số dùng để gán cho num của các đỉnh (tăng dần)

Tìm các bộ phận liên thông mạnh



num[v] và min_num[v]

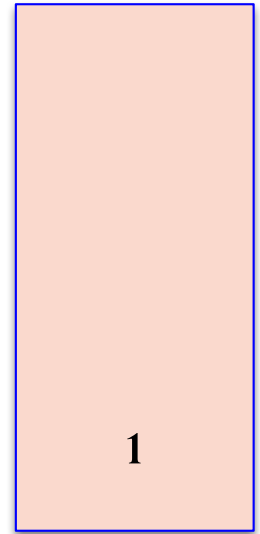
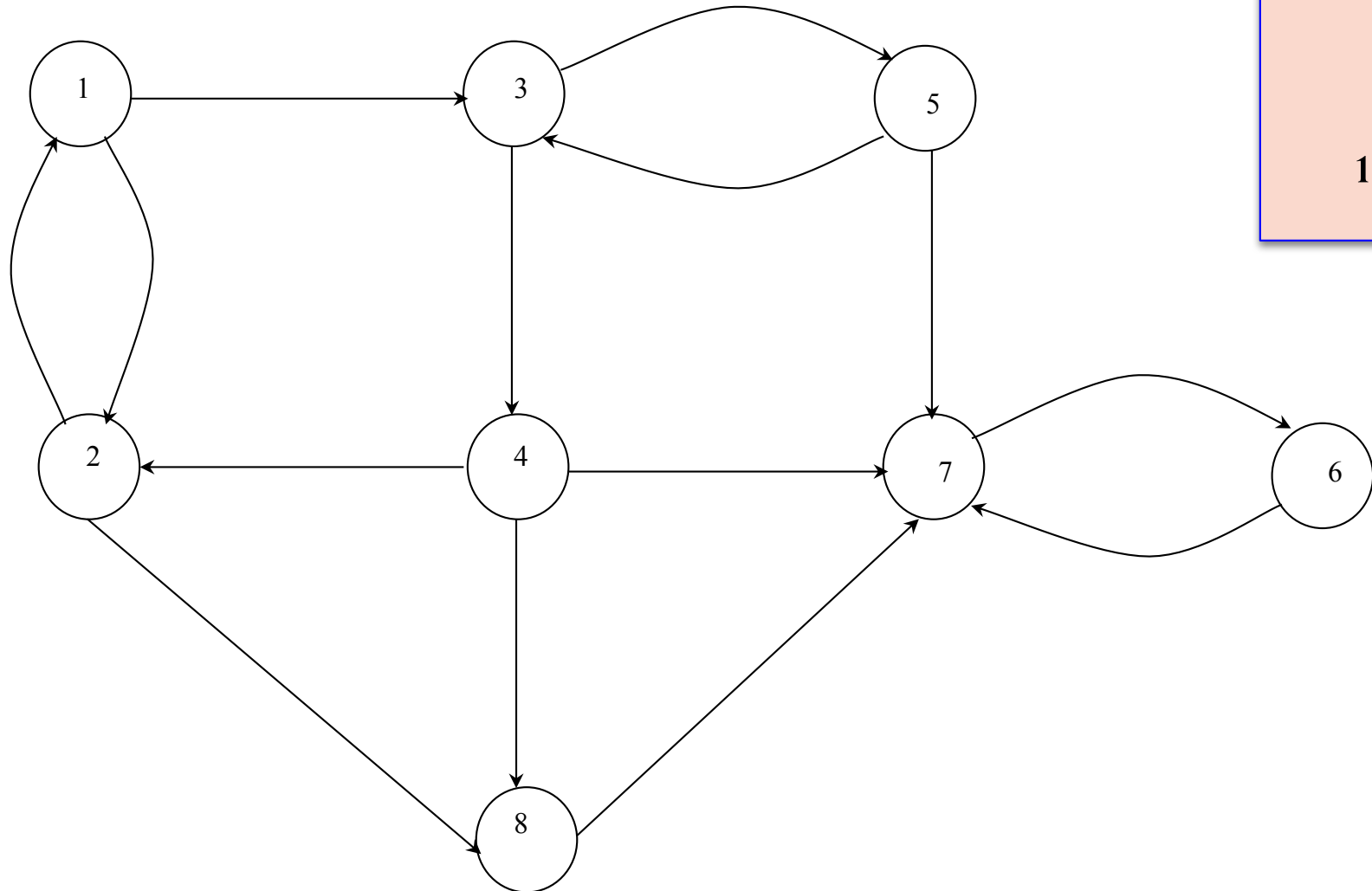
Tìm các bộ phận liên thông mạnh

```
void SCC(int u) {  
    //1. Đánh số cho đỉnh u, đưa u vào ngăn xếp  
    num[u] = k; min_num[u] = k; k++;  
    push(S, u); on_stack[u] = true;  
    //2. Lần lượt xét các đỉnh kề của u  
    for (v là các đỉnh kề của u)  
        if (v chưa duyệt) {  
            //2a. Duyệt v và cập nhật lại min_num[u]  
            SCC(v);  
            min_num[u] = min(min_id[u], min_num[v]);  
        } else if (v còn trên stack) {  
            //2b. cập nhật lại min_num[u]  
            min_num[u] = min(min_num[u], num[v]);  
        } else { //2c. bỏ qua, không làm gì cả }  
    //3. Kiểm tra num[u] == min_num[u]  
}
```

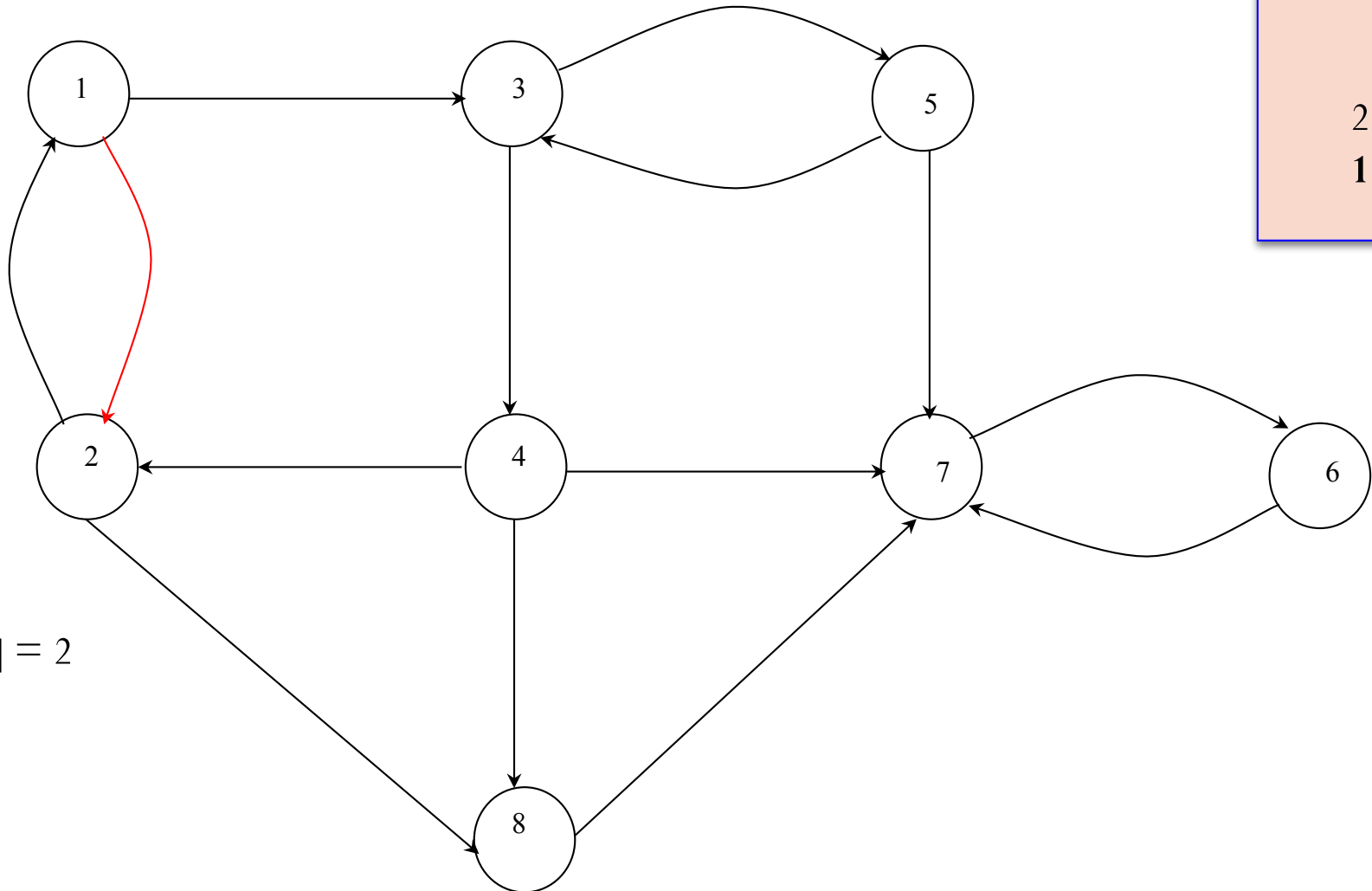
Tìm các bộ phận liên thông mạnh

```
void SCC(int u) {  
    //1. Đánh số cho đỉnh u, đưa u vào ngăn xếp  
    //2. Lần lượt xét các đỉnh kề của u  
    //3. Kiểm tra num[u] == min_num[u]  
    if (num[u] == min_num[u]) {  
        //Lấy các đỉnh trong stack ra cho đến khi gặp u  
        //Các đỉnh này thuộc về một thành phần liên thông  
  
        int w;  
        do {  
            w = top(&S); pop(&S);  
            on_stack[w] = 0;  
            //làm gì đó trên w, vd: in ra màn hình  
        } while (w != u);  
    }  
}
```

$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$

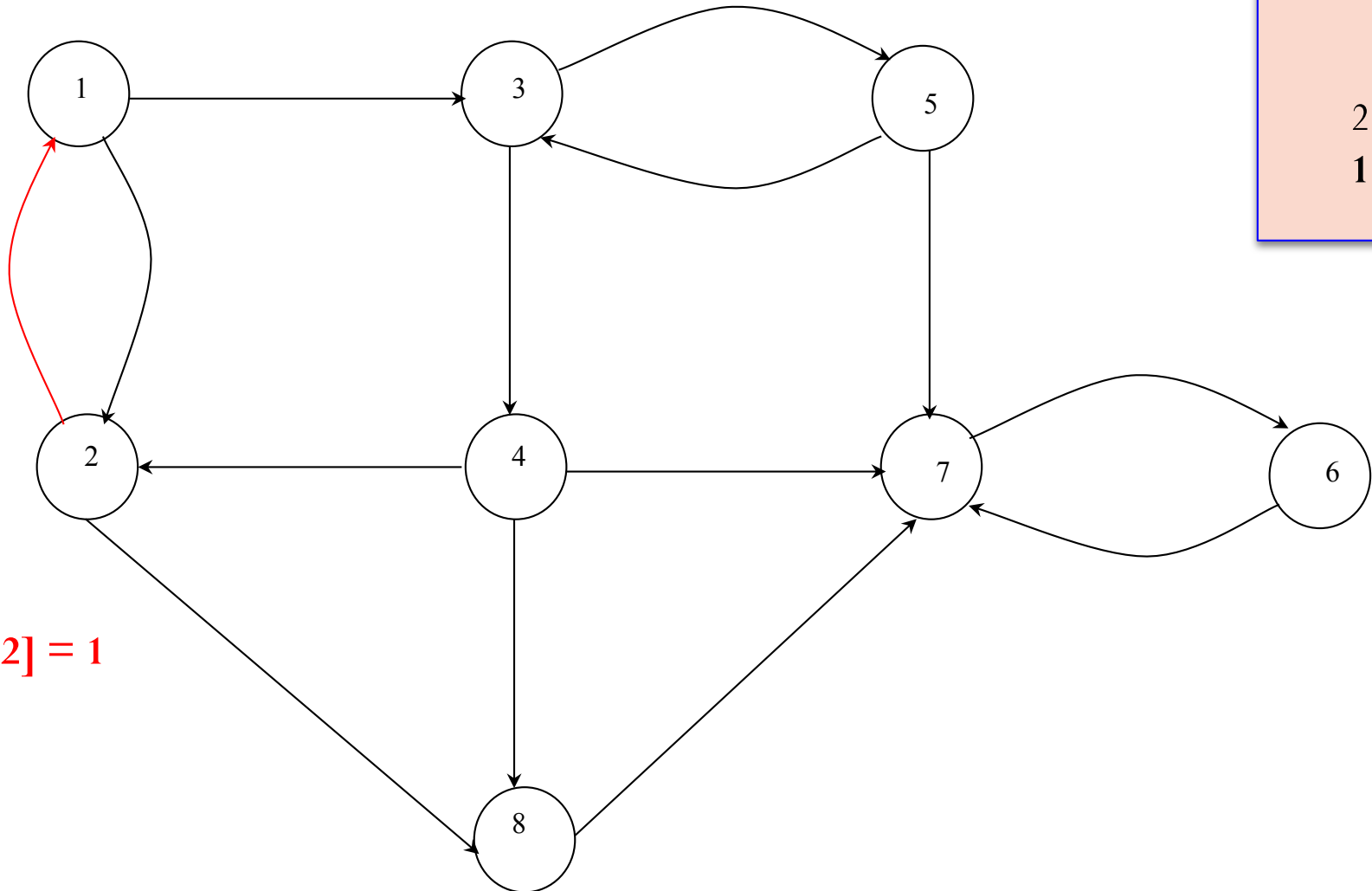


$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$



$\text{num}[2] = 2$
 $\text{min_num}[2] = 2$

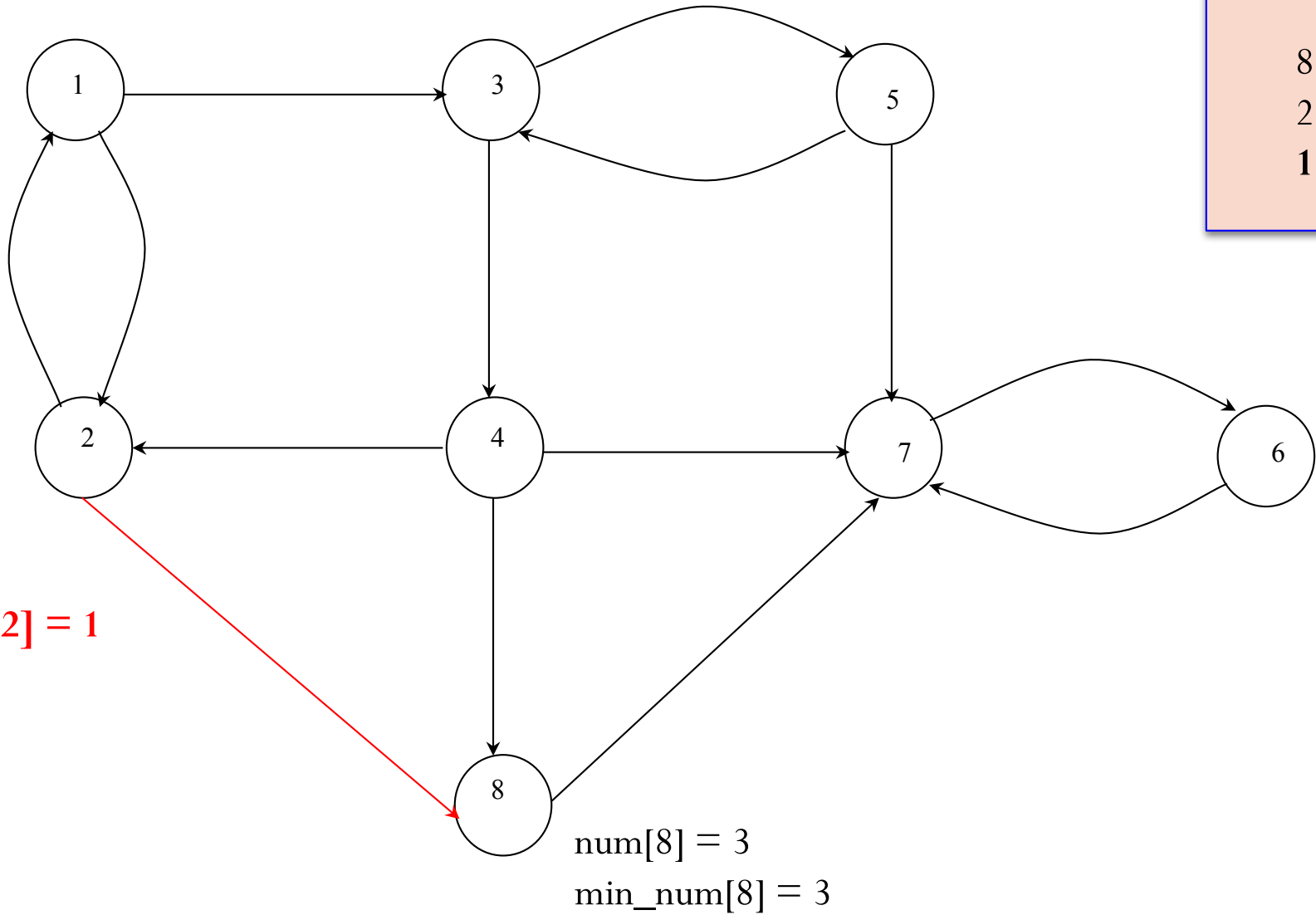
$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$



2
1

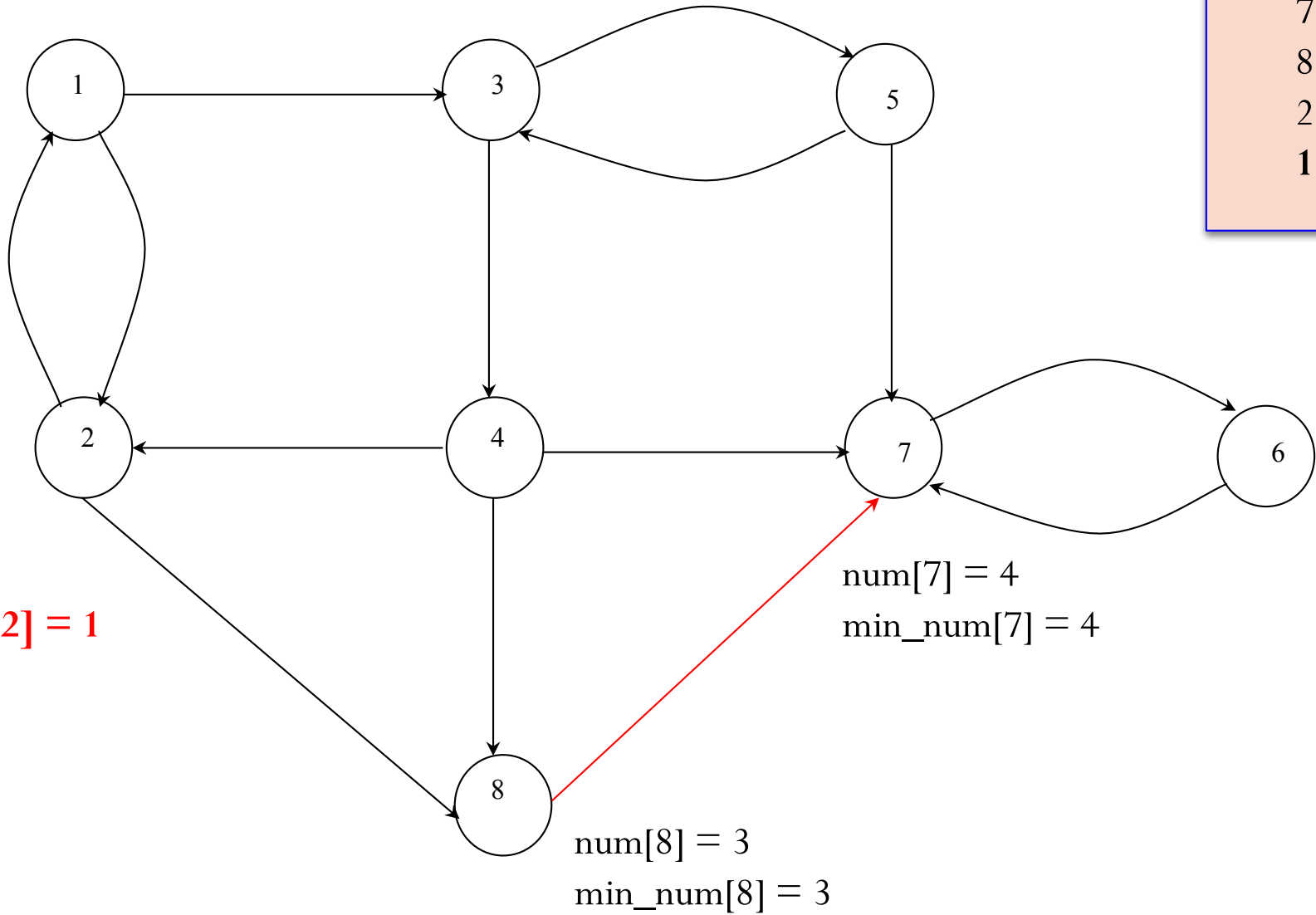
$\text{num}[2] = 2$
 $\text{min_num}[2] = 1$

$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$



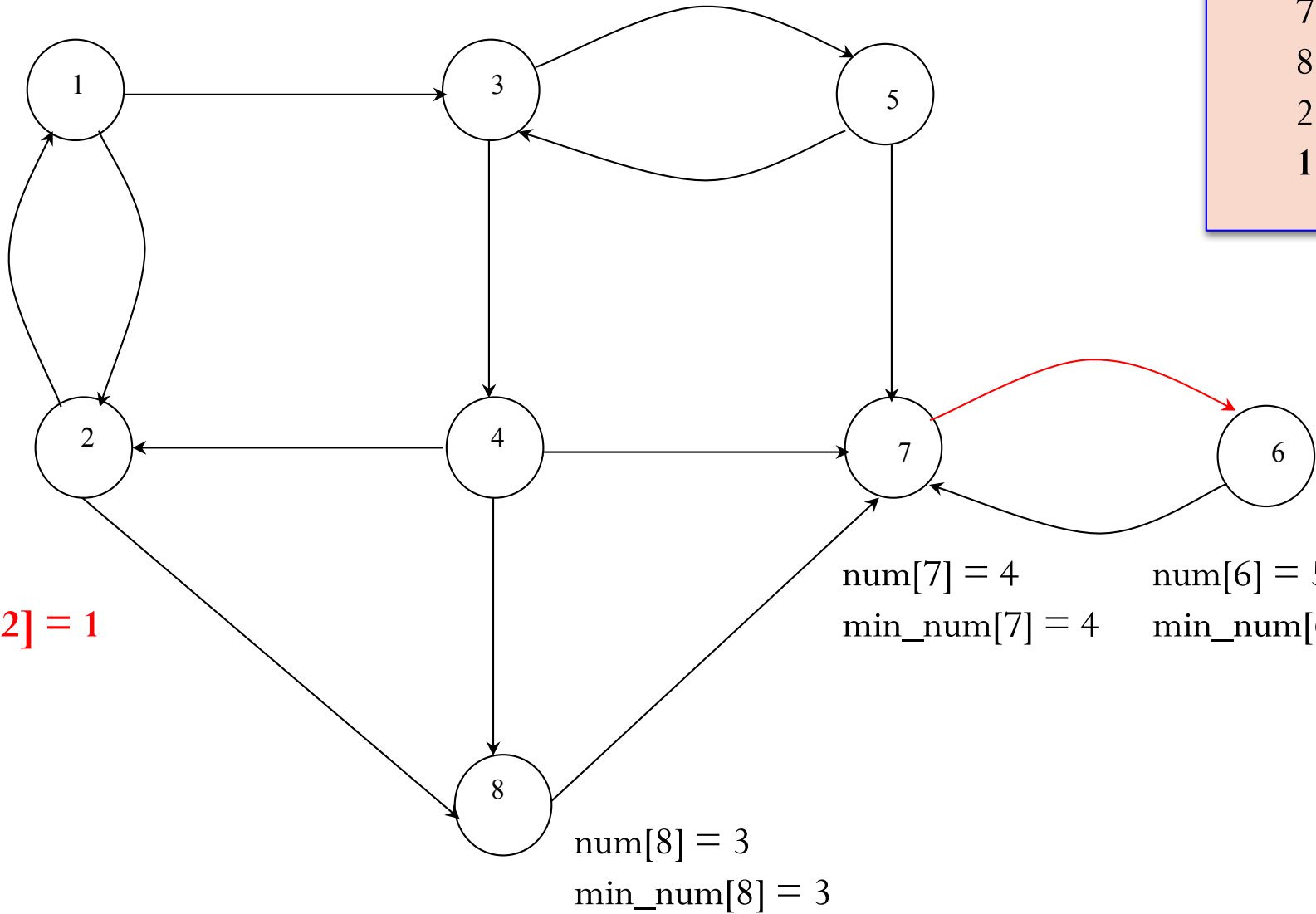
8
2
1

$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$



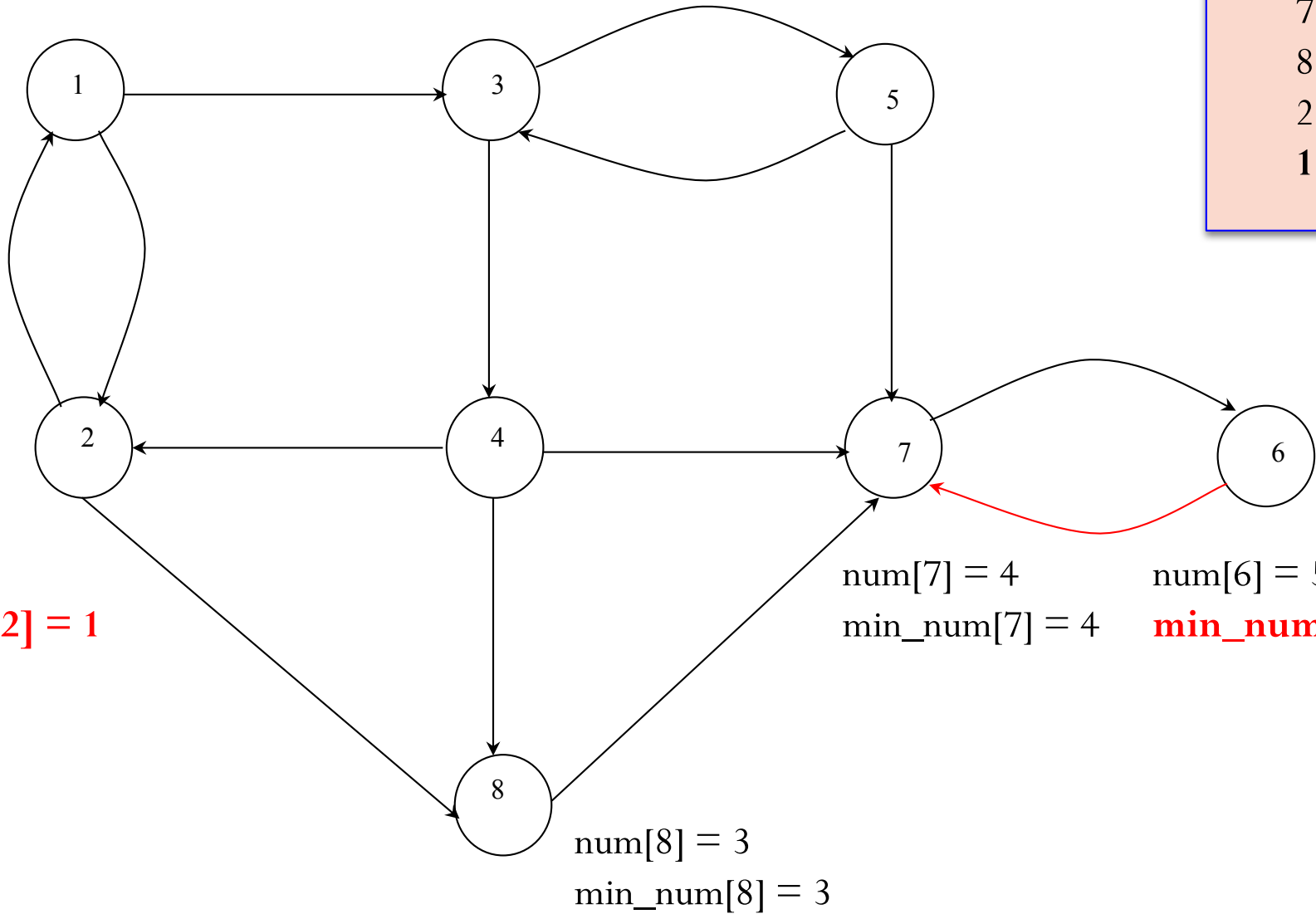
7
8
2
1

$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$



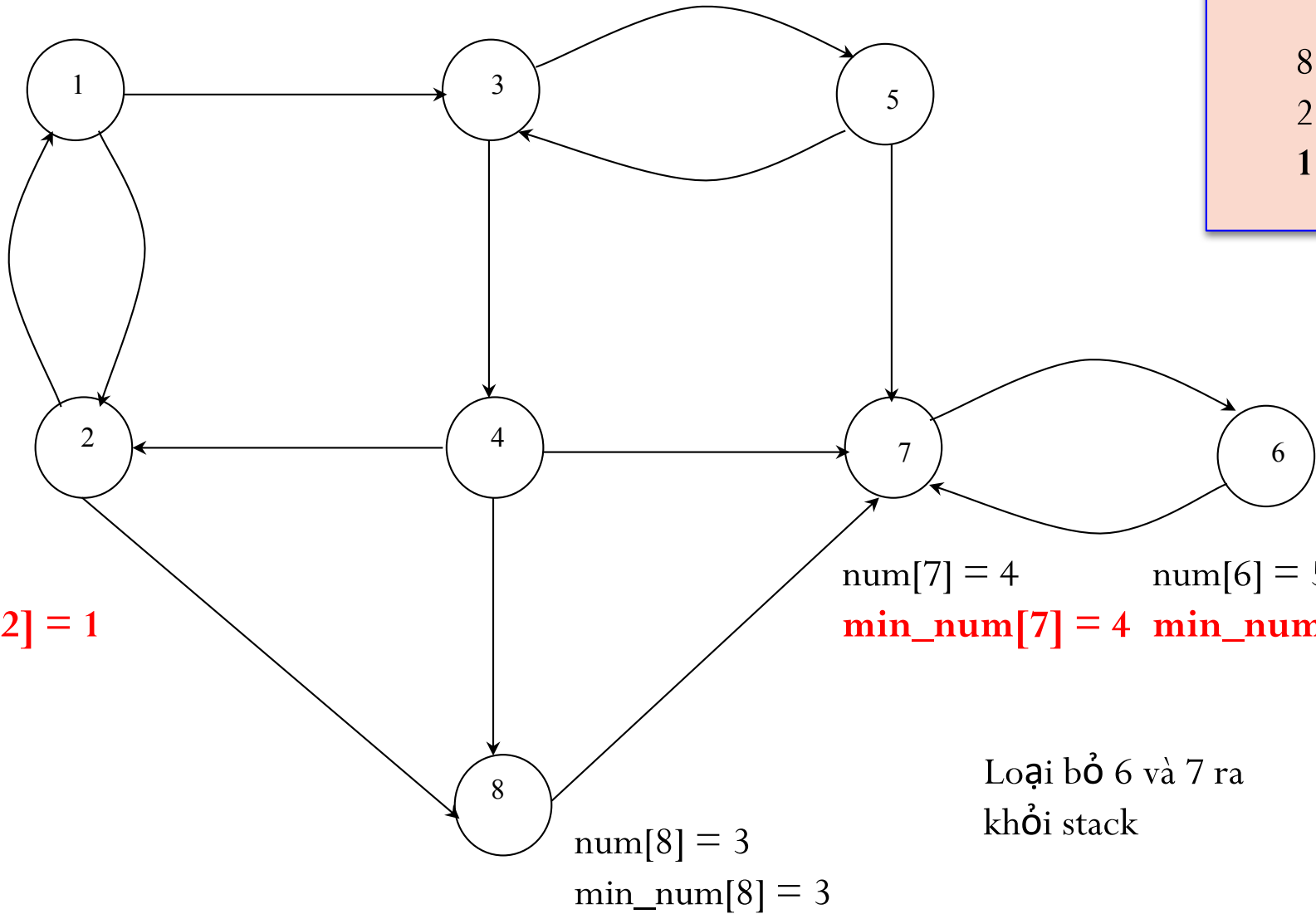
6
7
8
2
1

$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$



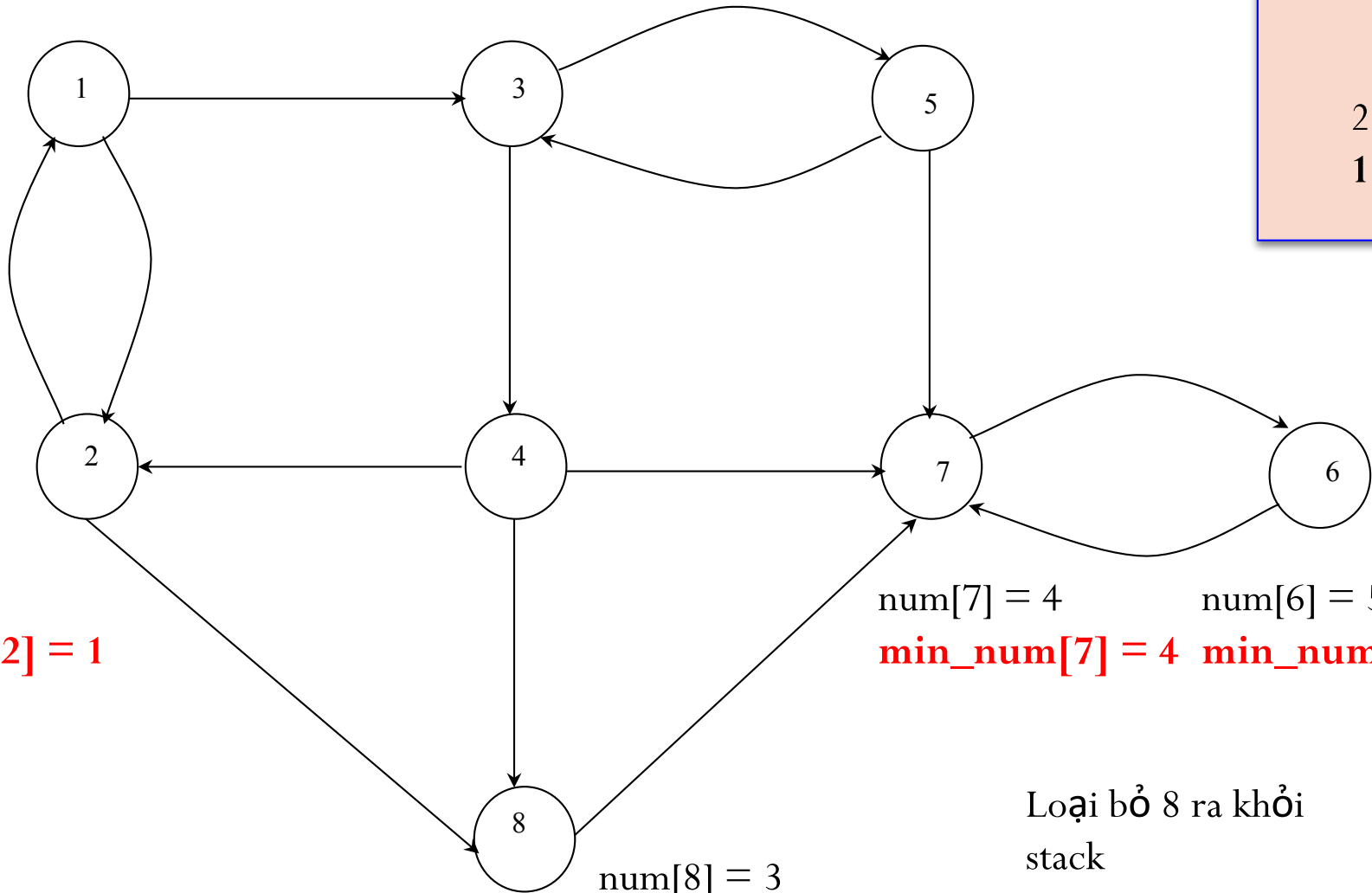
6
7
8
2
1

$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$



8
2
1

$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$



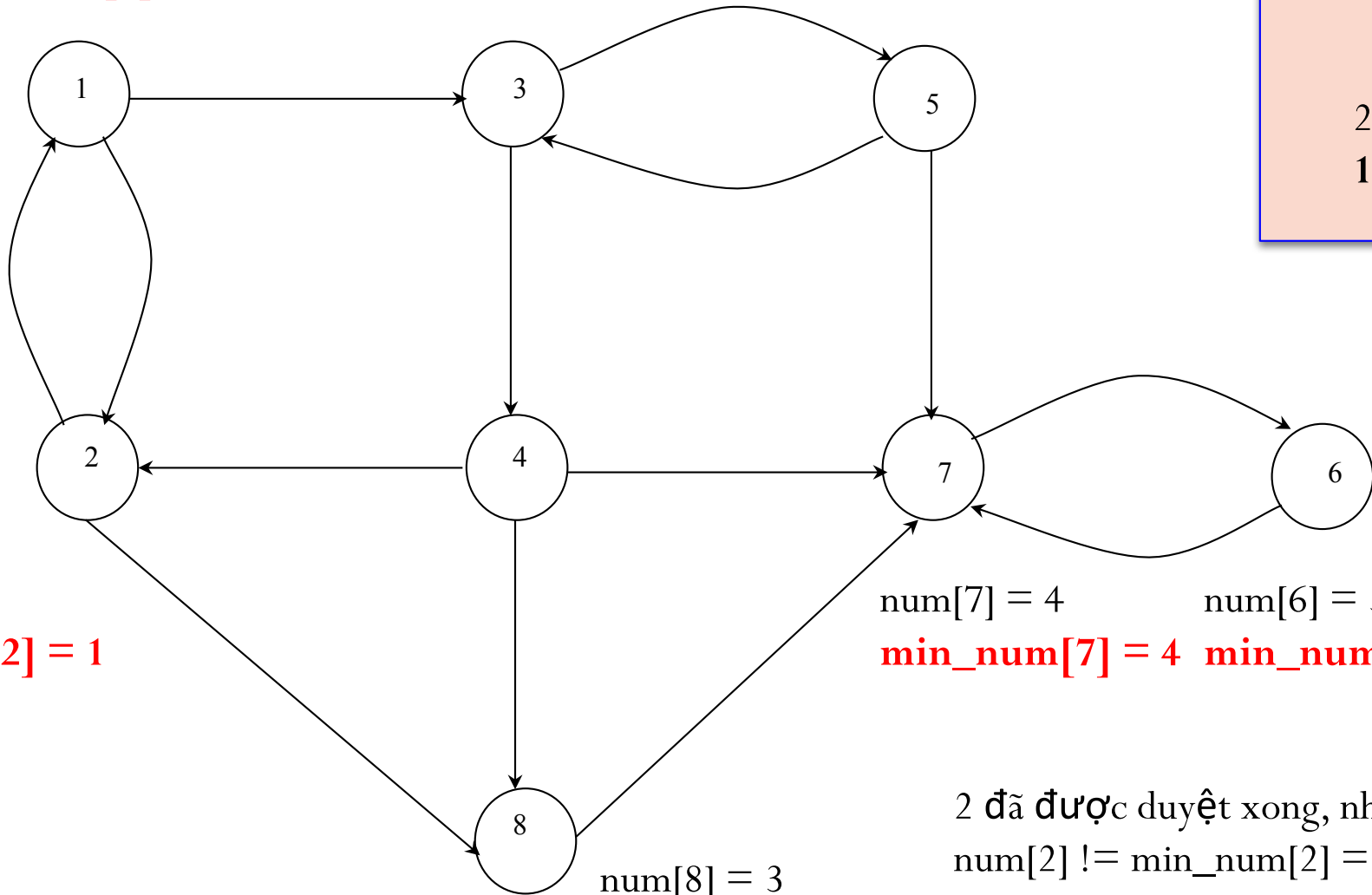
$\text{num}[2] = 2$
 $\text{min_num}[2] = 1$

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

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

Loại bỏ 8 ra khỏi
stack

$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$



$\text{num}[2] = 2$
 $\text{min_num}[2] = 1$

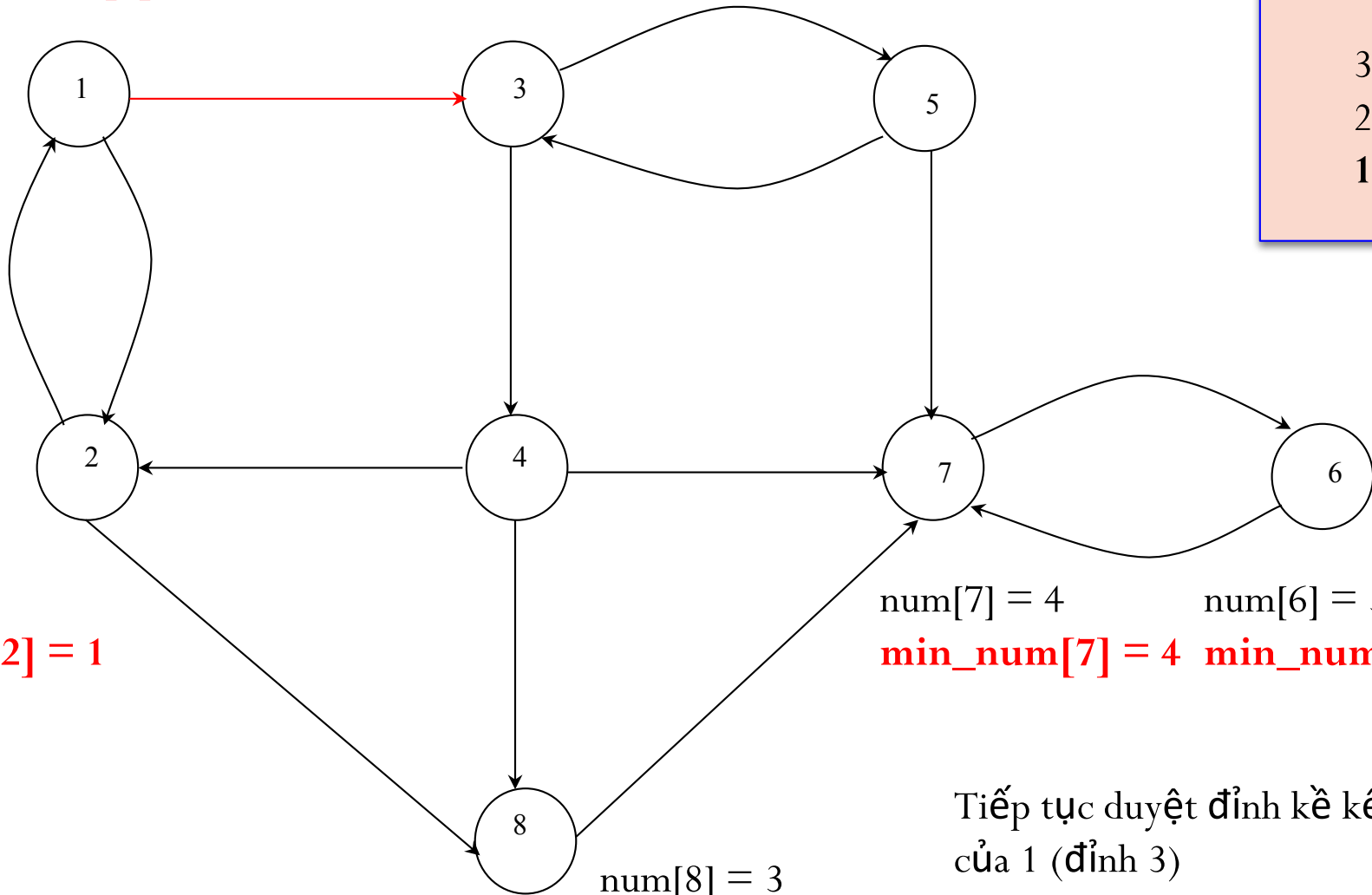
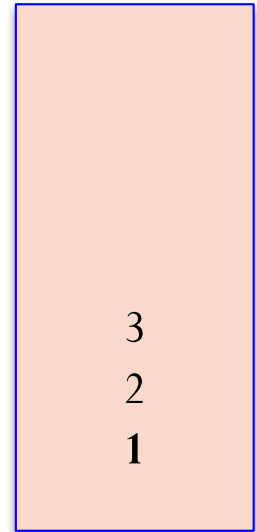
$\text{num}[7] = 4$ $\text{num}[6] = 5$
 $\text{min_num}[7] = 4$ $\text{min_num}[6] = 4$

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

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 kế tiếp
của 1 (đỉnh 3)

$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$

$\text{num}[3] = 6$
 $\text{min_num}[3] = 6$

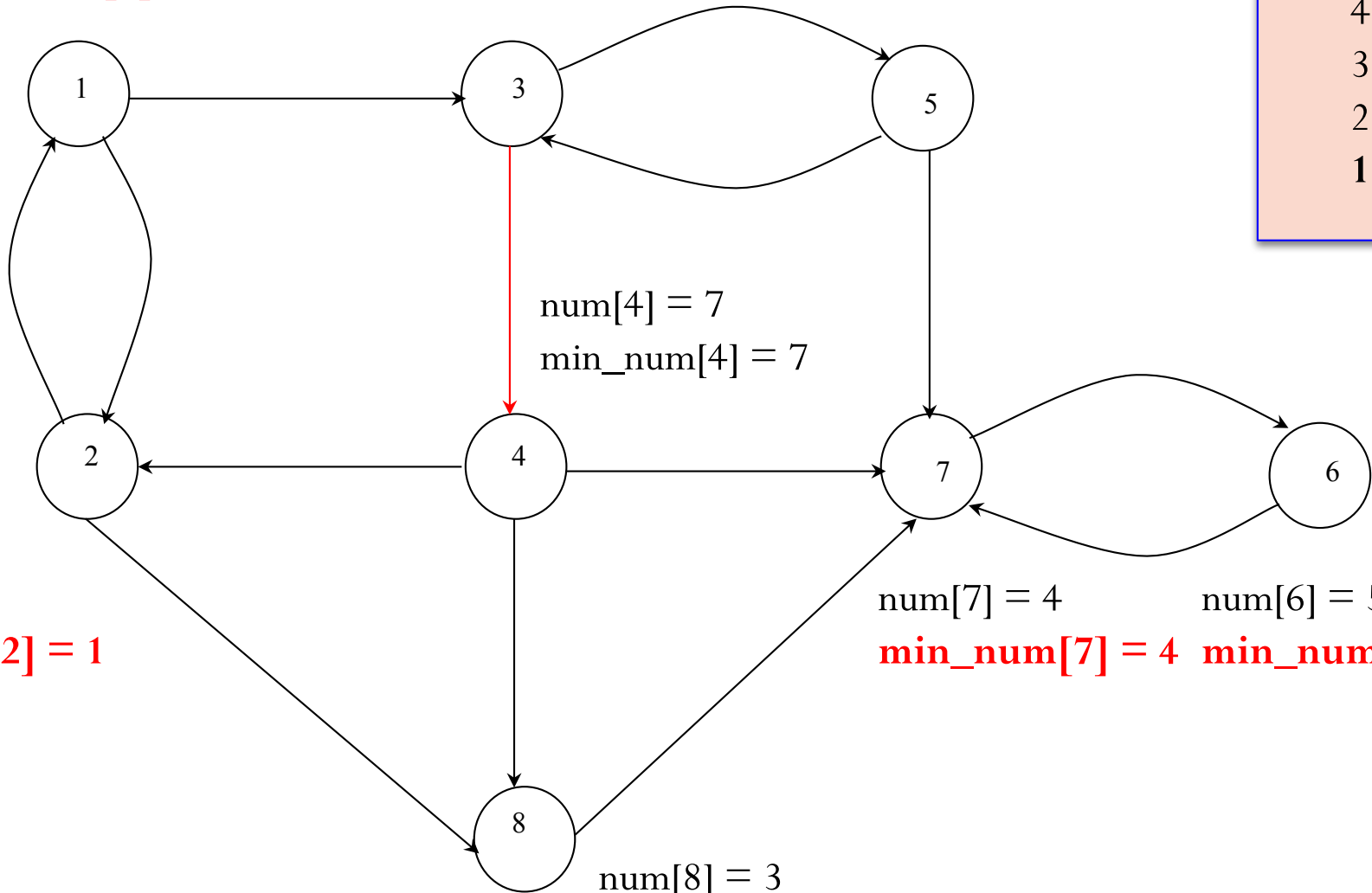
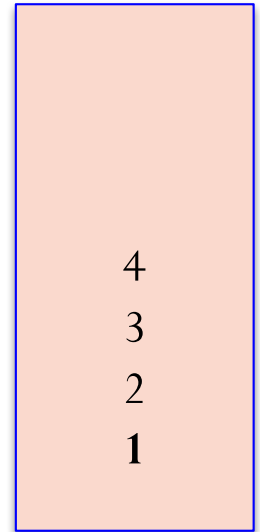
$\text{num}[4] = 7$
 $\text{min_num}[4] = 7$

$\text{num}[2] = 2$
 $\text{min_num}[2] = 1$

$\text{num}[7] = 4$
 $\text{min_num}[7] = 4$

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

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



$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$

$\text{num}[3] = 6$
 $\text{min_num}[3] = 6$

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

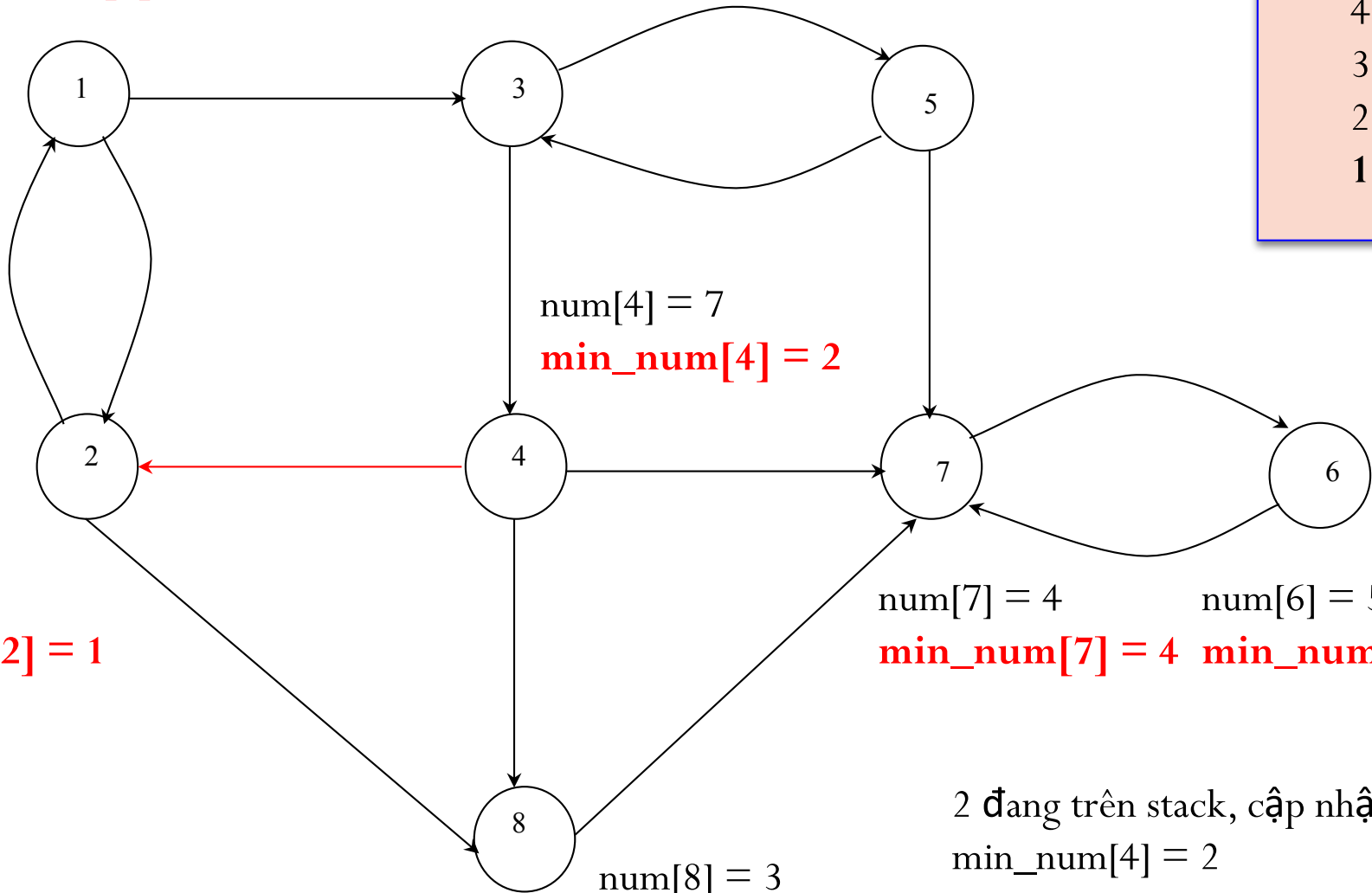
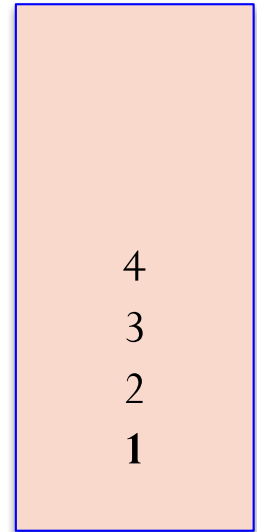
$\text{num}[2] = 2$
 $\text{min_num}[2] = 1$

$\text{num}[7] = 4$
 $\text{min_num}[7] = 4$

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

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

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



$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$

$\text{num}[3] = 6$
 $\text{min_num}[3] = 6$

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

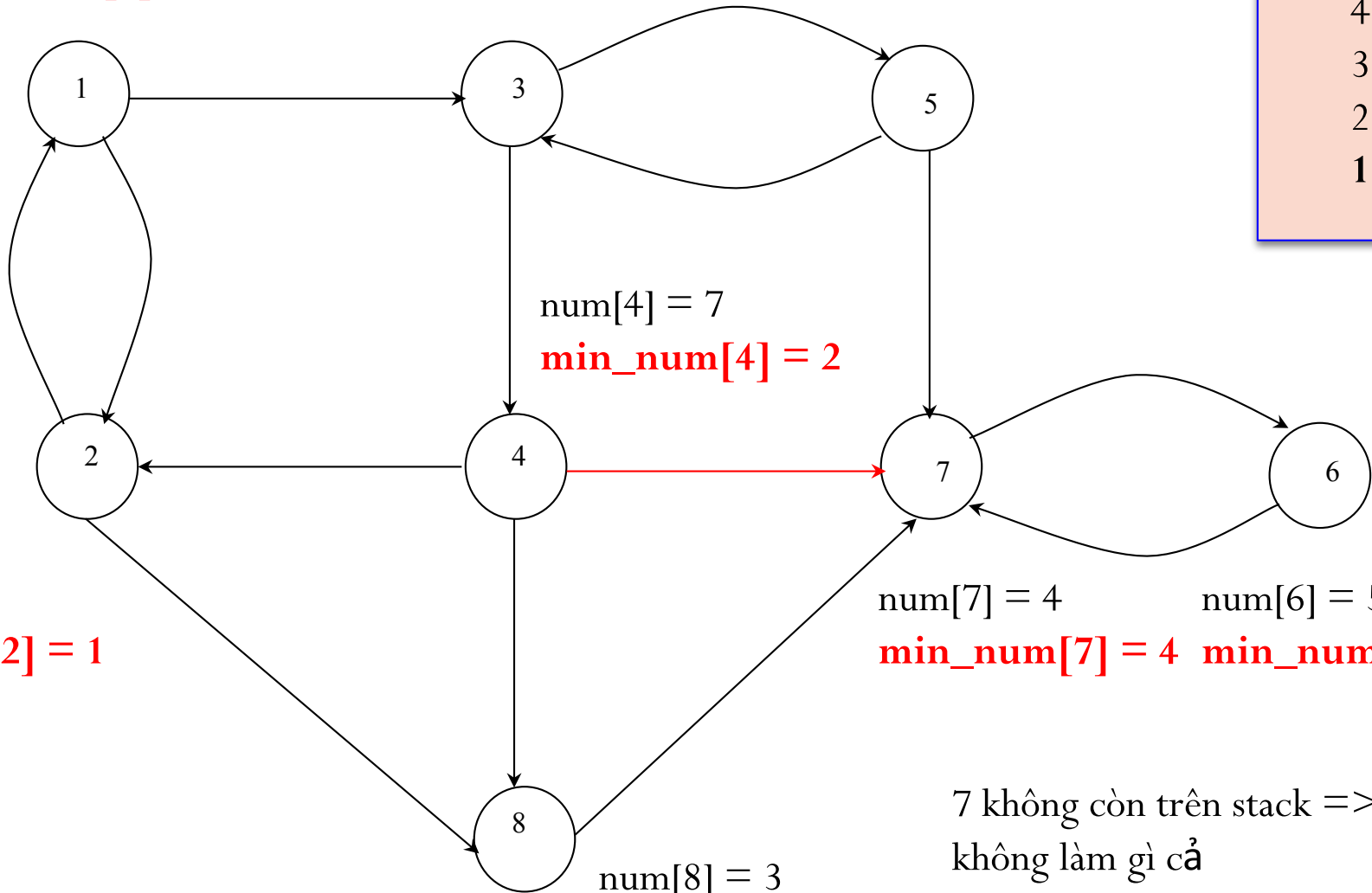
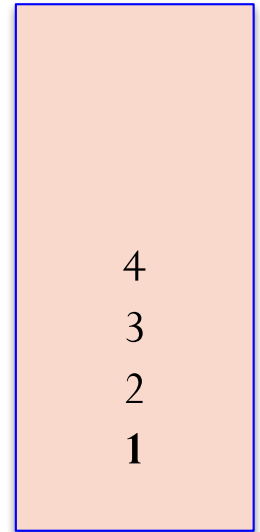
$\text{num}[2] = 2$
 $\text{min_num}[2] = 1$

$\text{num}[7] = 4$
 $\text{min_num}[7] = 4$

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

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

7 không còn trên stack =>
không làm gì cả



$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$

$\text{num}[3] = 6$
 $\text{min_num}[3] = 6$

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

$\text{num}[2] = 2$
 $\text{min_num}[2] = 1$

$\text{num}[7] = 4$
 $\text{min_num}[7] = 4$

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

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

8 không còn trên stack =>
không làm gì cả

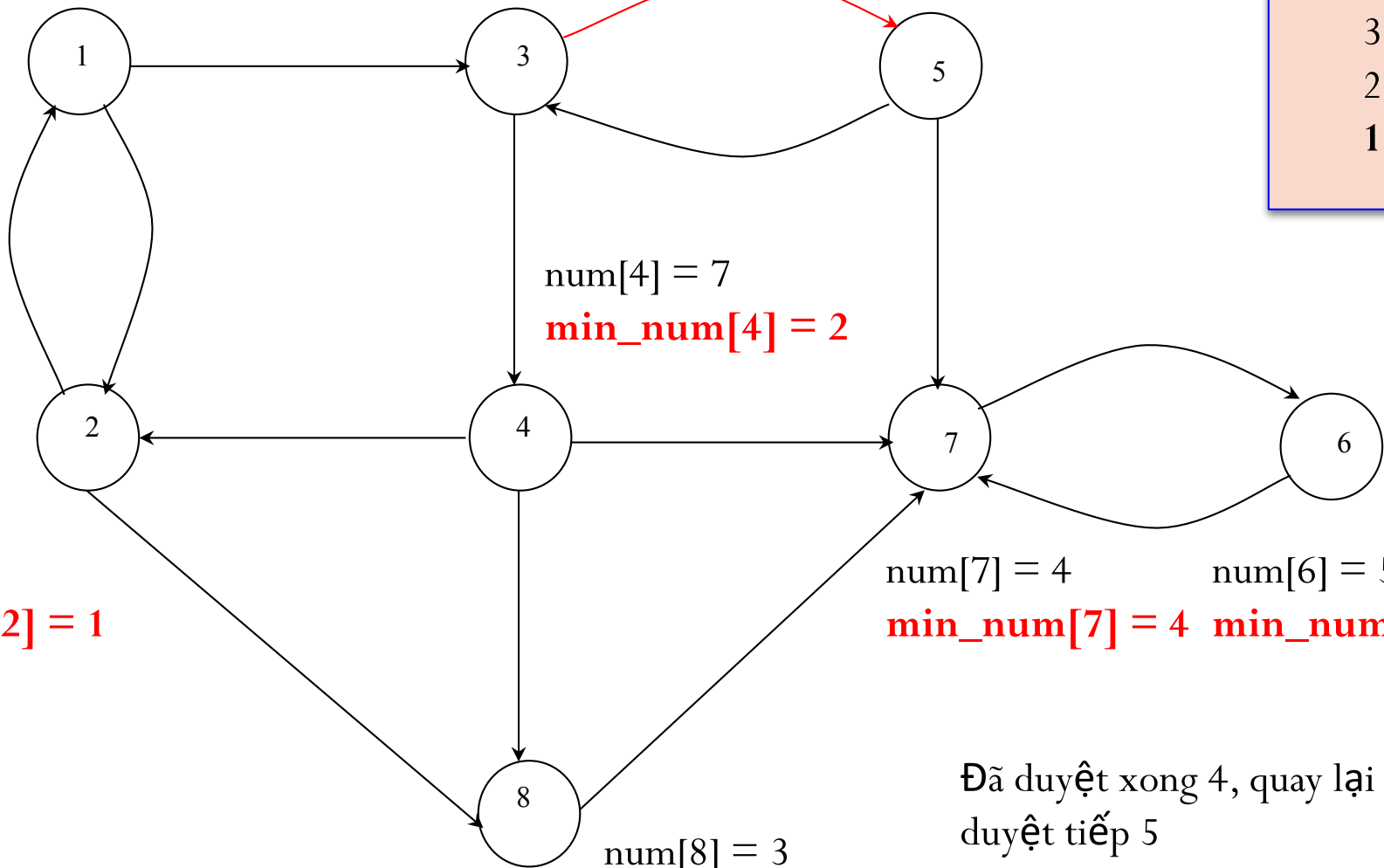
4
3
2
1

$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$

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

$\text{num}[5] = 8$
 $\text{min_num}[5] = 8$

5
4
3
2
1



$\text{num}[7] = 4$
 $\text{min_num}[7] = 4$

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

Đã duyệt xong 4, quay lại
duyệt tiếp 5

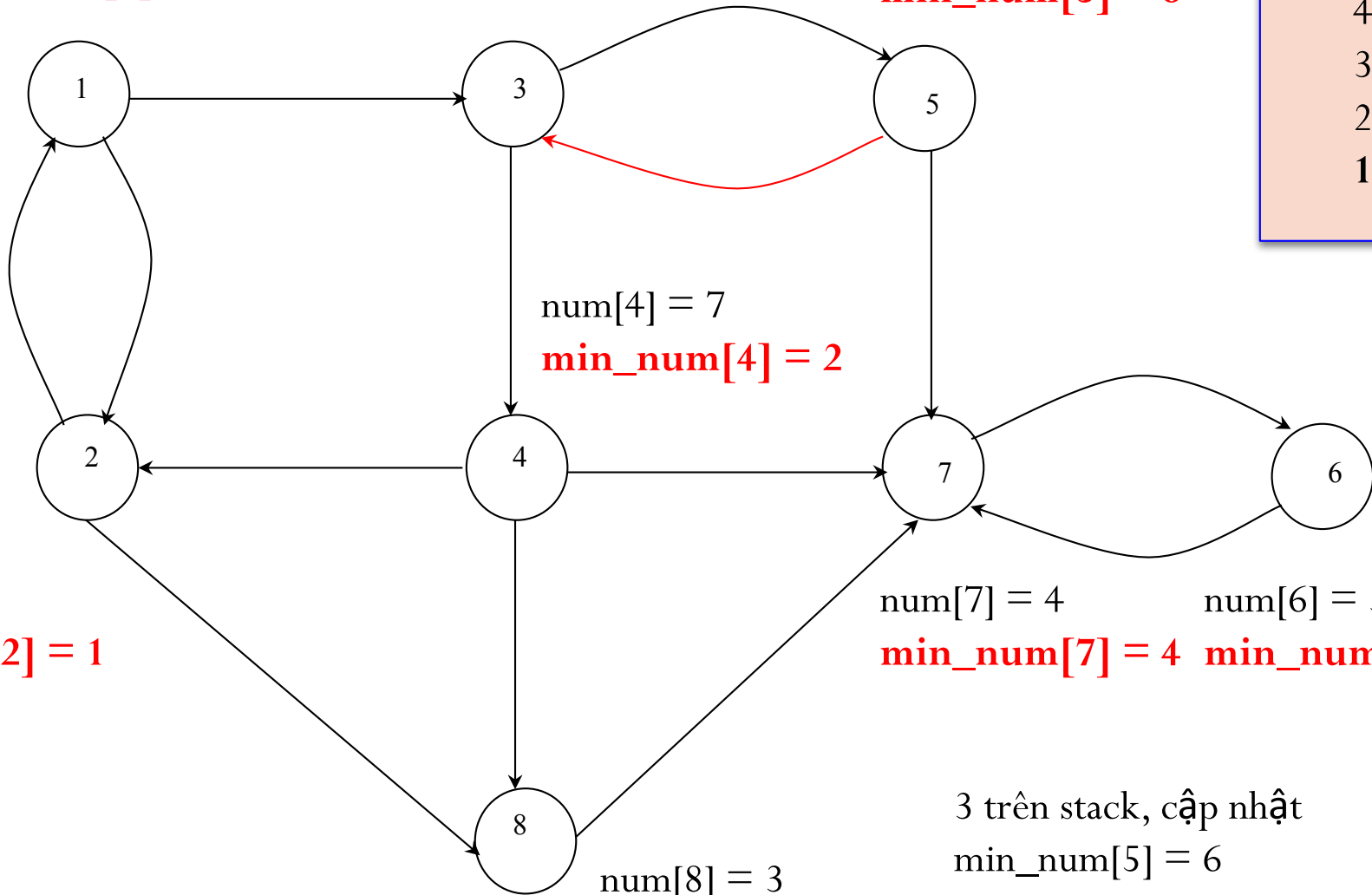
$\text{num}[8] = 3$
 $\text{min_num}[8] = 3$

$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$

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

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

5
4
3
2
1



$\text{num}[7] = 4$
 $\text{min_num}[7] = 4$

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

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

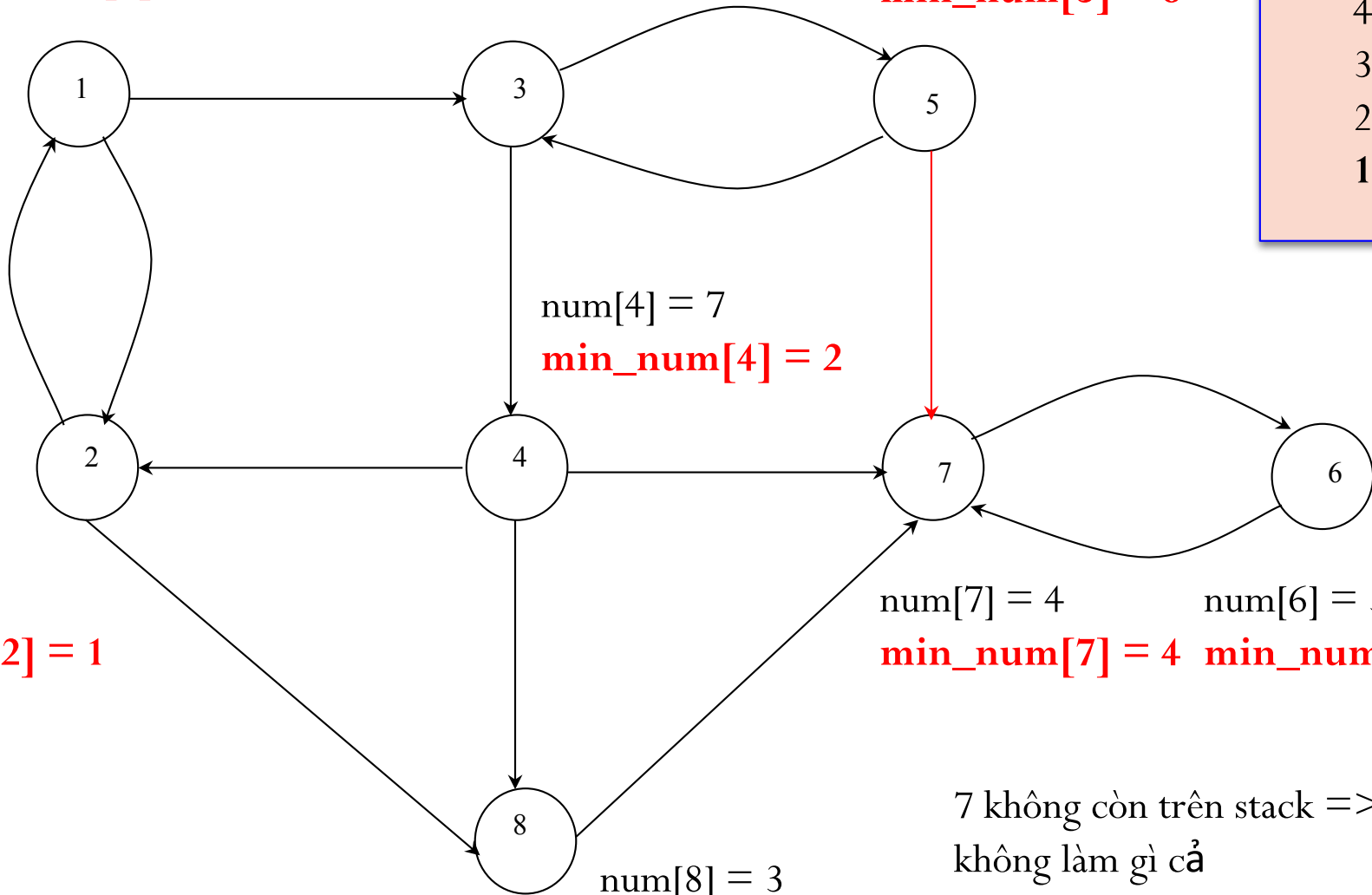
$\text{num}[8] = 3$
 $\text{min_num}[8] = 3$

$\text{num}[1] = 1$
 $\text{min_num}[1] = 1$

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

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

5
4
3
2
1



7 không còn trên stack =>
không làm gì cả

num[1] = 1
min_num[1] = 1

num[3] = 6
min_num[3] = 2

num[5] = 8
min_num[5] = 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

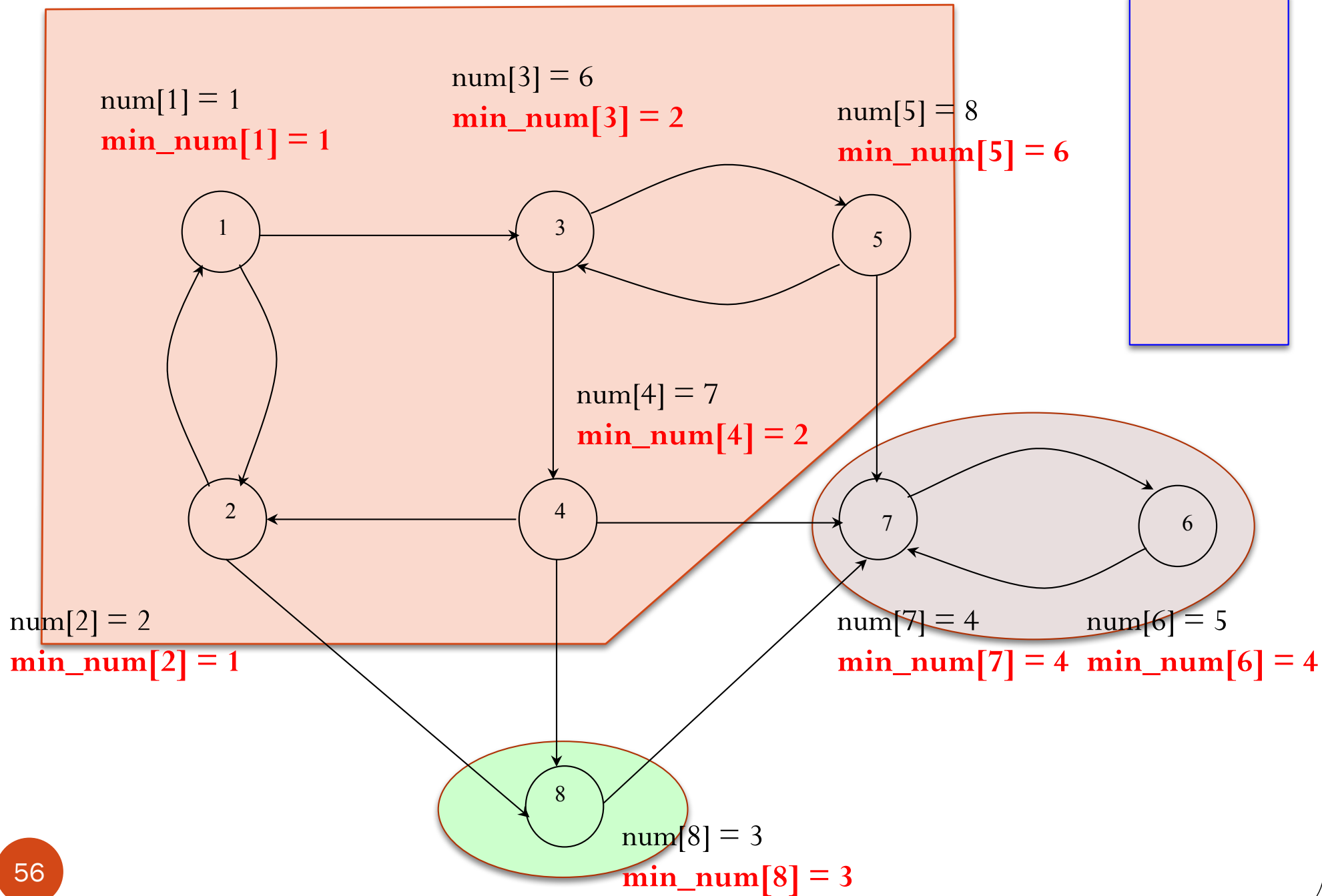
num[8] = 3
min_num[8] = 3

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

5
4
3
2
1

Lưu ý:

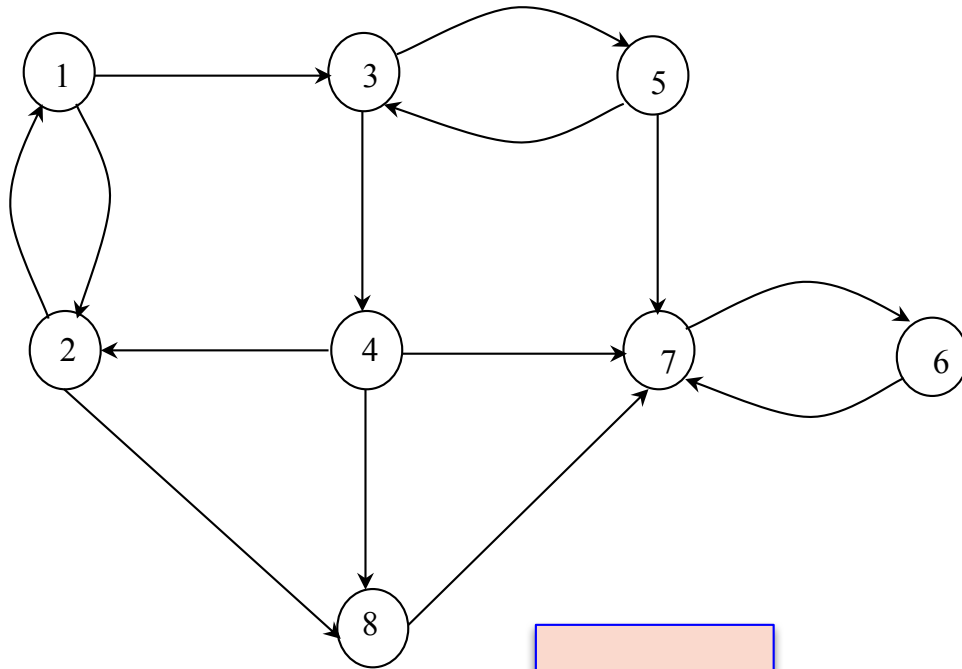
- Thứ tự các đỉnh kề của 1 đỉnh được sắp xếp từ bé đến lớn.
- Khi duyệt xong 1 đỉnh v , quay về đỉnh cha u (đỉnh trước), cập nhật lại $\text{min_num}[u]$ (so với **$\text{min_num}[v]$**)
- Khi xét 1 đỉnh kề v của u mà v đang có mặt trong stack \Rightarrow cập nhật lại $\text{min_num}[u]$ (so với **$\text{num}[v]$**).



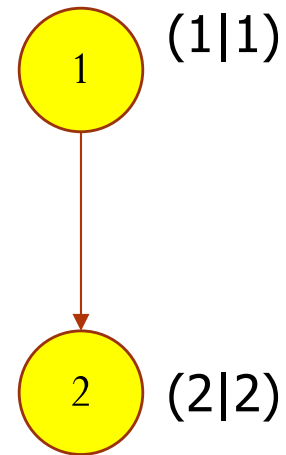
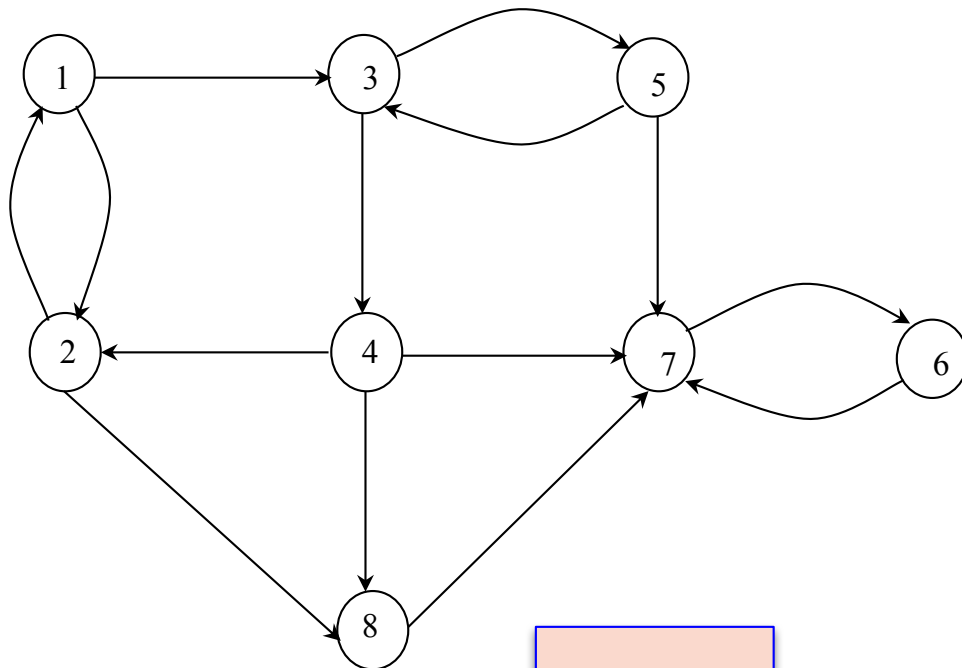
Cây duyệt theo chiều sâu

- Quá trình duyệt cây kết hợp với đánh số (num, min_num) tạo ra cây “duyet đồ thị”
- Phần tiếp theo minh họa quá trình chạy giải thuật và cây “duyet đồ thị tương ứng”
- Cung liên nét (cung thuận)
- Cung không liên nét: cung quay lui (minh họa việc cập nhật min_num)

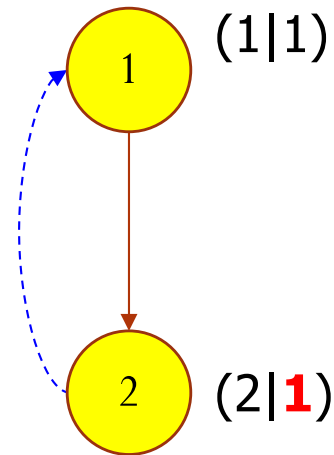
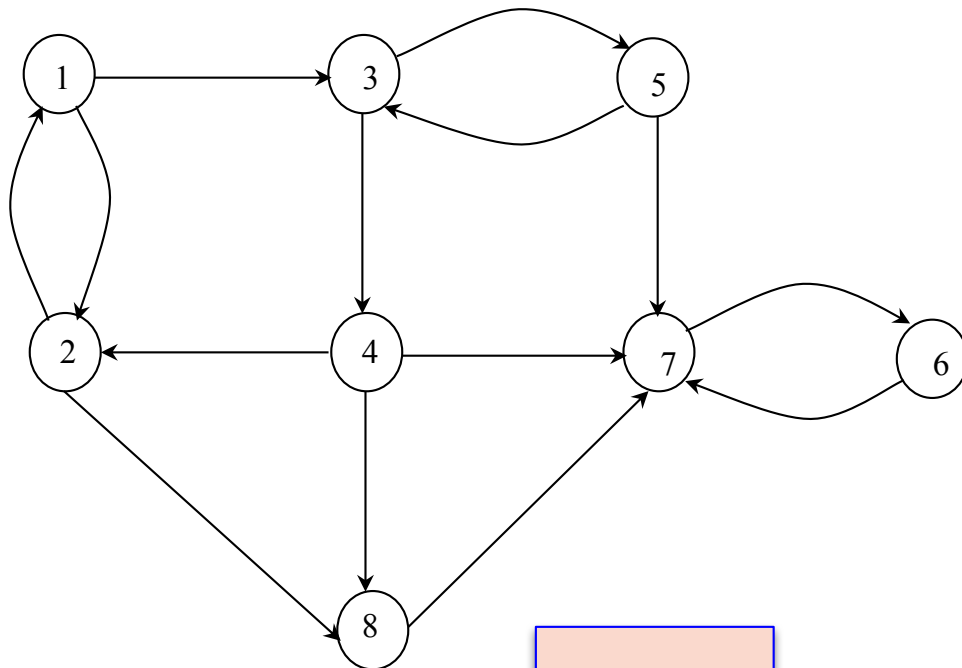
1 (1|1)



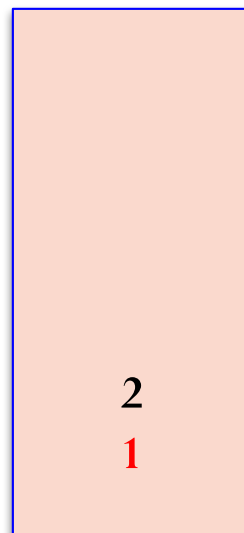
1

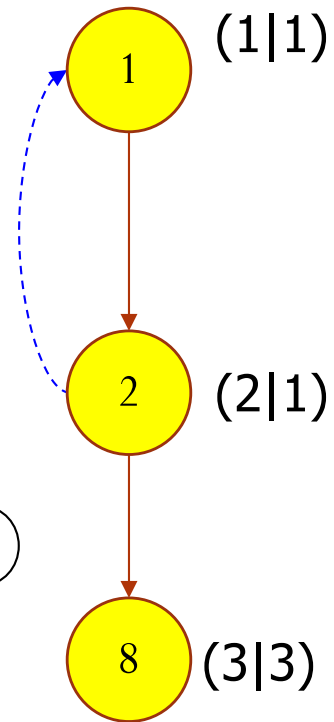
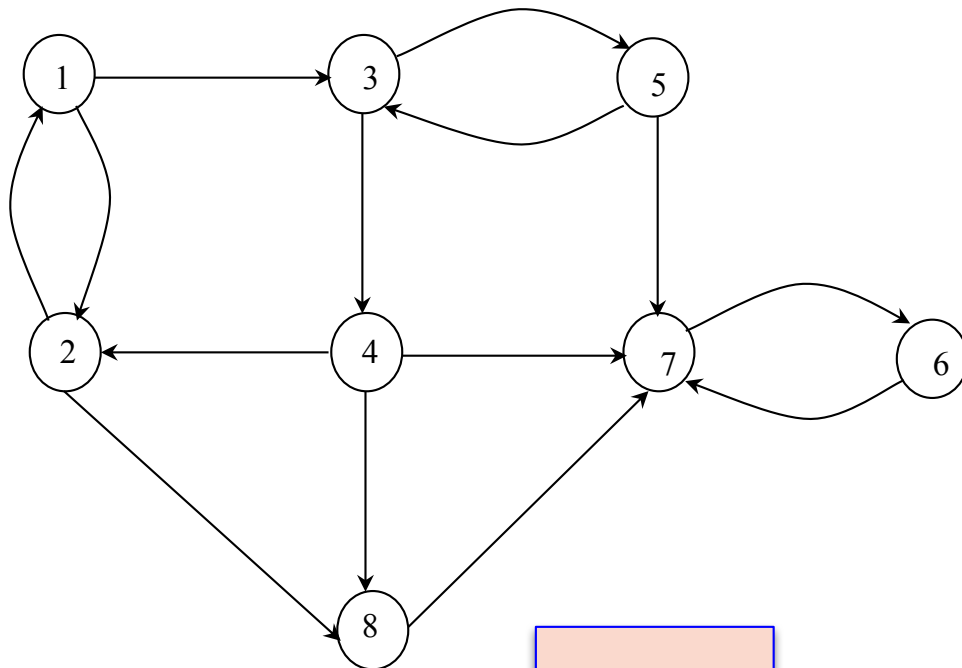


2
1

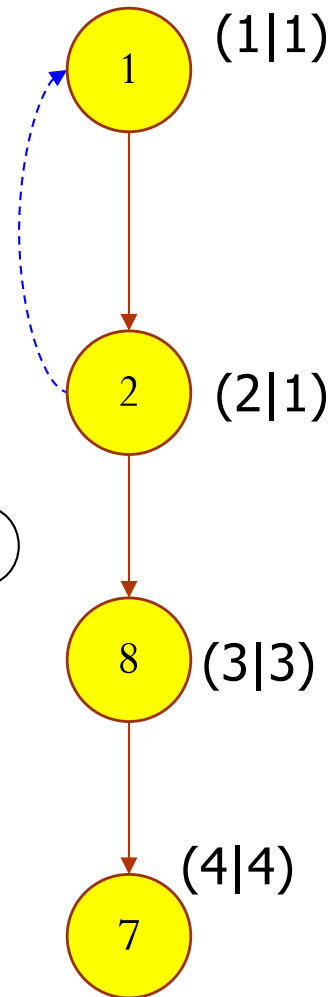
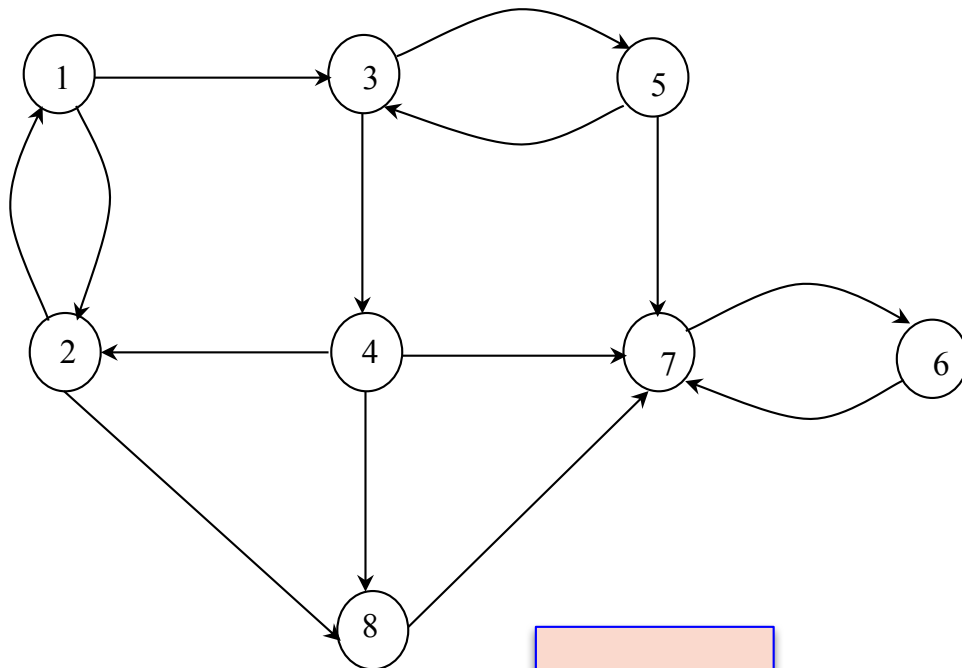


Cập nhật
 $\text{min_num}[2] = 1$

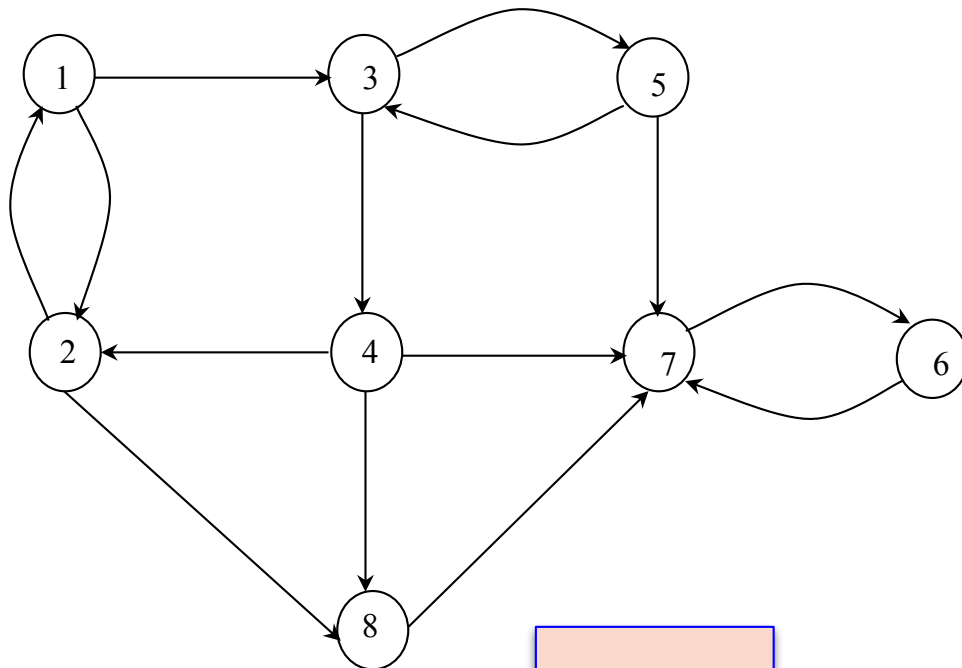




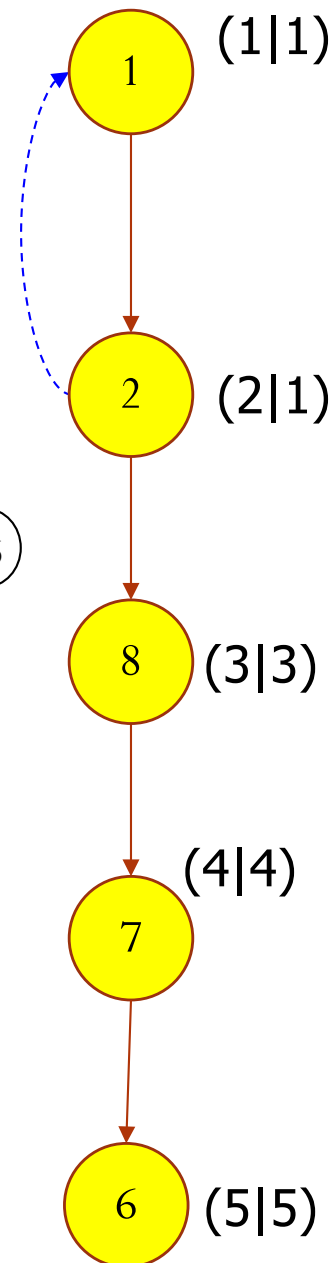
8
2
1

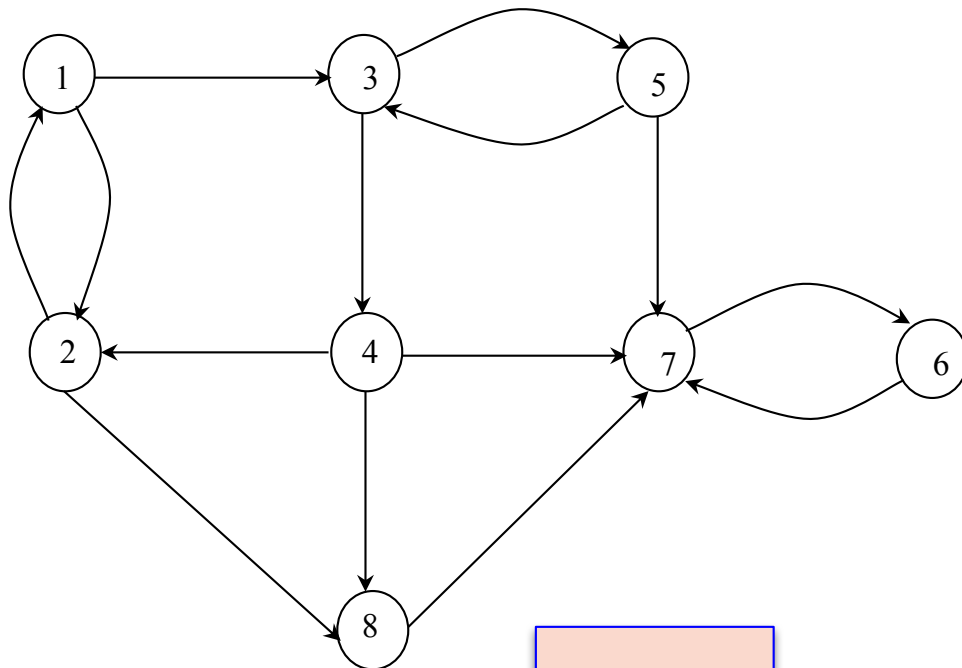


7
8
2
1



6
7
8
2
1

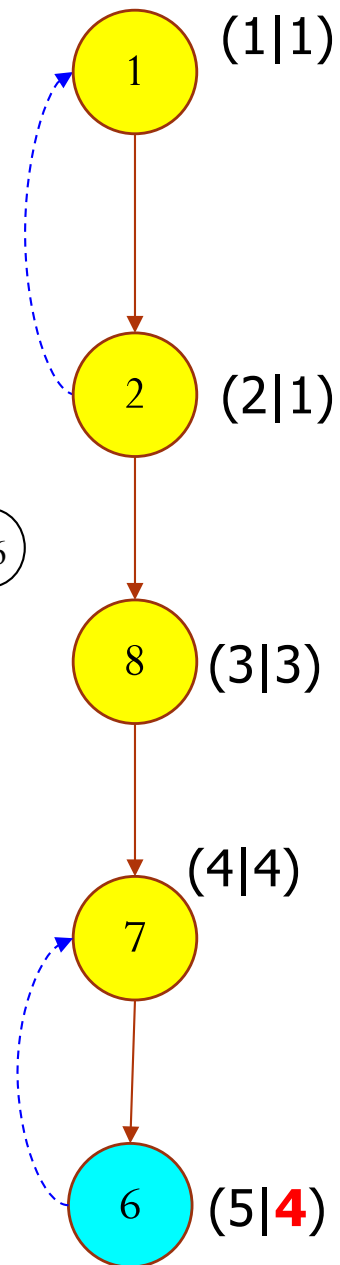


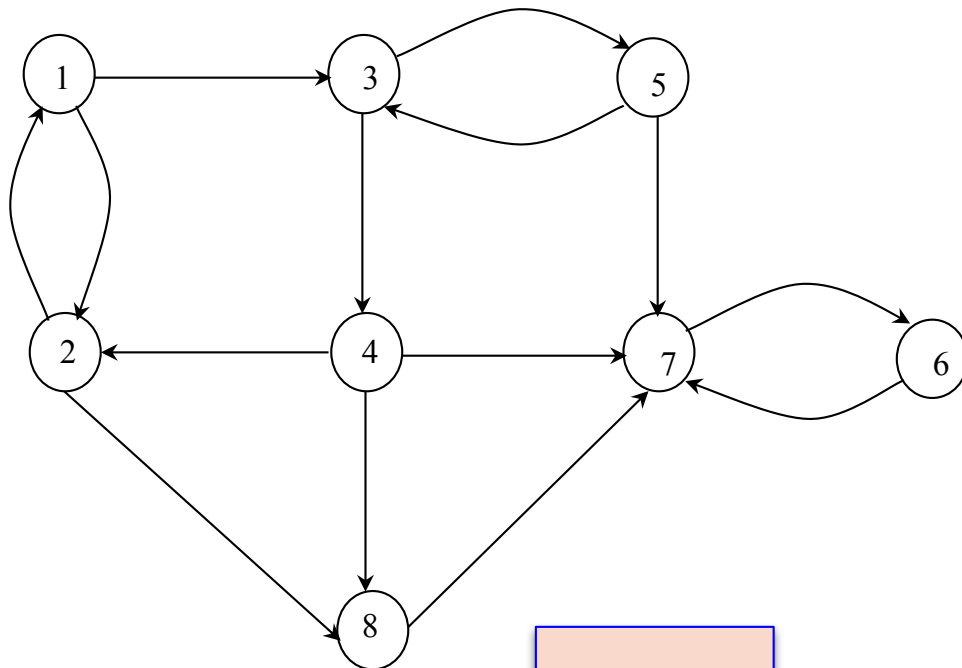


6 đã được duyệt
xong, nhưng 5 \neq 4

64

6
7
8
2
1

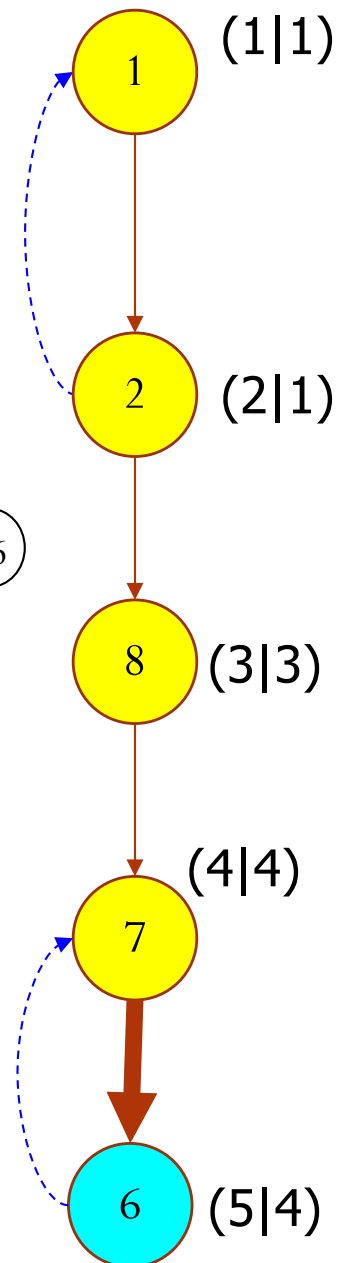




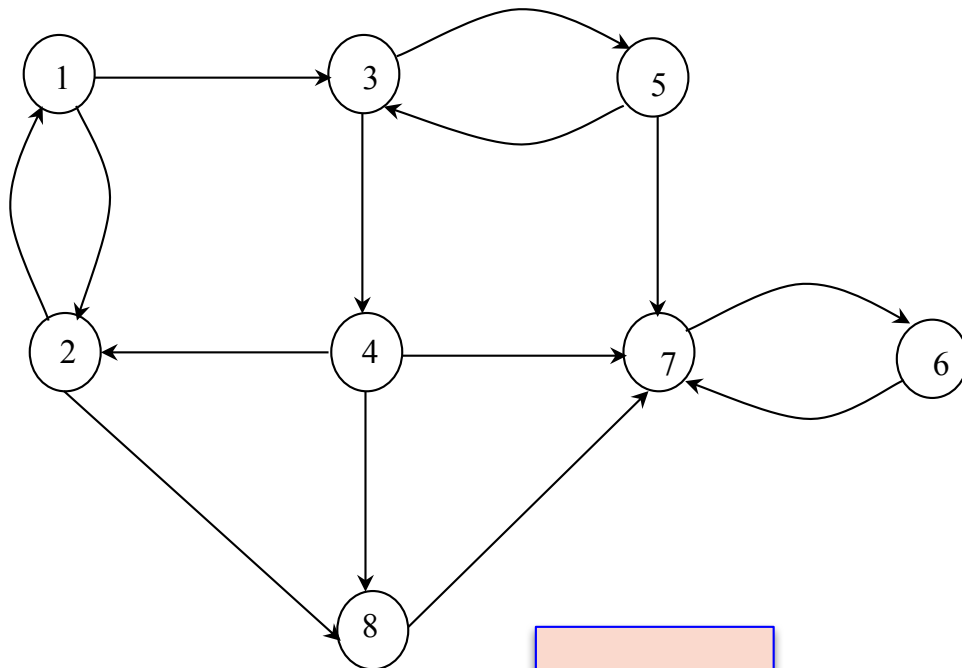
6 đã được duyệt
xong, nhưng $5 \neq 4$

65

6
7
8
2
1

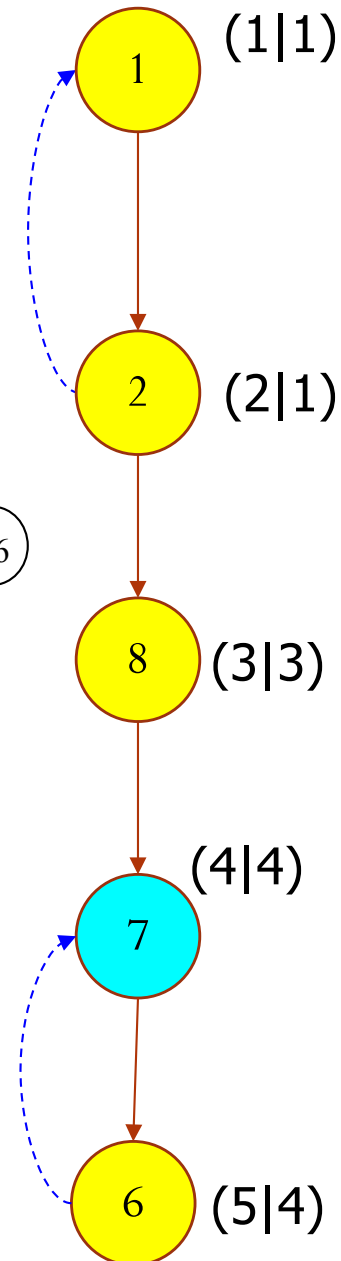


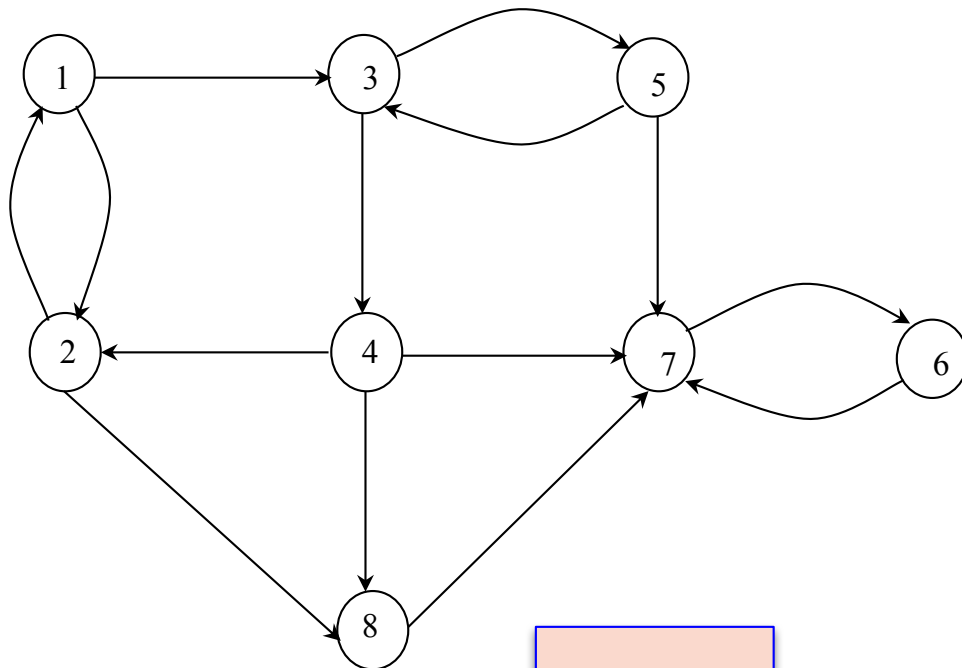
Kiểm tra cập nhật lại
 $\text{min_num}[7]$ (vì lúc này 7
gọi duyệt 6) => **Không
cần cập nhật vì $4 \leq 4$**



7 đã được duyệt
xong và có $4 == 4$

6
7
8
2
1

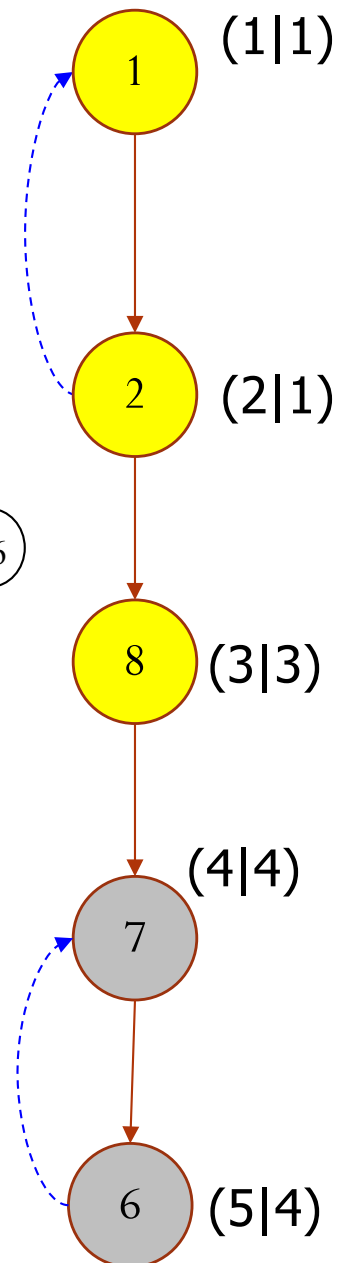


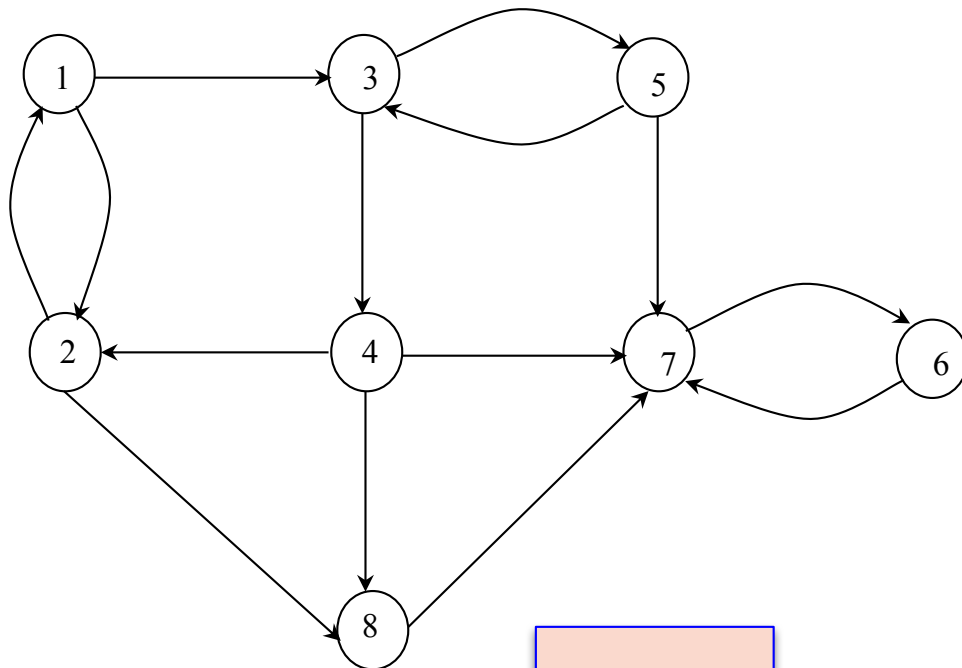


7 đã được duyệt
xong và có $4 == 4$
Loại bỏ 6, 7 ra
 \Rightarrow tìm được BPLT
(6, 7)

67

6
7
8
2
1

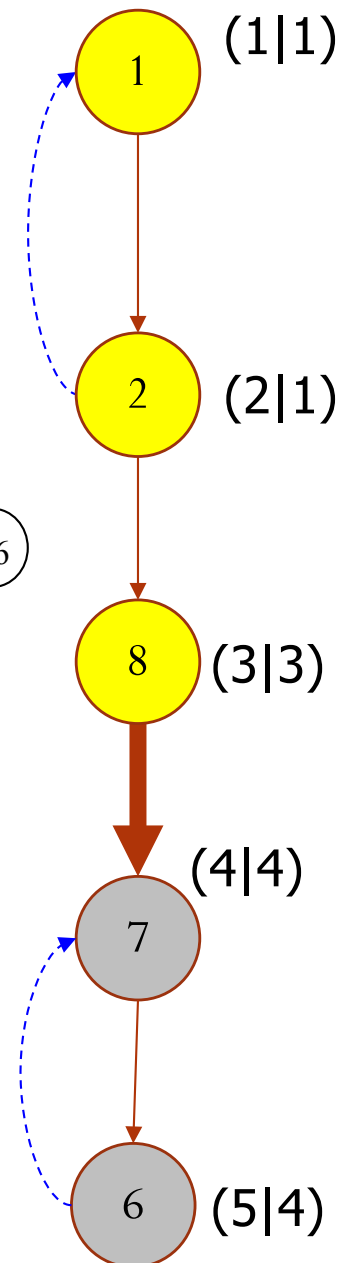




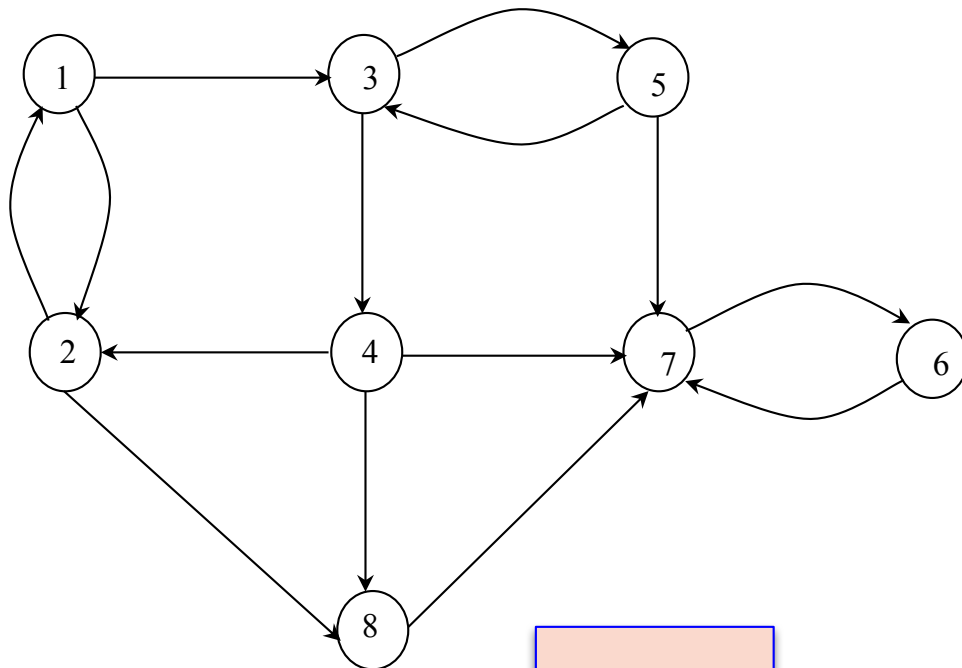
7 đã được duyệt
xong và có $4 == 4$
Loại bỏ 6, 7 ra
 \Rightarrow tìm được BPLT
(6, 7)

68

6
7
8
2
1

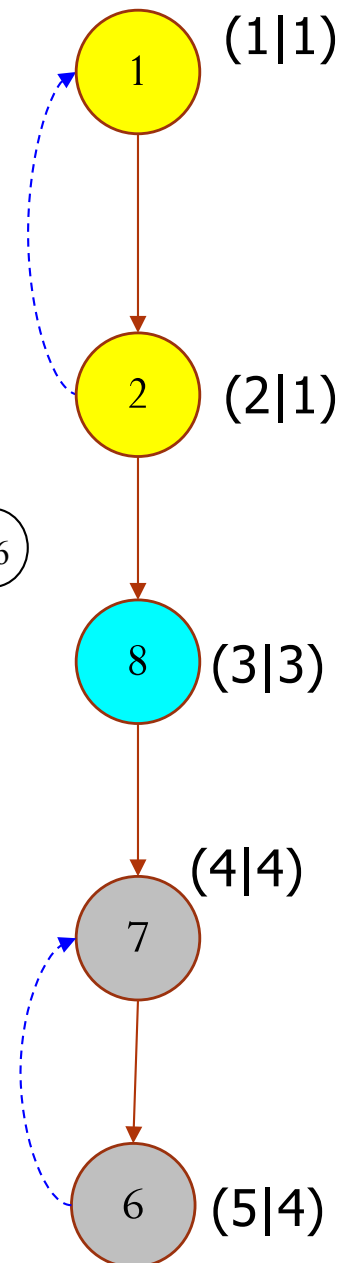


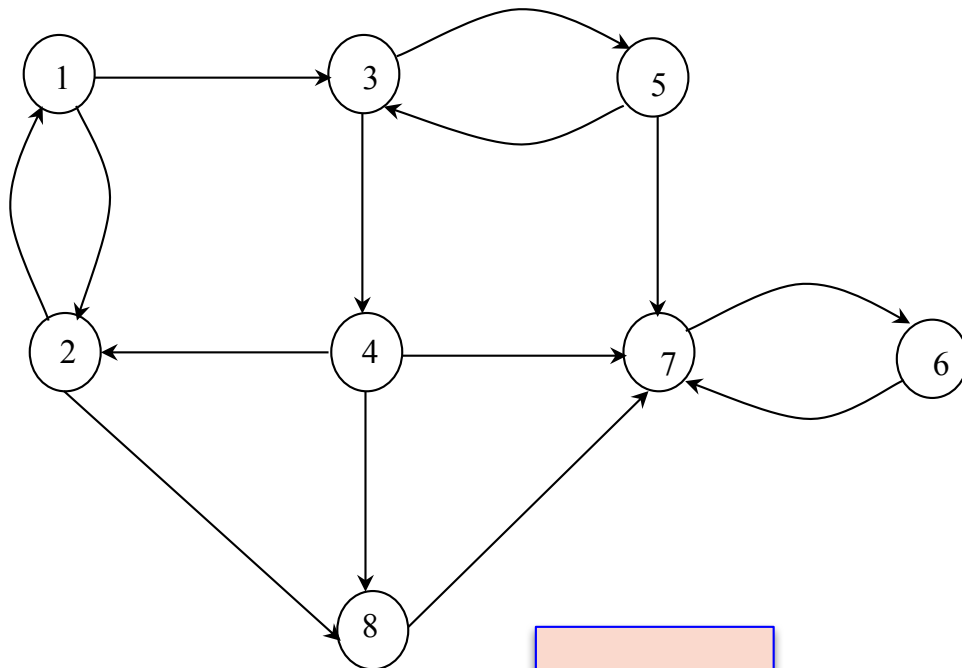
Kiểm tra cập nhật lại
 $\text{min_num}[8]$ (vì lúc này 8
gọi duyệt 7) \Rightarrow **Không
cần cập nhật vì $3 \leq 4$**



8 đã được duyệt
xong và có $3 == 3$

8
2
1

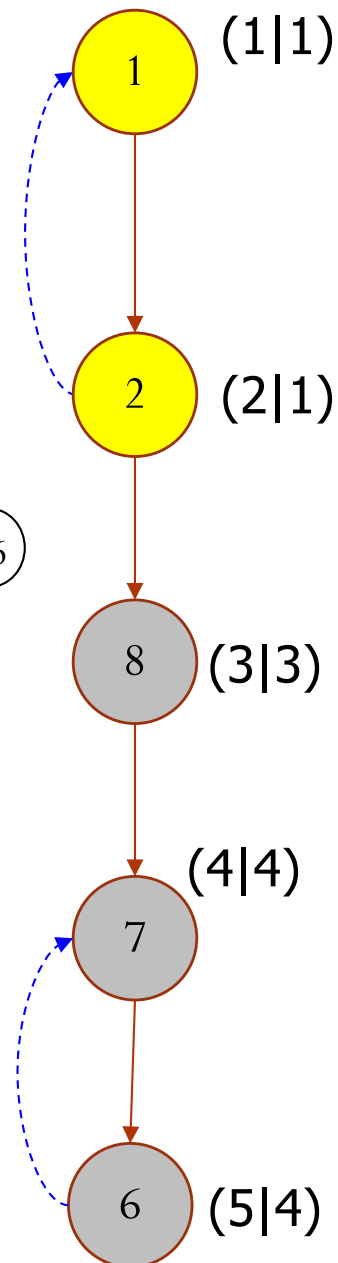


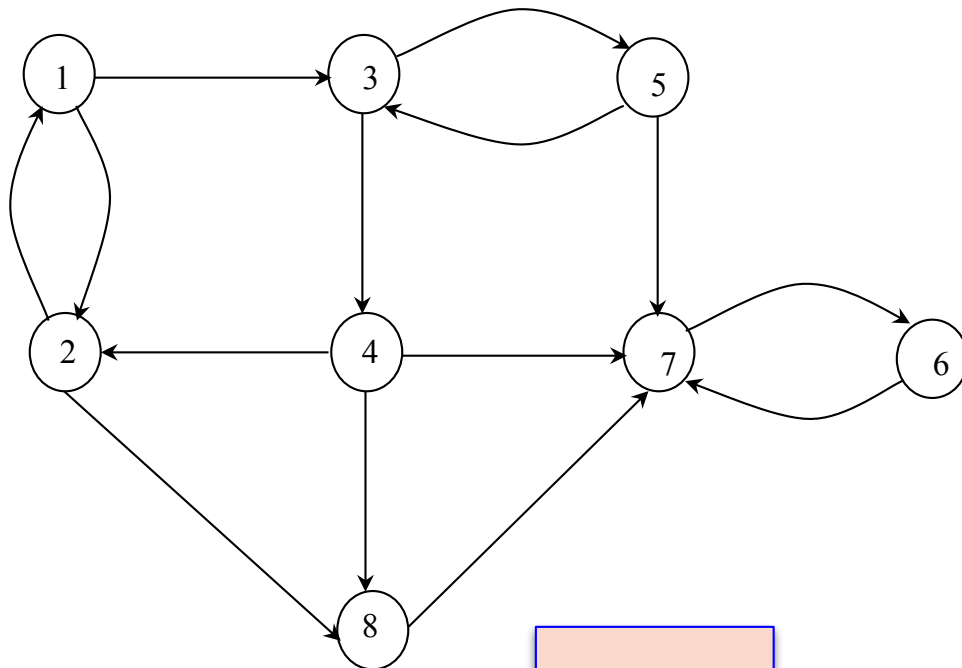


8 đã được duyệt
xong và có $3 == 3$
Loại bỏ 8 ra
 \Rightarrow tìm được BPLT
(8)

70

8
2
1

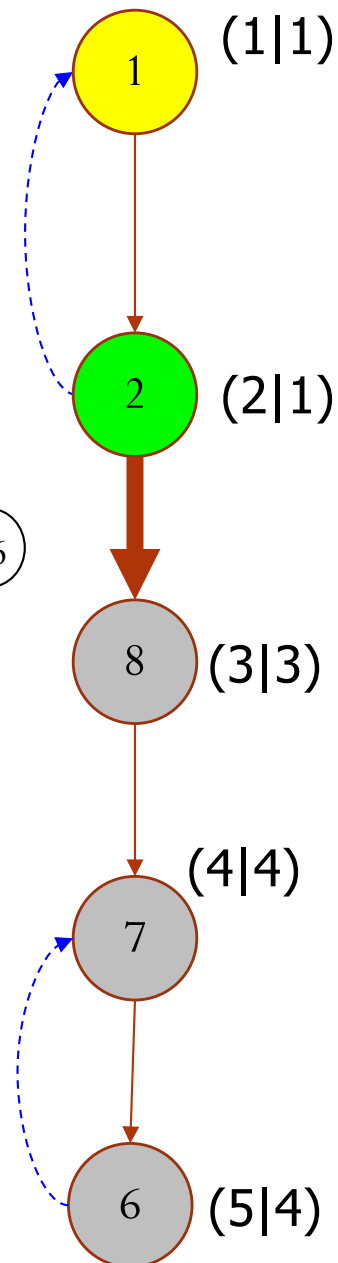




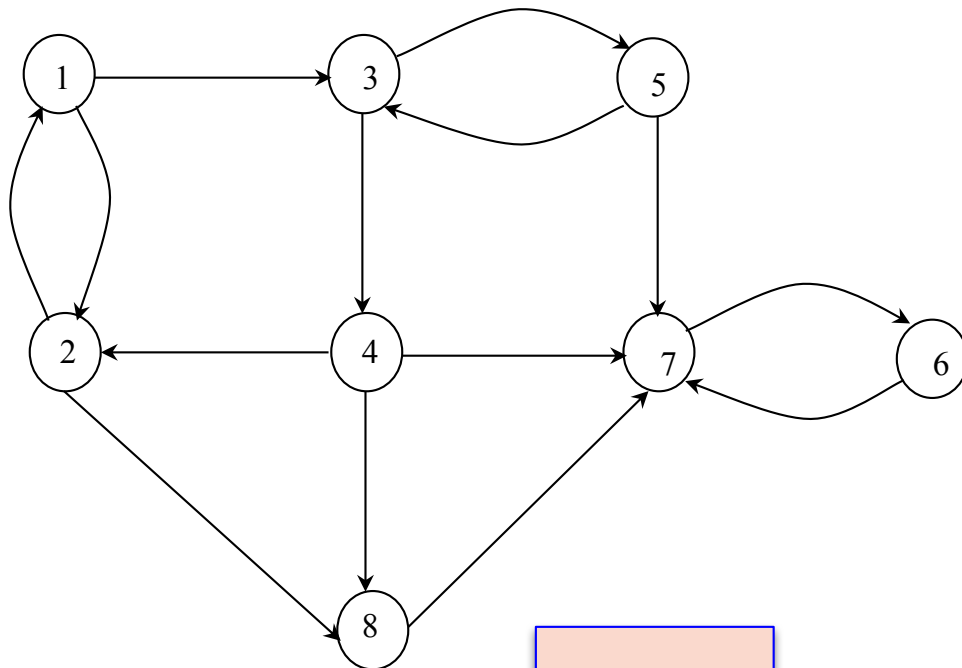
8 đã được duyệt
xong và có $3 == 3$
Loại bỏ 8 ra
 \Rightarrow tìm được BPLT
(8)

71

8
2
1

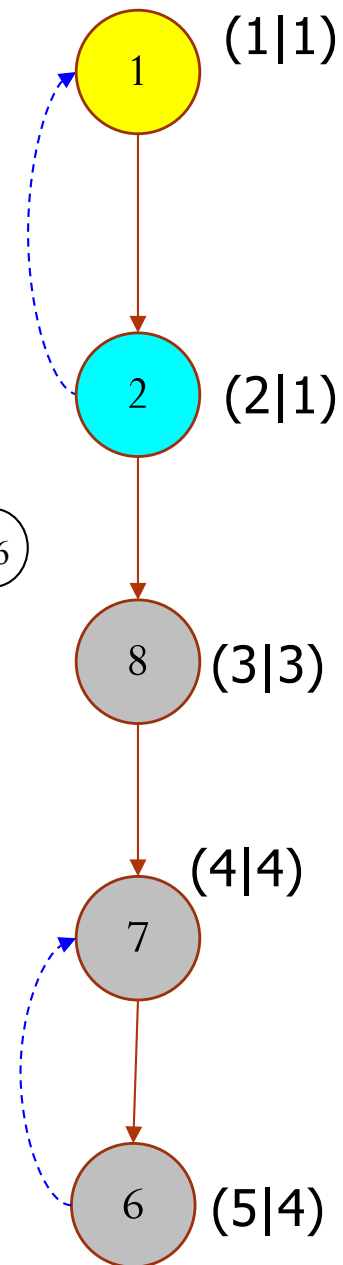


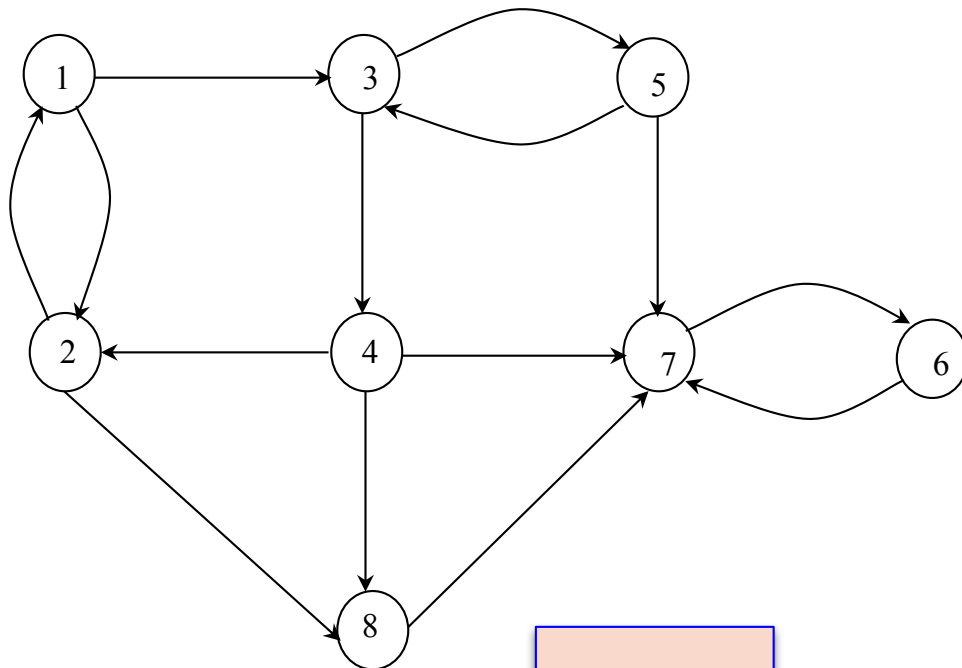
Kiểm tra cập nhật lại
 $\text{min_num}[2]$ (vì lúc này 2
gọi duyệt 8) \Rightarrow **Không
cần cập nhật vì $1 \leq 3$**



2 đã được duyệt
xong và có $2 \neq 1$

2
1

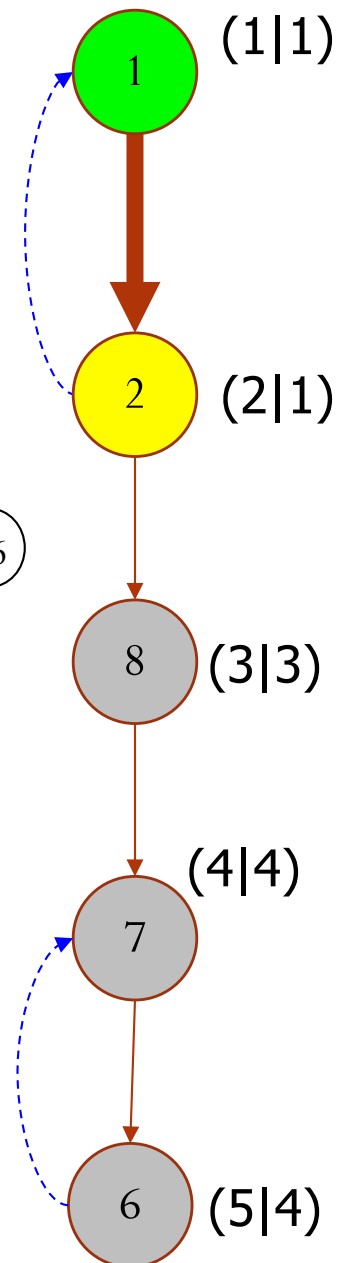




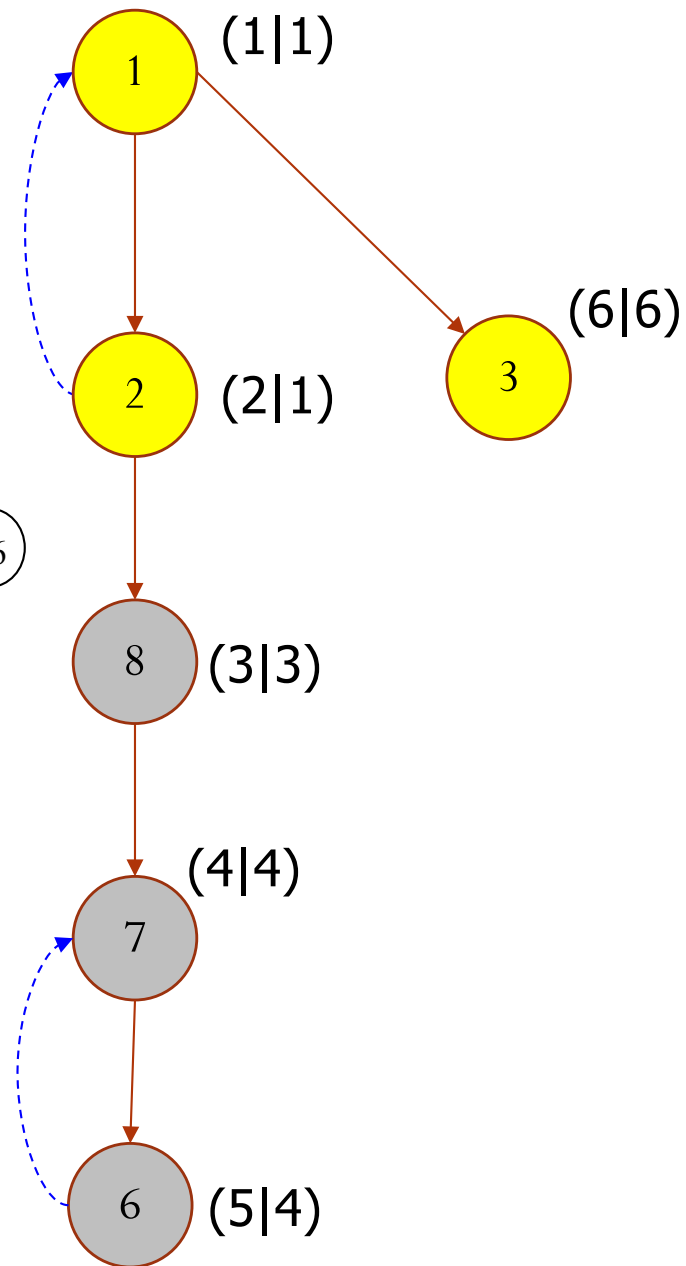
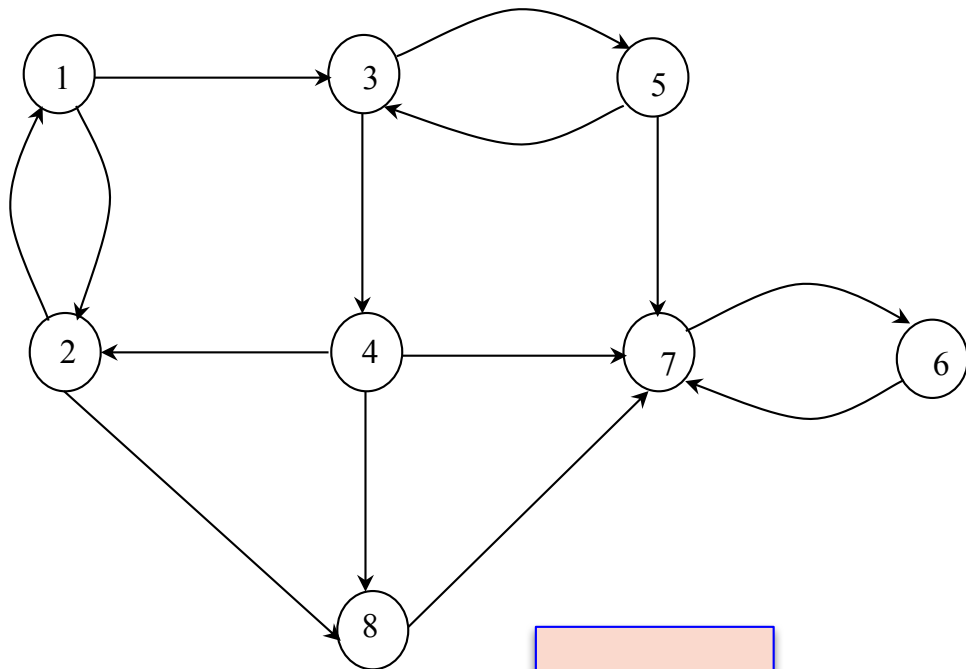
2 đã được duyệt
xong và có $2 \neq 1$

73

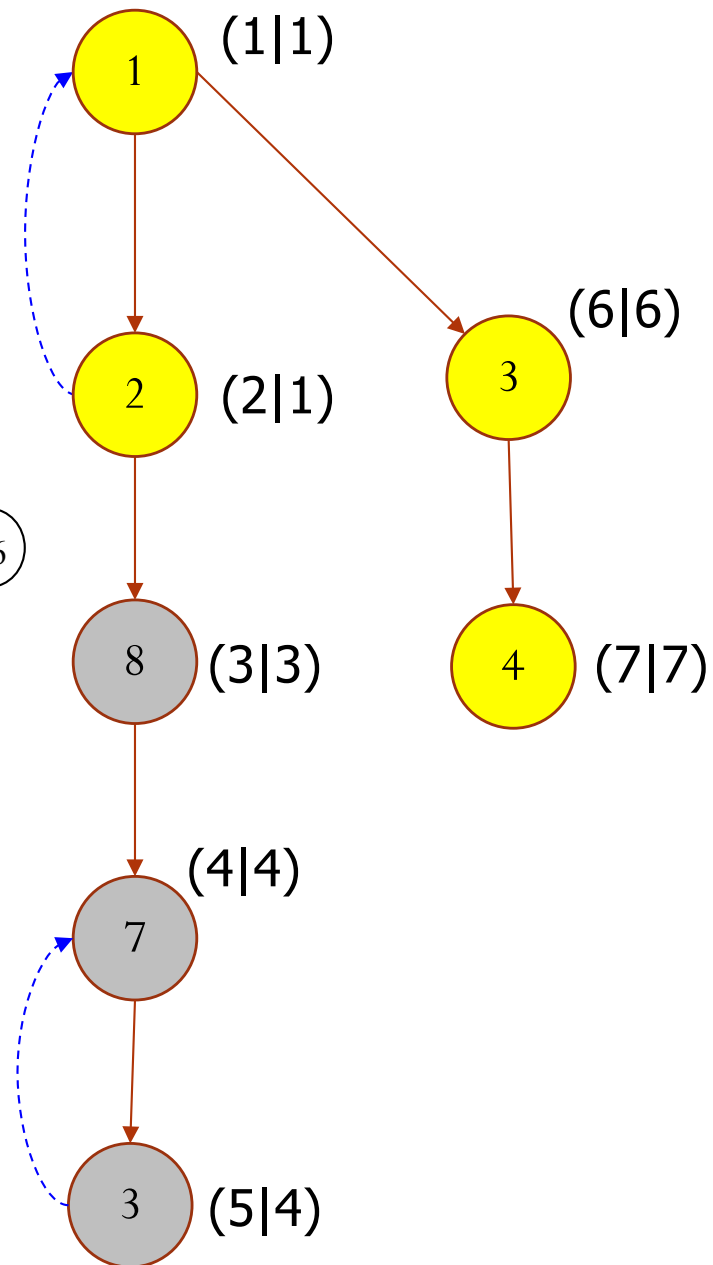
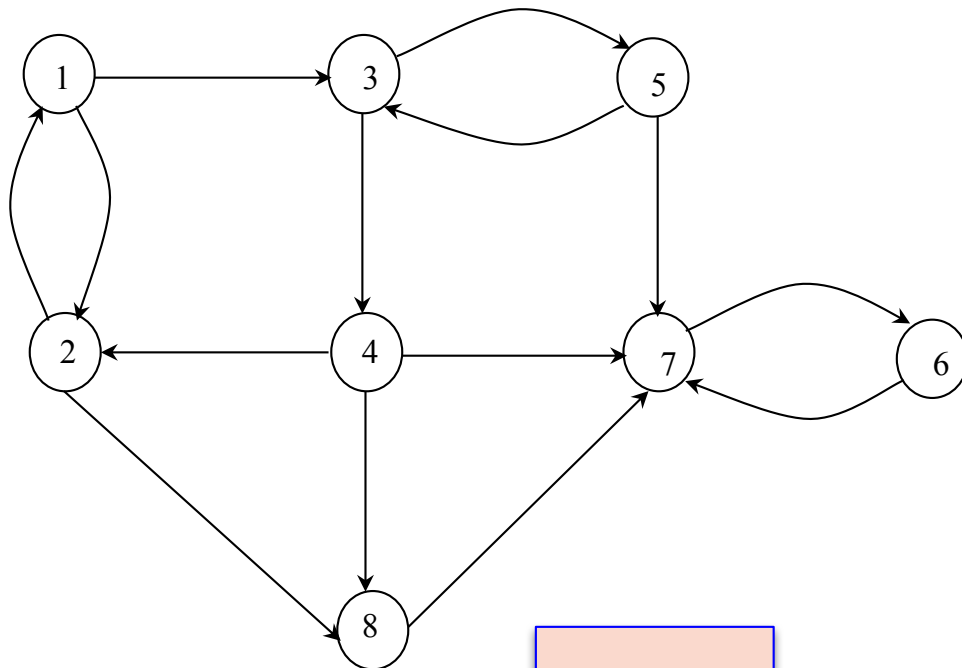
2
1



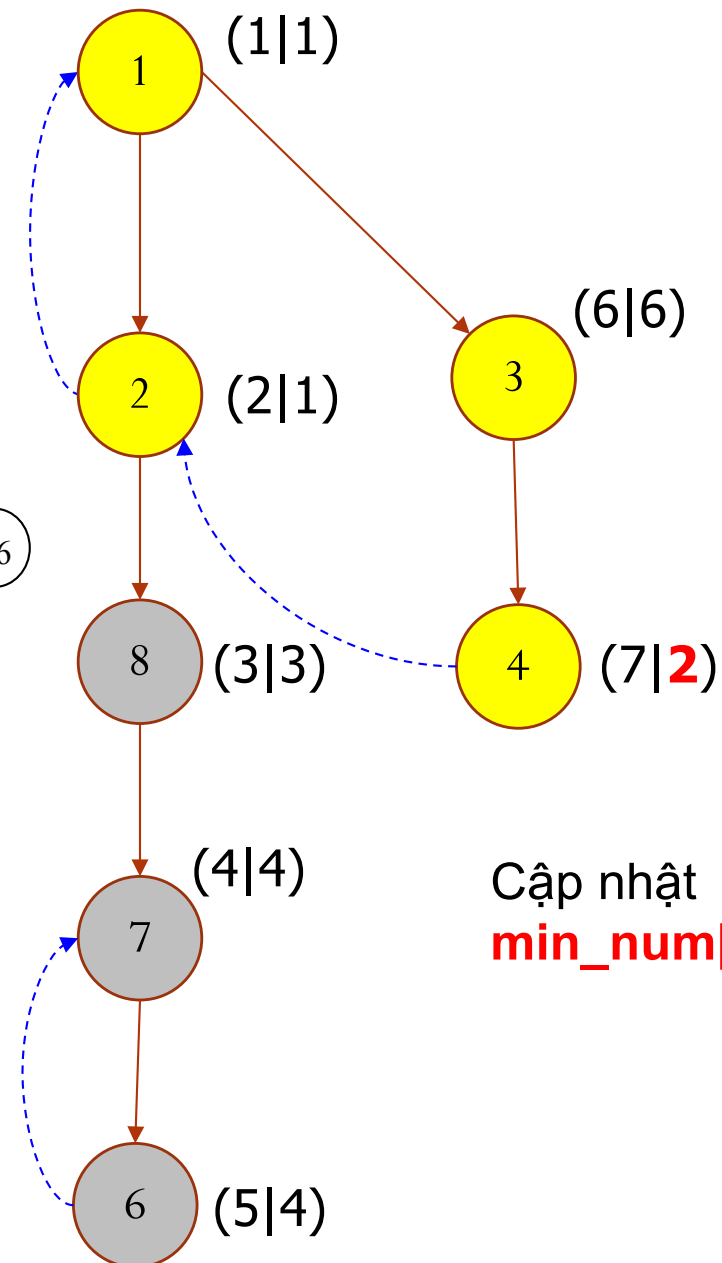
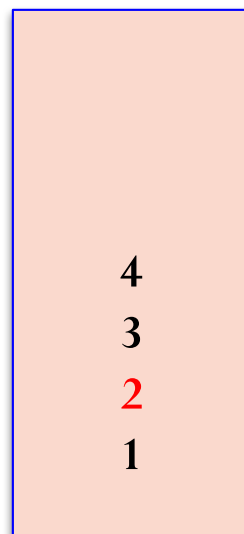
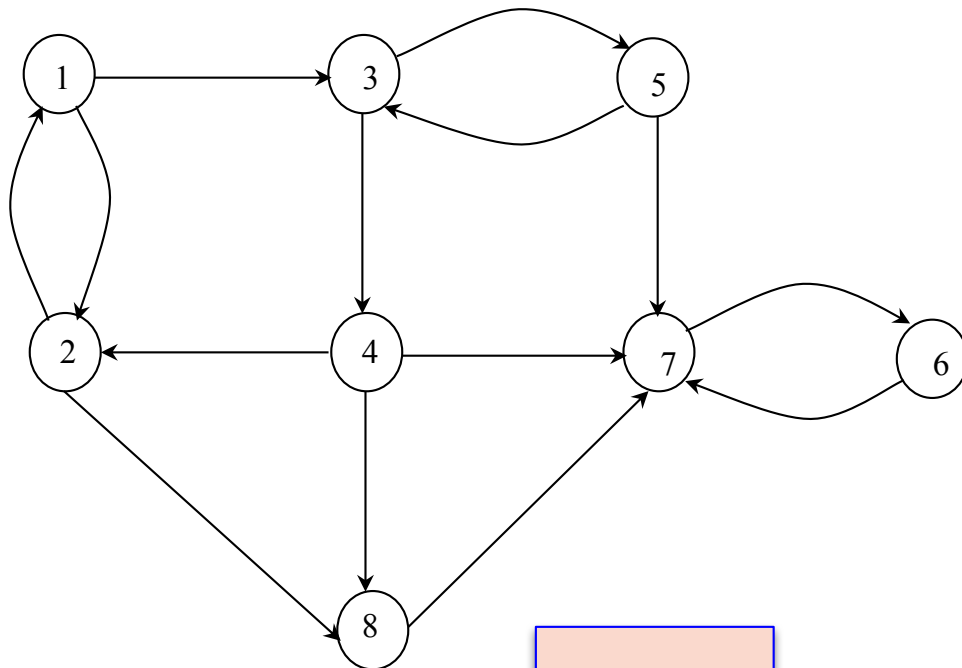
Kiểm tra cập nhật lại
 $\text{min_num}[1]$ (vì lúc này 1
gọi duyệt 2) => **Không
cần cập nhật vì $1 \leq 1$**



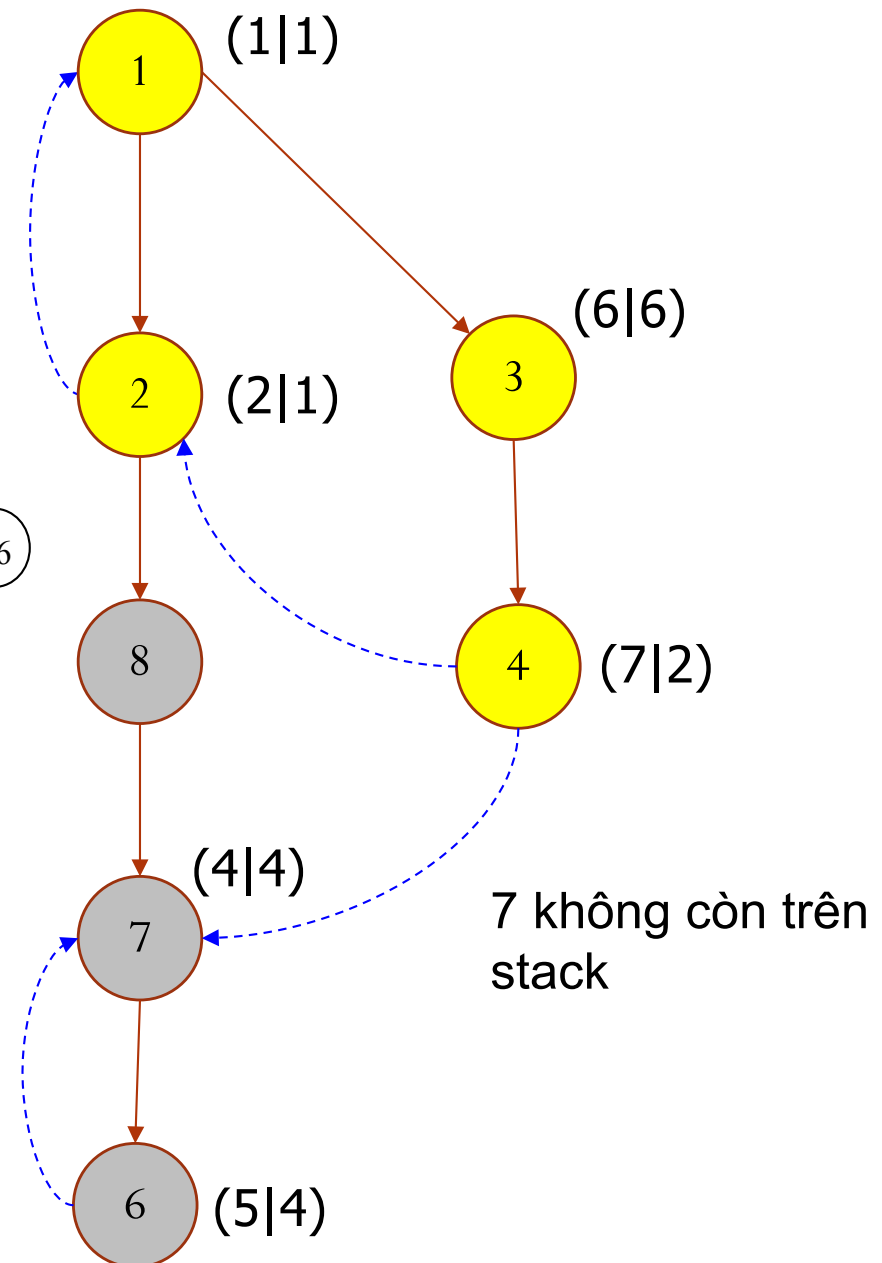
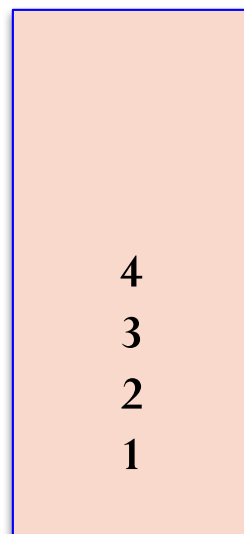
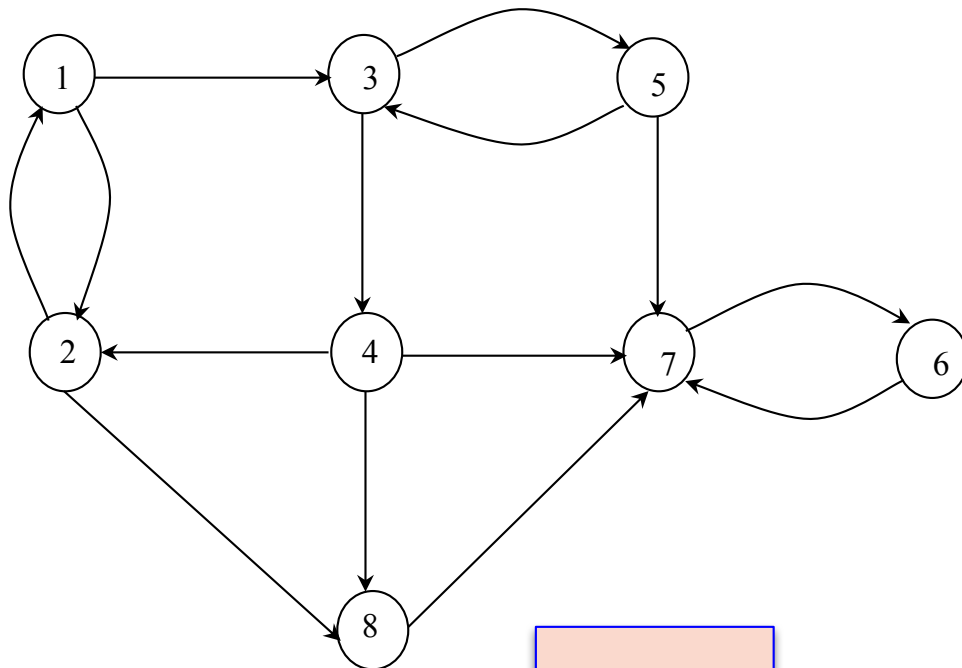
3
2
1

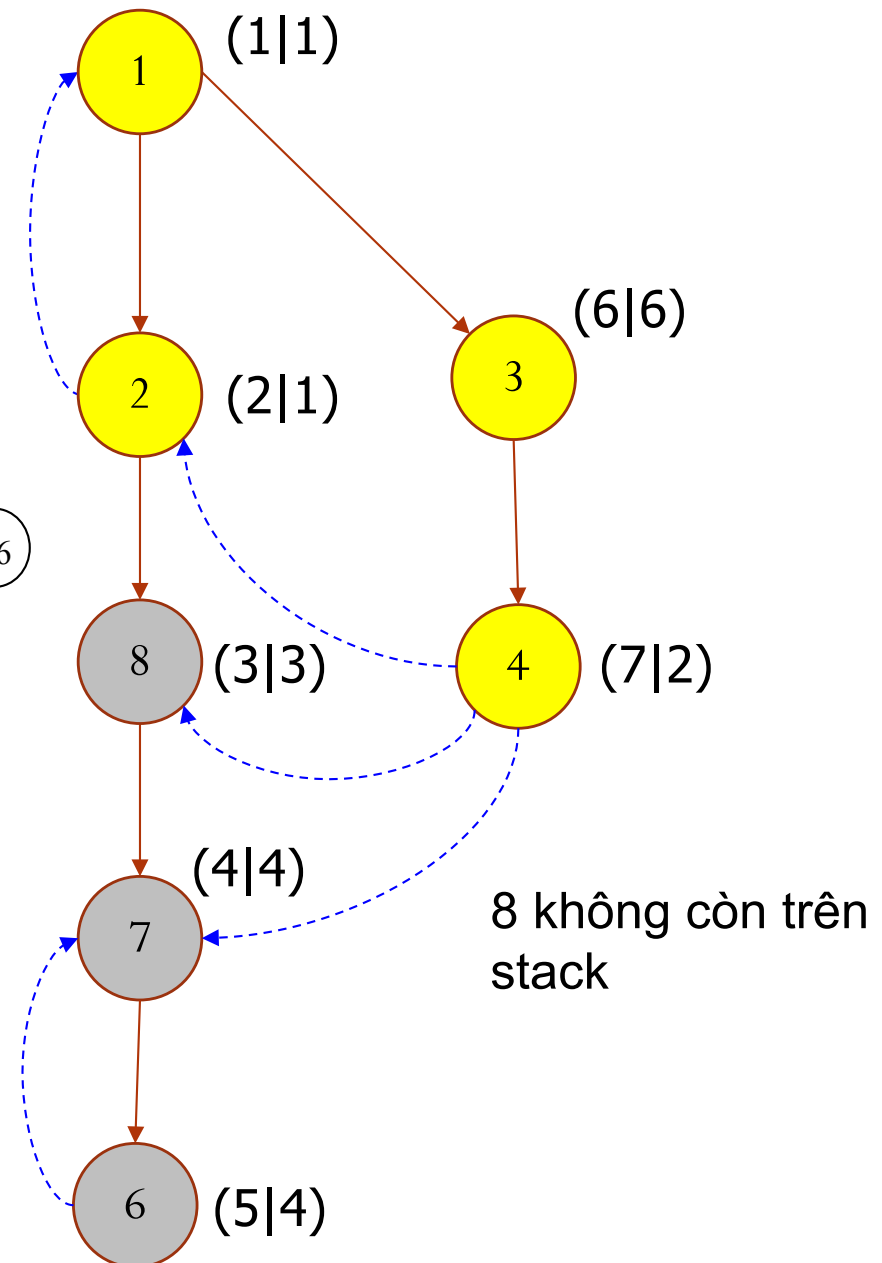
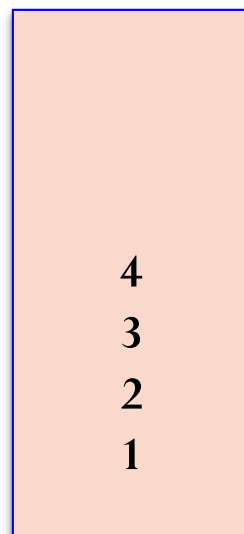
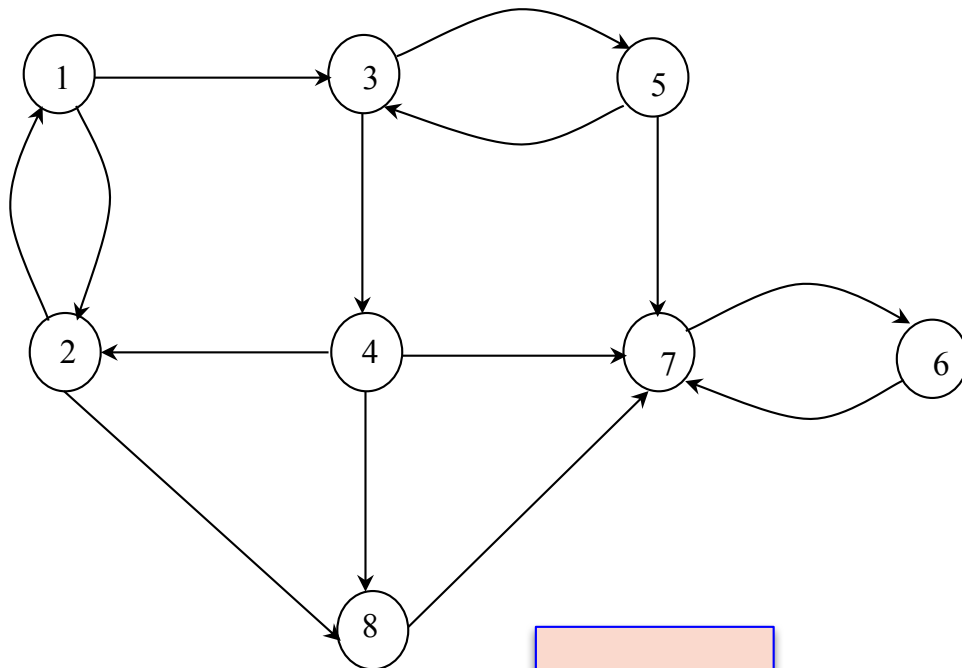


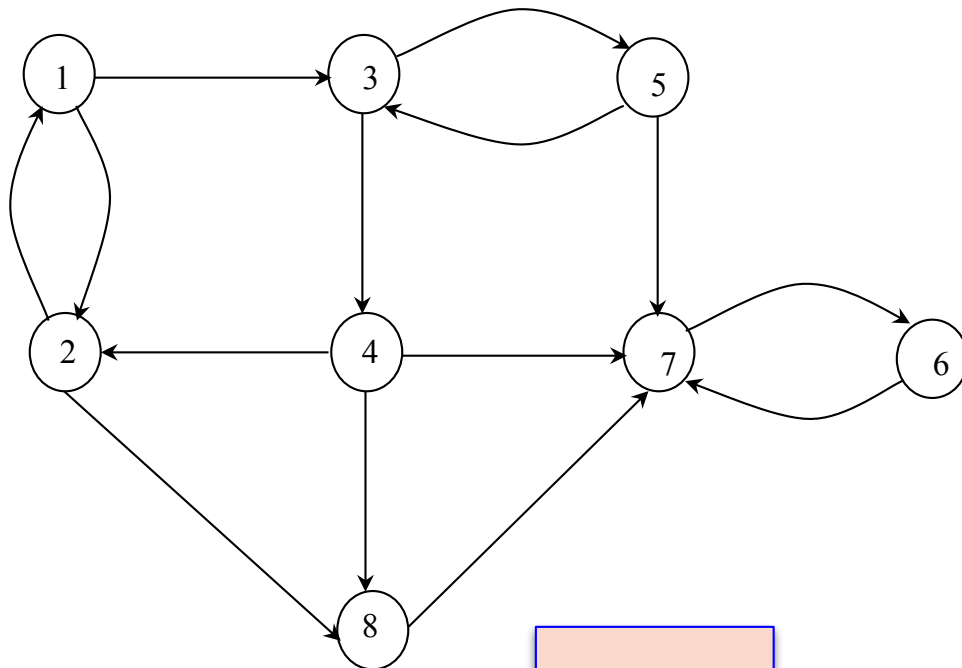
4
3
2
1



Cập nhật
min_num[4] = 2

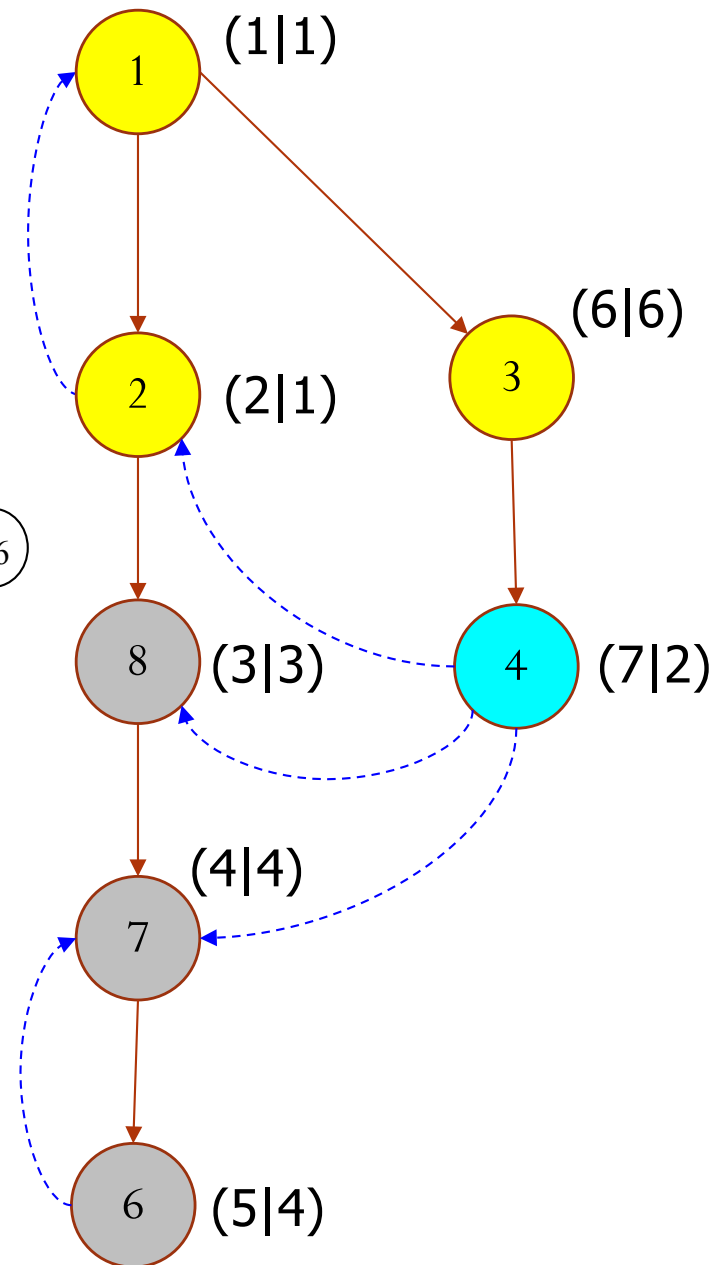


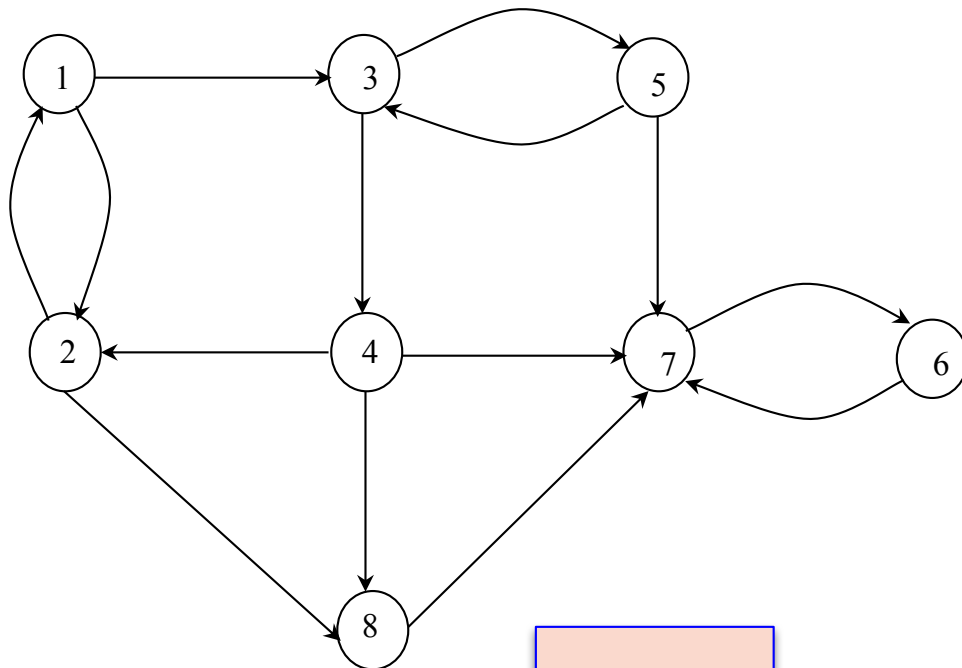




4 đã được duyệt
xong và có $7 \neq 2$

4
3
2
1

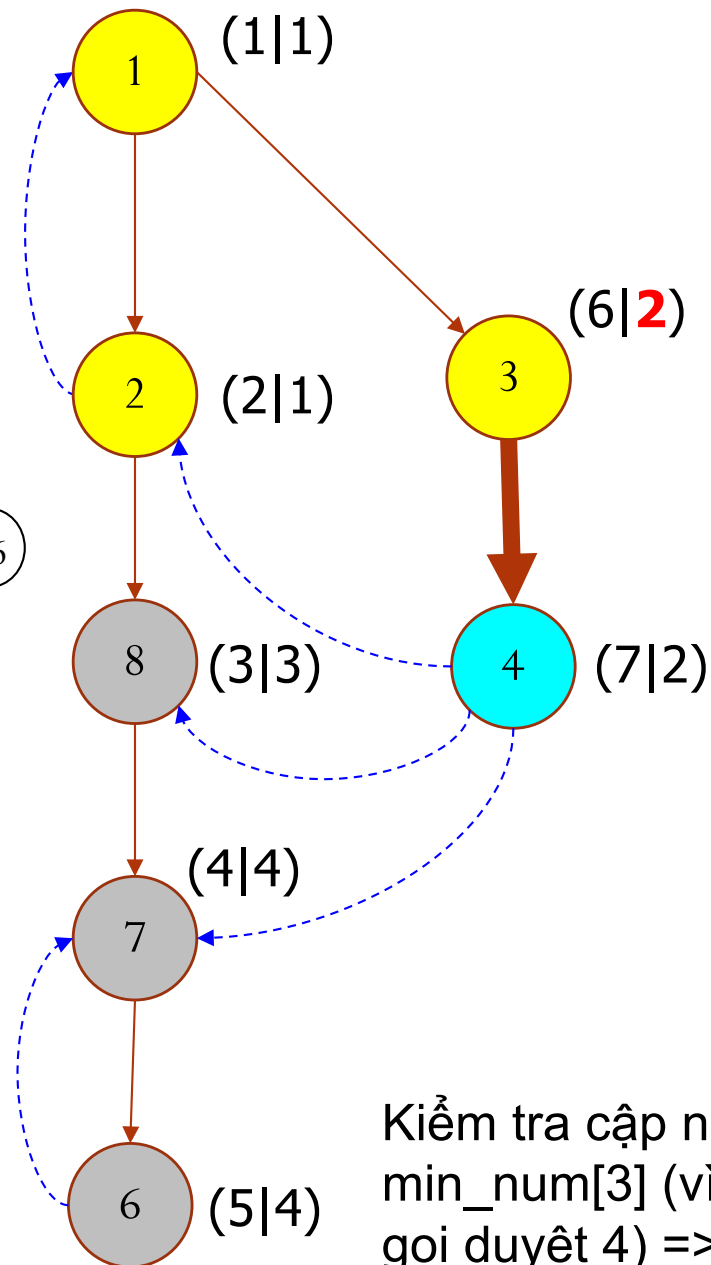




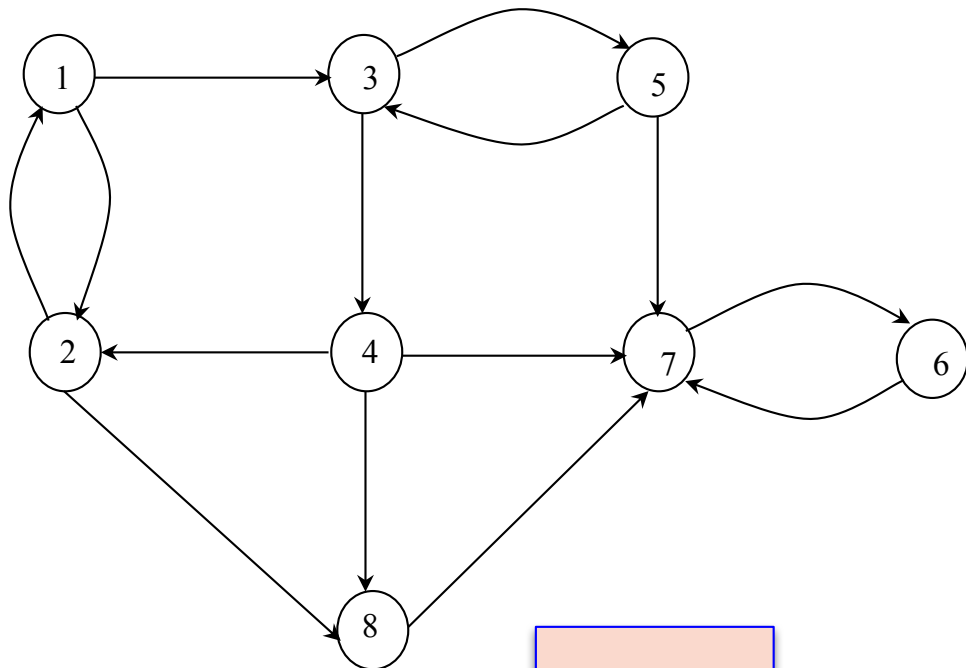
4 đã được duyệt
xong và có $7 \neq 2$

80

4
3
2
1

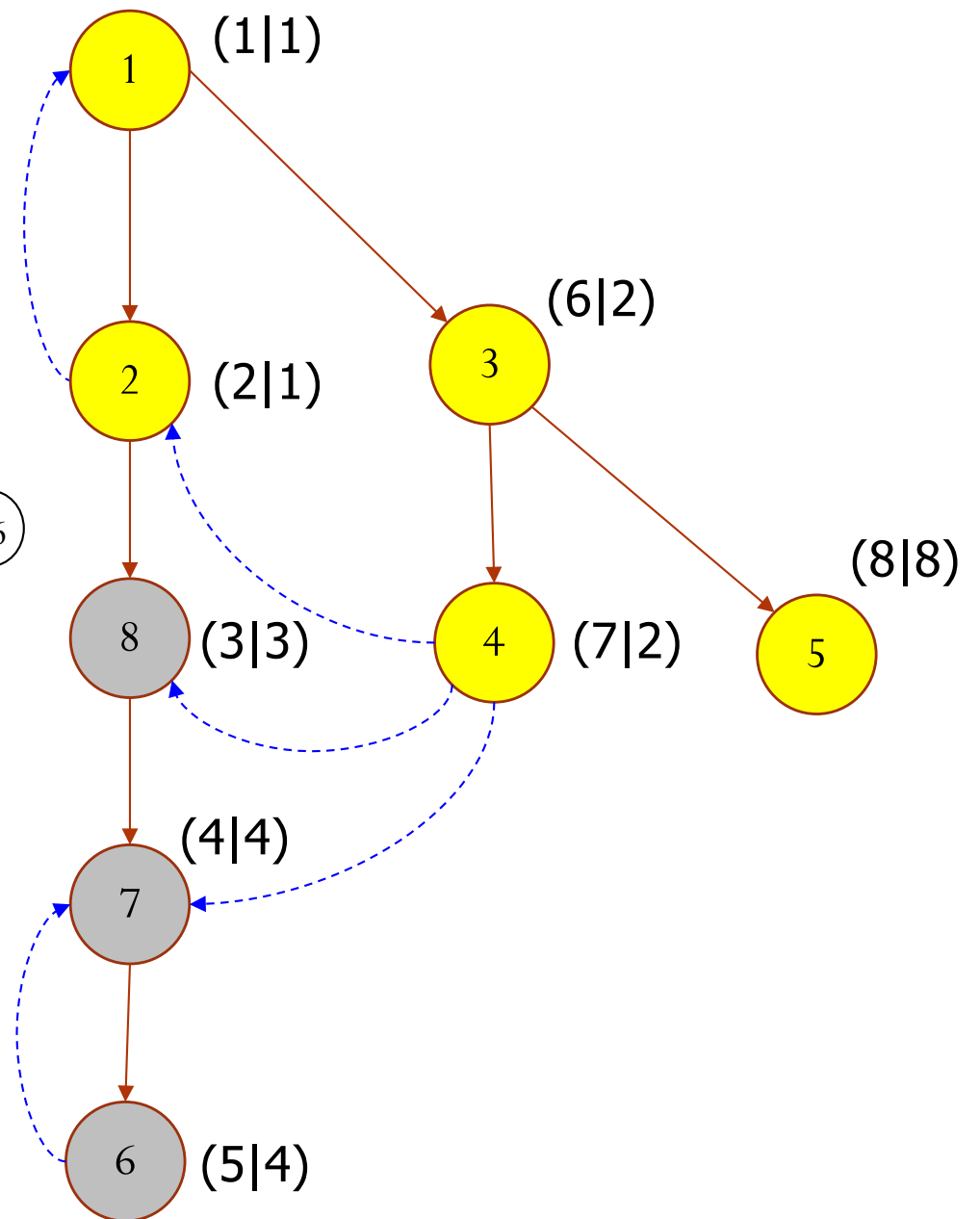


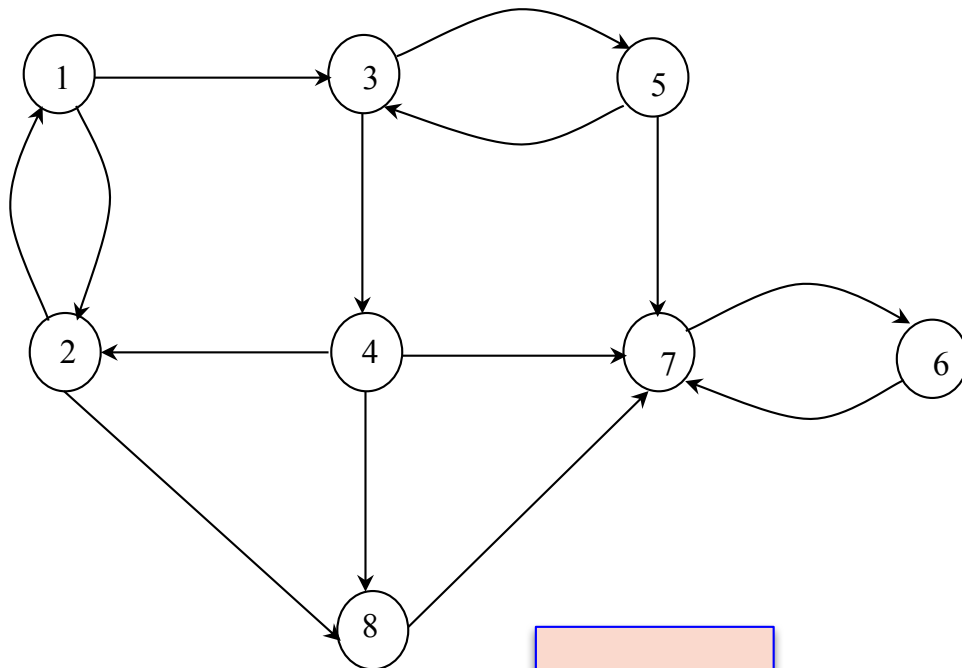
Kiểm tra cập nhật lại
 $\text{min_num}[3]$ (vì lúc này 3
gọi duyệt 4) => **Cập nhật
lại $\text{min_num}[3] = 2$**



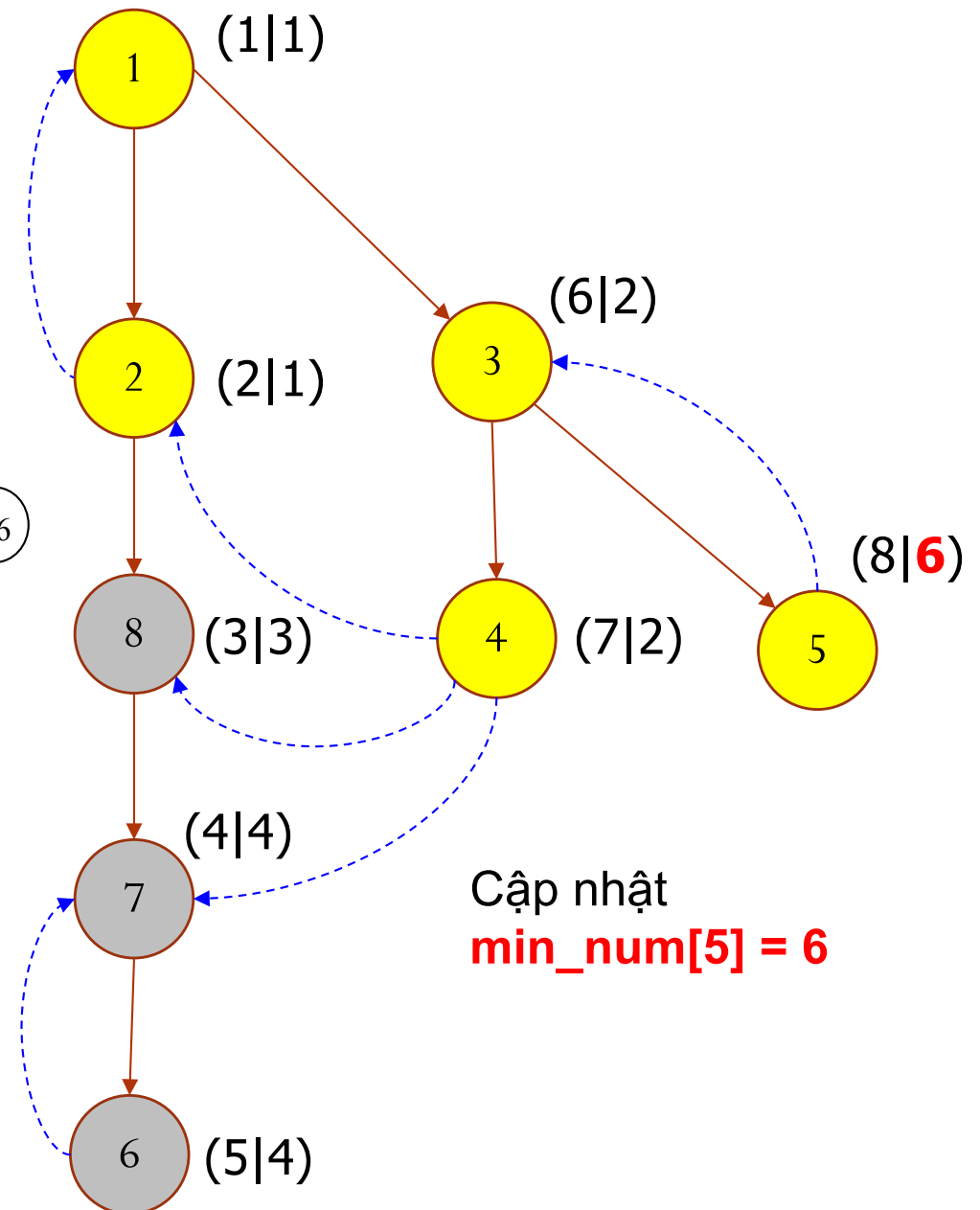
5
4
3
2
1

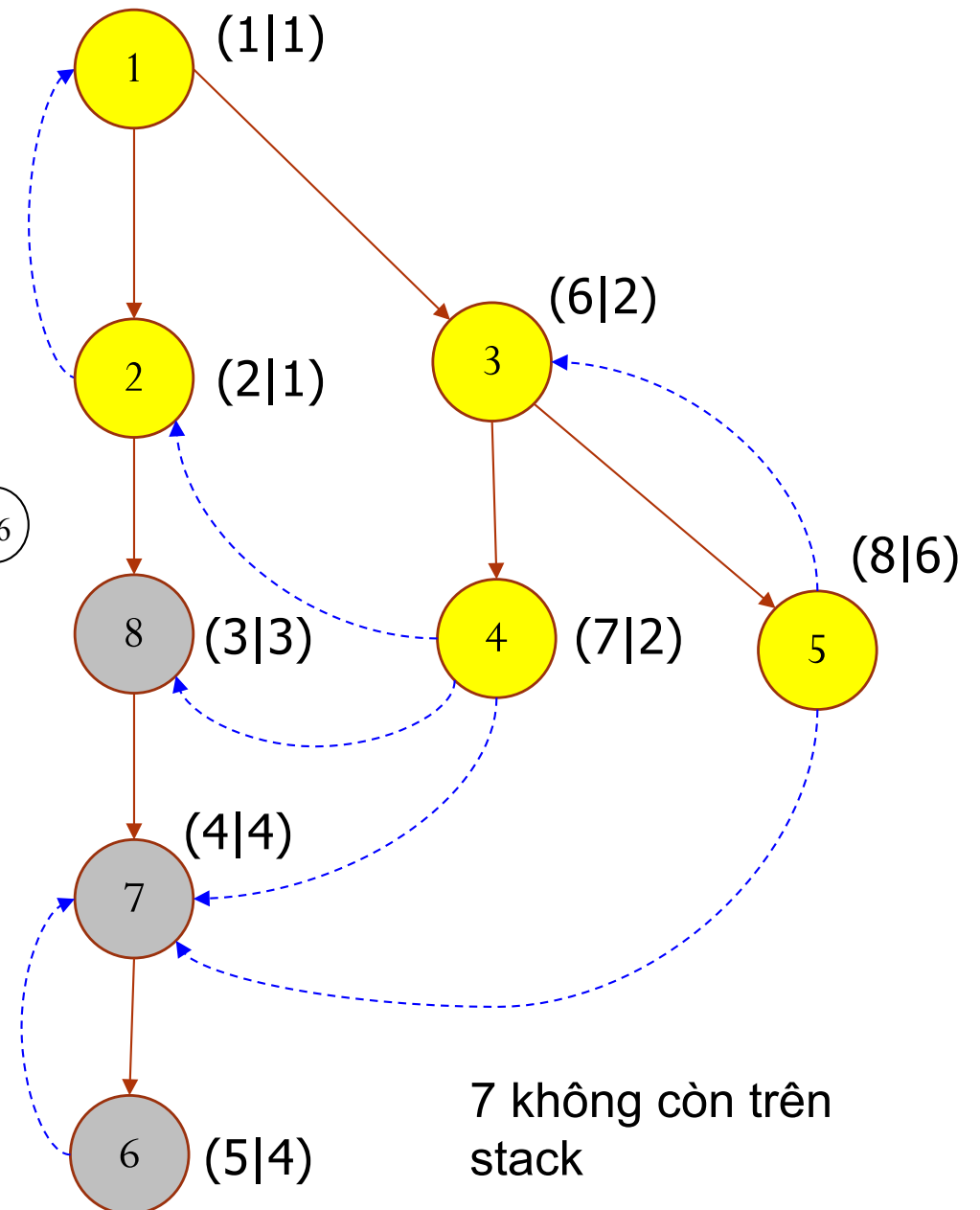
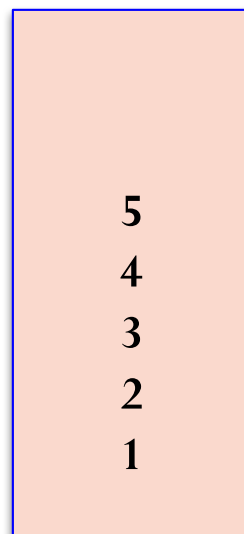
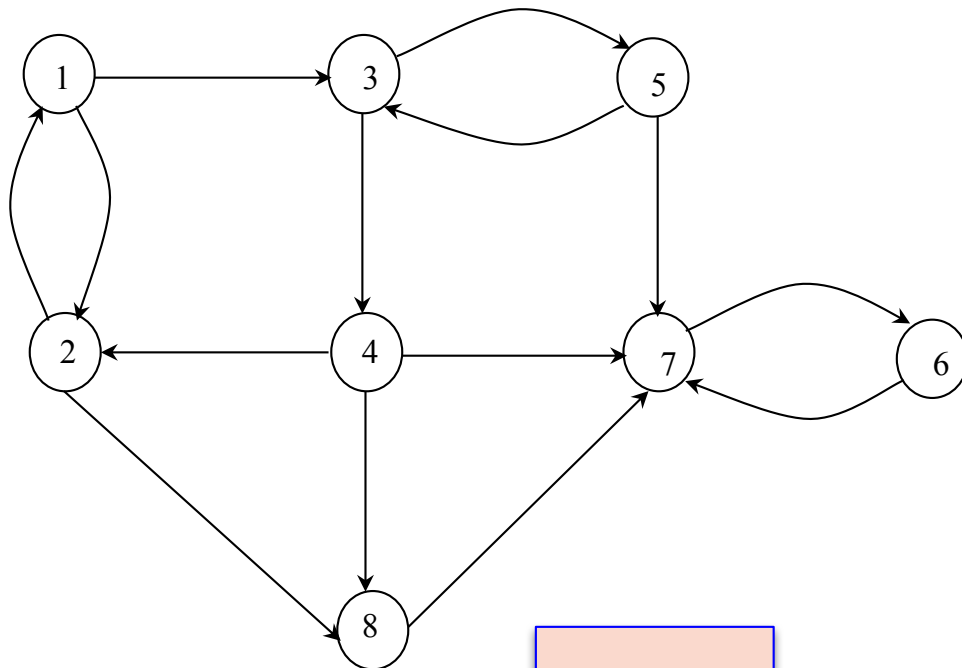
81

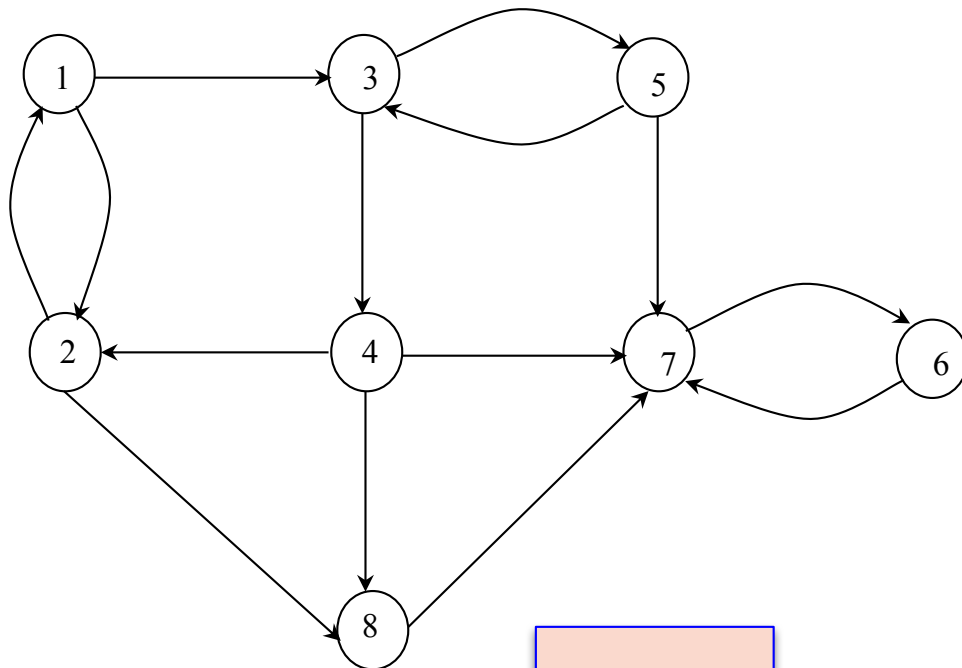




5
4
3
2
1

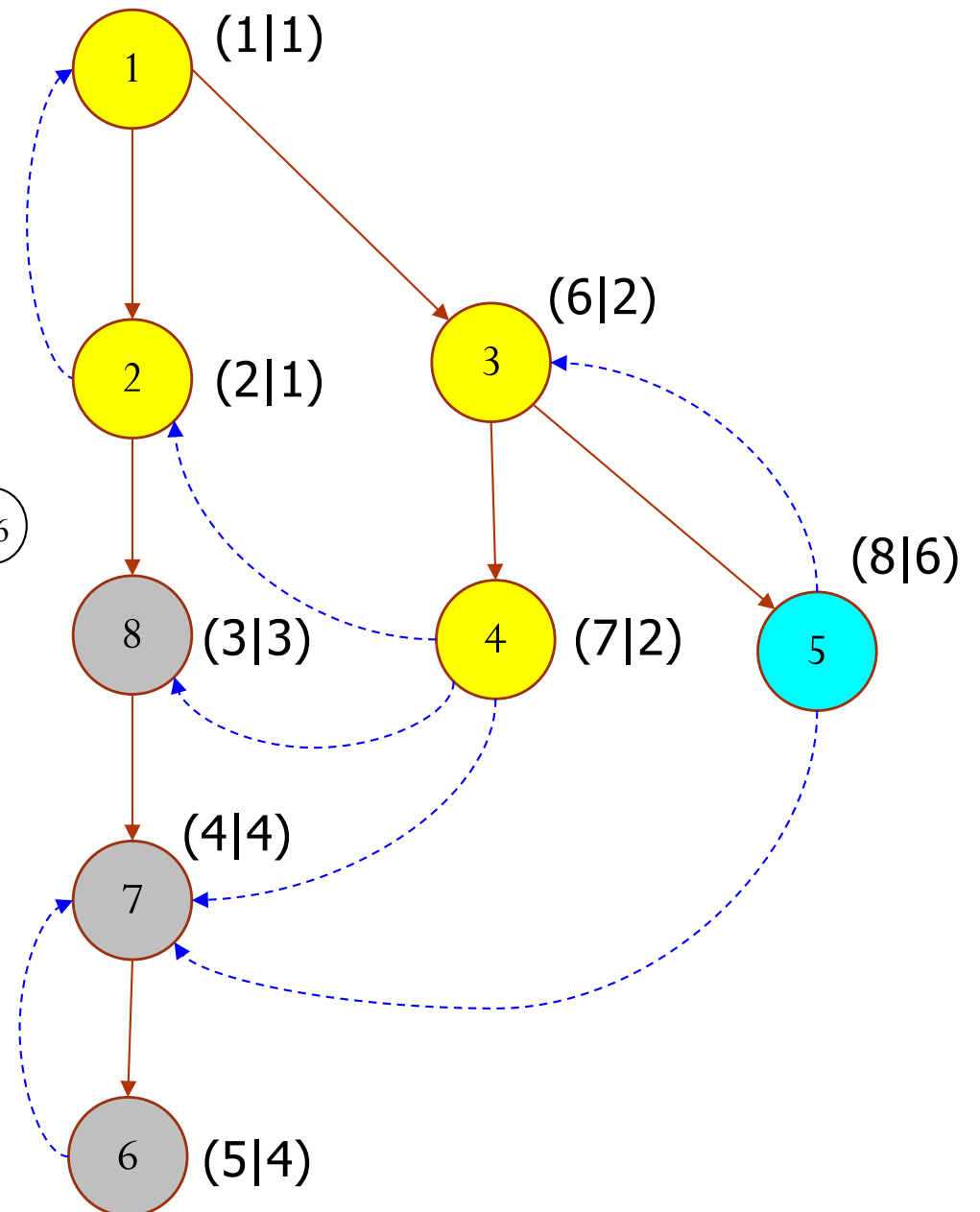


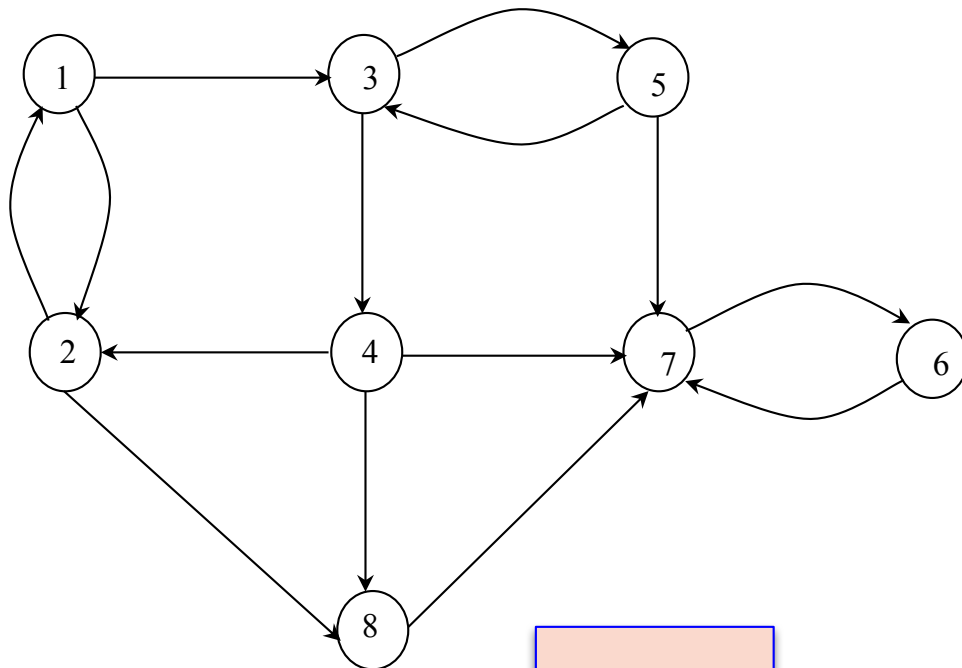




5 đã được duyệt
xong và có $8 \neq 6$

5
4
3
2
1

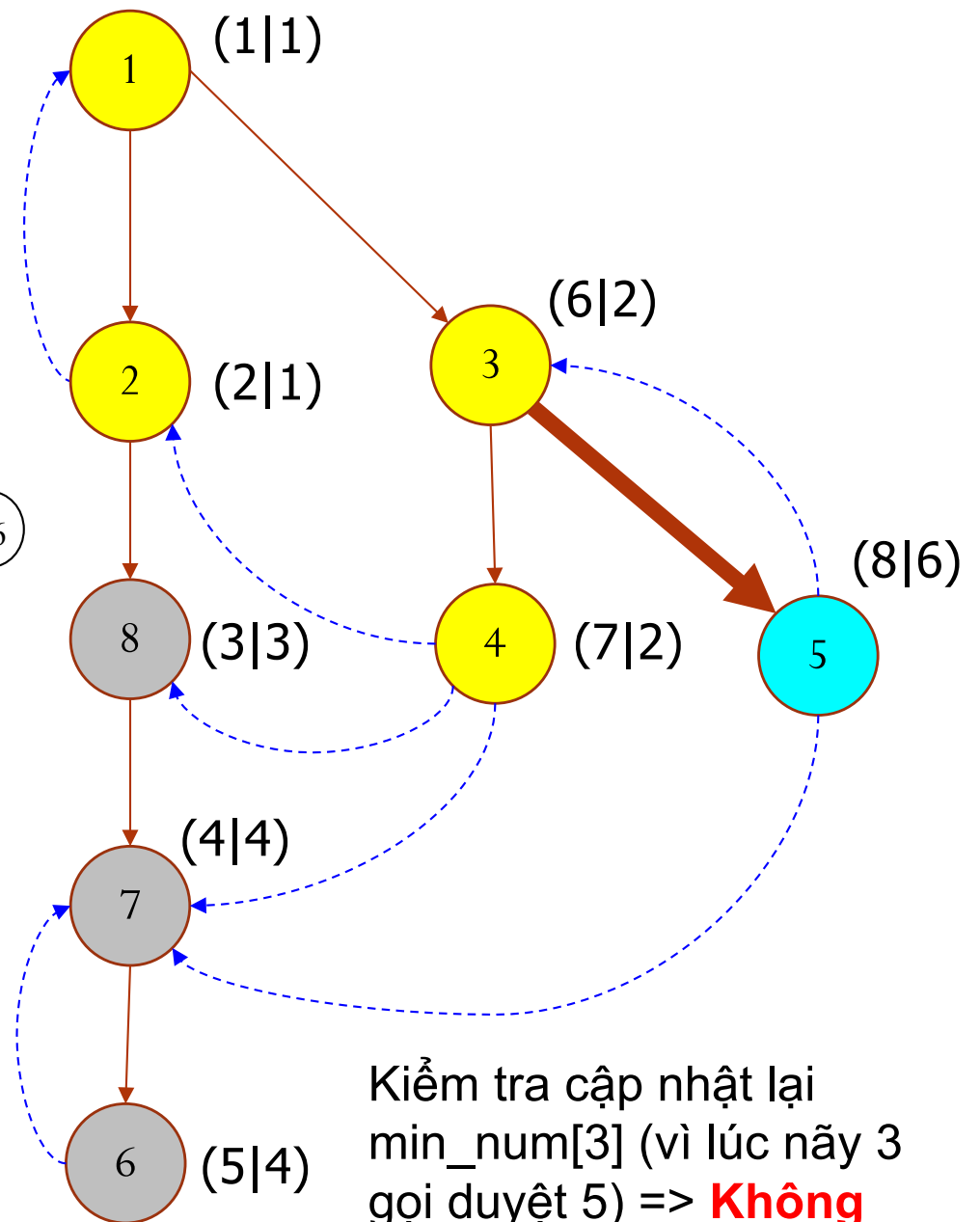




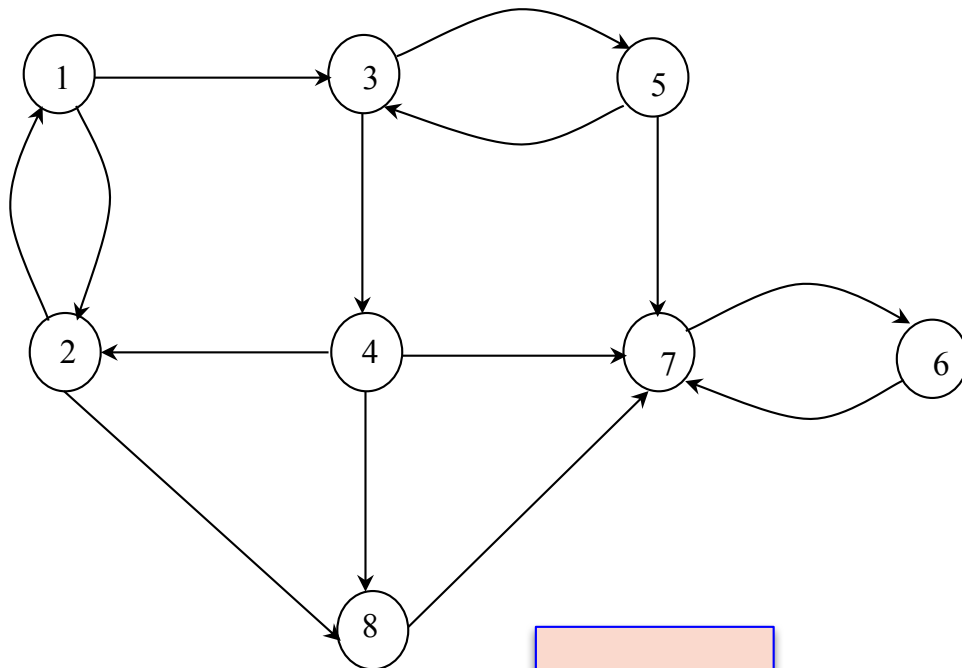
5 đã được duyệt
xong và có $8 \neq 6$

85

5
4
3
2
1

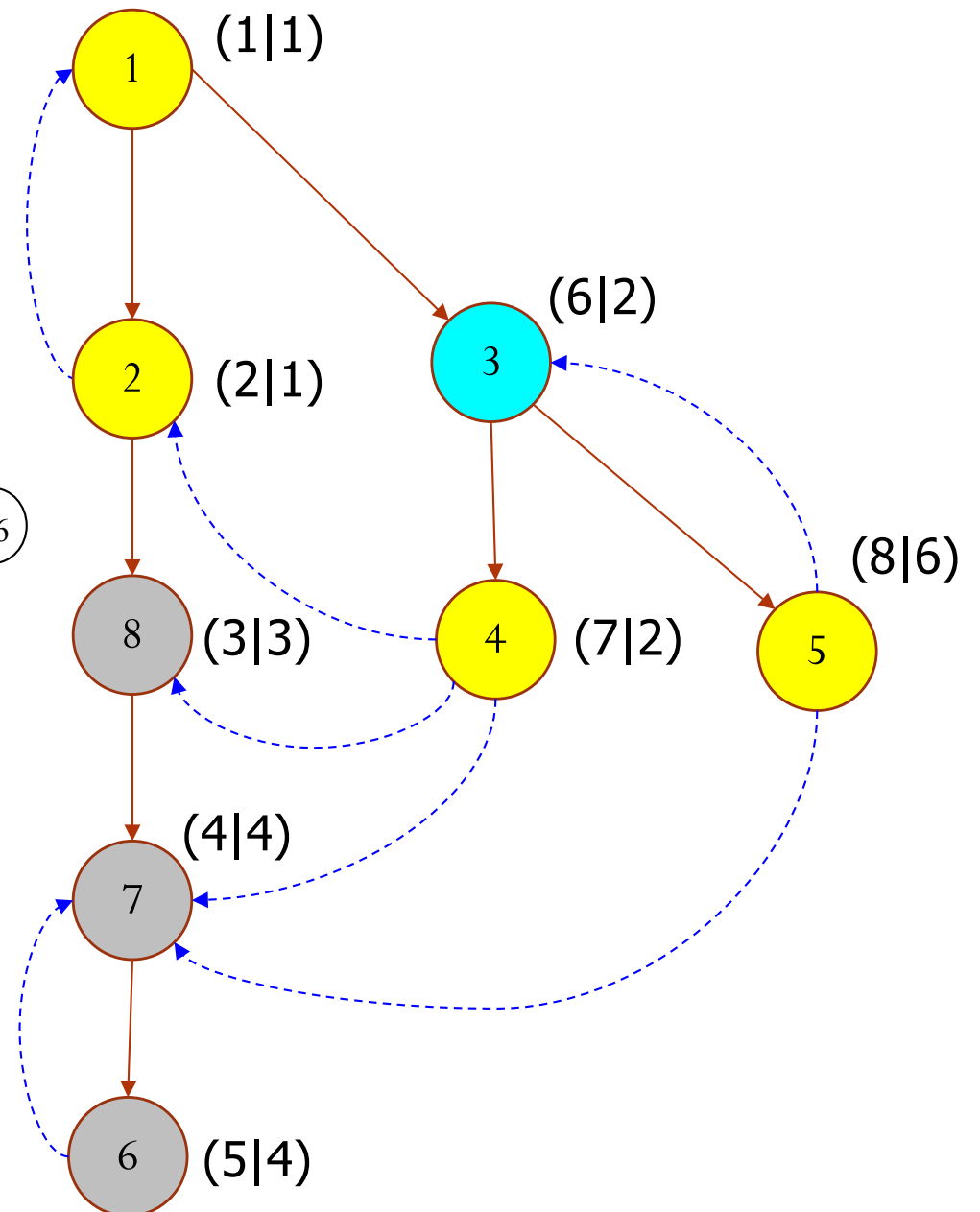


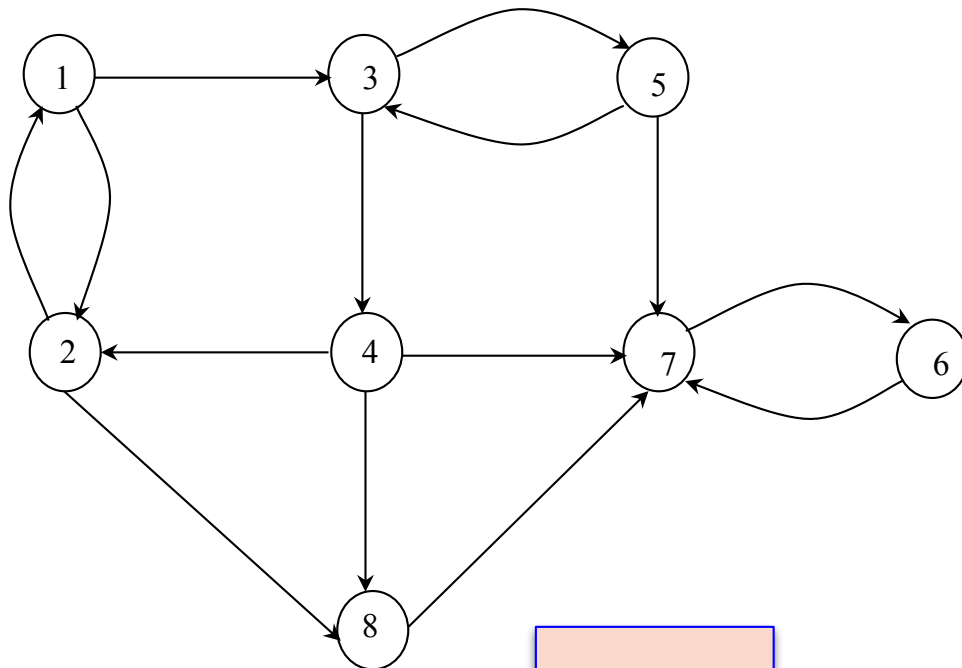
Kiểm tra cập nhật lại
 $\text{min_num}[3]$ (vì lúc này 3
gọi duyệt 5) => **Không**
cần CN vì $2 \leq 6$



3 đã được duyệt
xong và có $6 \neq 2$

5
4
3
2
1

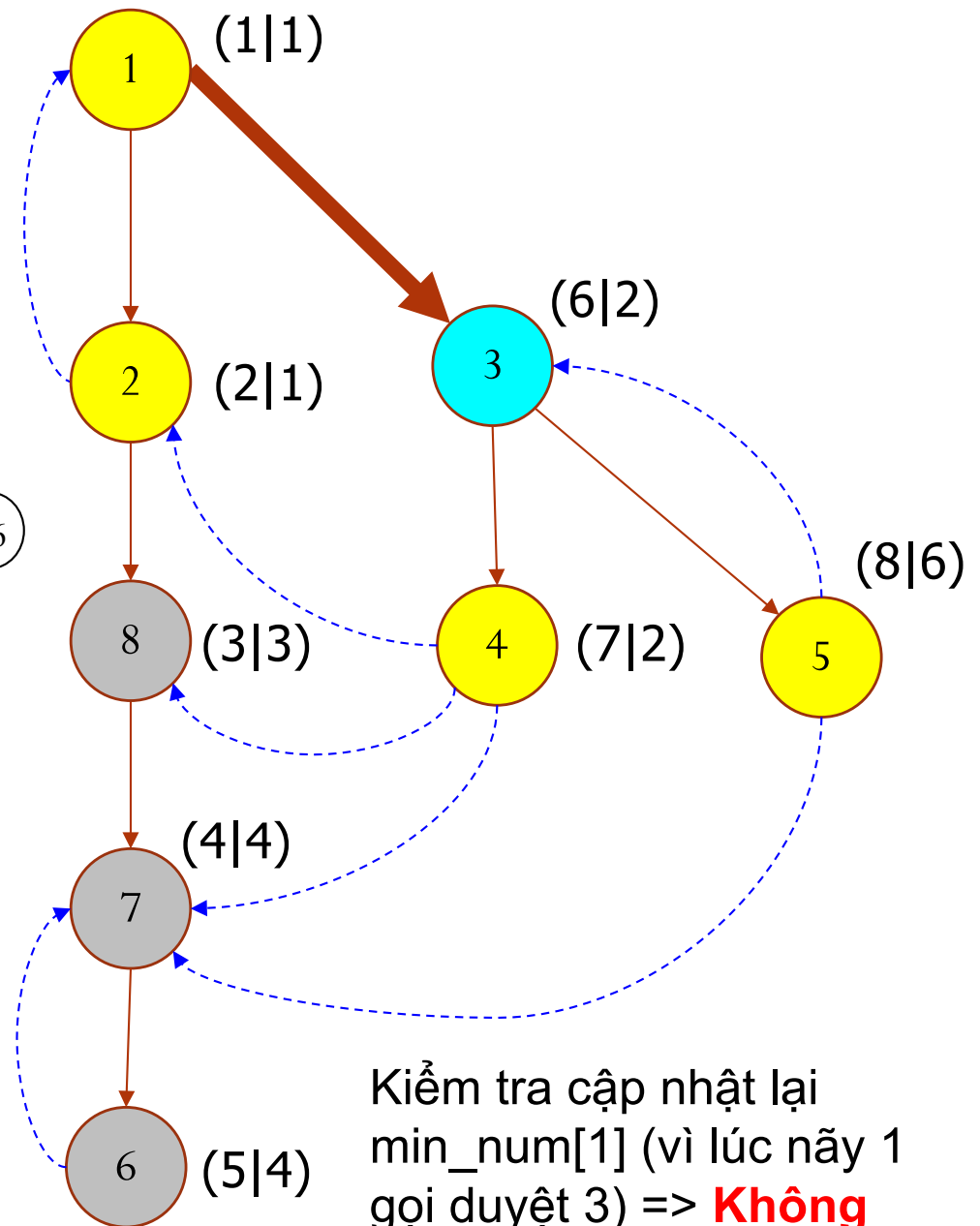




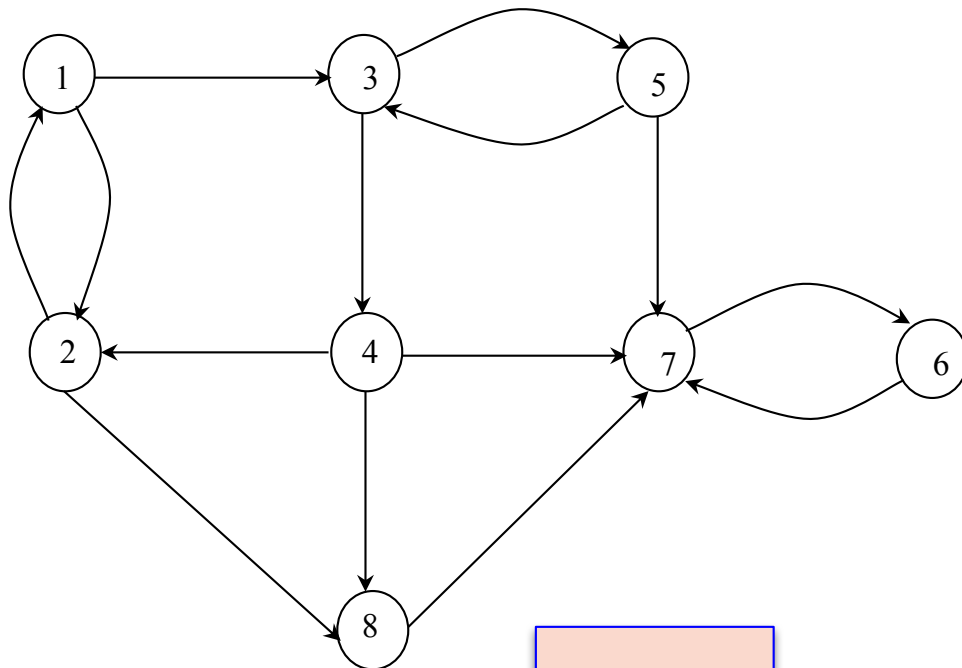
3 đã được duyệt
xong và có $6 \neq 2$

5
4
3
2
1

87

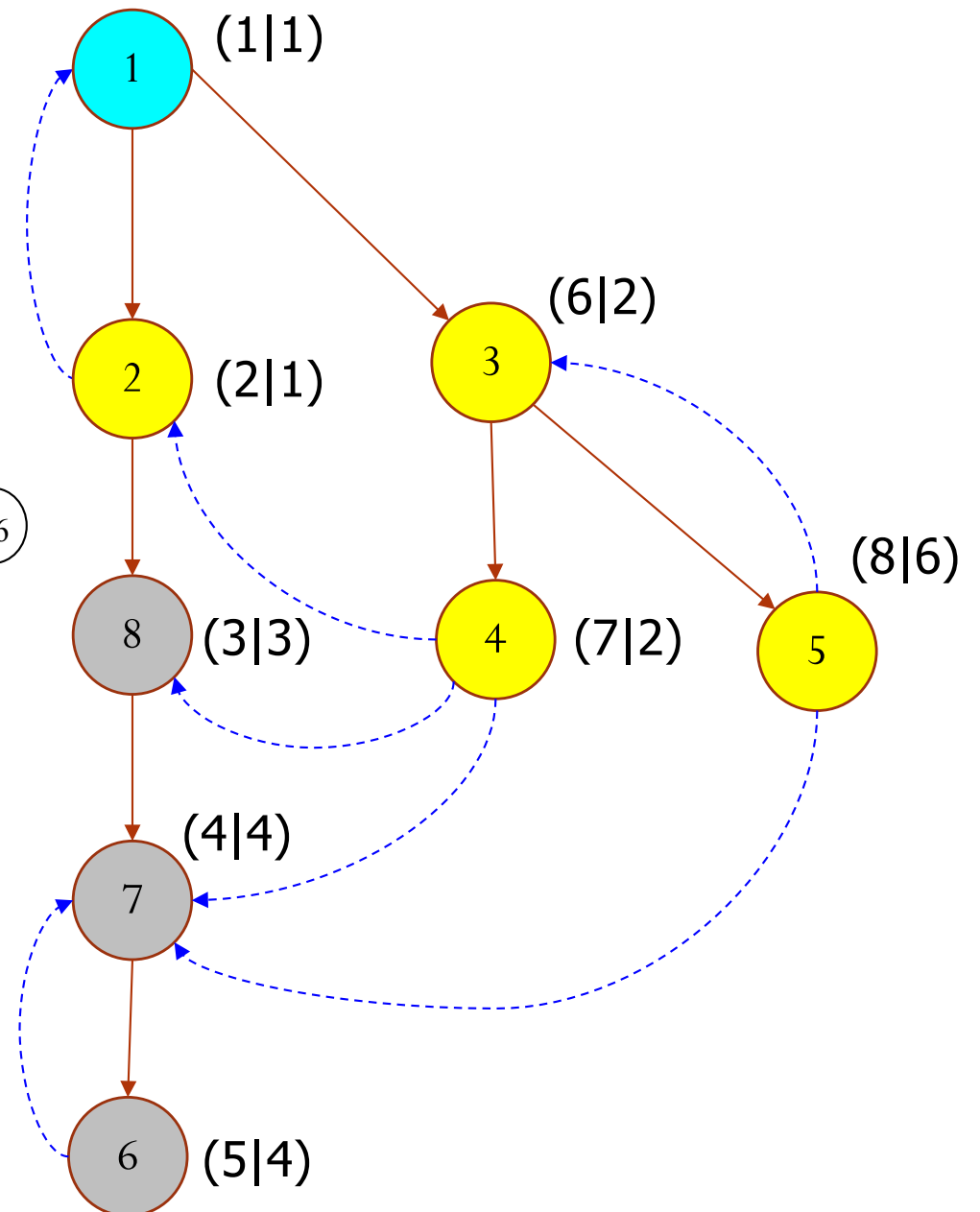


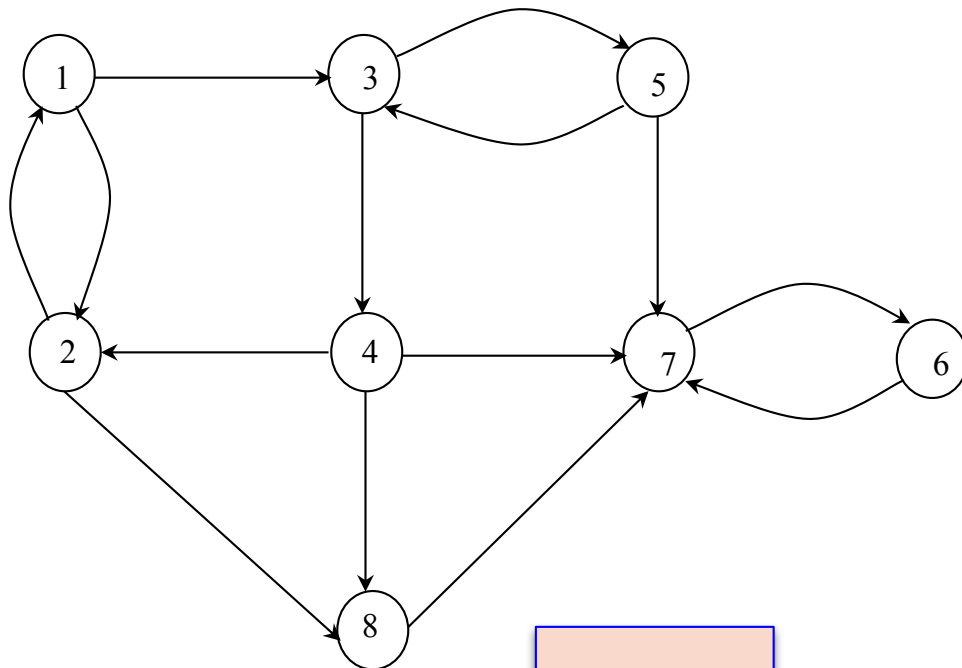
Kiểm tra cập nhật lại
 $\text{min_num}[1]$ (vì lúc này 1
gọi duyệt 3) => **Không**
cần CN vì $1 \leq 2$



1 đã được duyệt
xong và có $1 = 1$

5
4
3
2
1





1 đã được duyệt
xong và có $1 = 1$
Loại bỏ 5, 4, 3, 2, 1
ra \Rightarrow tìm được
BPLT (1,2,3,4,5)

89

5
4
3
2
1

