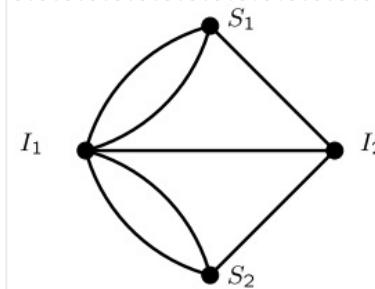
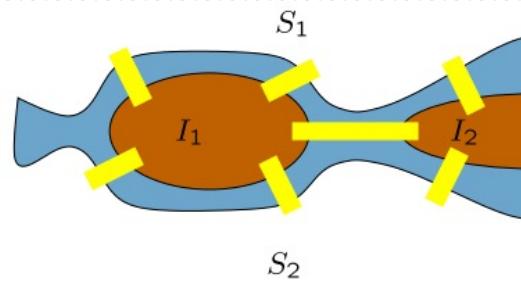


# LÝ THUYẾT ĐỒ THỊ

## Luồng cực đại trong mạng

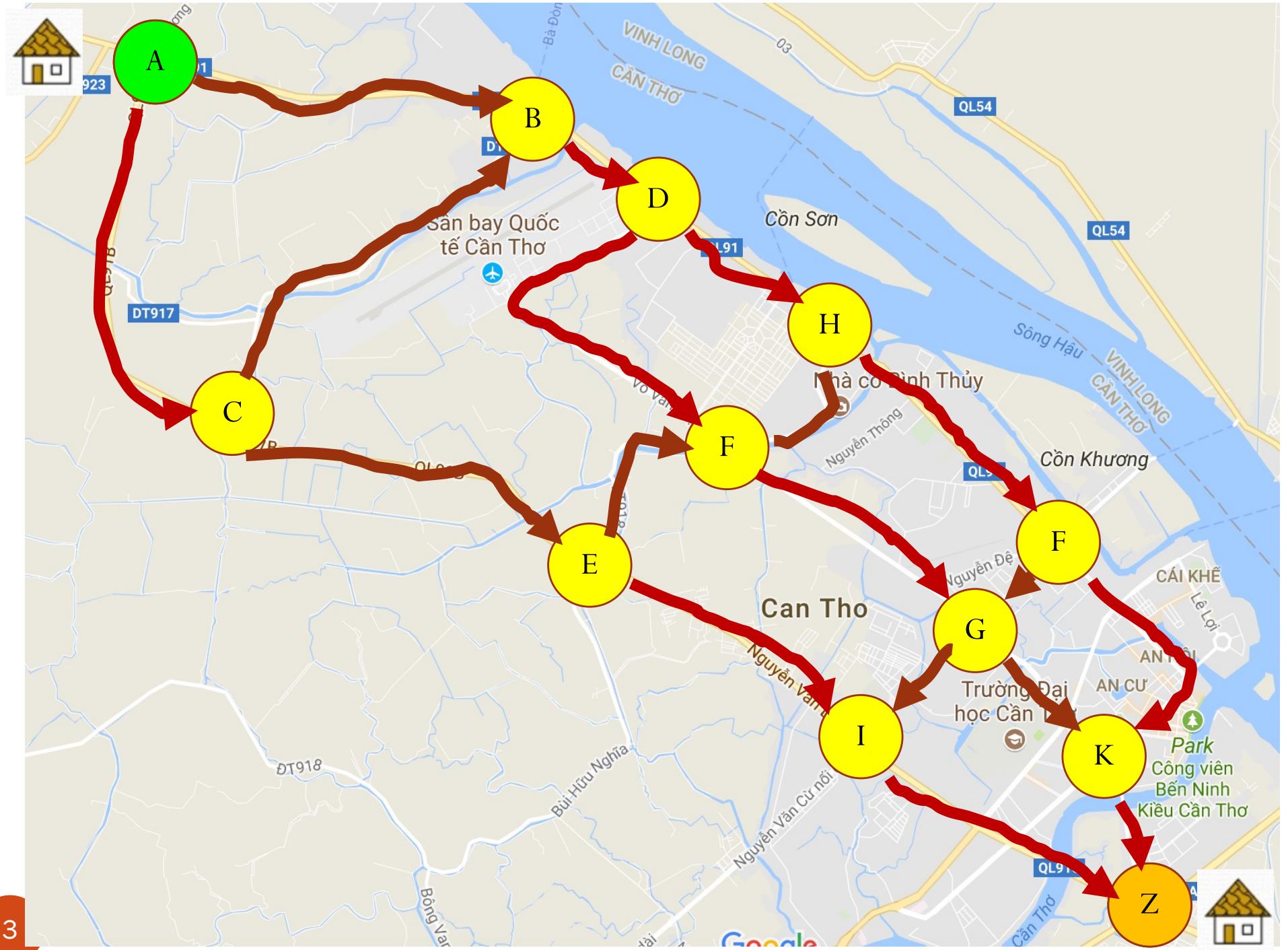
Phạm Nguyên Khang  
BM. Khoa học máy tính, CNTT  
[pnkhang@cit.ctu.edu.vn](mailto:pnkhang@cit.ctu.edu.vn)



Cần Thơ, 8/2021

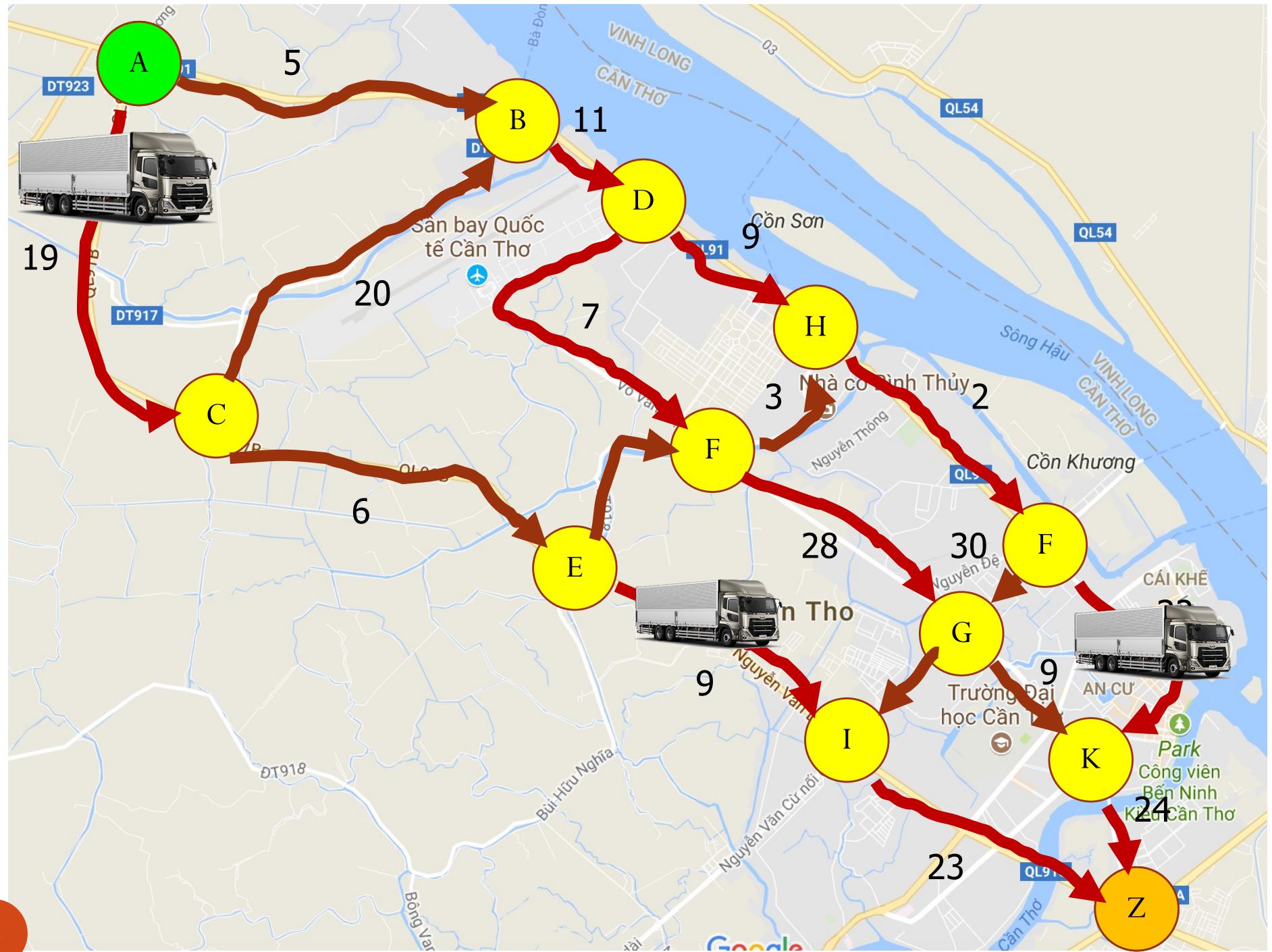
# Nội dung

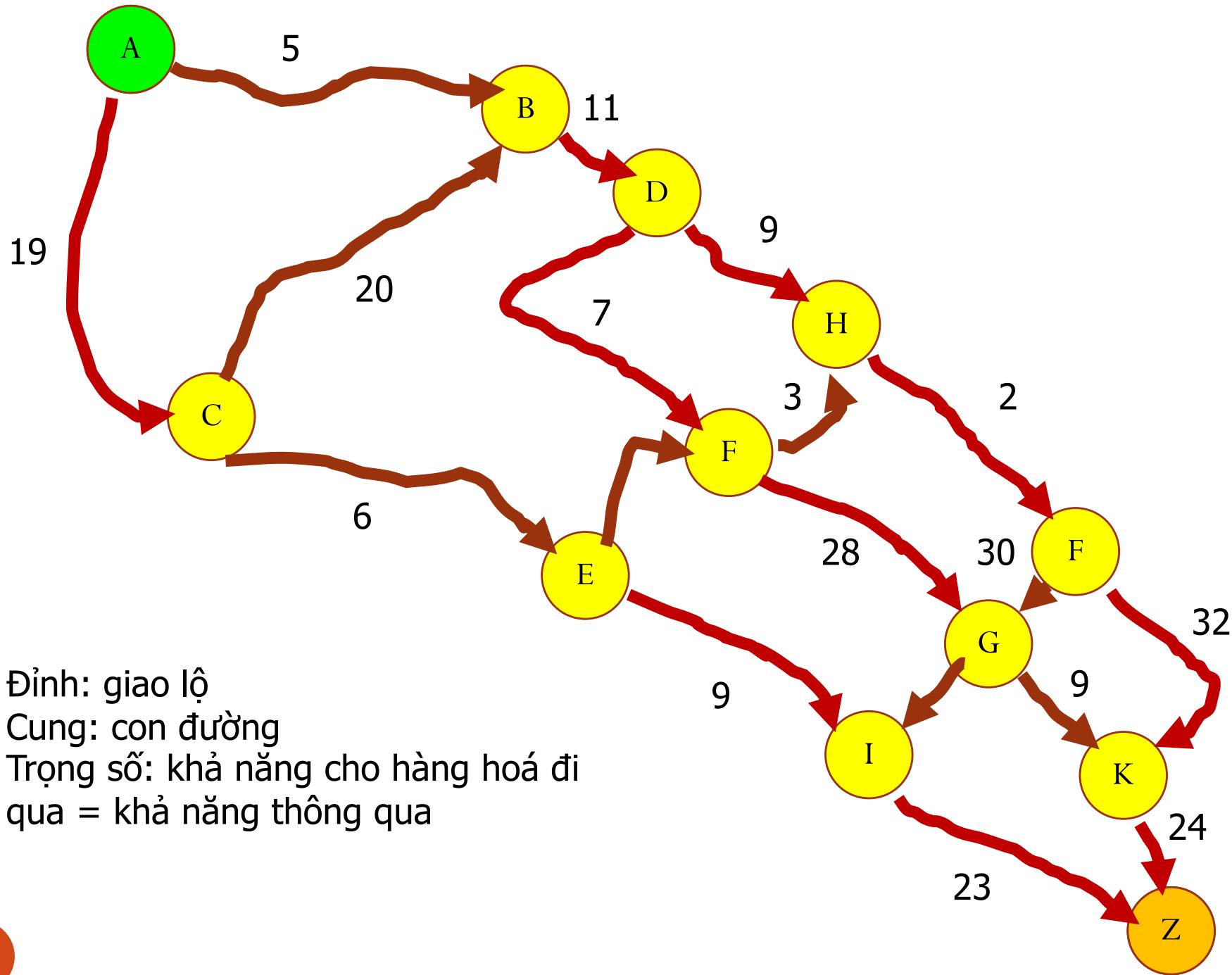
- Bài toán lưu thông hàng hóa
- Mạng (các luồng) & luồng trong mạng
- Luồng cực đại
- Lát cắt
- Thuật toán/Phương pháp Ford – Fulkerson
  - Thuật toán Edmonds – Karp











# Mạng (các luồng)

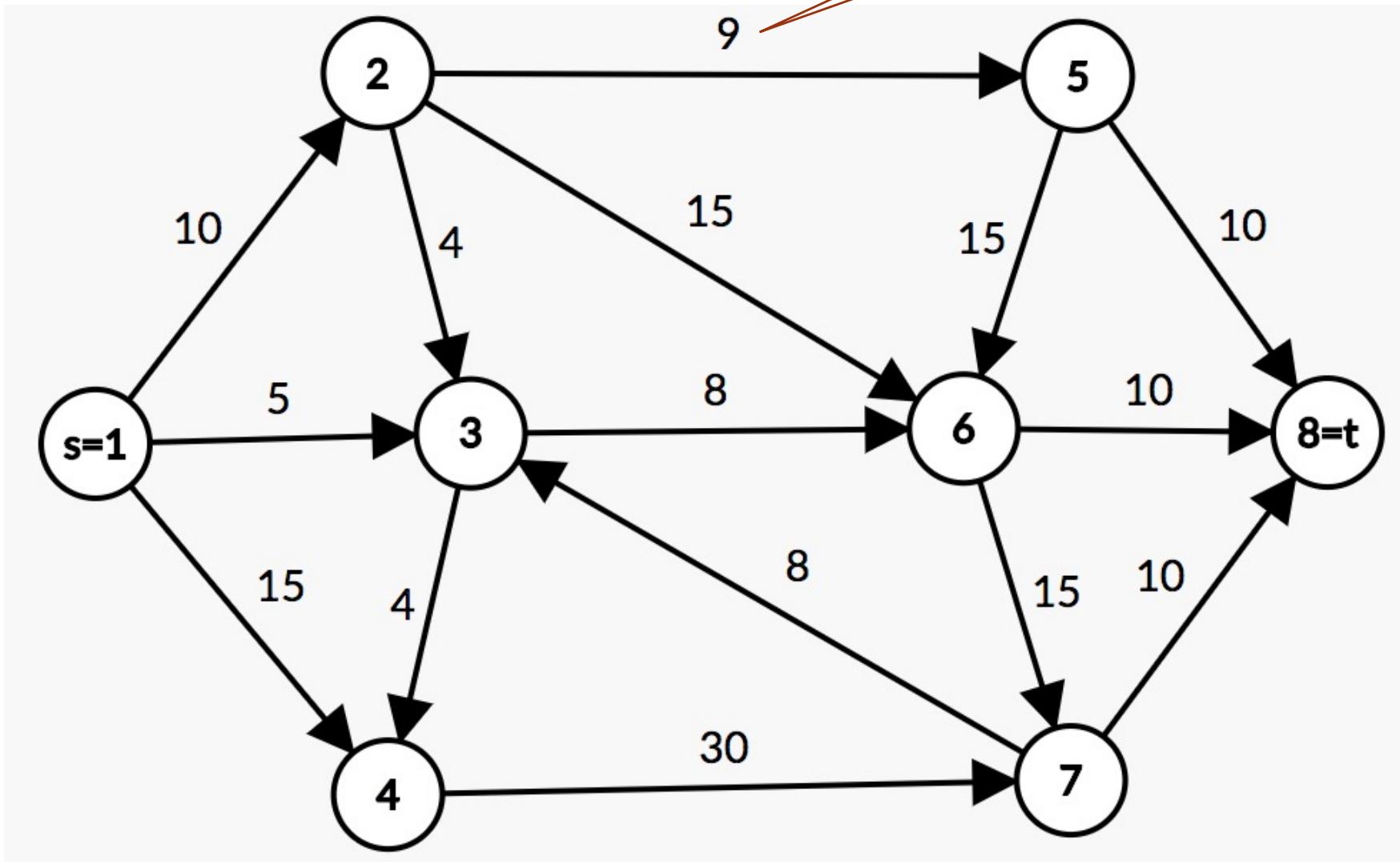
- Network (of flows)/Flow network
  - Mạng là 1 bộ 5:  $N = \langle V, E, s, t, c \rangle$  trong đó
    - $\langle V, E \rangle$  là đồ thị có hướng biểu diễn cho mạng
      - V: đỉnh/nút
      - E: cung
    - s: đỉnh phát (source)
    - t: đỉnh thu (sink)
    - c:  $(u,v) \rightarrow c(u,v)$  hàm mô tả khả năng thông qua của 1 cung
      - $c(u,v)$ : luồng lớn nhất có thể đi qua cung  $(u, v)$
- Ví dụ: mạng giao thông, trong đó các luồng giao thông (xe) chạy trên mạng.

# Luồng (trên mạng)

- (Network) Flow: **thứ lưu thông trên mạng**
  - Một luồng đi từ đỉnh phát ( $s$ ) đến đỉnh thu ( $t$ ) là 1 hàm  $f: (u, v) \rightarrow f(u, v)$
  - thoả mãn các điều kiện:
    - Với mỗi cung:  $0 \leq f(u, v) \leq c(u, v)$
    - Với mỗi đỉnh khác  $s$  và  $t$ : tổng luồng vào = tổng luồng ra
    - Tổng luồng ra khỏi  $s$  = tổng luồng vào  $t$
- **Giá trị luồng  $|f| =$  tổng luồng ra khỏi  $s$**
- Mẹo: luồng  $s-t$  là cách **gán các giá trị luồng cho từng cung** của mạng sao cho **thoả mãn điều kiện luồng**.

# Luồng (trên mạng)

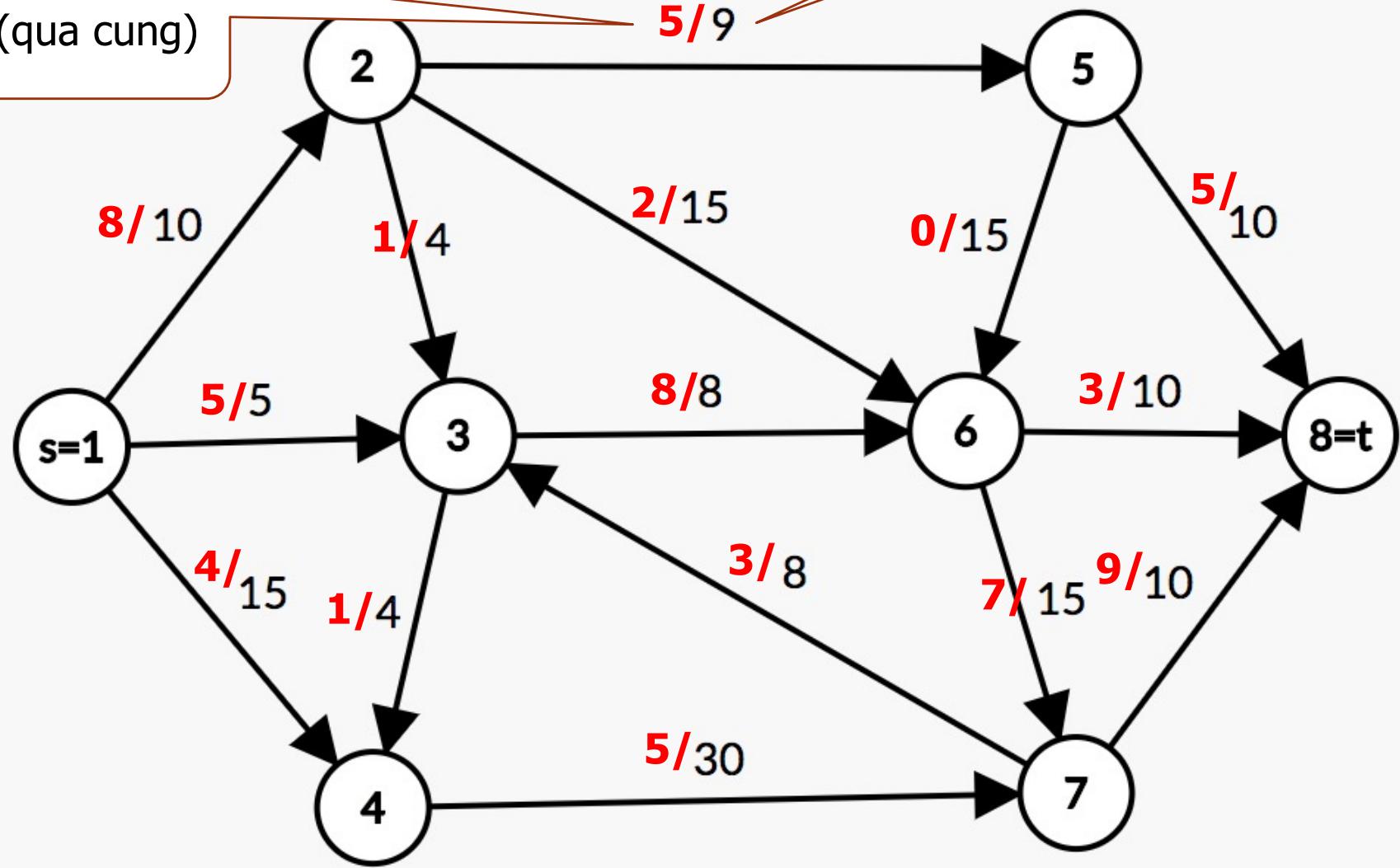
Khả năng thông qua của cung



# Luồng (trên mạng)

Khả năng thông qua của cung

Luồng (qua cung)



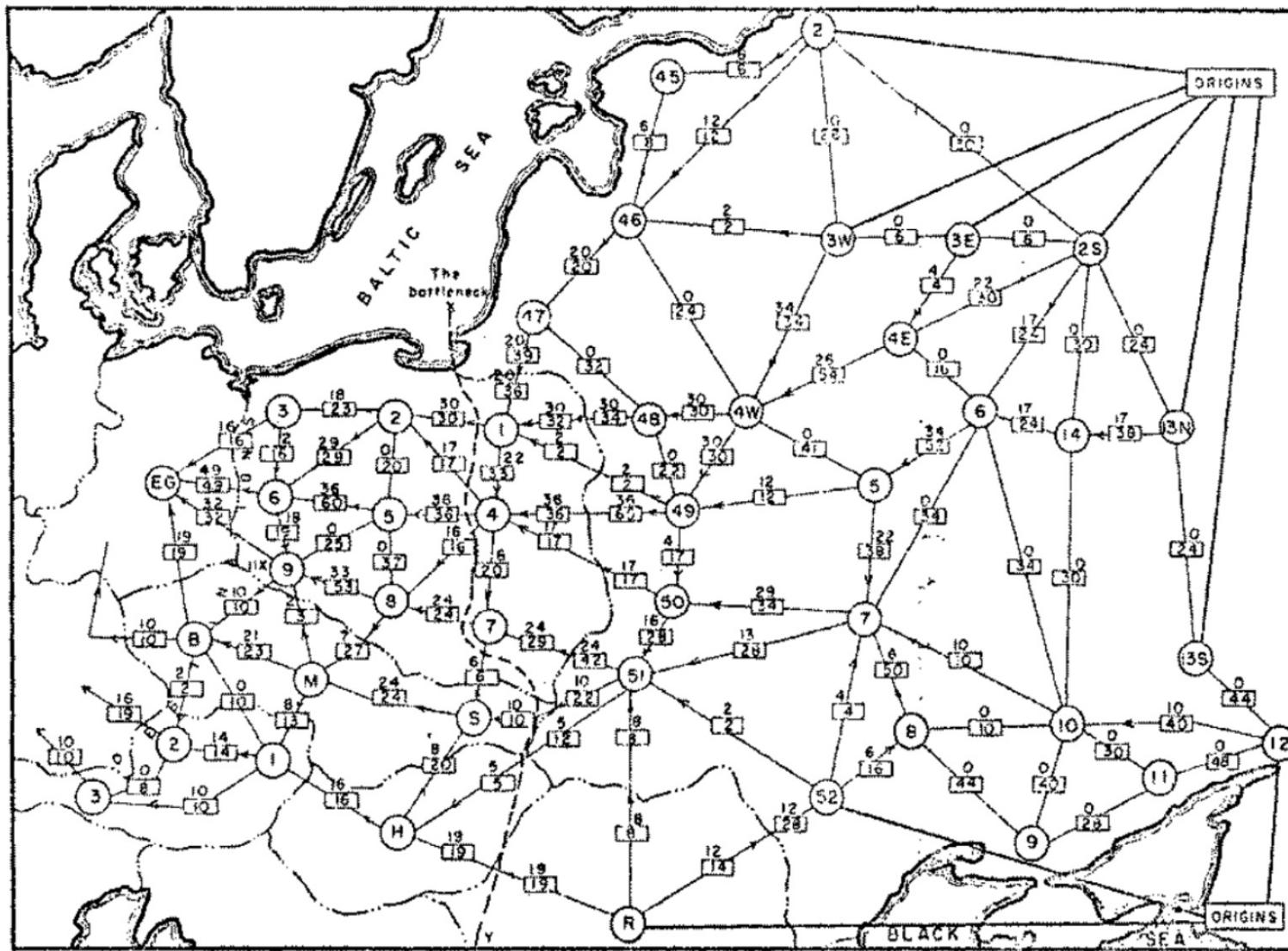
# Luồng (trên mạng)

# Luồng (trên mạng)

# Luồng trên mạng (Ứng dụng)

Mạng	Định/nút	Cung	Luồng
Truyền thông	Tổng đài điện thoại, máy tính, vệ tinh	Cáp, cáp quang, wifi	Âm thanh, video, gói tin
Mạch điện	Cổng, thanh ghi, vi xử lý	Dây dẫn	Dòng điện
Thuỷ lực	Trạm bơm, bể chứa, hồ	Ống dẫn	Dầu
Tài chính	Chứng khoán, tiền tệ	Giao dịch	Tiền
Giao thông	Sân bay, nhà ga, giao lộ	Con đường	Xe, hành khách
Hoá học	Nguyên tử	Liên kết hoá học	Năng lượng

# Mạng đường sắt vận chuyển hàng hoá từ Liên Xô (cũ) sang Đông Âu



# Luồng cực đại

- Cho mạng  $N = \langle V, E, s, t, c \rangle$ , tìm luồng  $f$  (từ  $s$  đến  $t$ ) có **giá trị luồng  $|f|$  lớn nhất**
- Bài tập:
  - Hãy tìm 1 luồng khác có giá trị lớn hơn luồng vừa rồi (17)

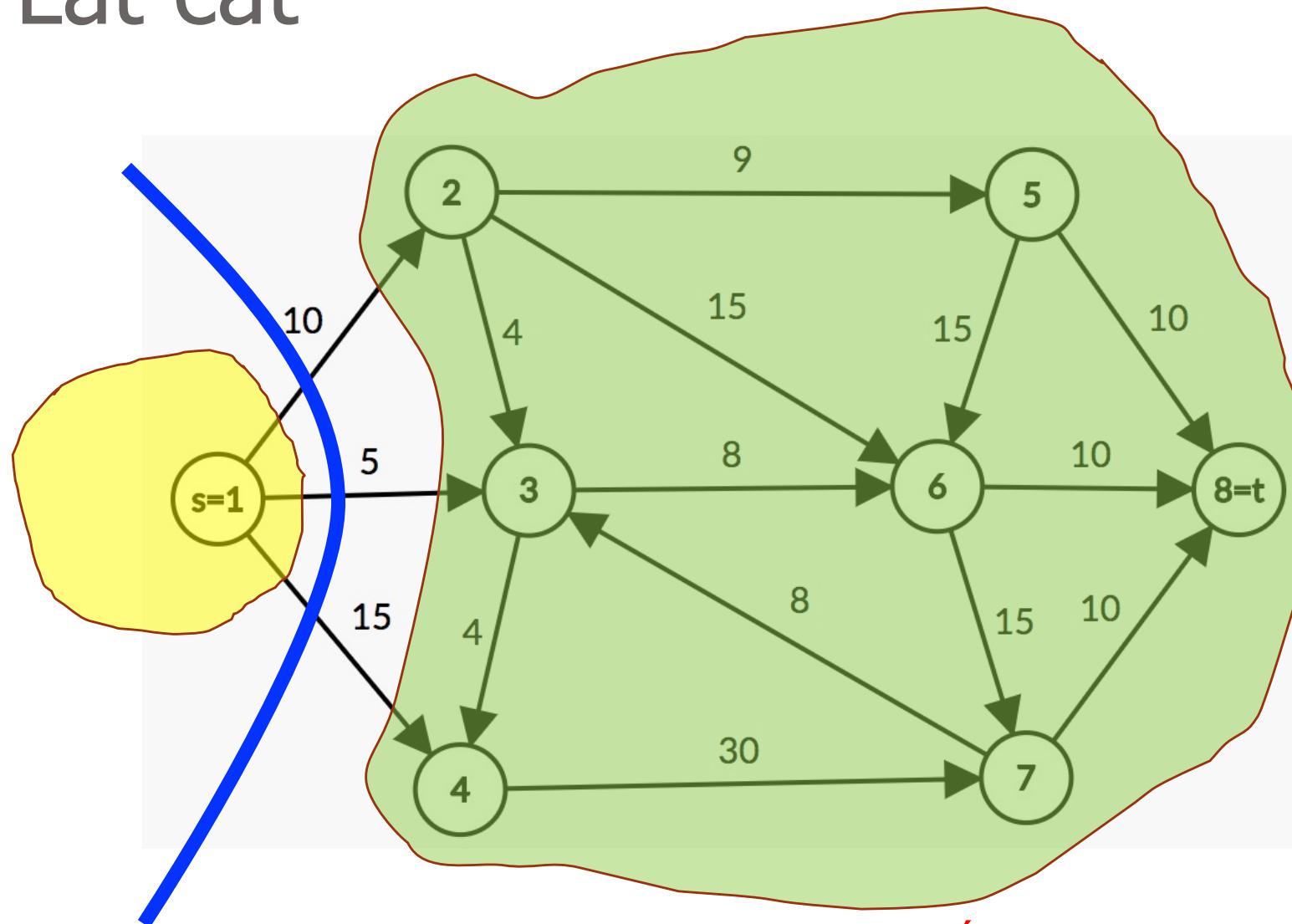
# Lát cắt

- Một lát cắt cắt tách s và t, (gọi là:  $s-t$  cut)
  - Là một cách chia (phân hoạch) tập đỉnh V thành hai phần rời nhau ( $S, T$ ) sao cho s nằm trong  $S$  và t nằm trong  $T$
- Các cung của một lát cắt (tách s và t)
  - Tập các cung xuất phát từ  $S$  đi đến  $T$ 
$$\{(u, v) \mid u \in S \text{ và } v \in T\}$$
- Khả năng thông qua của 1 lát cắt (tách s và t)

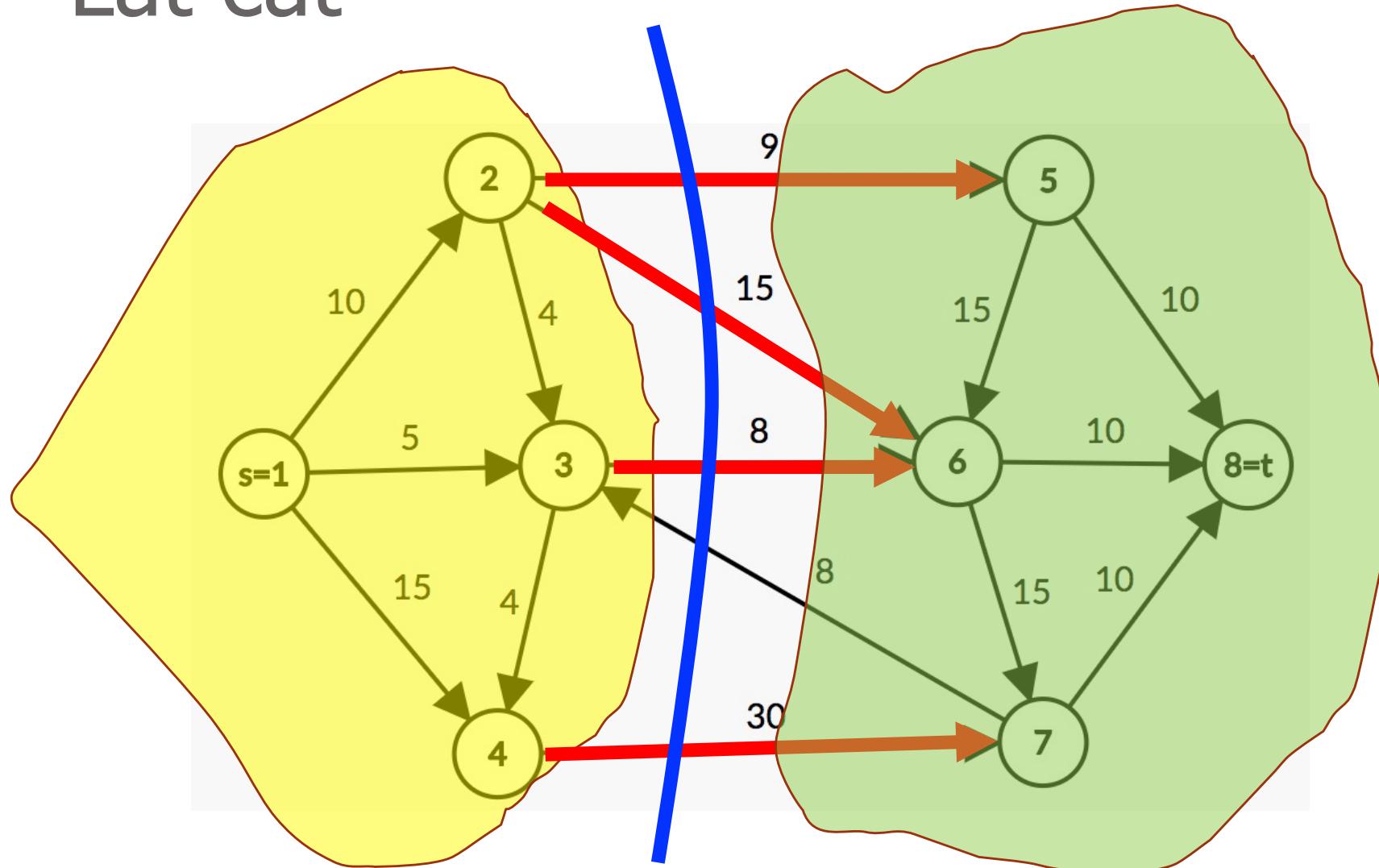
$$c(S, T) = \sum_{u \in S, v \in T} c(u, v)$$

- Tổng khả năng thông qua của các cung của lát cắt

# Lát cắt

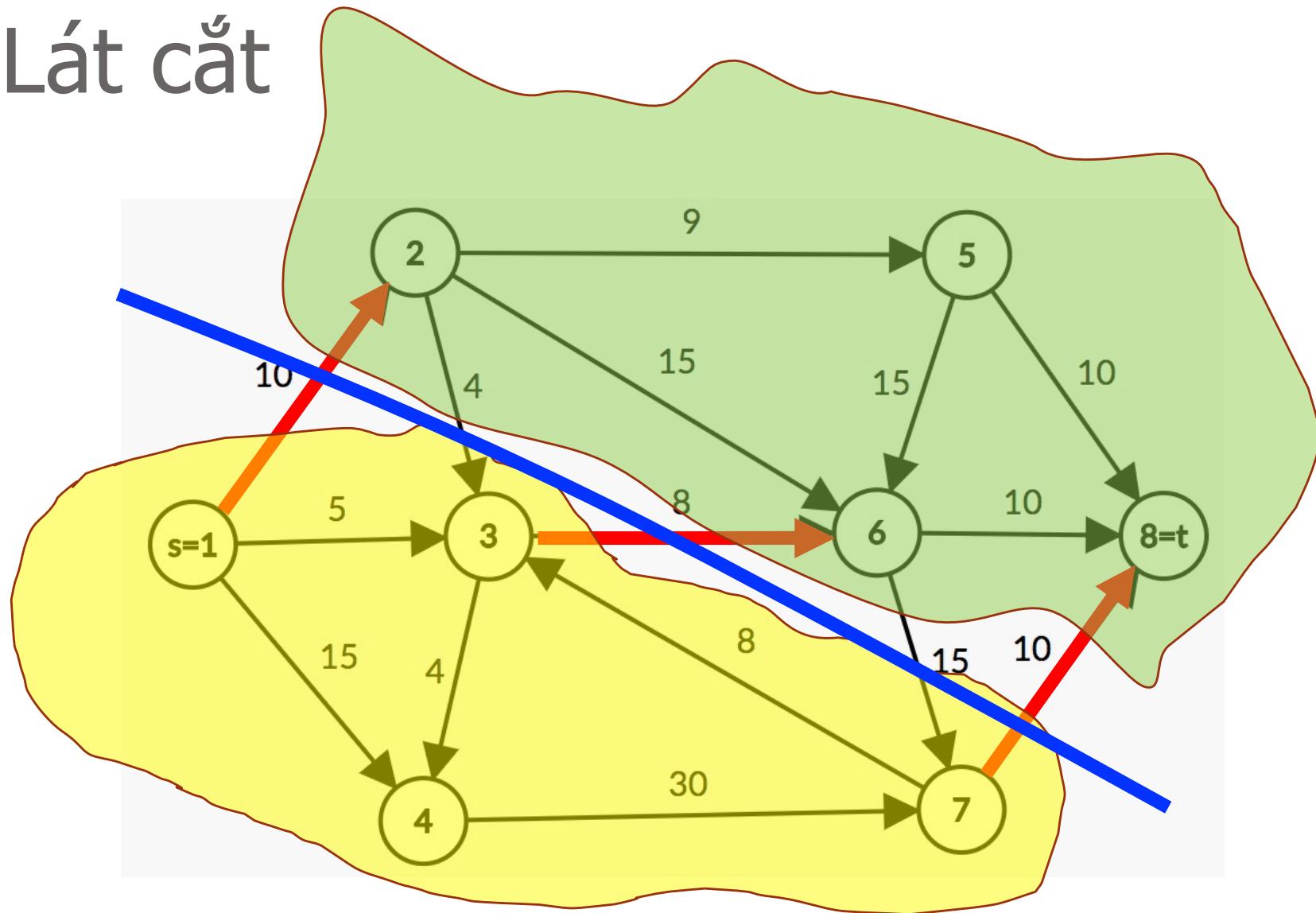


# Lát cắt



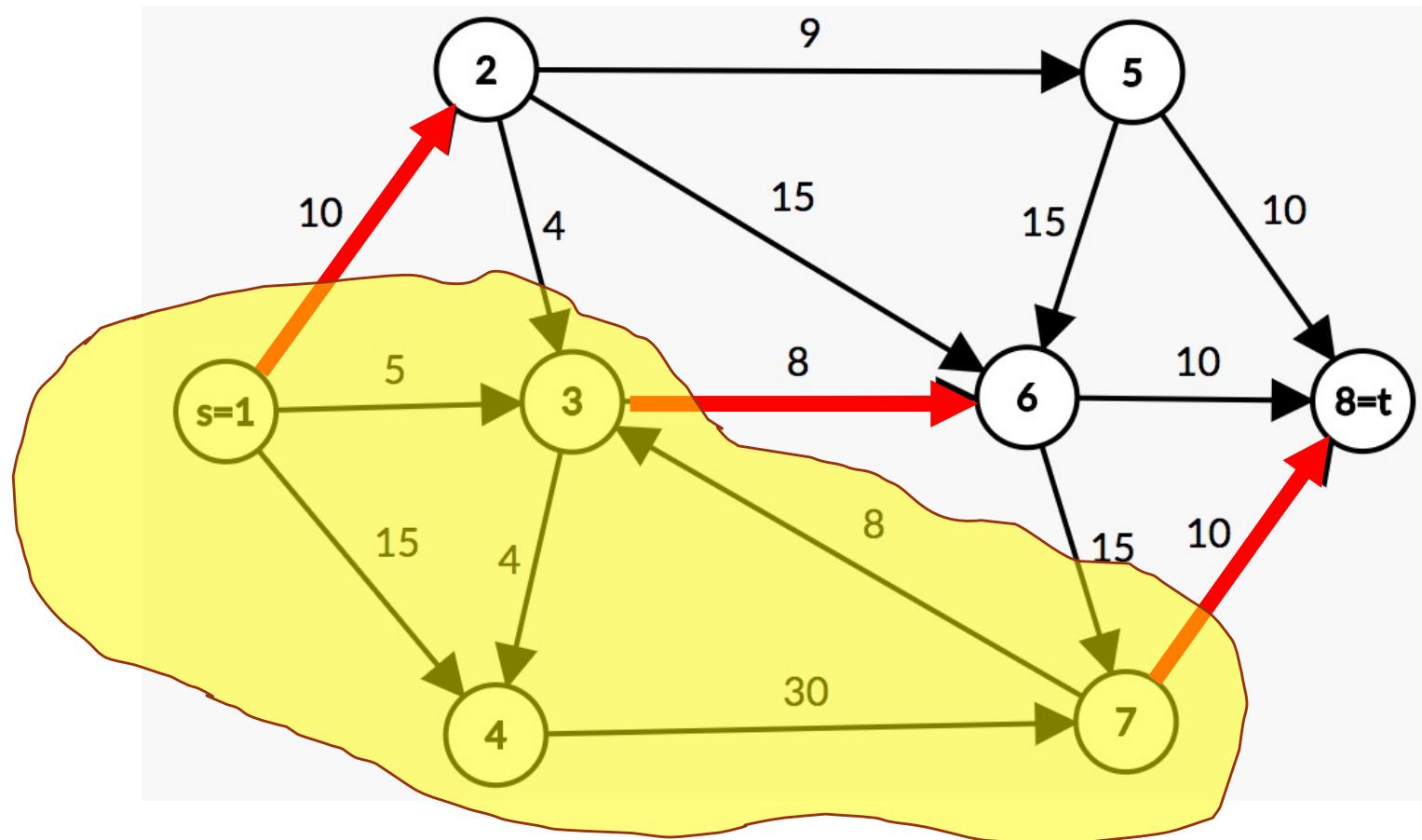
**Khả năng thông qua của lát cắt =  $9+15+8+30 = 62$**

# Lát cắt



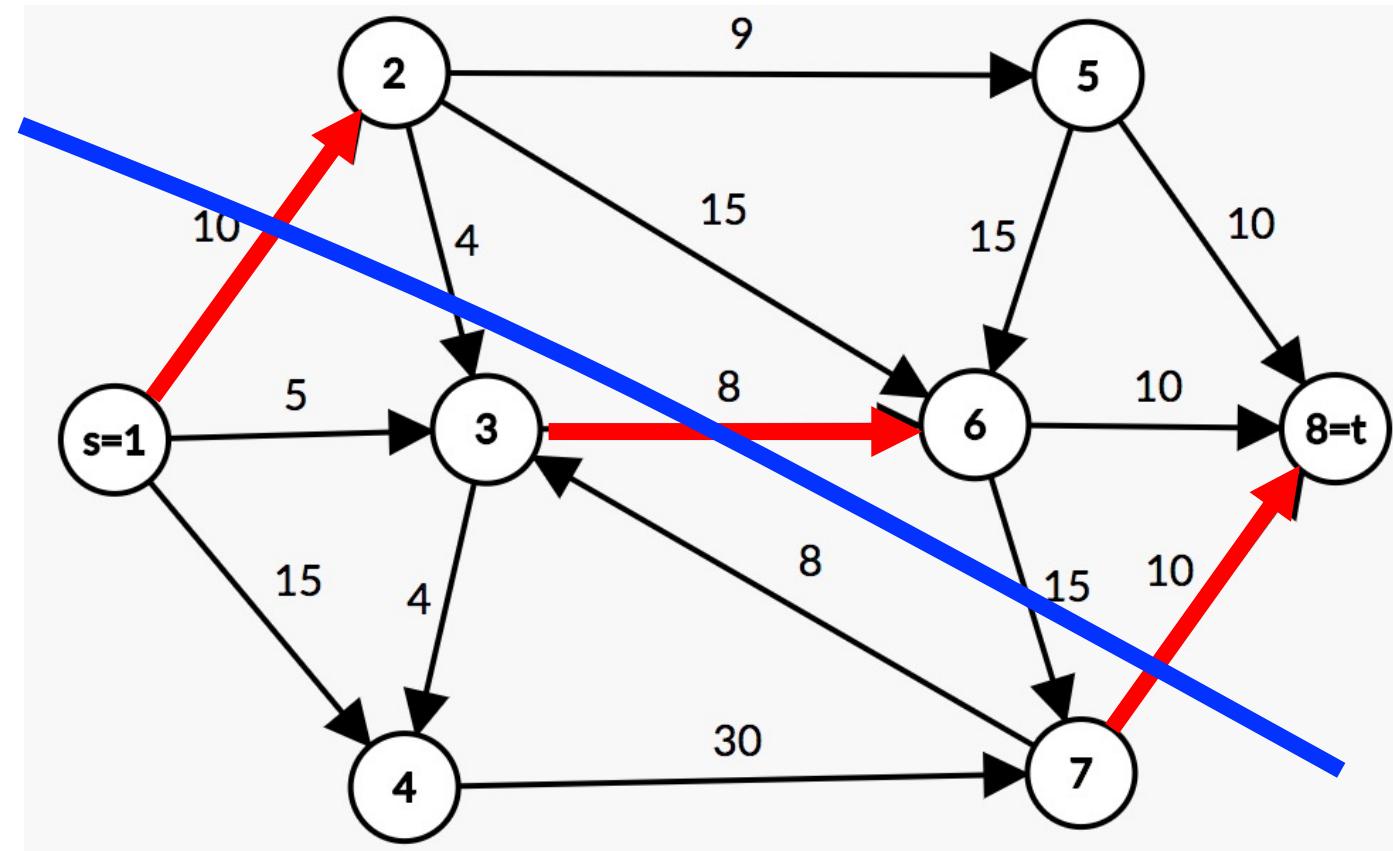
**Khả năng thông qua của lát cắt =  $10+8+10 = 28$**

# Lát cắt



**Khả năng thông qua của lát cắt =  $10+8+10 = 28$**

# Lát cắt



**Khả năng thông qua của lát cắt =  $10+8+10 = 28$**

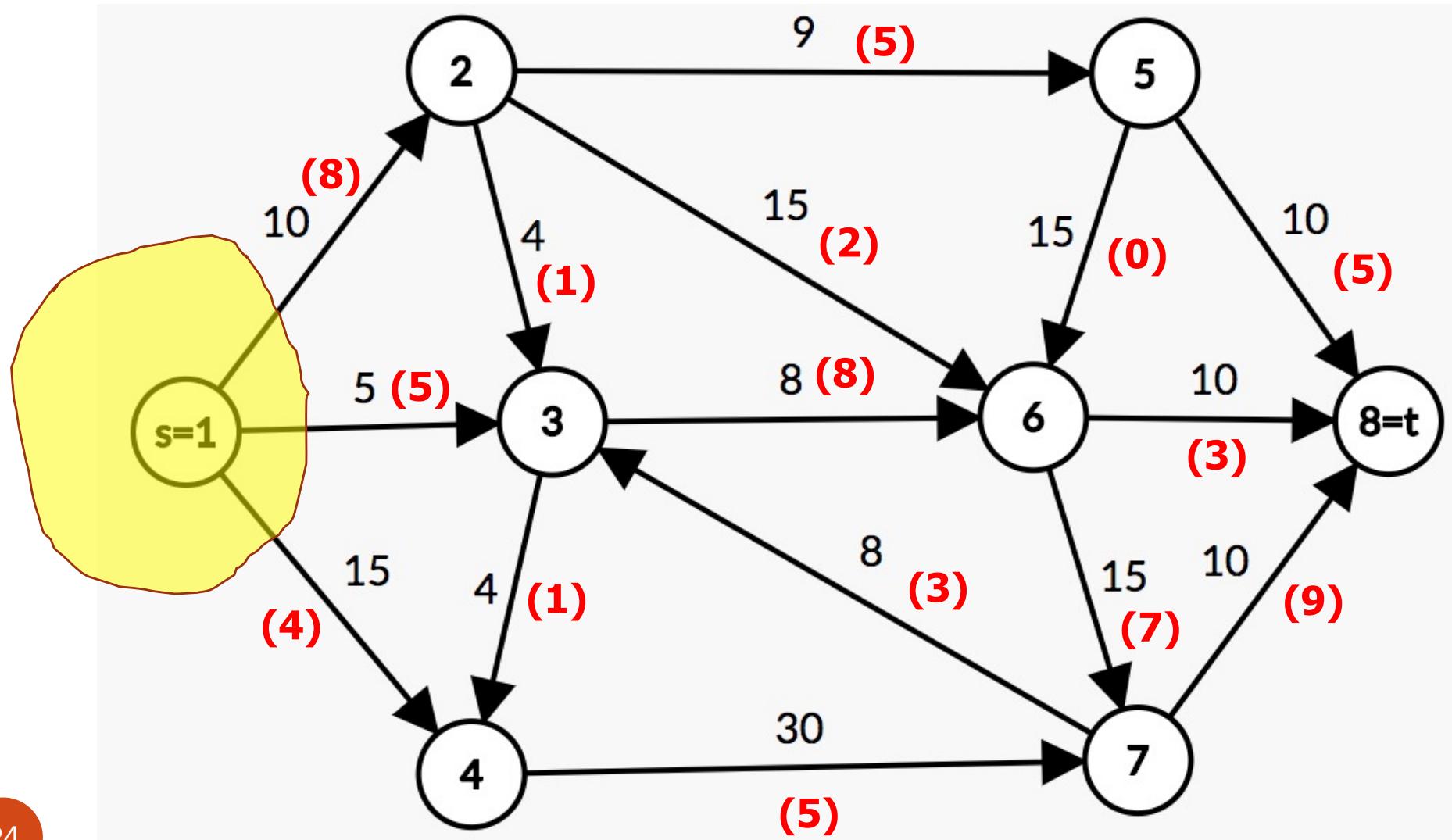
# Luồng và lát cắt

- **Bổ đề 1**
- Nếu gọi
  - $f$  là luồng đi qua mạng  $N$
  - $(S, T)$  là 1 lát cắt tách  $s$  và  $t$
- thì
  - Luồng đi qua lát cắt = luồng ra khỏi  $s$   
= luồng đi vào  $t$
  - Tổng luồng đi ra khỏi  $S$  - tổng luồng vào  $S$  = Giá trị luồng

$$\sum_{e \text{ out of } S} f(e) - \sum_{e \text{ in to } S} f(e) = \sum_{e \text{ out of } s} f(e) = |f|$$

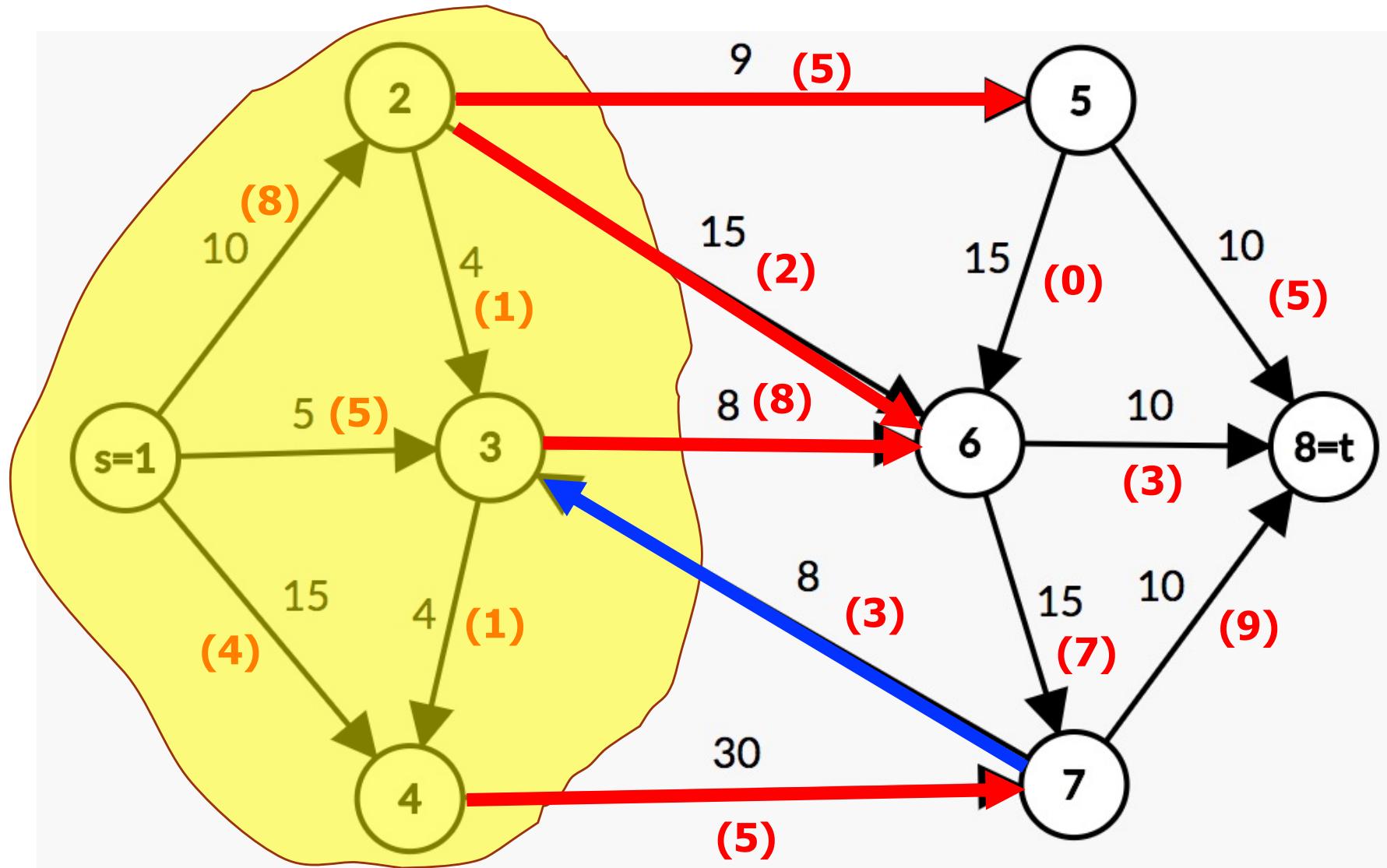
# Luồng và lát cắt

$$17 - 0 = 17$$



# Luồng và lát cắt

$$20 - 3 = 17$$



# Luồng và lát cắt

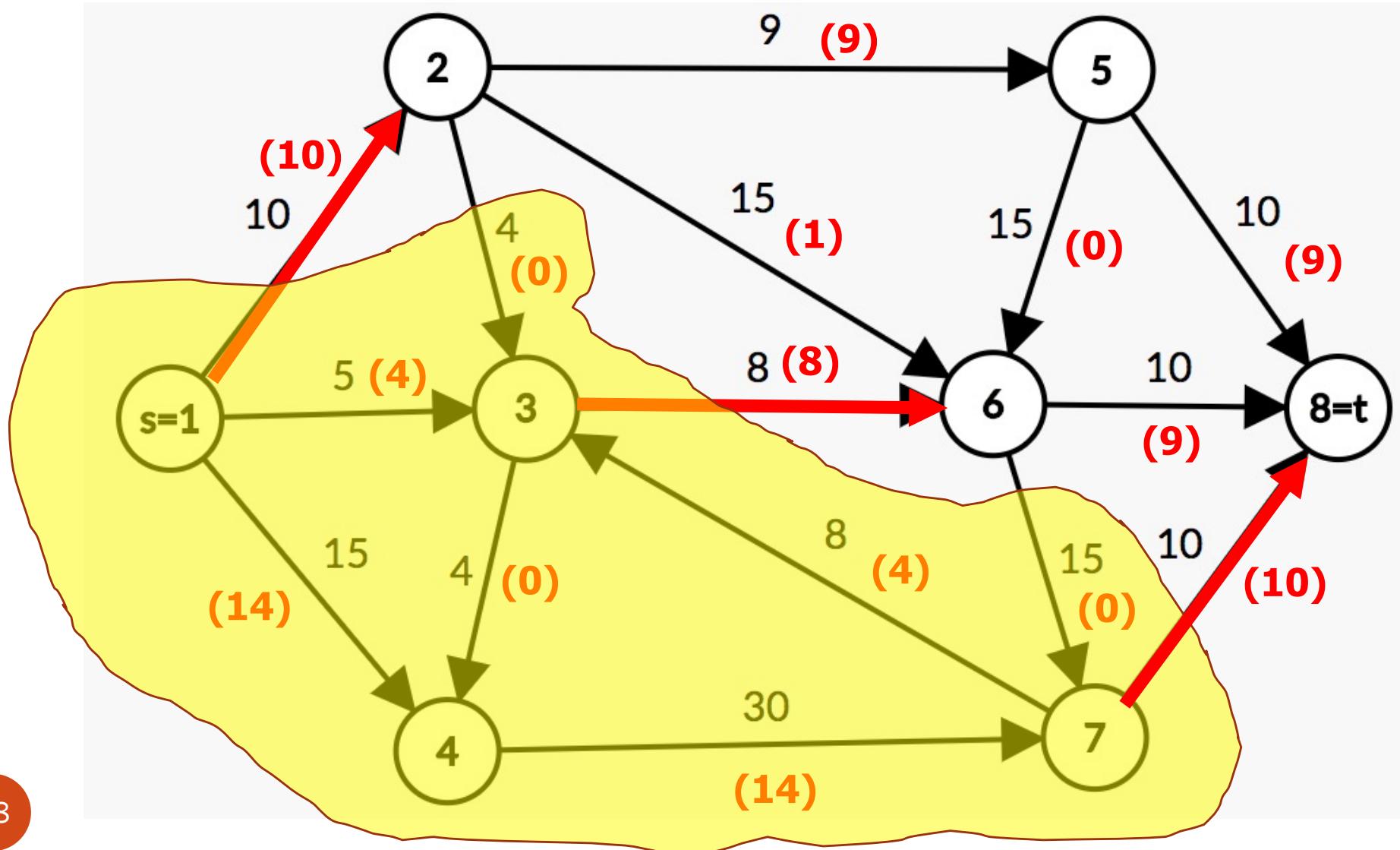
- **Bổ đề 2**
- Nếu gọi:
  - $f$  là luồng đi qua mạng  $N$
  - $(S, T)$  là 1 lát cắt tách  $s$  và  $t$
- thì
  - Giá trị luồng không vượt quá **khả năng thông qua** của lát cắt

$$|f| \leq c(S, T)$$

# Luồng và lát cắt

- **Hệ quả**
- **Gọi:**
  - $f$  là luồng đi qua mạng  $N$
  - $(S, T)$  là 1 lát cắt tách  $s$  và  $t$
- Nếu  $|f| = c(S, T)$  thì  $f$  là **luồng lớn nhất (luồng cực đại)** và  $(S, T)$  là **lát cắt hẹp nhất.**

# Luồng và lát cắt



# Luồng và lát cắt

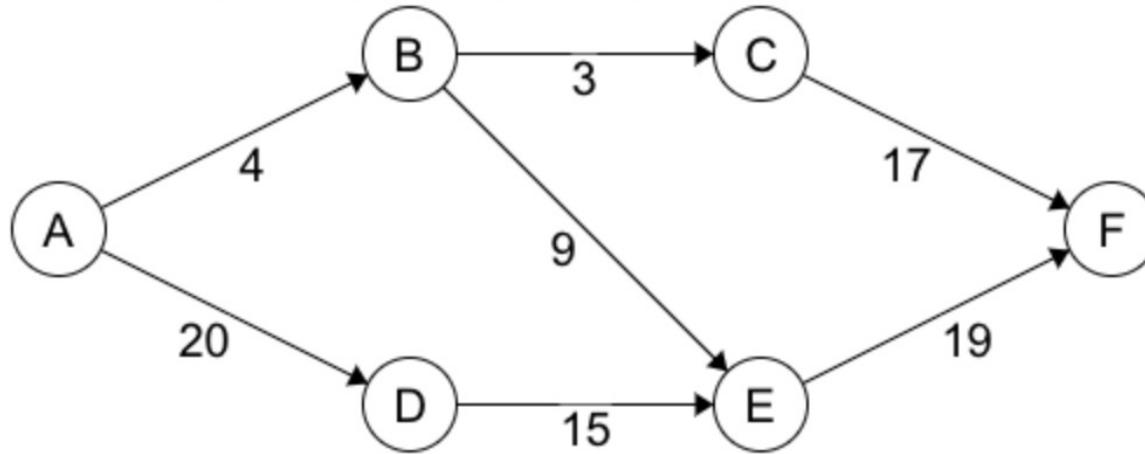
- Định lý luồng cực đại lát cắt hẹp nhất (Ford-Fulkerson, 1956)
  - Giá trị luồng cực đại = khả năng thông qua của lát cắt hẹp nhất.

# Phương pháp Ford-Fulkerson

- Ý tưởng: **tìm lát cắt hẹp nhất bằng cách tăng luồng đi qua mạng**
  - Khởi 1 luồng hợp lệ (thường là luồng có giá trị 0)
  - Tìm cách tăng luồng đi qua mạng cho đến khi bị ngẽn => lát cắt hẹp nhất => giá trị luồng cực đại.
- Tìm **đường tăng luồng** (augmenting path):
  - Tìm đường đi từ s đến t sao cho có thể tăng thêm luồng trên đường đi đó.

## Đồ thị biểu diễn mạng (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

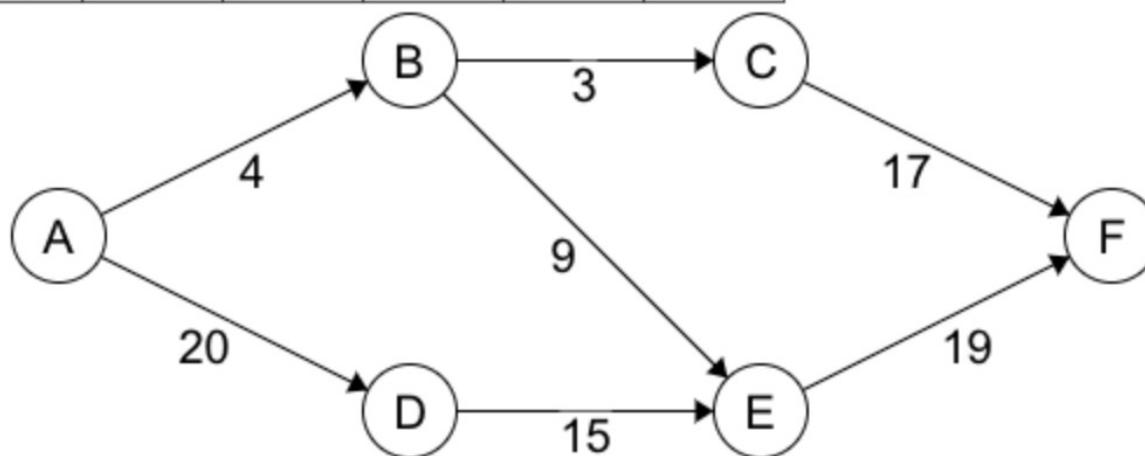
Help Clear shift Delete Edit Undo



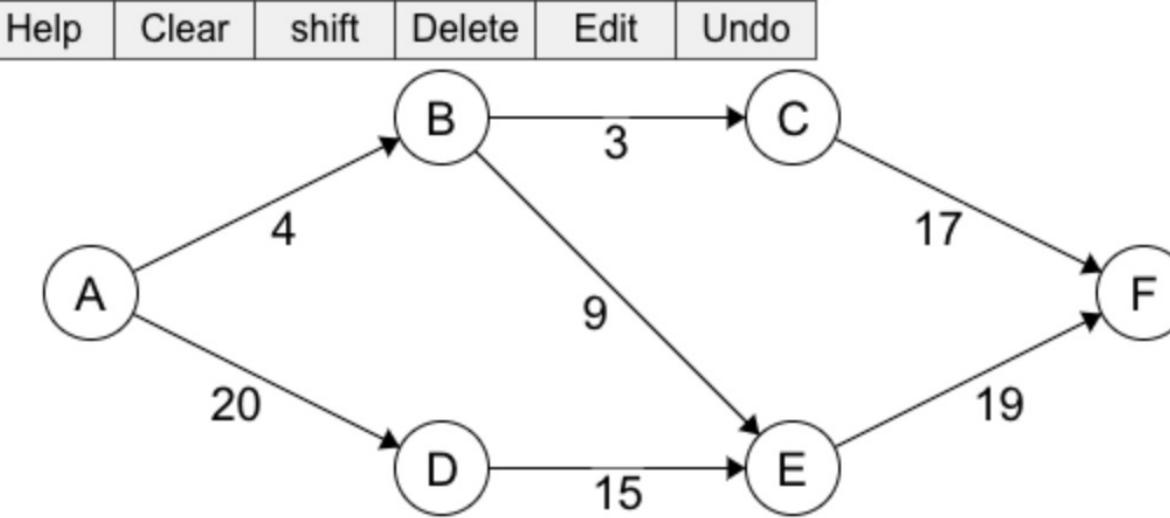
Áp dụng phương pháp Ford-Fulkerson tìm luồng cực đại trong mạng trên.

I. Khởi tạo một luồng hợp lệ bất kỳ có giá trị không vượt quá 3

Help Clear shift Delete Edit Undo



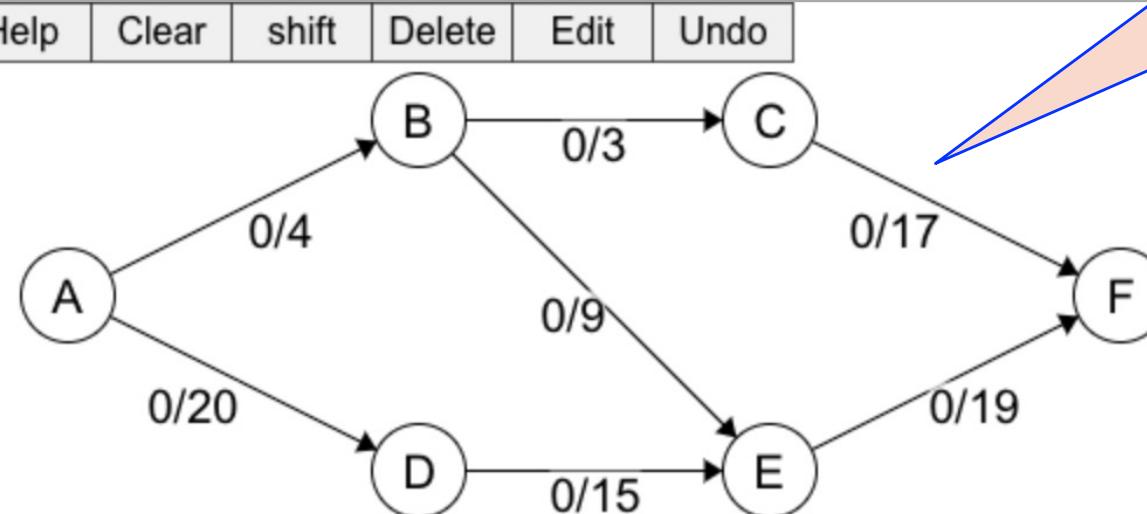
## Đồ thị biểu diễn mạng (Dùng chuột để thay đổi vị trí của các đỉnh/cung)



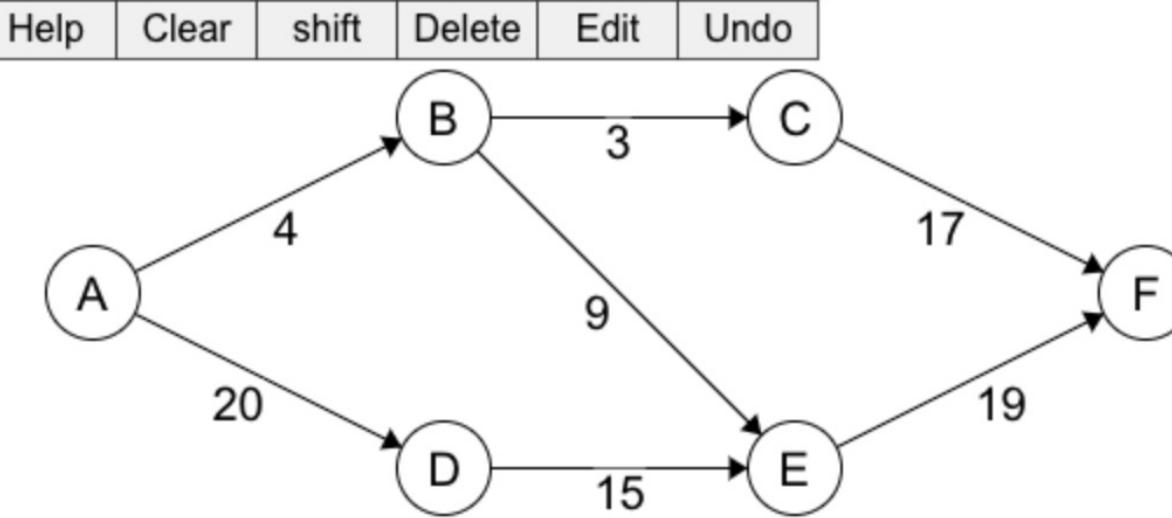
Áp dụng phương pháp Ford-Fulkerson tìm luồng cực đại trong mạng trên.

I. Khởi tạo một luồng hợp lệ bất kỳ có giá trị không vượt quá 3

Luồng có giá trị  
bằng 0 (trivial flow)



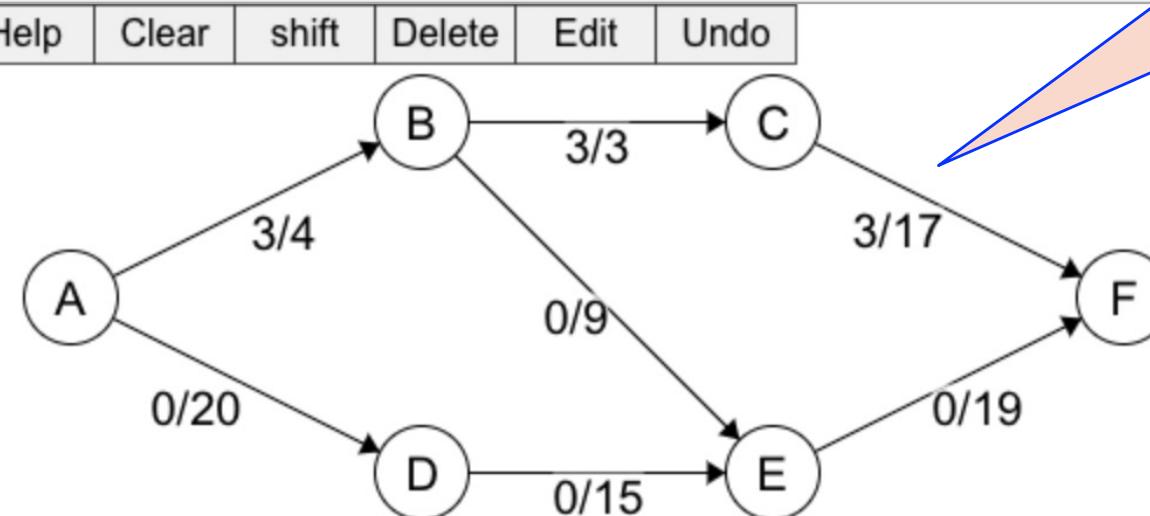
## Đồ thị biểu diễn mạng (Dùng chuột để thay đổi vị trí của các đỉnh/cung)



Áp dụng phương pháp Ford-Fulkerson tìm luồng cực đại trong mạng trên.

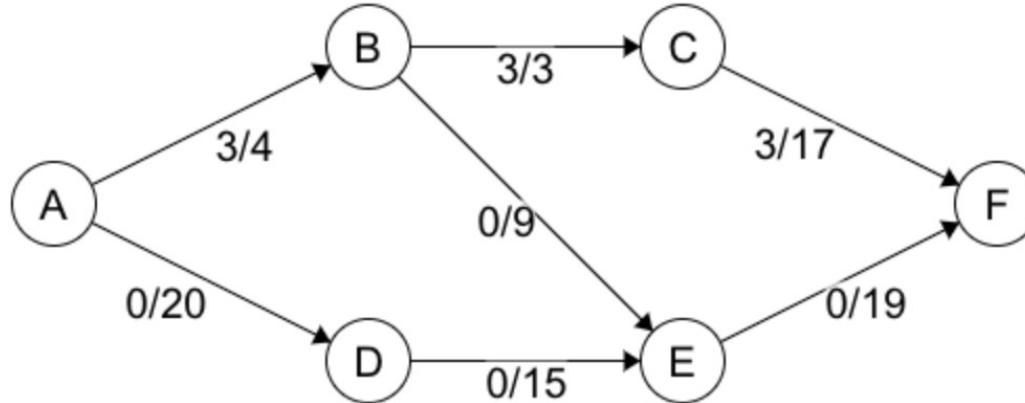
I. Khởi tạo một luồng hợp lệ bất kỳ có giá trị không vượt quá 3

Luồng có giá trị  
bằng 3



### I. Khởi tạo một luồng hợp lệ bất kỳ có giá trị không vượt quá 3

Help Clear shift Delete Edit Undo



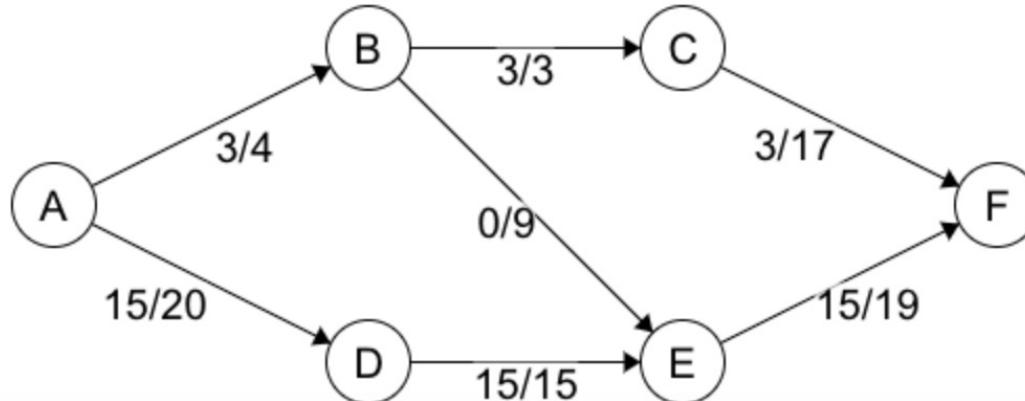
Tìm đường tăng luồng & lượng luồng tăng thêm

### II. Lặp tìm đường tăng luồng + tăng luồng

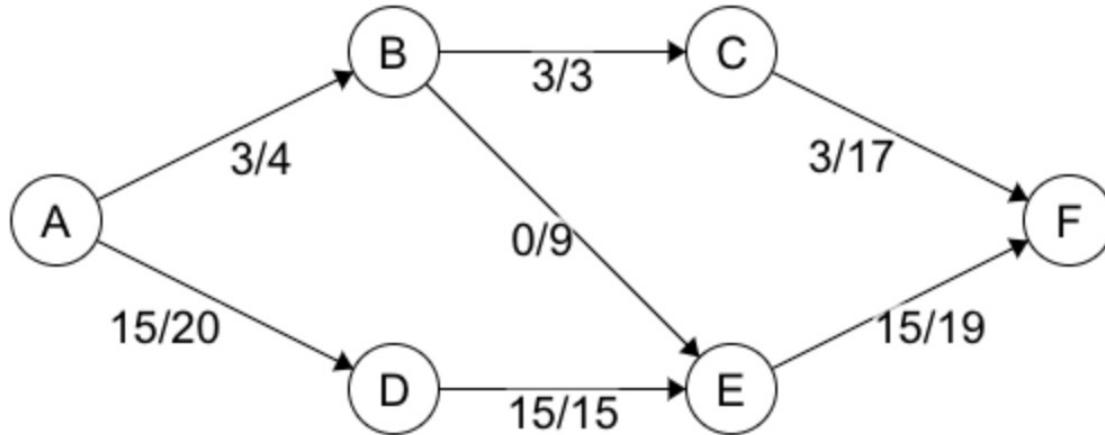
#### Lần lặp 1

- Tìm một đường tăng luồng (augmenting path) bất kỳ:  ví dụ: A -> C -> F
- Lượng luồng tăng thêm (bottle neck capacity):
- Mạng sau khi tăng luồng  Copy mạng ở bước trên

Help Clear shift Delete Edit Undo



Help Clear shift Delete Edit Undo

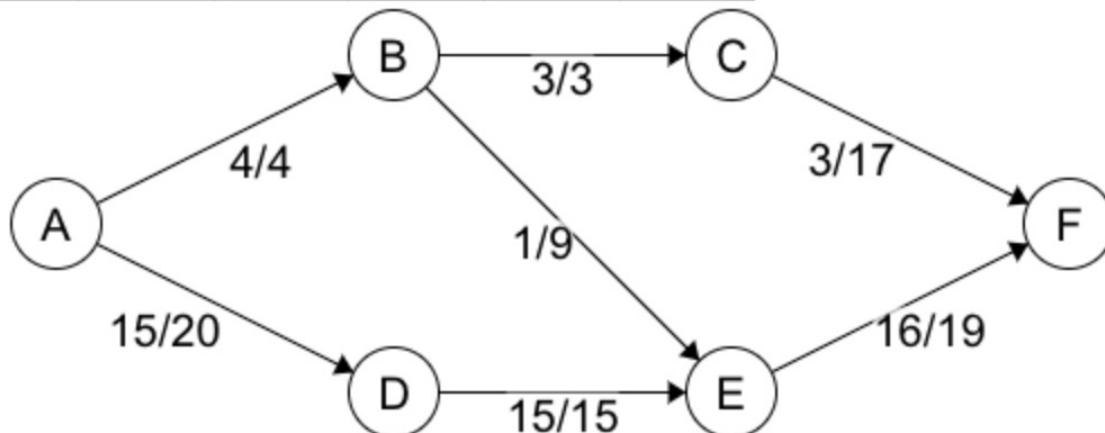


Tìm đường tăng luồng & lượng luồng tăng thêm (1 lần nữa)

### Lần lặp 2

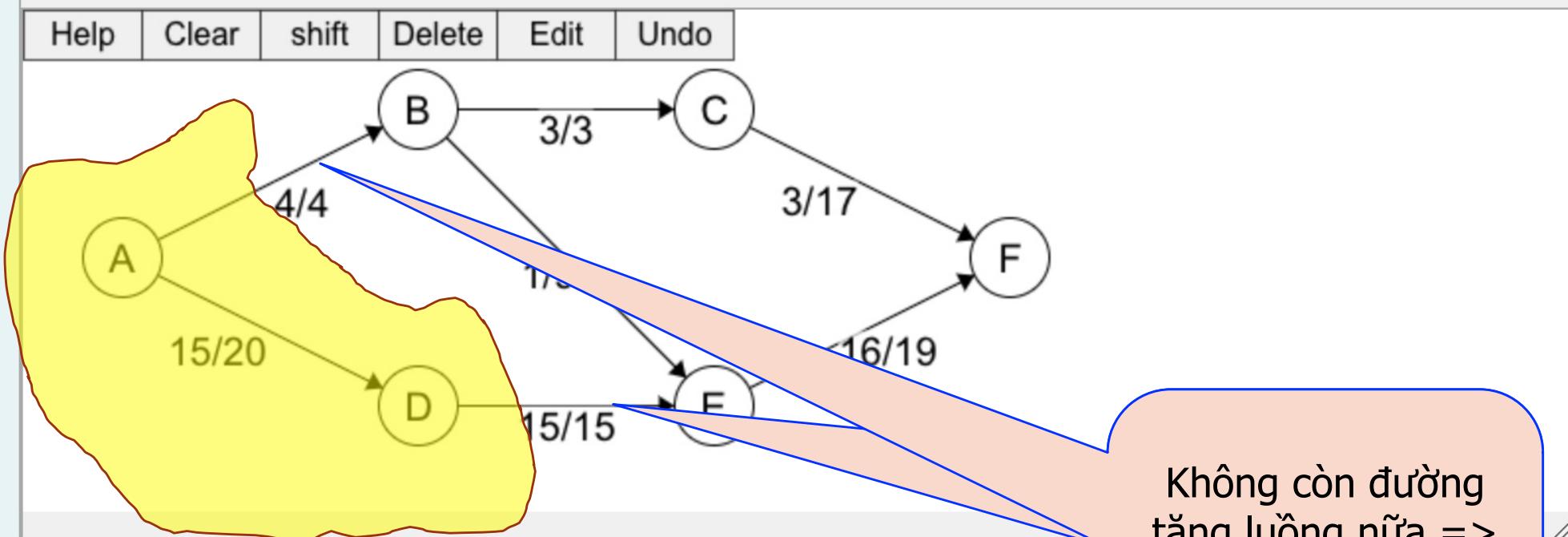
- Tìm một đường tăng luồng (augmenting path) bất kỳ: A -> B -> E -> F ví dụ: A -> C -> F
- Lượng luồng tăng thêm (bottle neck capacity): 1
- Mạng sau khi tăng luồng  Copy mạng ở bước trên

Help Clear shift Delete Edit Undo



## Lần lặp 2

- Tìm một đường tăng luồng (augmenting path) bất kỳ: A -> B -> E -> F ví dụ: A -> C -> F
- Lượng luồng tăng thêm (bottle neck capacity): 1
- Mạng sau khi tăng luồng  Copy mạng ở bước trên



### III. Kết quả

Luồng cực đại: 19

Lát cắt hẹp nhất tách s và t (ngăn cách các đỉnh bằng dấu phẩy), ví dụ A, C, D :

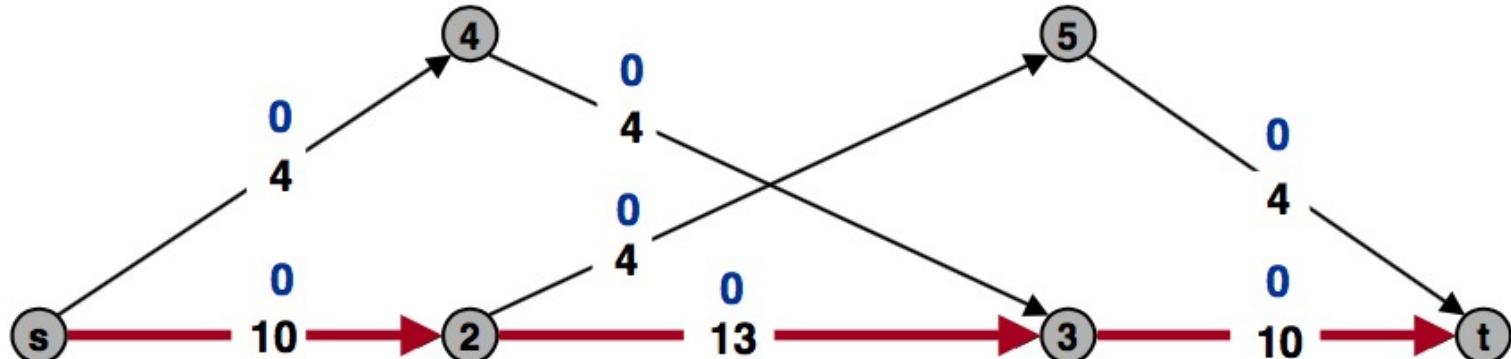
S = A,D

và T = B,C,E,F

Không còn đường  
tăng luồng nữa =>  
các cung nghẽn gần  
với s nhất tạo thành  
lát cắt hẹp nhất

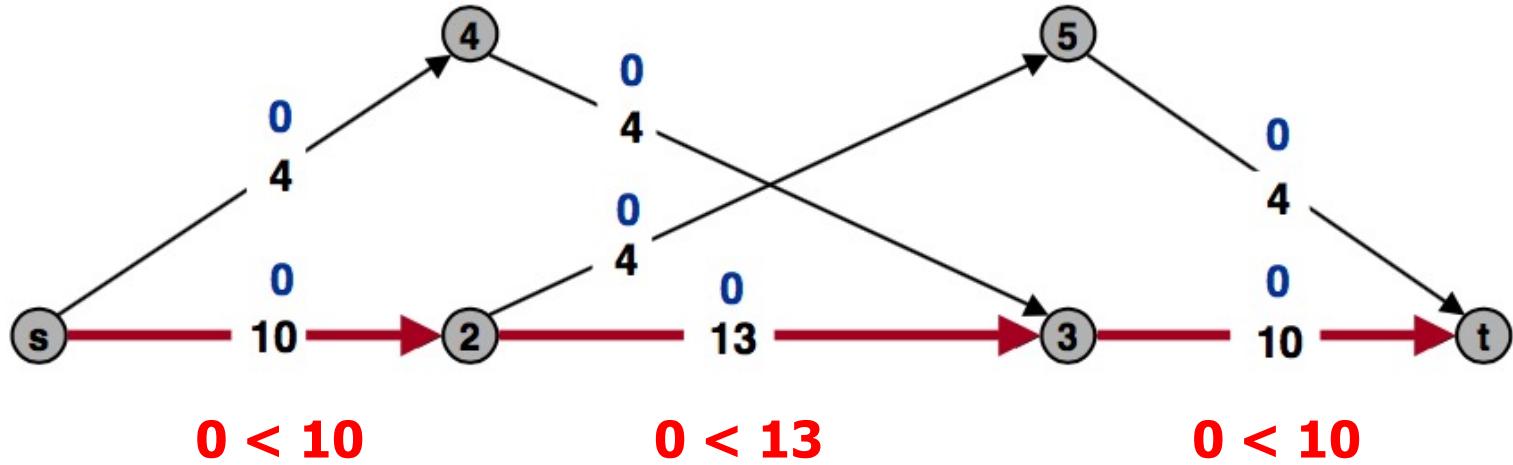
# Phương pháp Ford-Fulkerson

- Tìm đường tăng luồng:
  - Tìm đường đi từ s đến t sao cho có thể tăng thêm luồng trên đường đi đó.
  - **QS1: Tìm đường đi từ s đến t sao cho  $c(e) > f(e)$  với tất cả các cung trên đường đi**



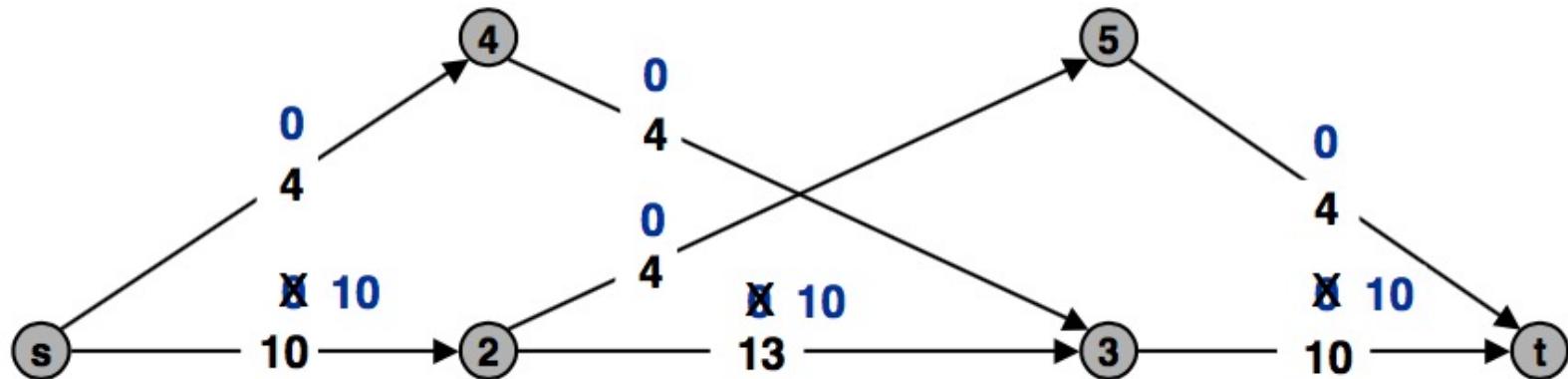
# Phương pháp Ford-Fulkerson

- Tìm đường tăng luồng:
  - Tìm đường đi từ s đến t sao cho có thể tăng thêm luồng trên đường đi đó.
  - **QS1: Tìm đường đi từ s đến t sao cho  $c(e) > f(e)$  với tất cả các cung trên đường đi**



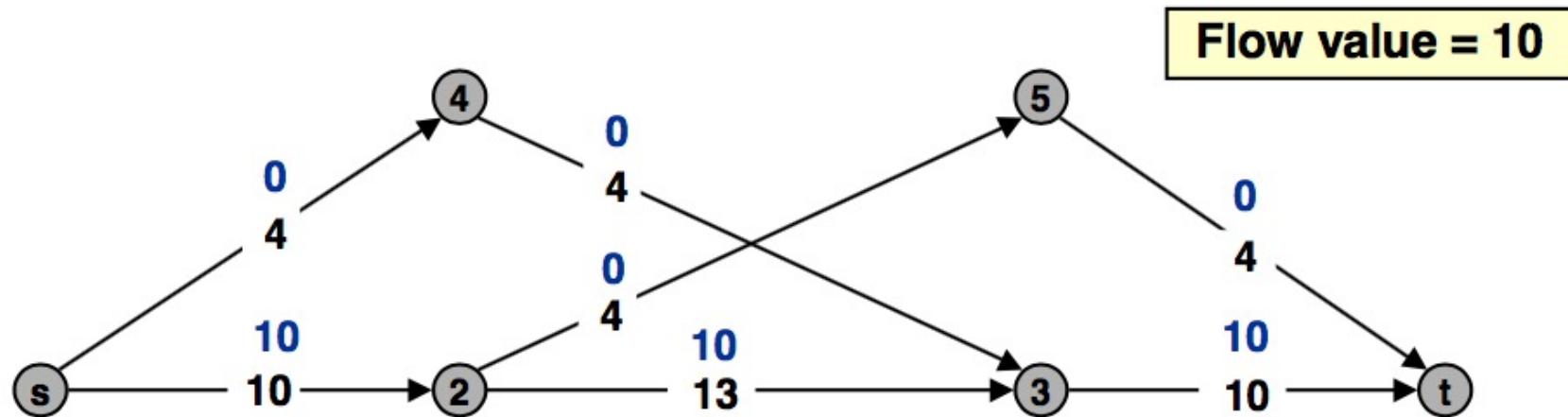
# Phương pháp Ford-Fulkerson

- Tìm đường tăng luồng:
  - Tìm đường đi từ s đến t sao cho có thể tăng thêm luồng trên đường đi đó.
  - **QS1: Tìm đường đi từ s đến t sao cho  $c(e) > f(e)$  với tất cả các cung trên đường đi**



# Phương pháp Ford-Fulkerson

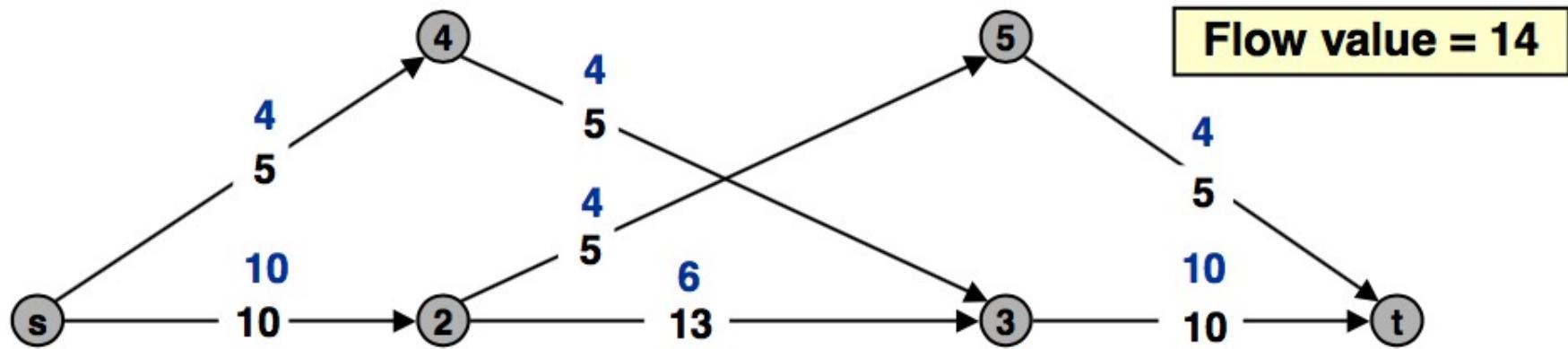
- Tìm đường tăng luồng:
  - Tìm đường đi từ s đến t sao cho có thể tăng thêm luồng trên đường đi đó.
  - **QS1: Tìm đường đi từ s đến t sao cho  $c(e) > f(e)$  với tất cả các cung trên đường đi**



**Không tìm được đường đi từ s đến t để tăng luồng !!!**

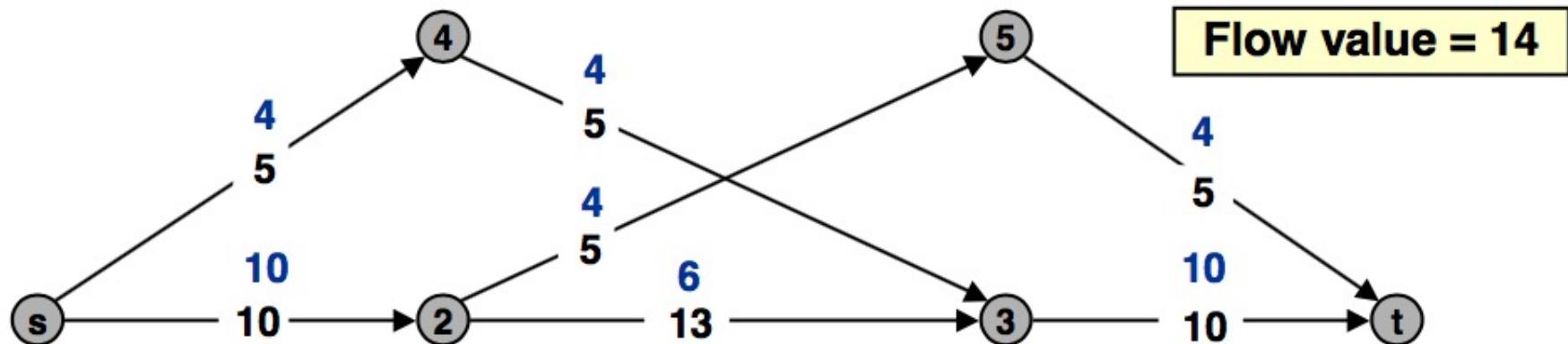
# Phương pháp Ford-Fulkerson

- Tìm đường tăng luồng:
  - Tìm đường đi từ s đến t sao cho có thể tăng thêm luồng trên đường đi đó.
  - **QS1: Tìm đường đi từ s đến t sao cho  $c(e) > f(e)$  với tất cả các cung trên đường đi**



# Phương pháp Ford-Fulkerson

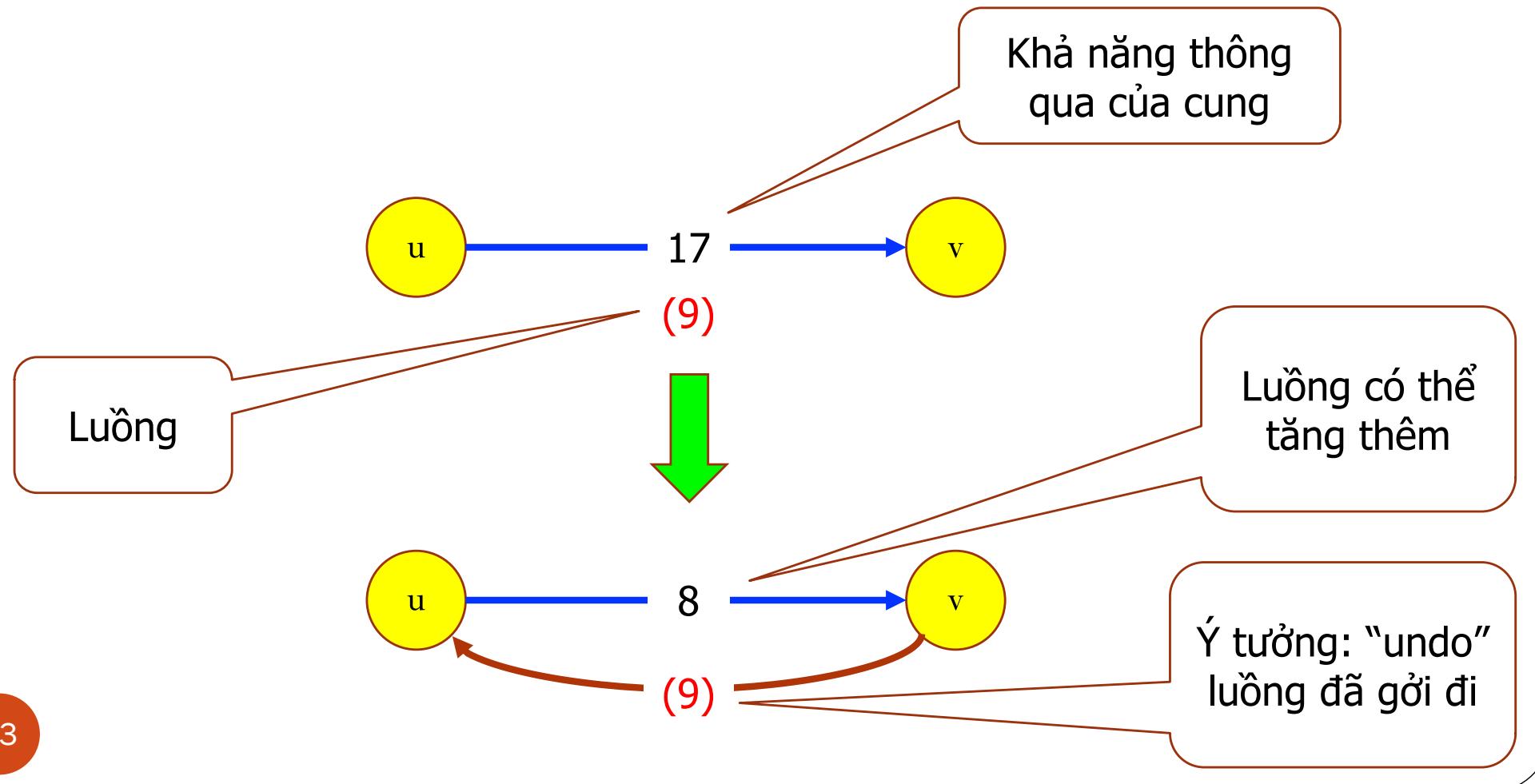
- Tìm đường tăng luồng:
  - Tìm đường đi từ s đến t sao cho có thể tăng thêm luồng trên đường đi đó.
  - **QS1: Tìm đường đi từ s đến t sao cho  $c(e) > f(e)$  với tất cả các cung trên đường đi**



Cần tìm cách khác để tăng luồng

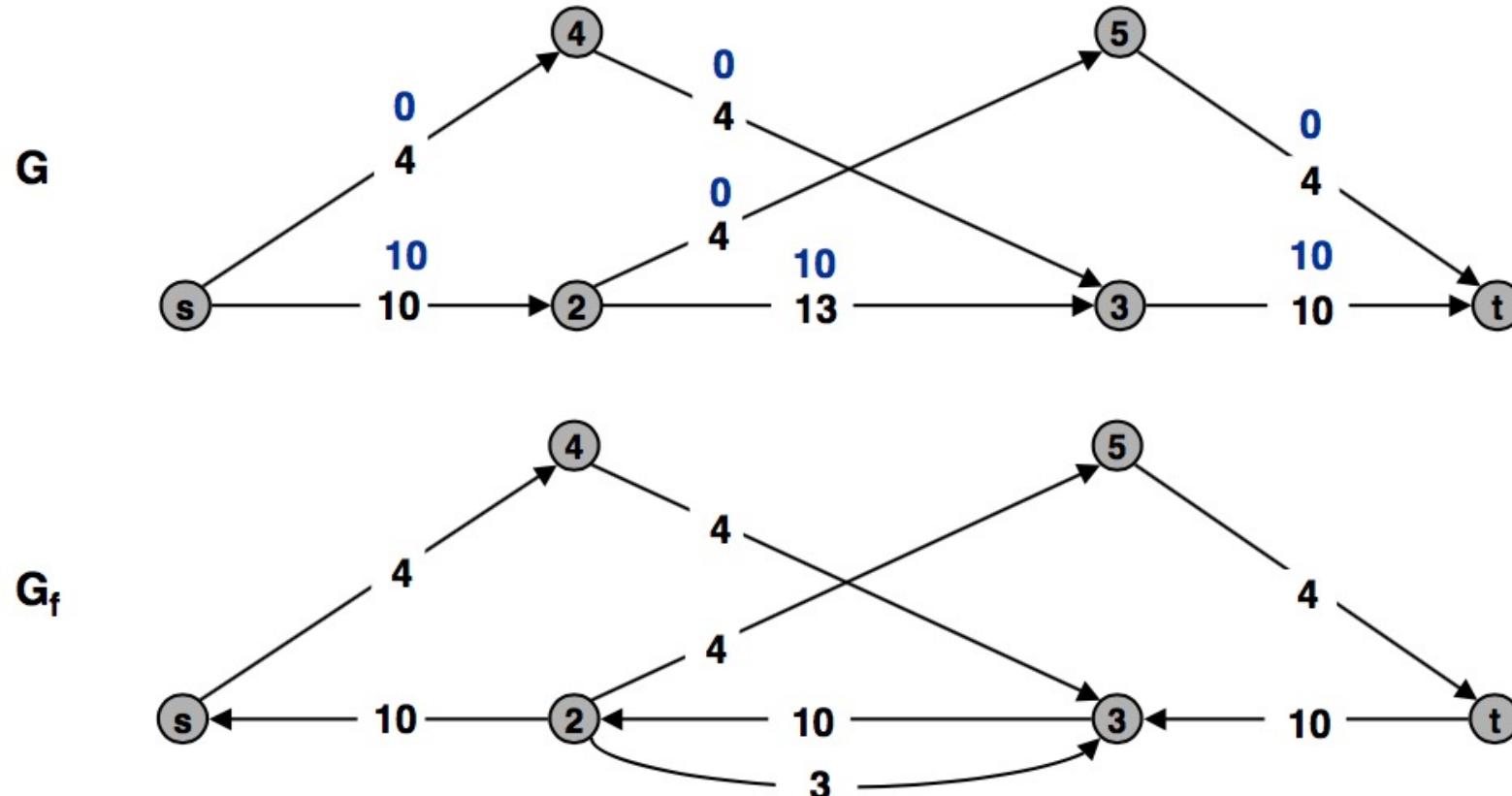
# Phương pháp Ford-Fulkerson

- Tìm đường tăng luồng:
  - Xây dựng đồ thị còn dư/thặng dư (residual graph)



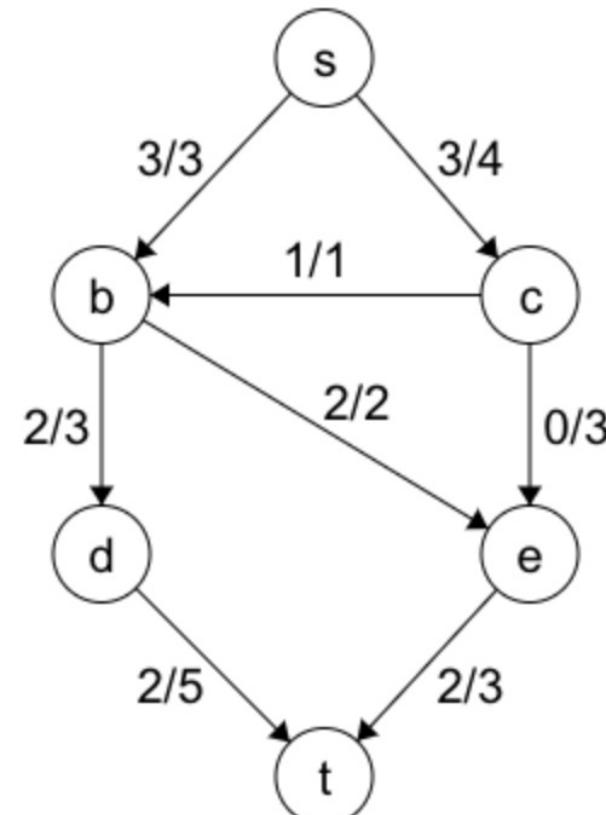
# Phương pháp Ford-Fulkerson

- Tìm đường tăng luồng:
  - Xây dựng đồ thị còn dư/thặng dư (residual graph)



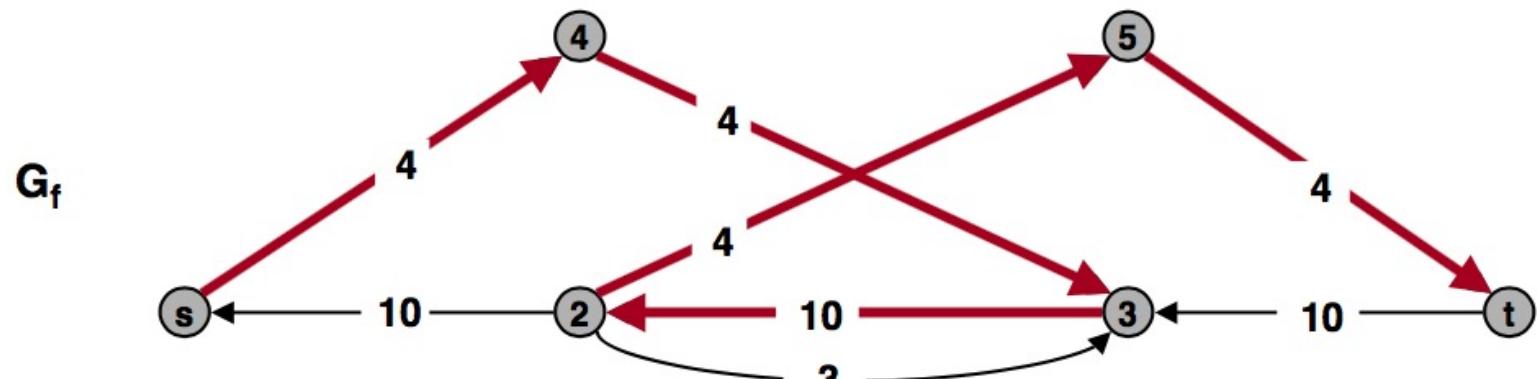
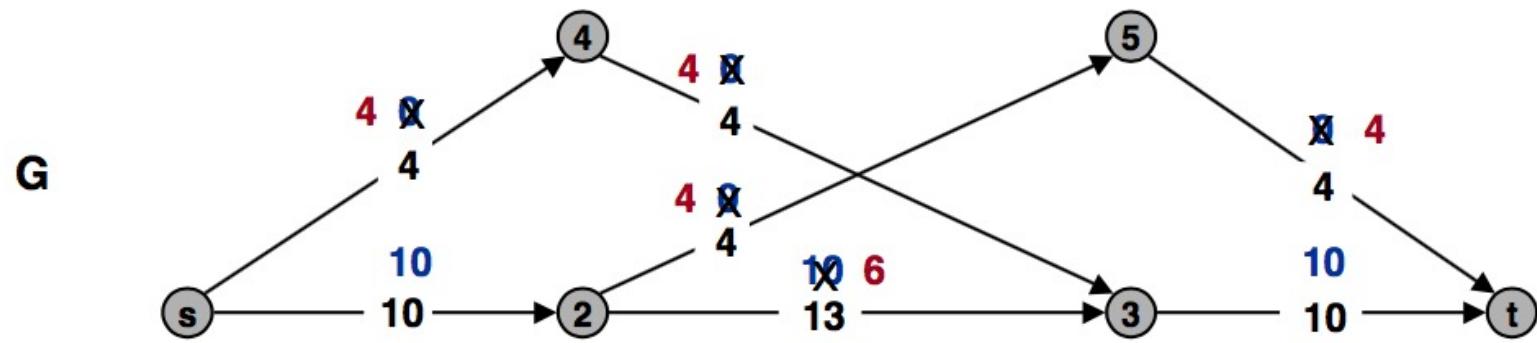
# Phương pháp Ford-Fulkerson

- **Bài tập**
- Cho mạng và luồng như hình vẽ, hãy tìm đồ thị thặng dư
  - Số đứng trước là luồng
  - Số đứng sau là khả năng thông qua)



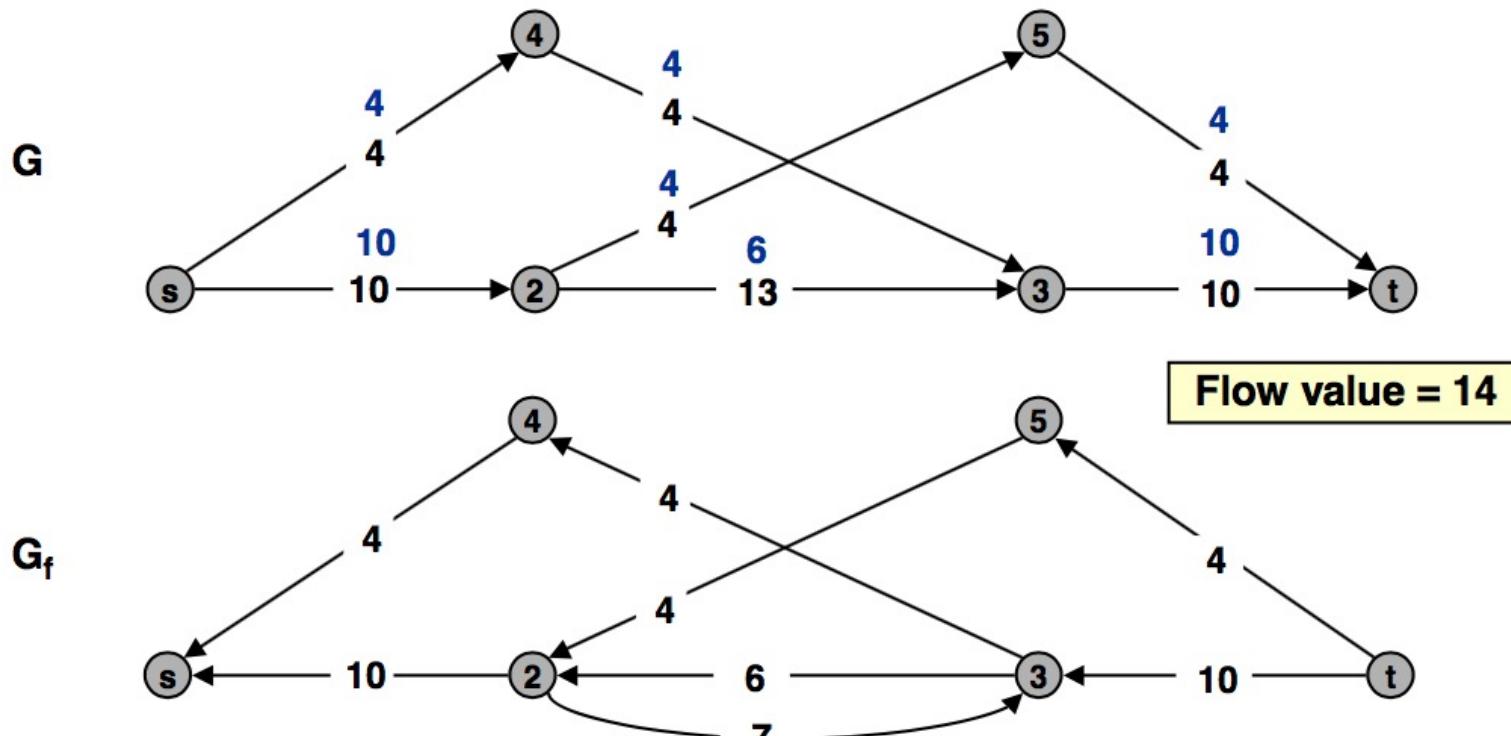
# Đường tăng luồng

- Đường tăng luồng = đường đi từ s đến t trên **đồ thị thặng dư**



# Đường tăng luồng

- Đường tăng luồng = đường đi từ s đến t trên **đồ thị thặng dư**
- Luồng cực đại  $\Leftrightarrow$  không còn đường tăng luồng



# Phương pháp Ford-Fulkerson

- **Gán nhãn (dánh dấu)** các đỉnh để tìm đường tăng luồng trên đồ thị thặng dư
  - Không cần xây dựng tường minh đồ thị thặng dư
  - Sử dụng **duyệt đồ thị** để tìm đường đi từ s đến t
- Mỗi đỉnh u được đánh dấu/gán nhãn với các thông tin sau:
  - **d[u]**: hướng của cung (+: cung thuận, -: cung nghịch “undo” luồng)
  - **p[u]**: đỉnh trước đỉnh u
  - **σ[u]**: lượng tăng luồng lớn nhất có thể tăng

# Phương pháp Ford-Fulkerson

- **Tìm đường tăng luồng từ s đến t**
  - Đánh dấu s: (+, s, oo)
  - Đánh dấu các đỉnh kề của s
  - Đánh dấu **các đỉnh kề của đỉnh kề** của s
  - ...
  - Cho đến khi t được đánh dấu => tìm được đường đi từ s đến t
- Có thể cài đặt bằng đệ quy hoặc vòng lặp sử dụng ngăn xếp, hàng đợi hoặc hàng đợi ưu tiên
- **Thuật toán Edmonds – Karp** sử dụng tìm kiếm theo chiều rộng để tìm đường tăng luồng

# Phương pháp Ford-Fulkerson

- **Tìm đường tăng luồng từ s đến t dùng hàng đợi**

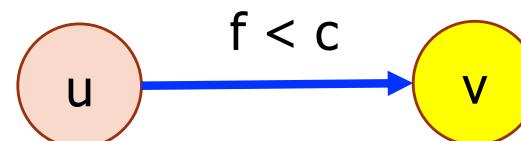
1. Gán nhãn s: (+, s,  $\infty$ )
2. Push s vào hàng đợi Q
3. while (Q không rỗng) {
  - a.  $u = \text{front}(Q); \text{dequeue}(Q);$
  - b. for (các đỉnh kề v của u và có  $f(u, v) < c(u, v)$ )  
Gán nhãn v: (+, u,  $\min\{\sigma[u], c(u, v) - f(u, v)\}$ )  
Đưa v vào Q
  - c. for (các đỉnh x có cung đi đến u có  $f(x, u) > 0$ )  
Gán nhãn x: (-, u,  $\min\{\sigma[u], f(x, u)\}$ )  
Đưa x vào Q
  - d. Nếu t được đánh dấu  $\Rightarrow$  thoát vòng lặp while}

# Phương pháp Ford-Fulkerson

## • Quy ước gán nhãn

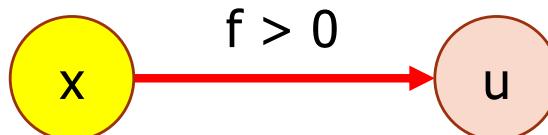
- Khi lấy  $u$  ra khỏi hàng đợi
  - Gán nhãn các đỉnh kề với nó trước (cung thuận)

(+,  $u$ ,  $\min(\sigma[u], c - f)$ )



- Gán nhãn các đỉnh có cung đi đến nó sau (cung nghịch)

(-,  $u$ ,  $\min(\sigma[u], f)$ )



# Phương pháp Ford-Fulkerson

- **Tăng luồng theo đường tăng luồng**

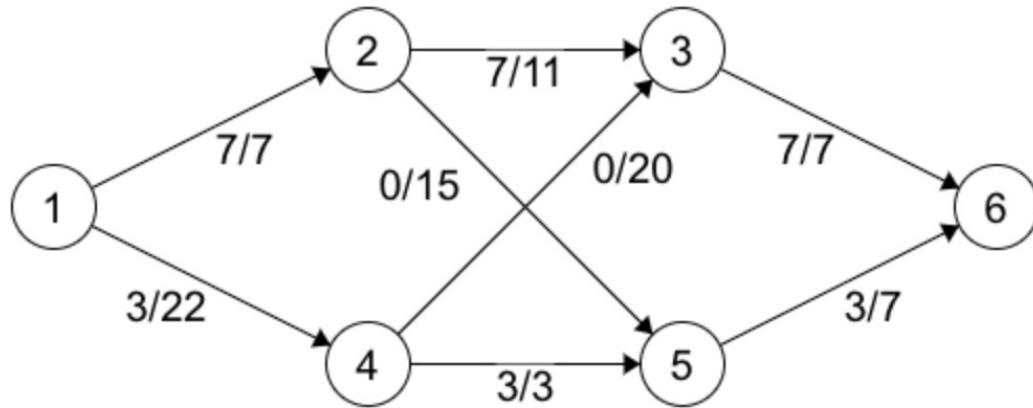
```
u = t;  
sigma = σ[t]  
while (u != s) {  
    if (d[u] == '+') //Cung thuận  
        f(p[u], u) += sigma; //TĂNG LUÔNG  
    else //Cung nghịch  
        f(u, p[u]) -= sigma; //GIẢM LUÔNG  
    u = p[u];  
}
```

# Phương pháp Ford-Fulkerson

- Khởi tạo  $f(u, v) = 0$  với mọi cung  $(u, v)$
- **while** (1) {
  - **Tìm đường tăng luồng**
  - Nếu không tìm thấy => thoát vòng lặp (break)
  - **Tăng luồng theo đường tăng luồng**}
- Lát cắt hẹp nhất =  $(S, T)$ 
  - S: các đỉnh đã có nhãn
  - T: là các đỉnh chưa có nhãn

### Mạng và luồng khởi tạo (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo

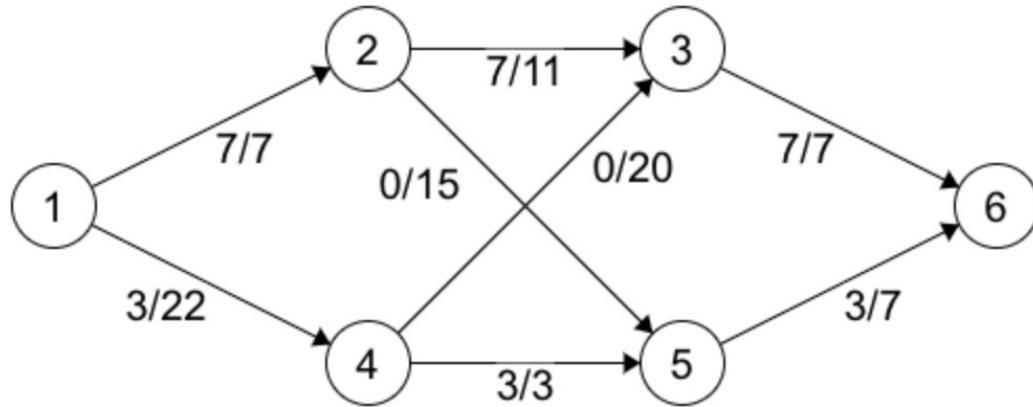


### Gán nhãn các đỉnh dùng hàng đợi

	1	2	3	4	5	6	Hàng đợi
Khởi tạo							
1							
2							
3							
4							
5							

### Mạng và luồng khởi tạo (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo

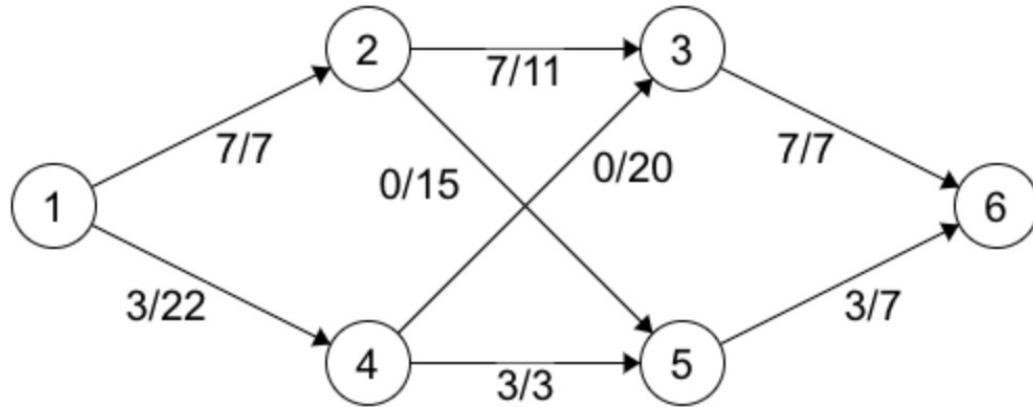


### Gán nhãn các đỉnh dùng hàng đợi

	1	2	3	4	5	6	Hàng đợi
Khởi tạo	(+, 1, oo)						1
1							
2							
3							
4							
5							

### Mạng và luồng khởi tạo (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo

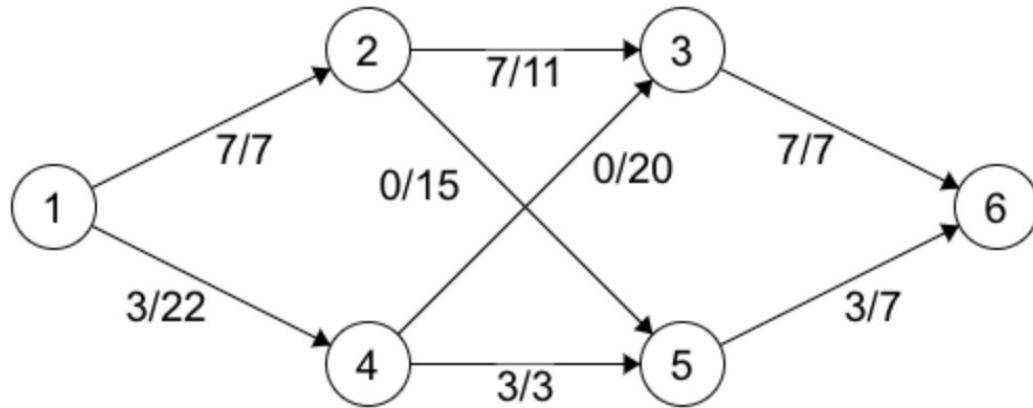


### Gán nhãn các đỉnh dùng hàng đợi

	1	2	3	4	5	6	Hàng đợi
Khởi tạo	(+, 1, oo)						1
1				(+, 1, 19)			4
2							
3							
4							
5							

### Mạng và luồng khởi tạo (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo

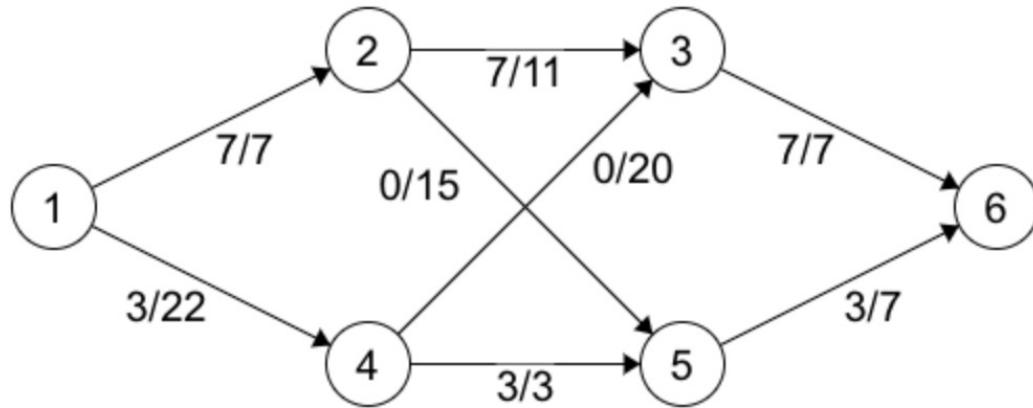


### Gán nhãn các đỉnh dùng hàng đợi

	1	2	3	4	5	6	Hàng đợi
Khởi tạo	(+, 1, oo)						1
1				(+, 1, 19)			4
2			(+, 4, 19)				3
3							
4							
5							

### Mạng và luồng khởi tạo (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo

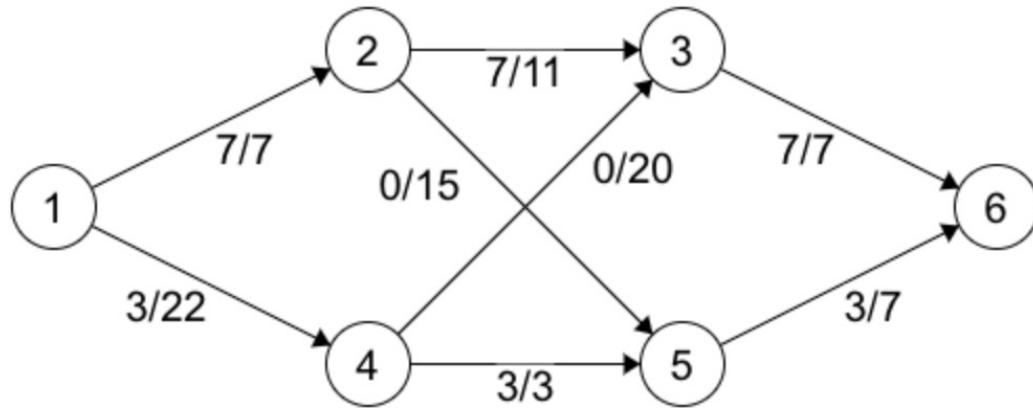


### Gán nhãn các đỉnh dùng hàng đợi

	1	2	3	4	5	6	Hàng đợi
Khởi tạo	(+, 1, oo)						1
1				(+, 1, 19)			4
2			(+, 4, 19)				3
3		(-, 3, 7)					2
4							
5							

### Mạng và luồng khởi tạo (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo

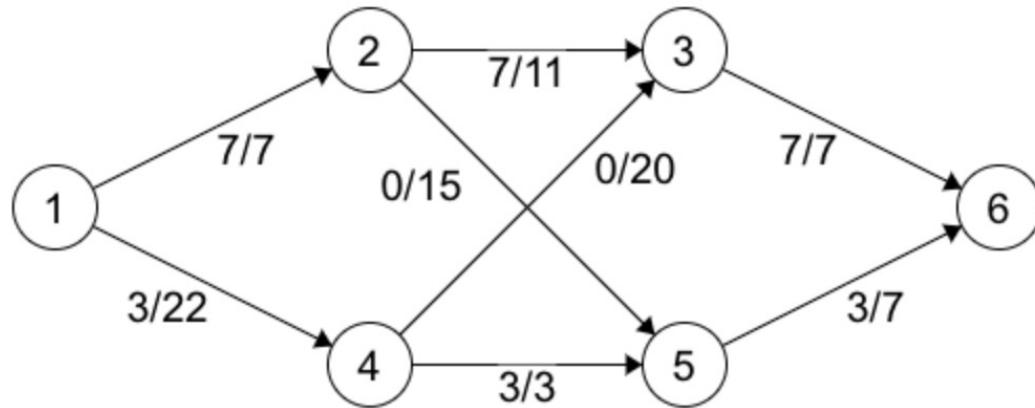


### Gán nhãn các đỉnh dùng hàng đợi

	1	2	3	4	5	6	Hàng đợi
Khởi tạo	(+, 1, oo)						1
1				(+, 1, 19)			4
2			(+, 4, 19)				3
3		(-, 3, 7)					2
4					(+, 2, 7)		5
5							

### Mạng và luồng khởi tạo (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo



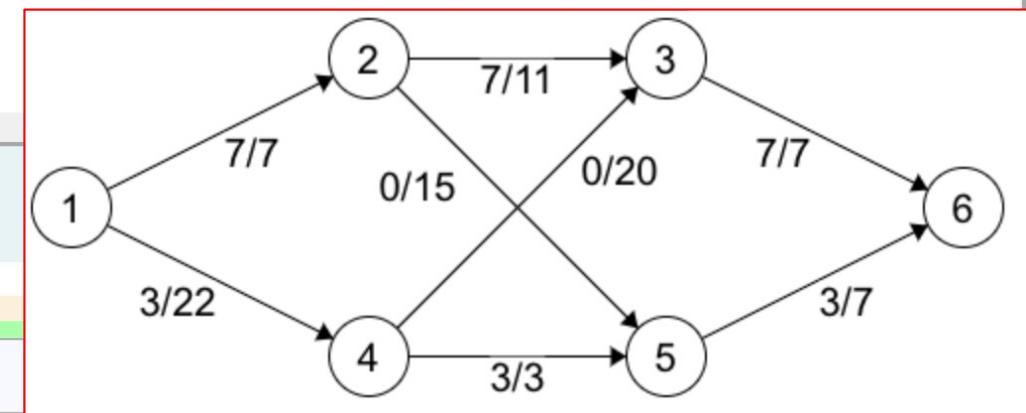
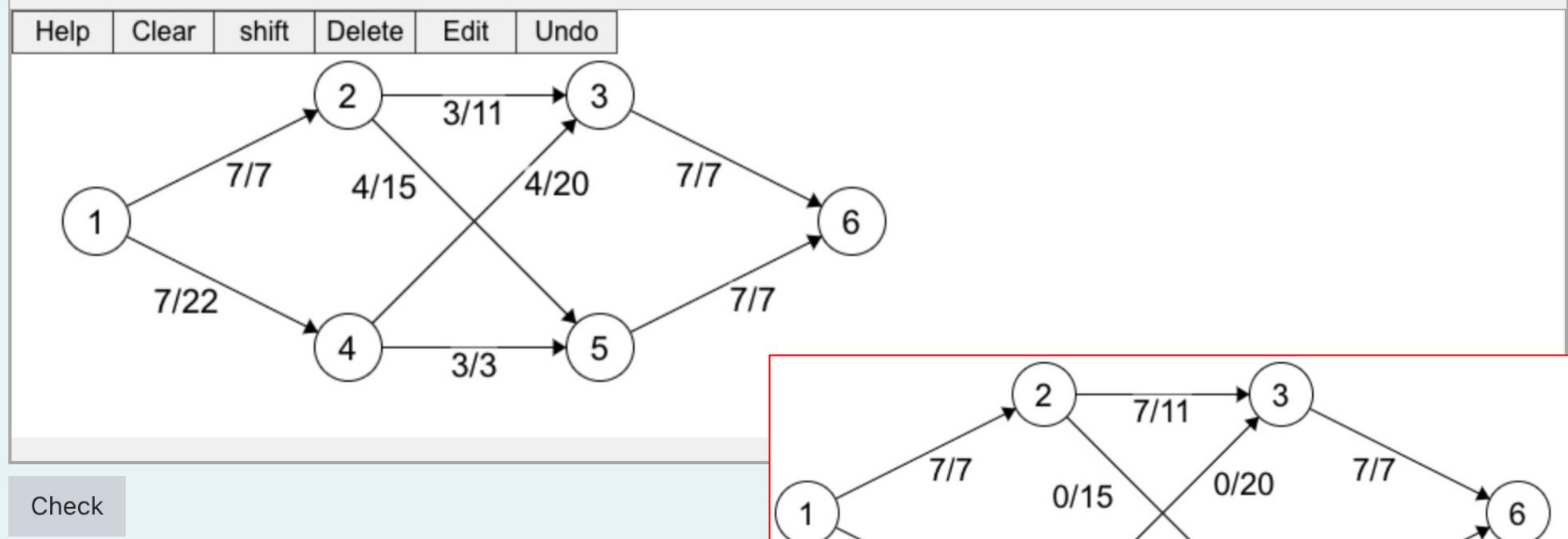
### Gán nhãn các đỉnh dùng hàng đợi

	1	2	3	4	5	6	Hàng đợi
Khởi tạo	(+, 1, oo)						1
1				(+, 1, 19)			4
2			(+, 4, 19)				3
3		(-, 3, 7)					2
4					(+, 2, 7)		5
5						(+, 5, 4)	6

- Đường tăng luồng (augmenting path): 1->4->3->2->5->6 ví dụ: 1 -> 3 -> 6

- Lượng luồng tăng thêm (bottle neck capacity): 4

### Mạng sau khi tăng luồng



#### Test

- ✓ 1. Gán nhãn (đánh dấu) các đỉnh (80%)  
2. Đường tăng luồng và lượng luồng tăng thêm (10%)  
3. Mạng sau khi tăng luồng (10%)

1. Kiểm tra việc gán nhãn (đánh dấu) các đỉnh ✓

Khởi tạo:

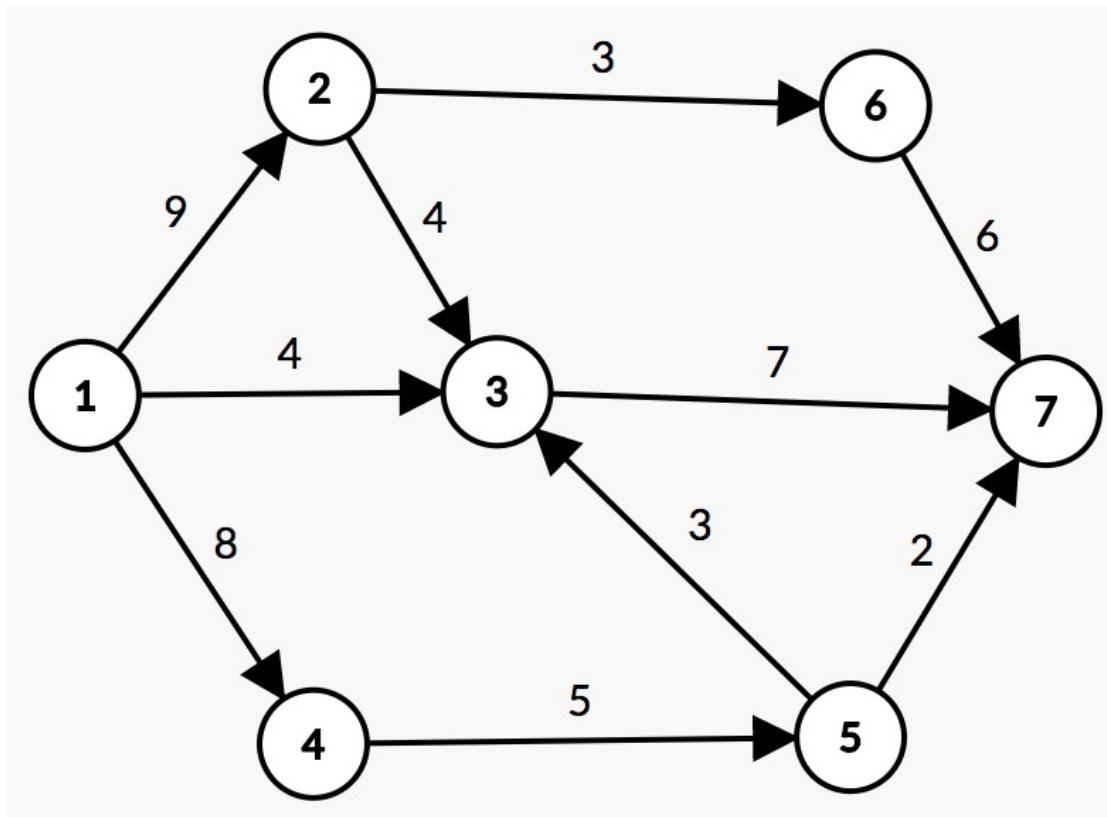
[I] Nhãn của các đỉnh okie.

[I] Nội dung hàng đợi okie.

Lần lặp 1:

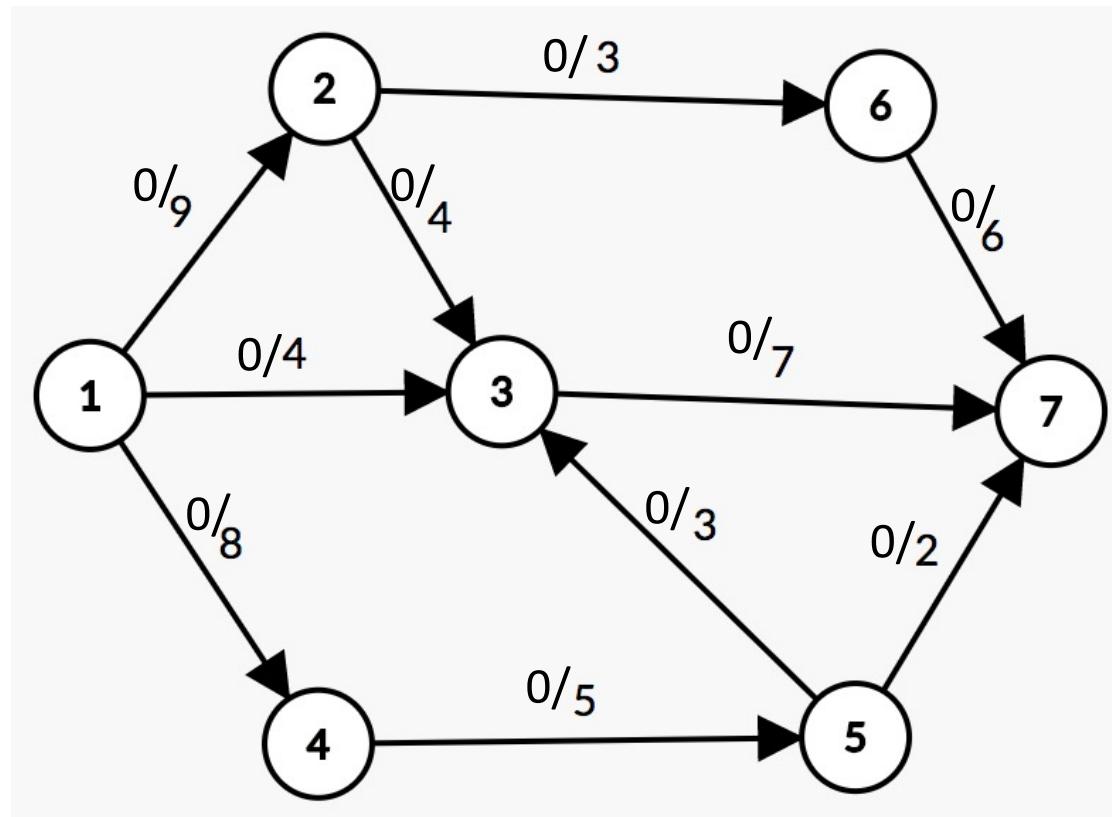
# Ví dụ

- Tìm luồng cực đại và lát cắt hẹp nhất



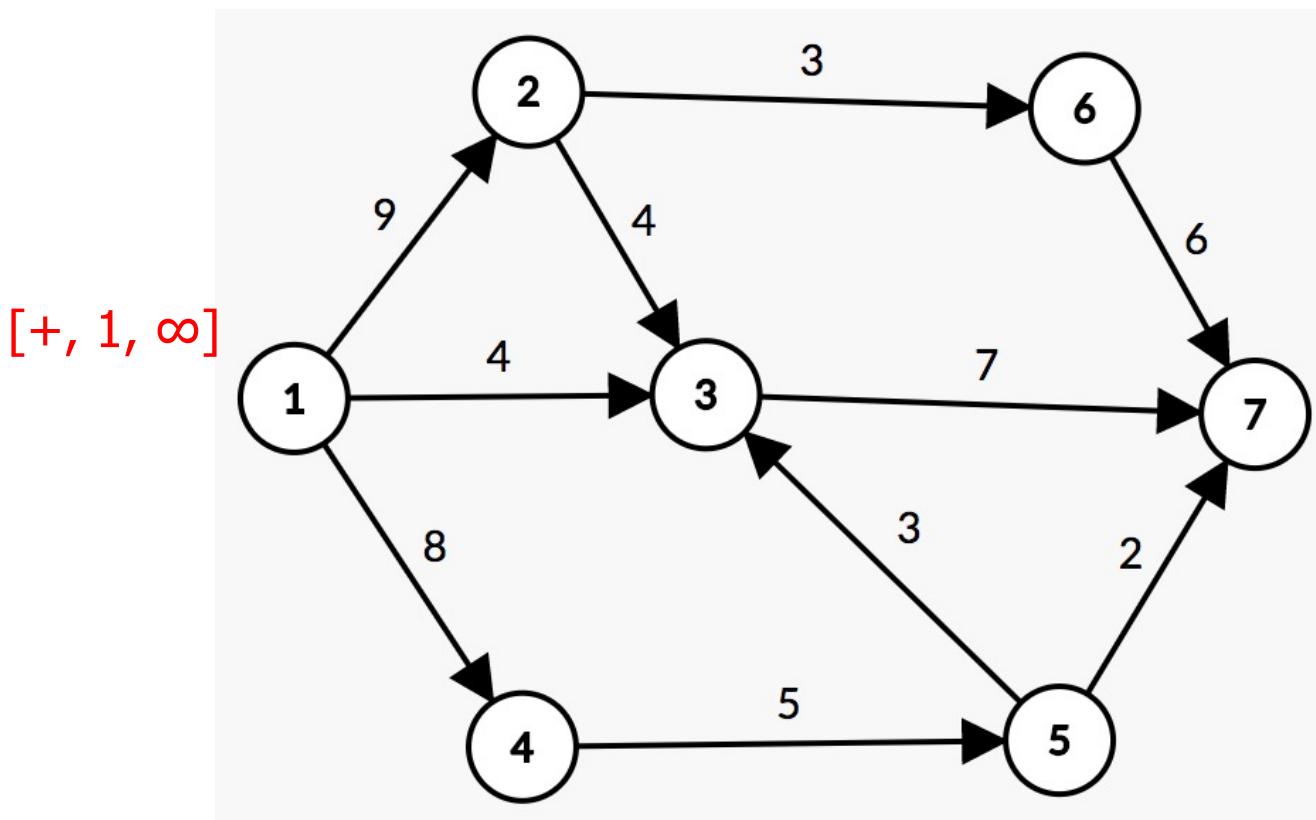
# Khởi tạo

- Khởi tạo một luồng hợp lệ bất kỳ (thường là 0)



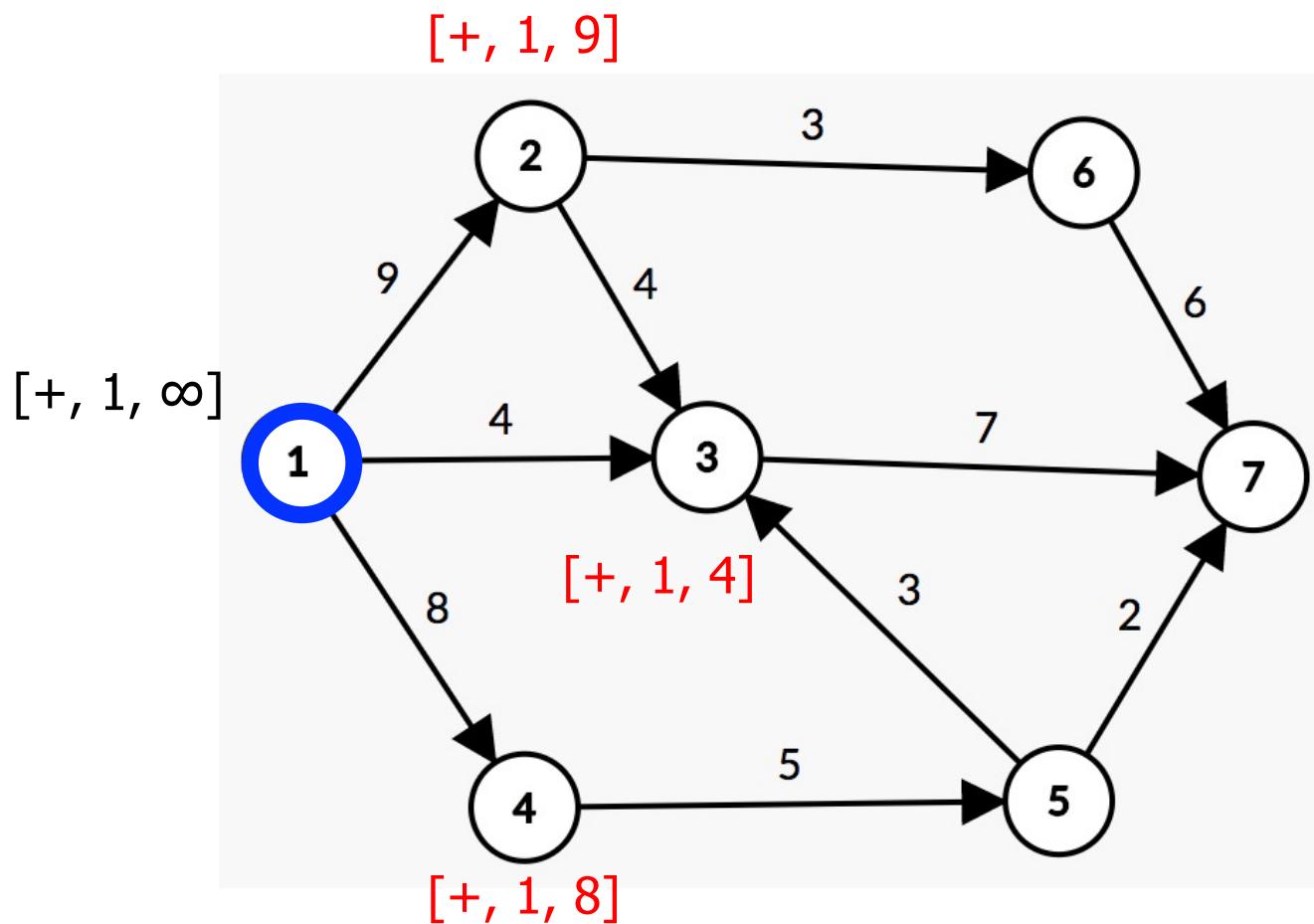
# Lắp 1

- Tìm đường tăng luồng



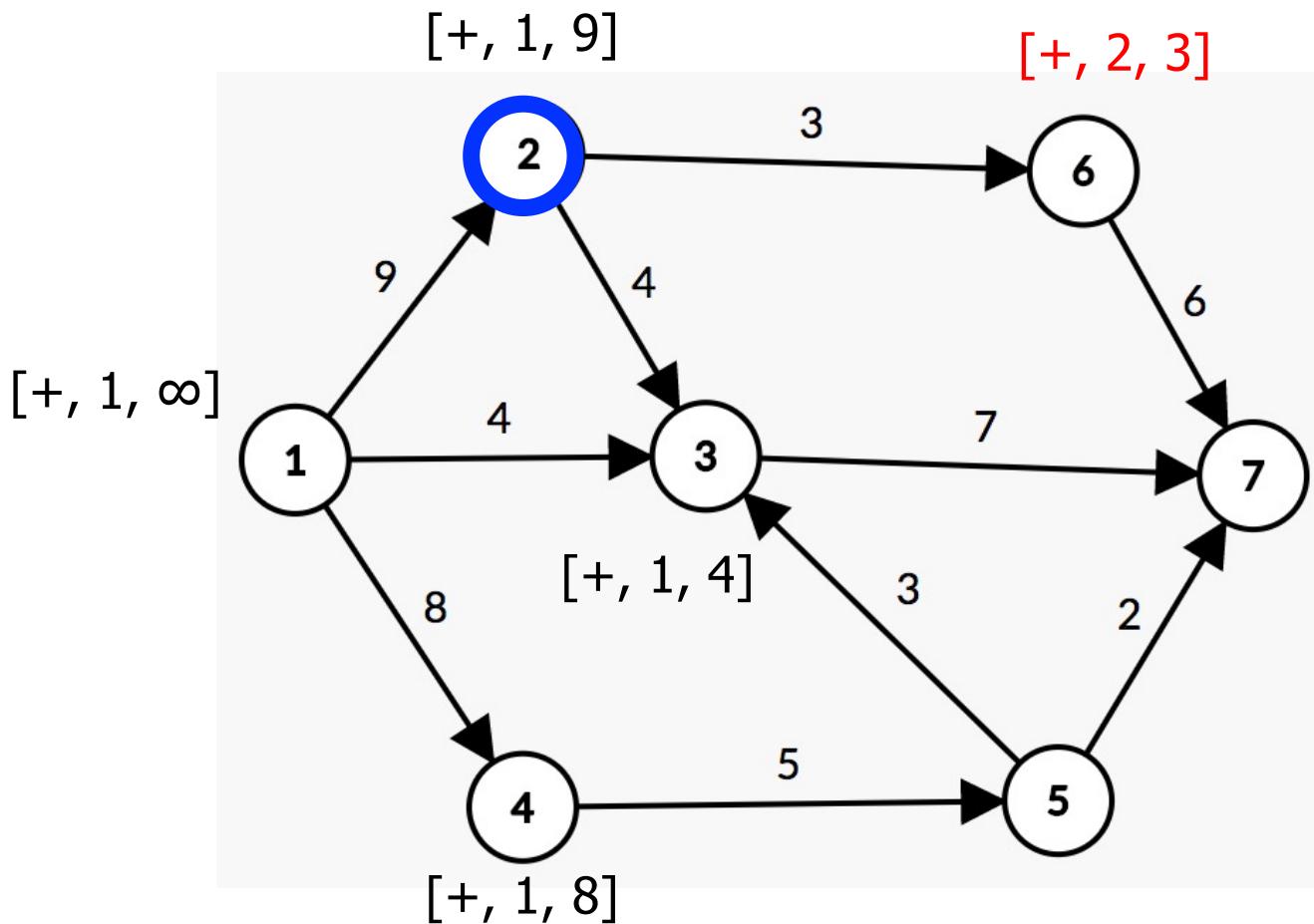
# Lắp 1

- Tìm đường tăng luồng



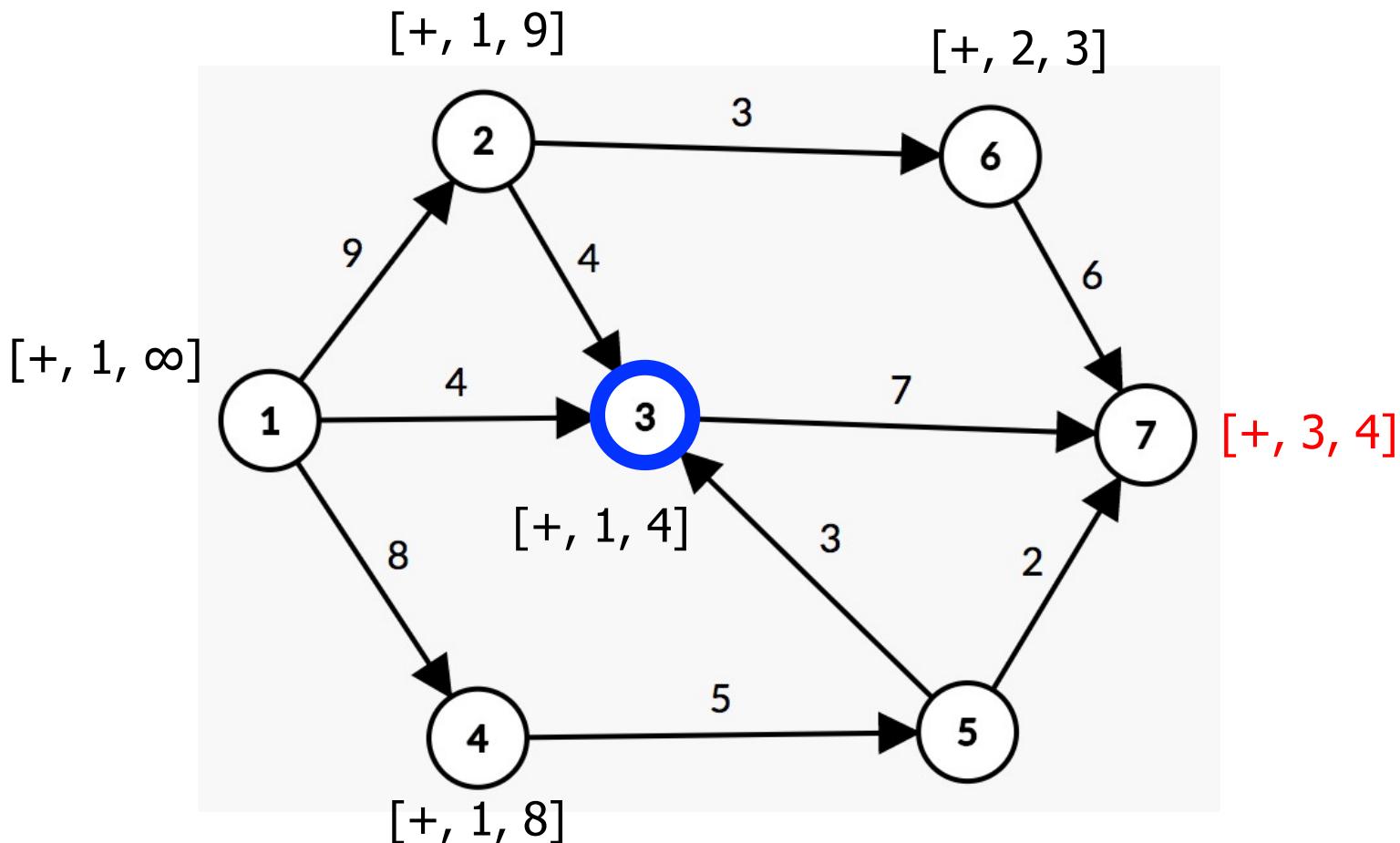
# Lắp 1

- Tìm đường tăng luồng



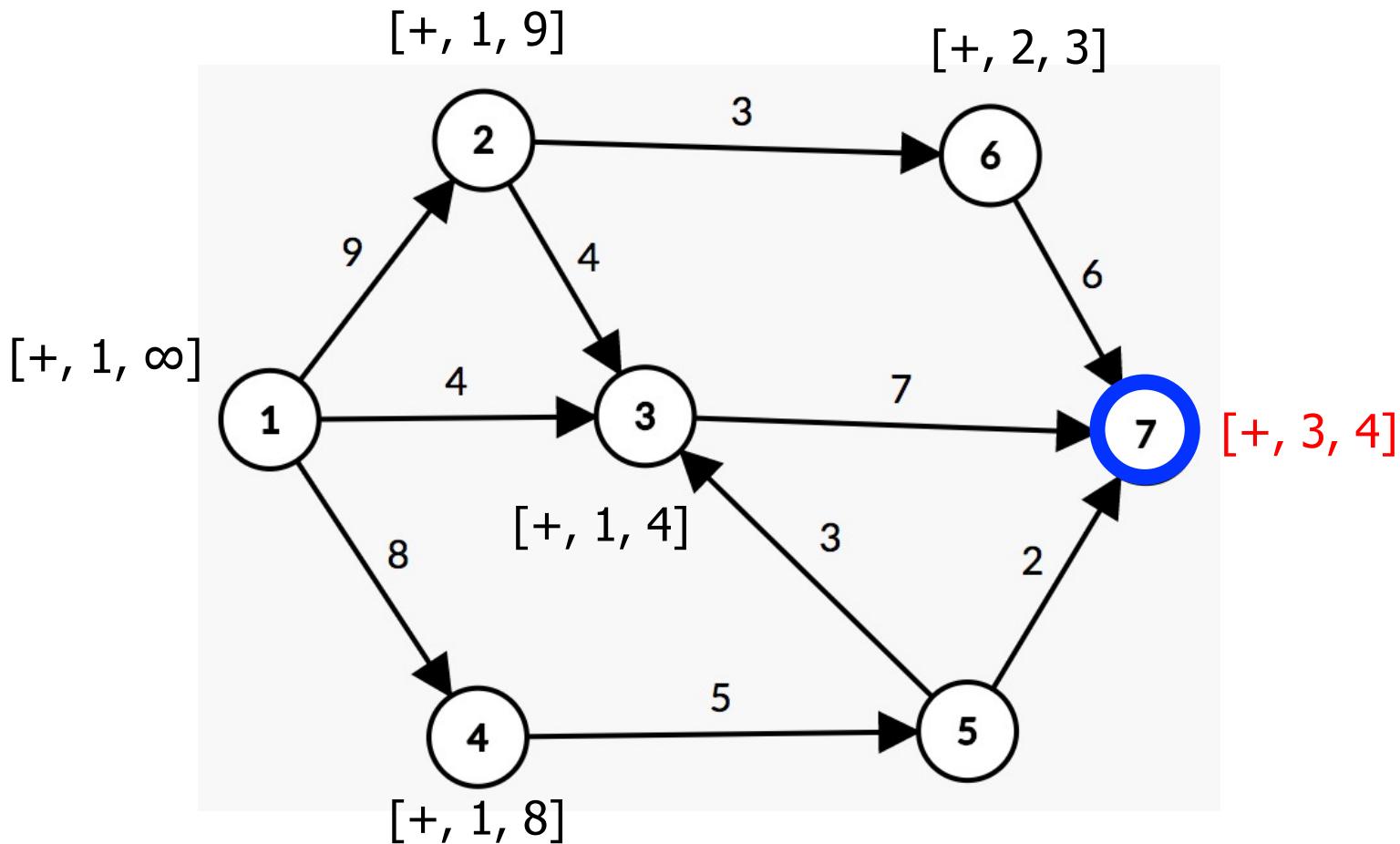
# Lắp 1

- Tìm đường tăng luồng



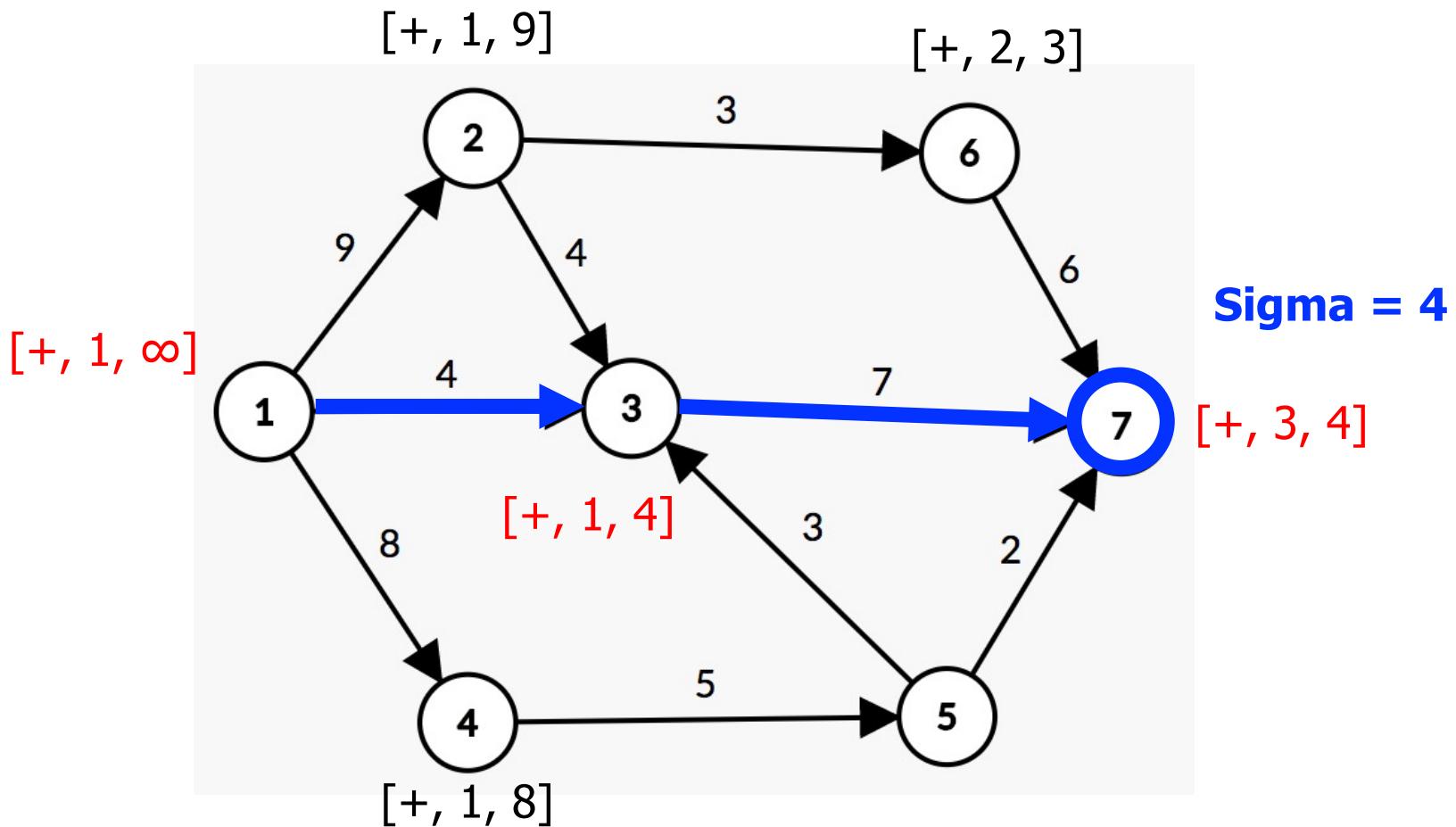
# Lắp 1

- Tìm đường tăng luồng



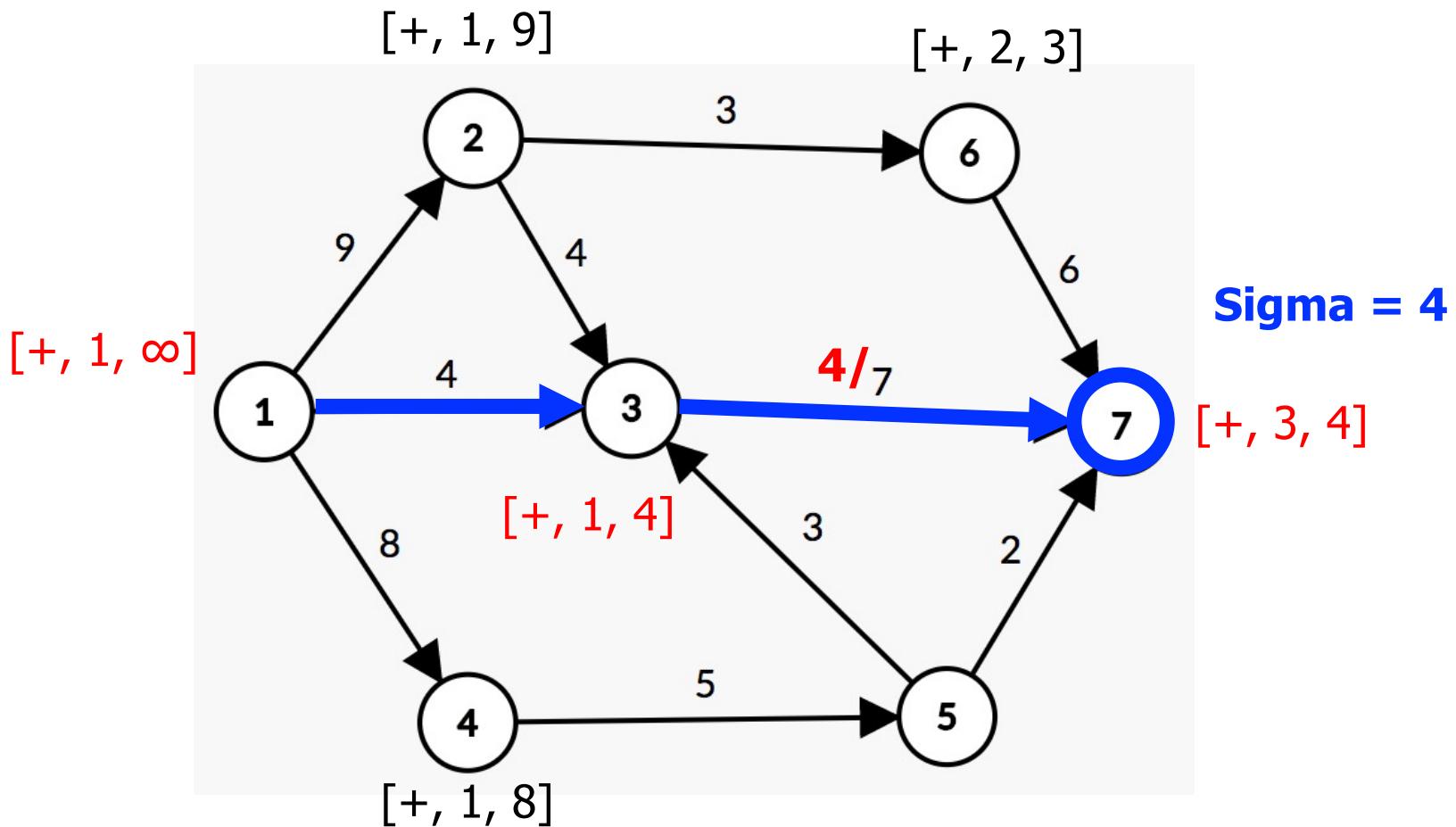
# Lắp 1

- Đường tăng luồng:  $1 \rightarrow 3 \rightarrow 7$



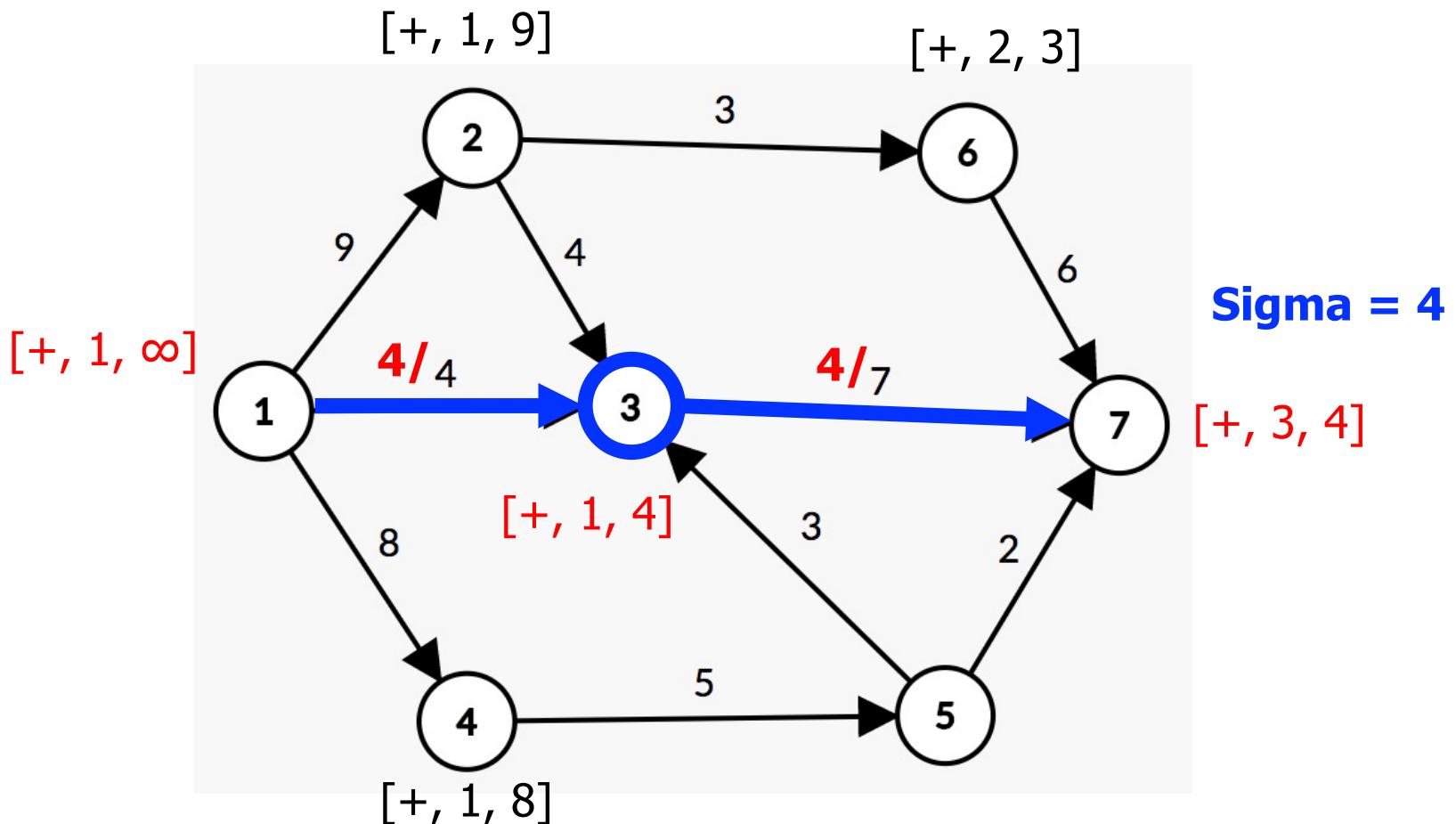
# Lắp 1

- Tăng luồng



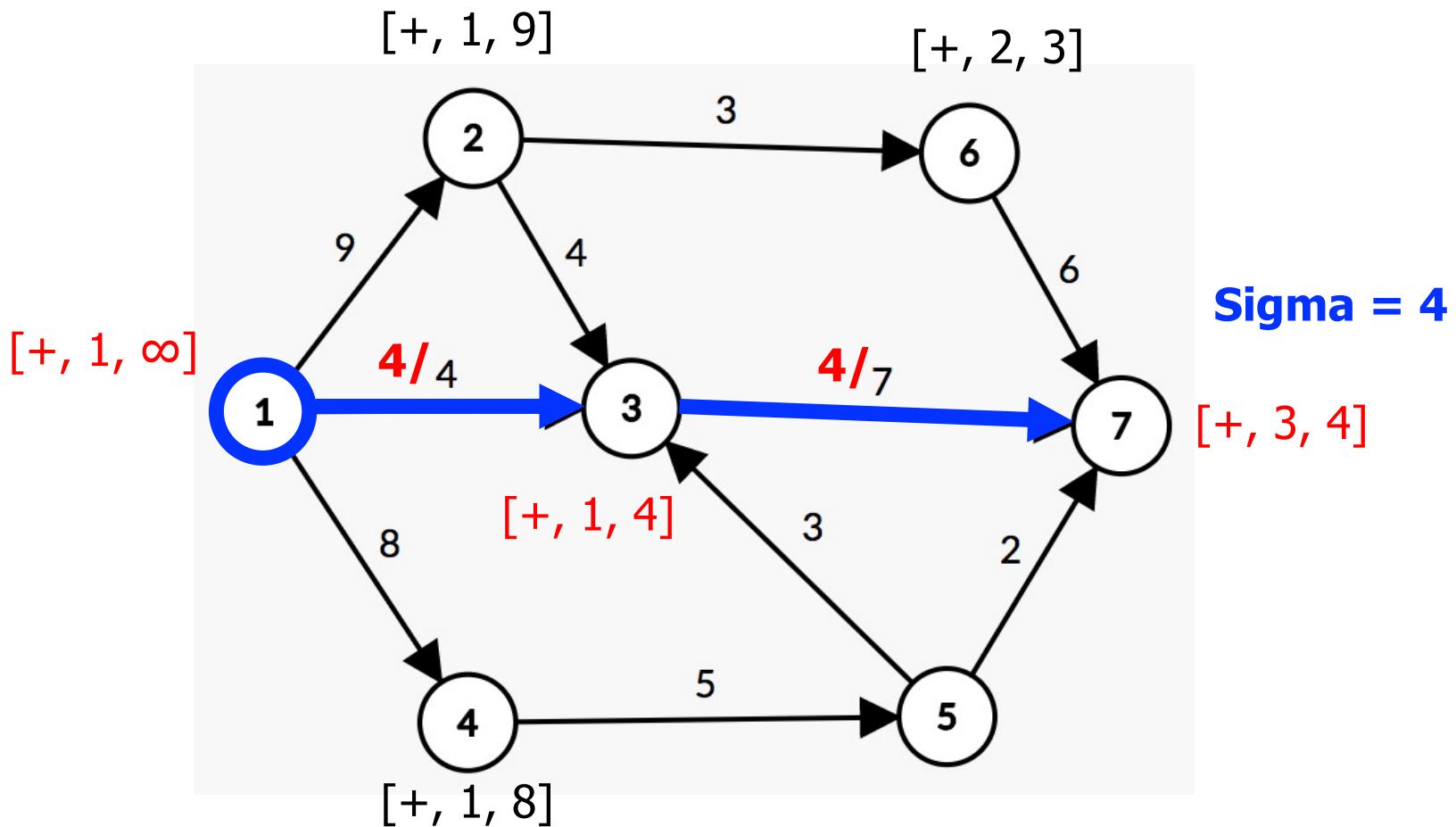
# Lắp 1

- Tăng luồng



# Lắp 1

- Tăng luồng

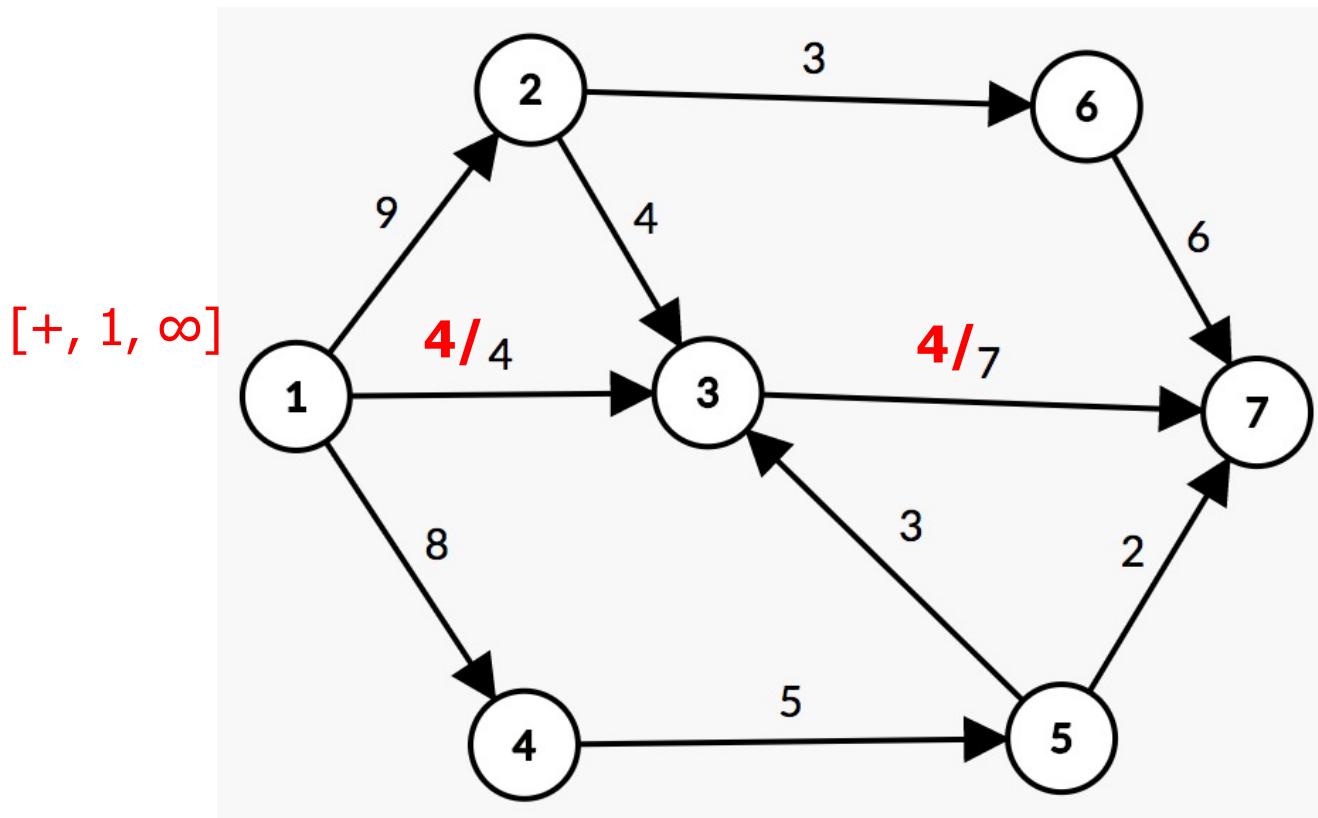


# Mẹo

- Tất cả các cung trên đường tăng luồng sẽ được tăng/giảm 1 lượng giống nhau =  **$\sigma[t]$**
- Trình bày:
  - Lắp 1:
    - Đánh dấu trên đồ thị gốc
    - Vẽ đồ thị mới và ghi kết quả tăng luồng trên đồ thị mới
  - Lắp 2:
    - Tiếp tục đánh dấu trên đồ thị mới
    - Vẽ đồ thị mới nữa và ghi kết quả tăng luồng
  - ...

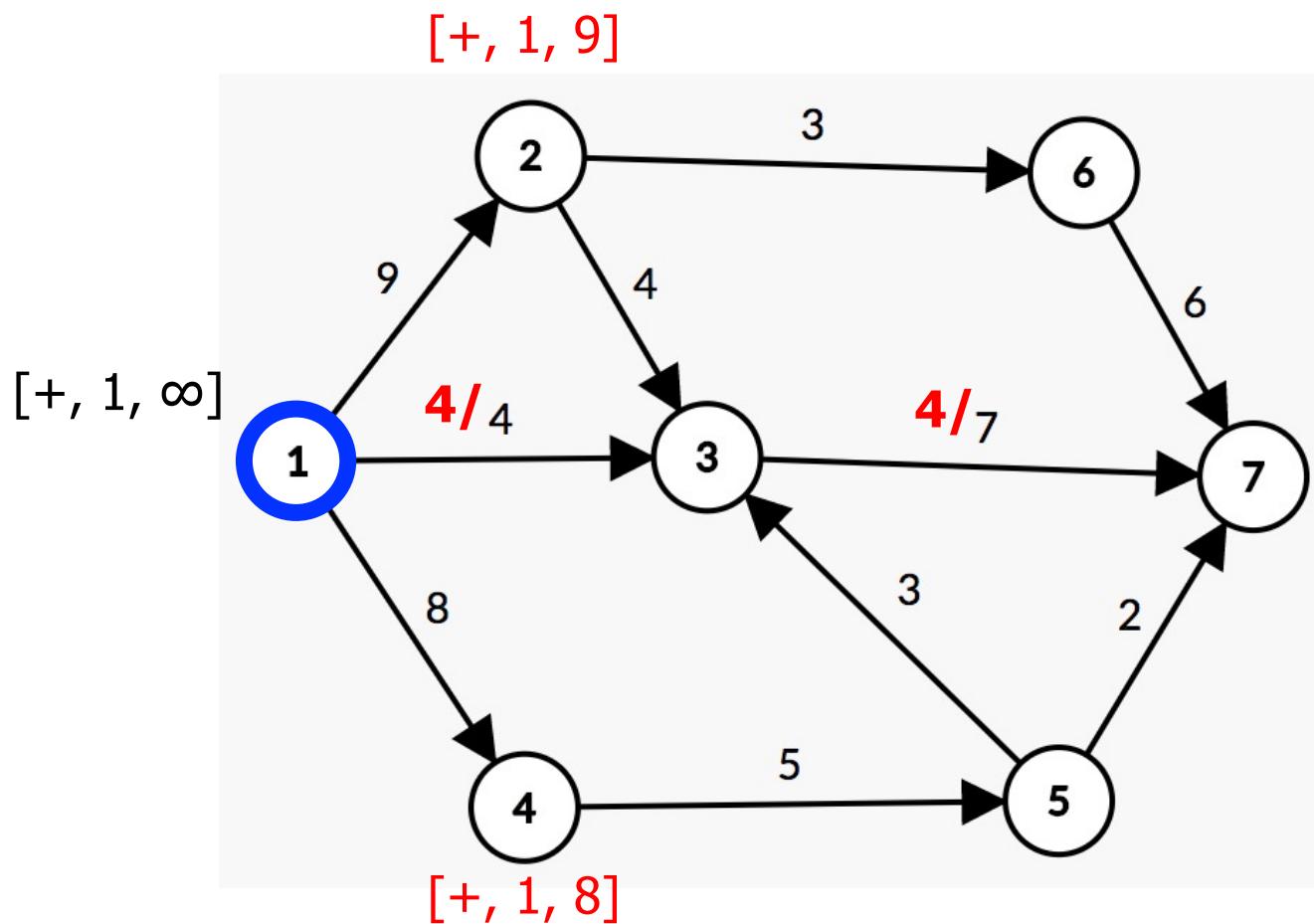
# Lắp 2

- Tìm đường tăng luồng



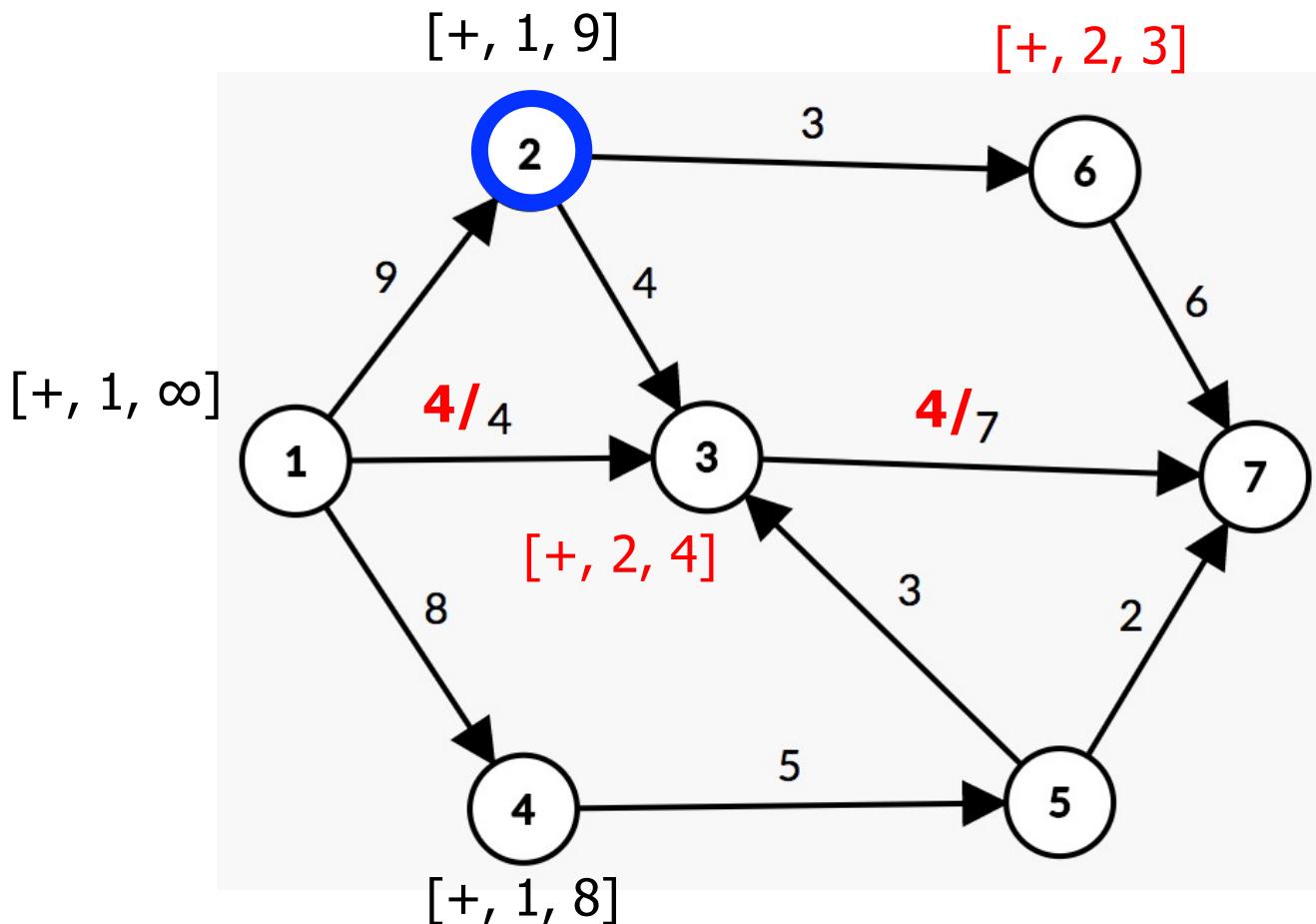
# Lắp 2

- Tìm đường tăng luồng



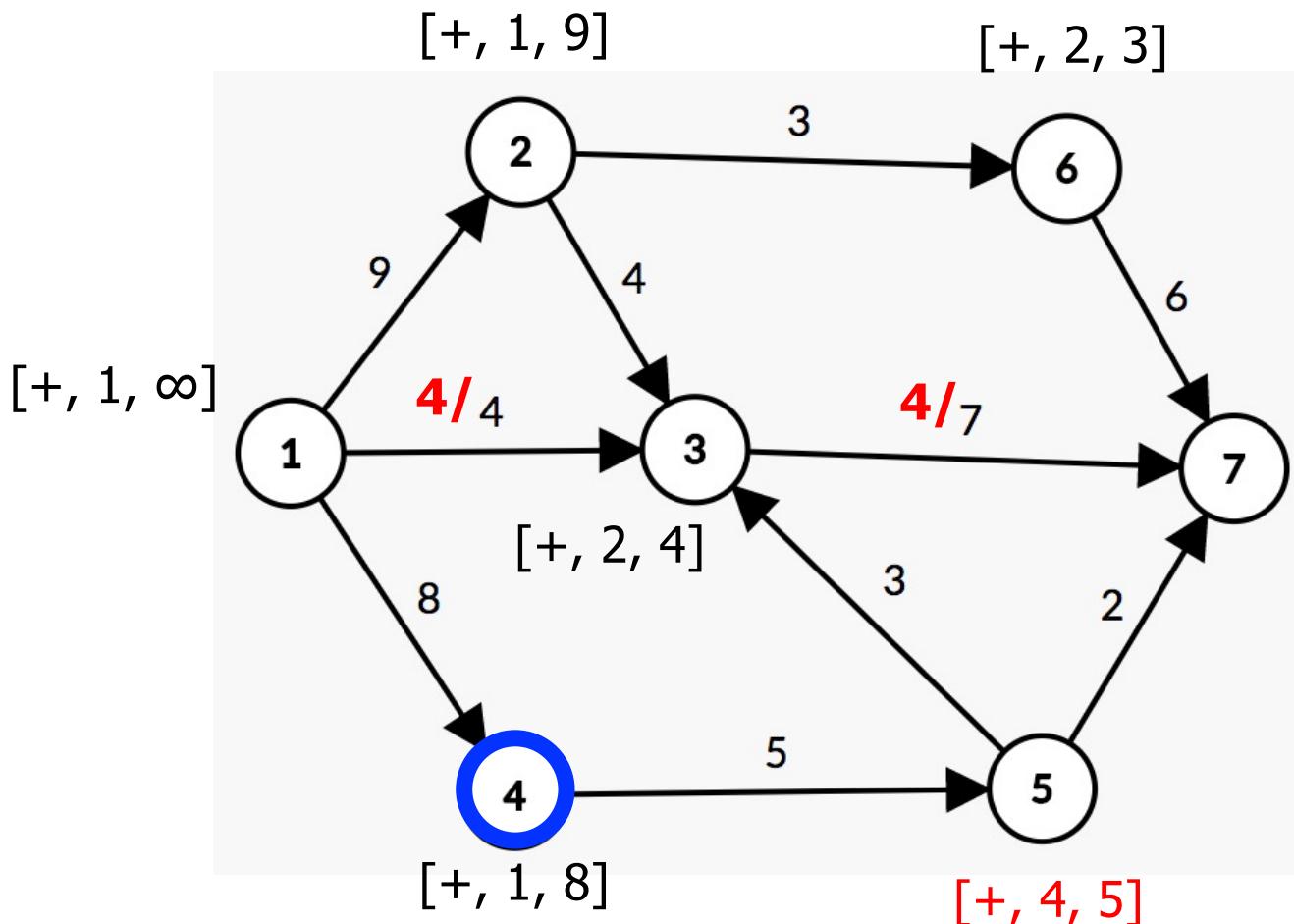
# Lắp 2

- Tìm đường tăng luồng



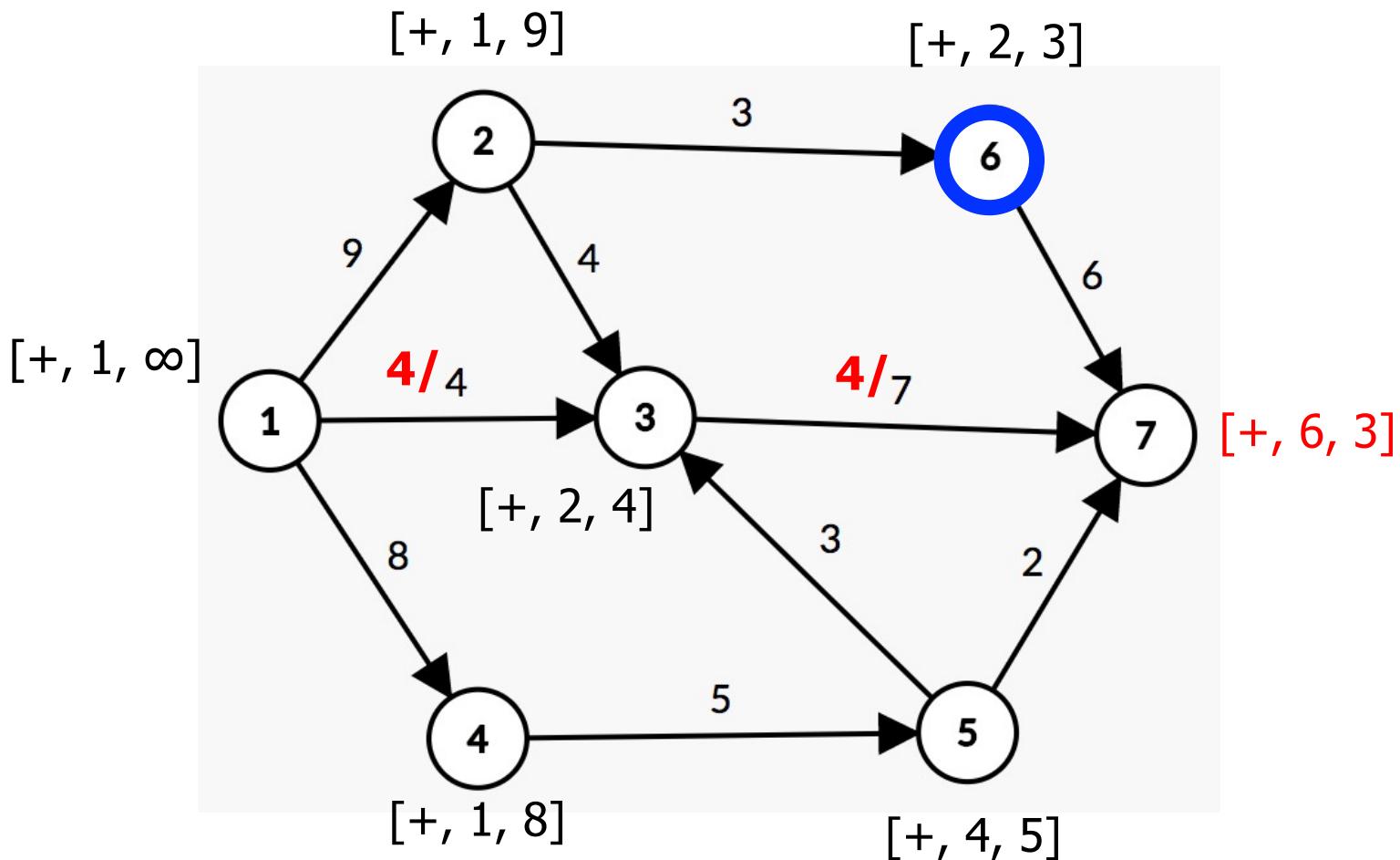
# Lắp 2

- Tìm đường tăng luồng



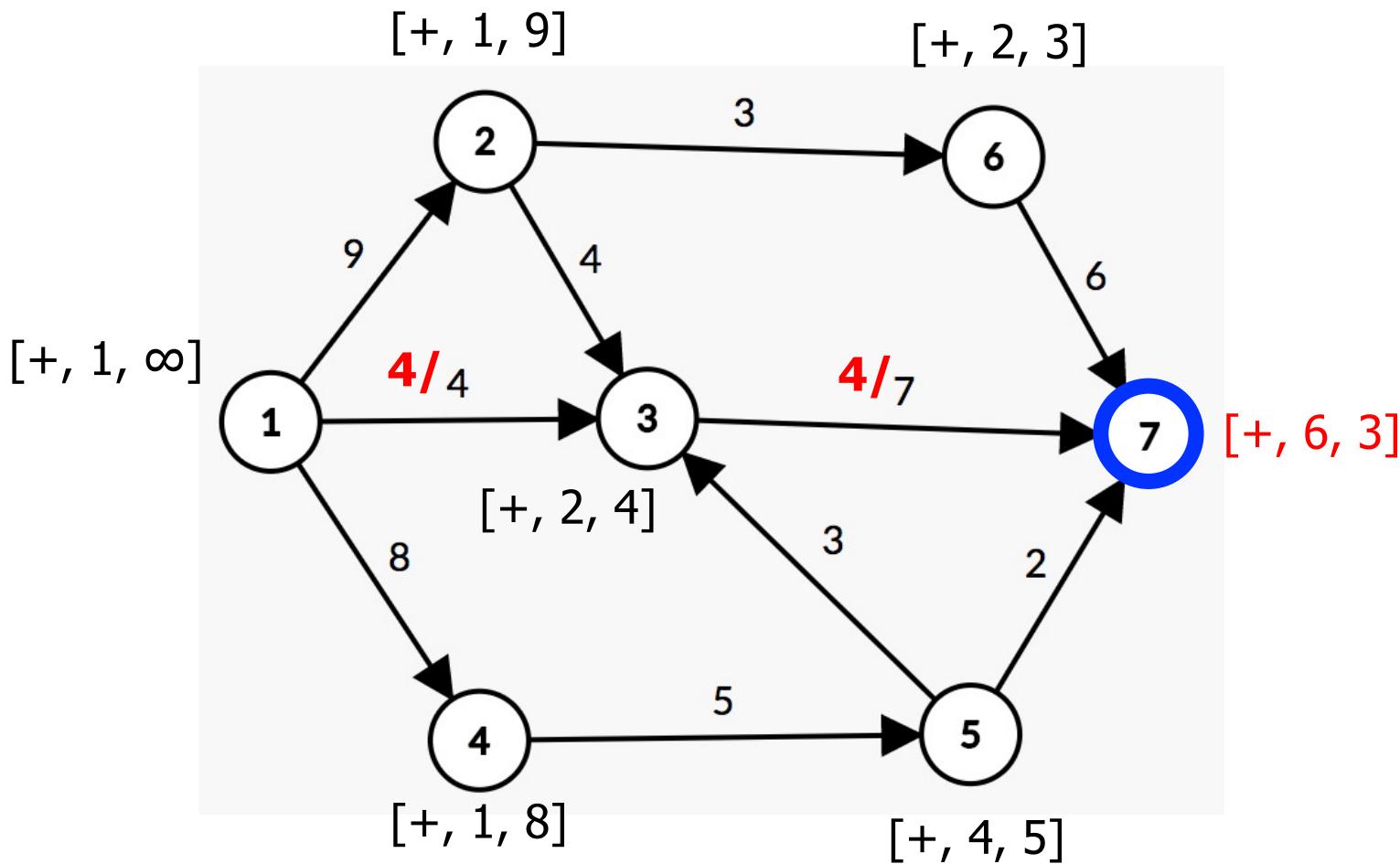
# Lắp 2

- Tìm đường tăng luồng



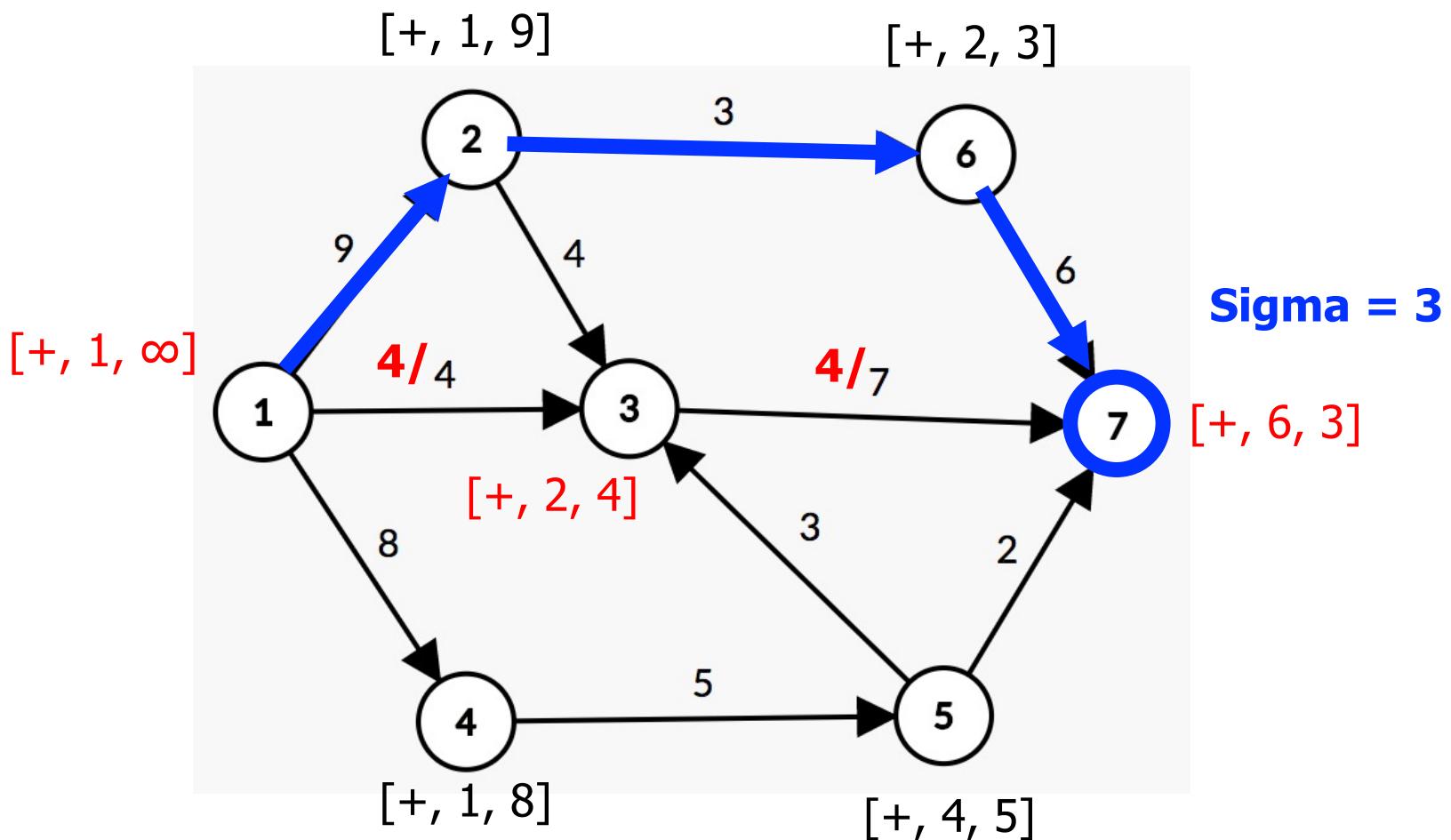
# Lặp 2

- t có nhãn  $\Rightarrow$  dừng đánh dấu



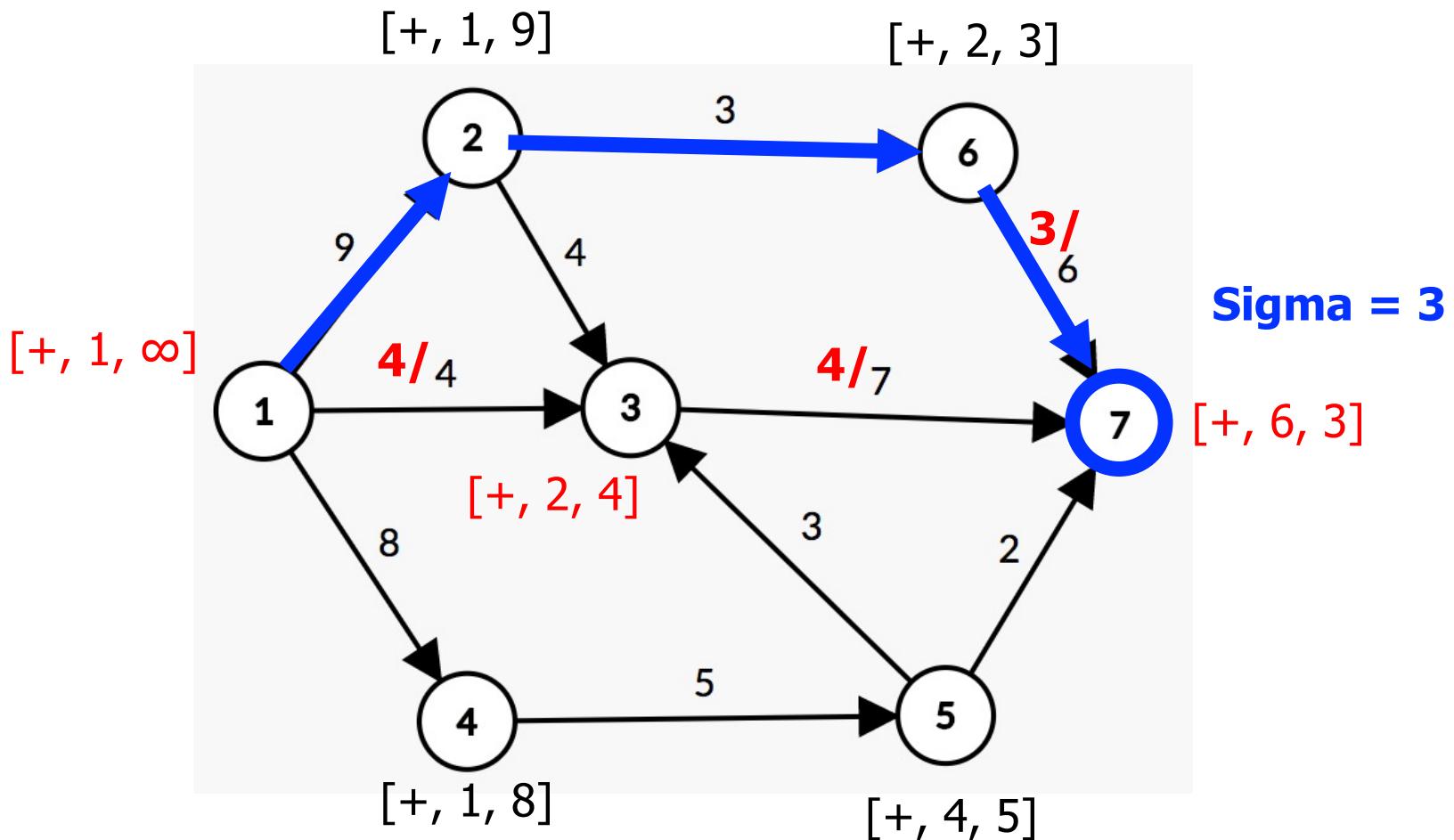
# Lắp 2

- Đường tăng luồng:  $1 \rightarrow 2 \rightarrow 6 \rightarrow 7$



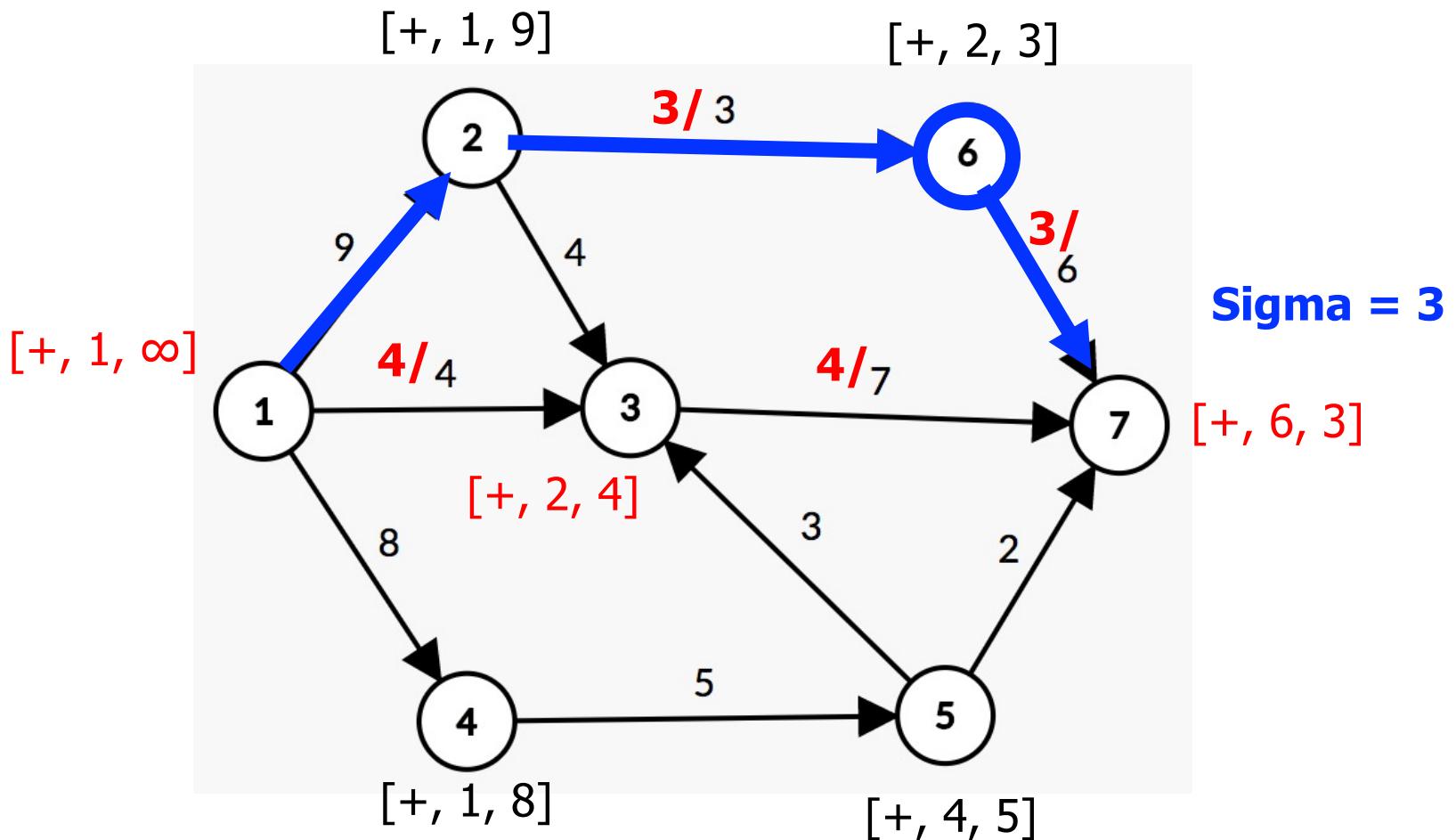
# Lắp 2

- Tăng luồng



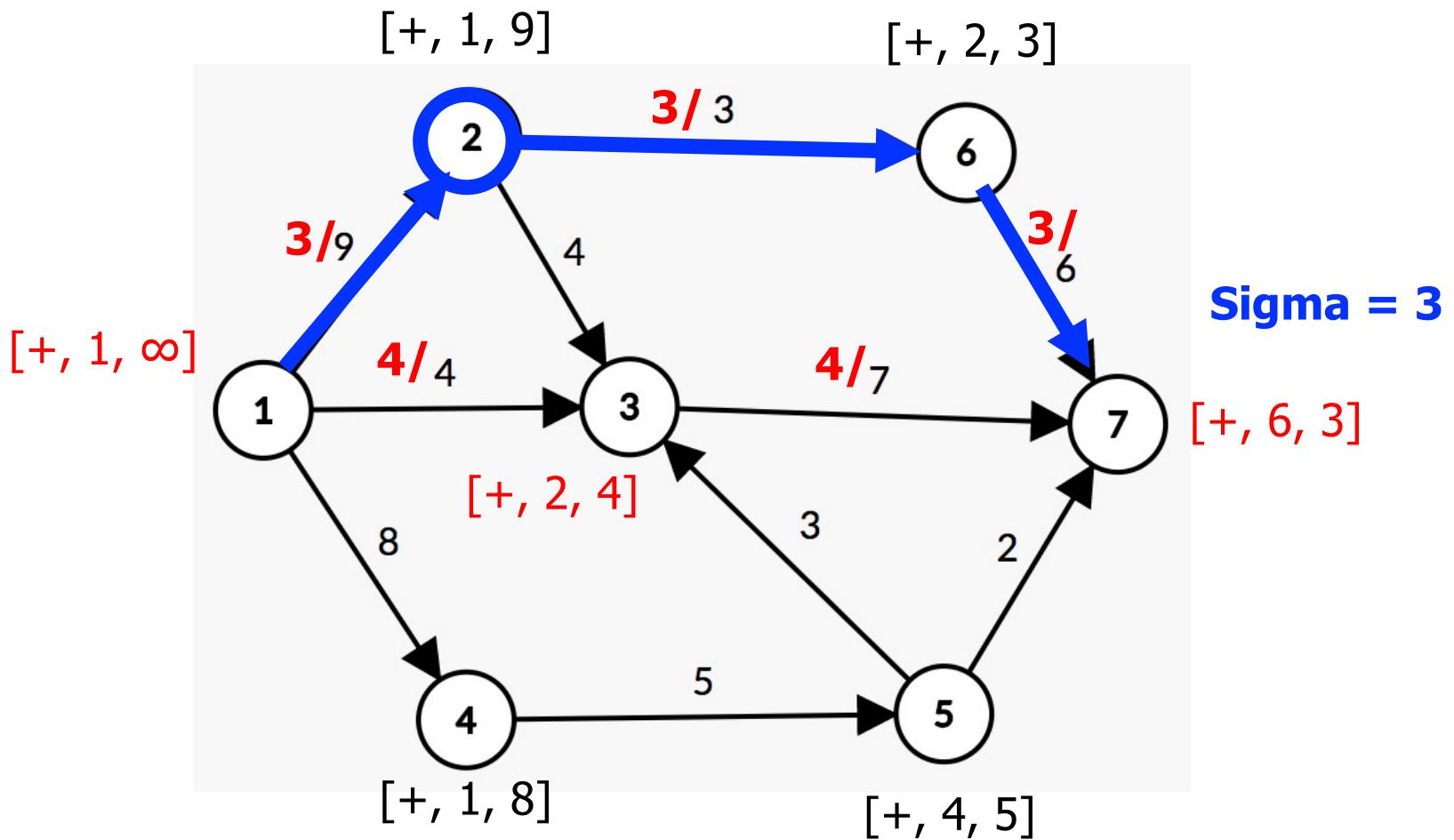
# Lắp 2

- Tăng luồng



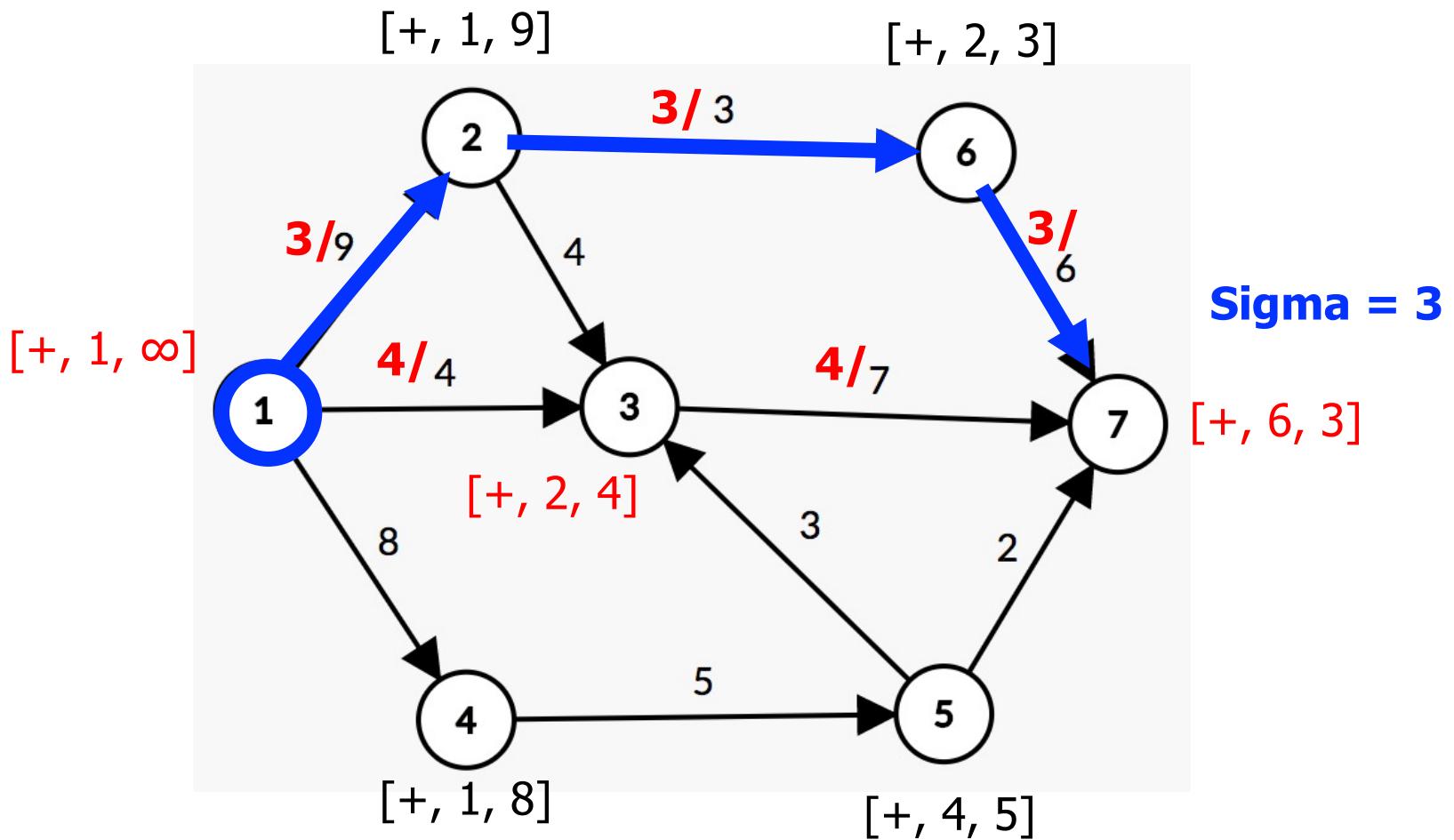
# Lắp 2

- Tăng luồng



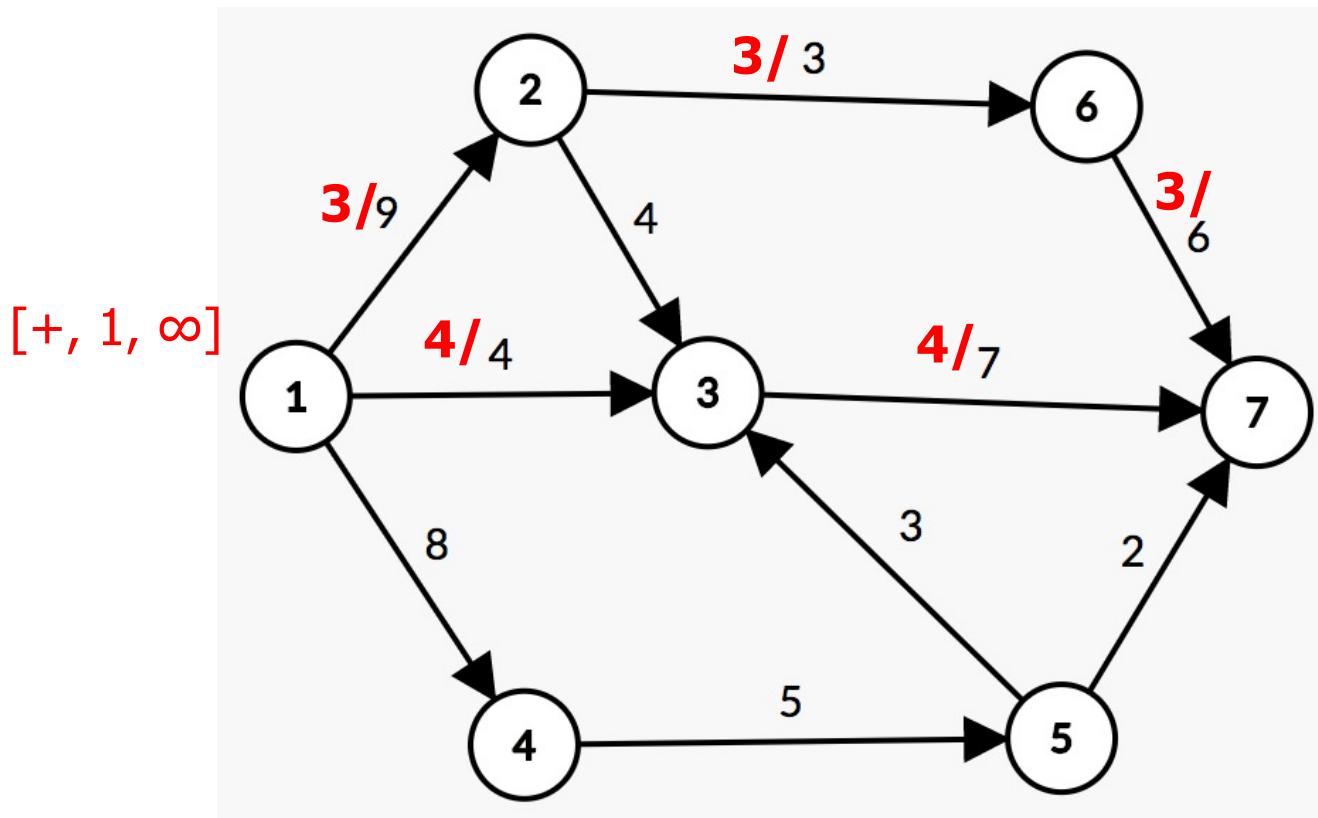
# Lắp 2

- Tăng luồng



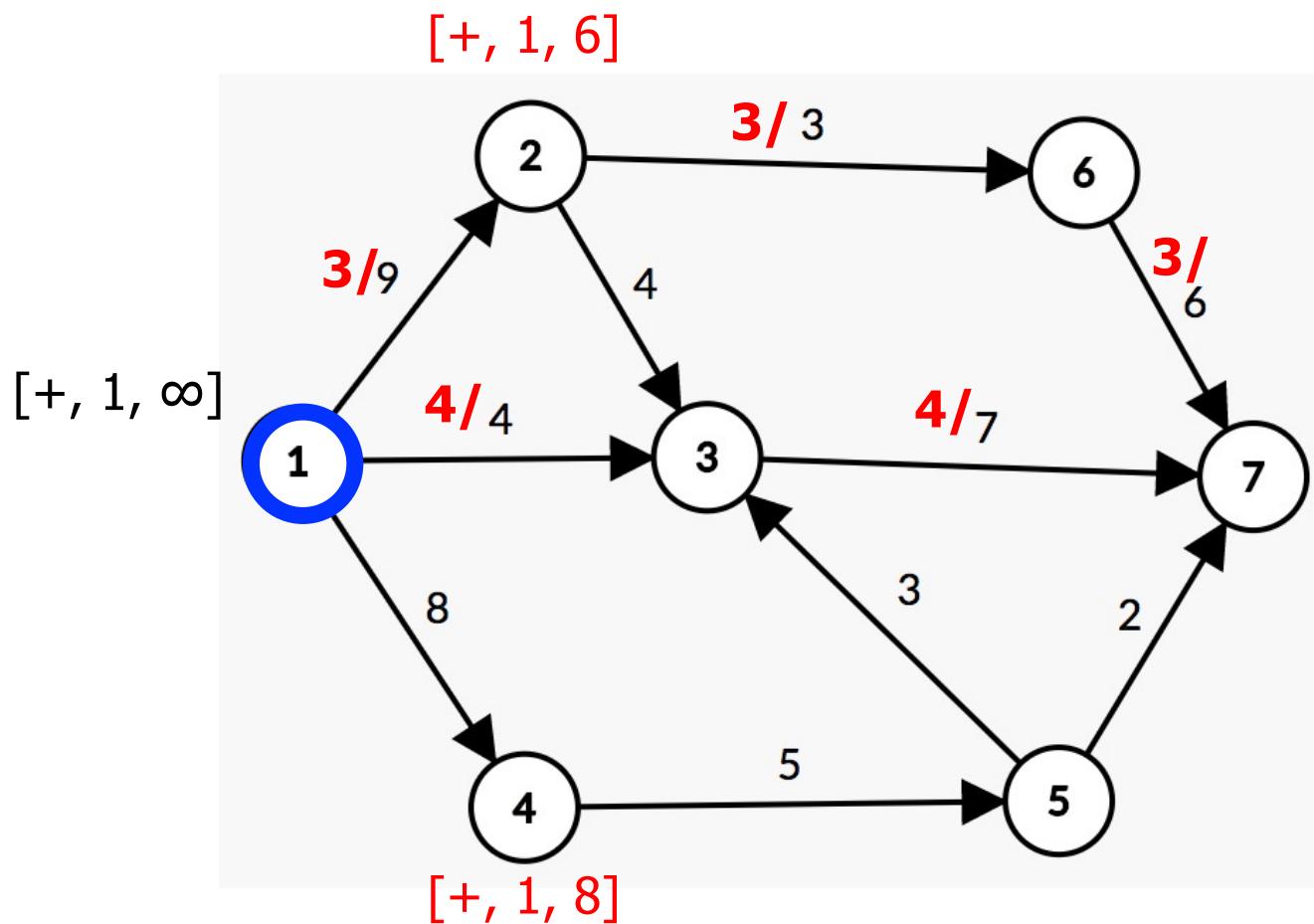
# Lắp 3

- Tìm đường tăng luồng



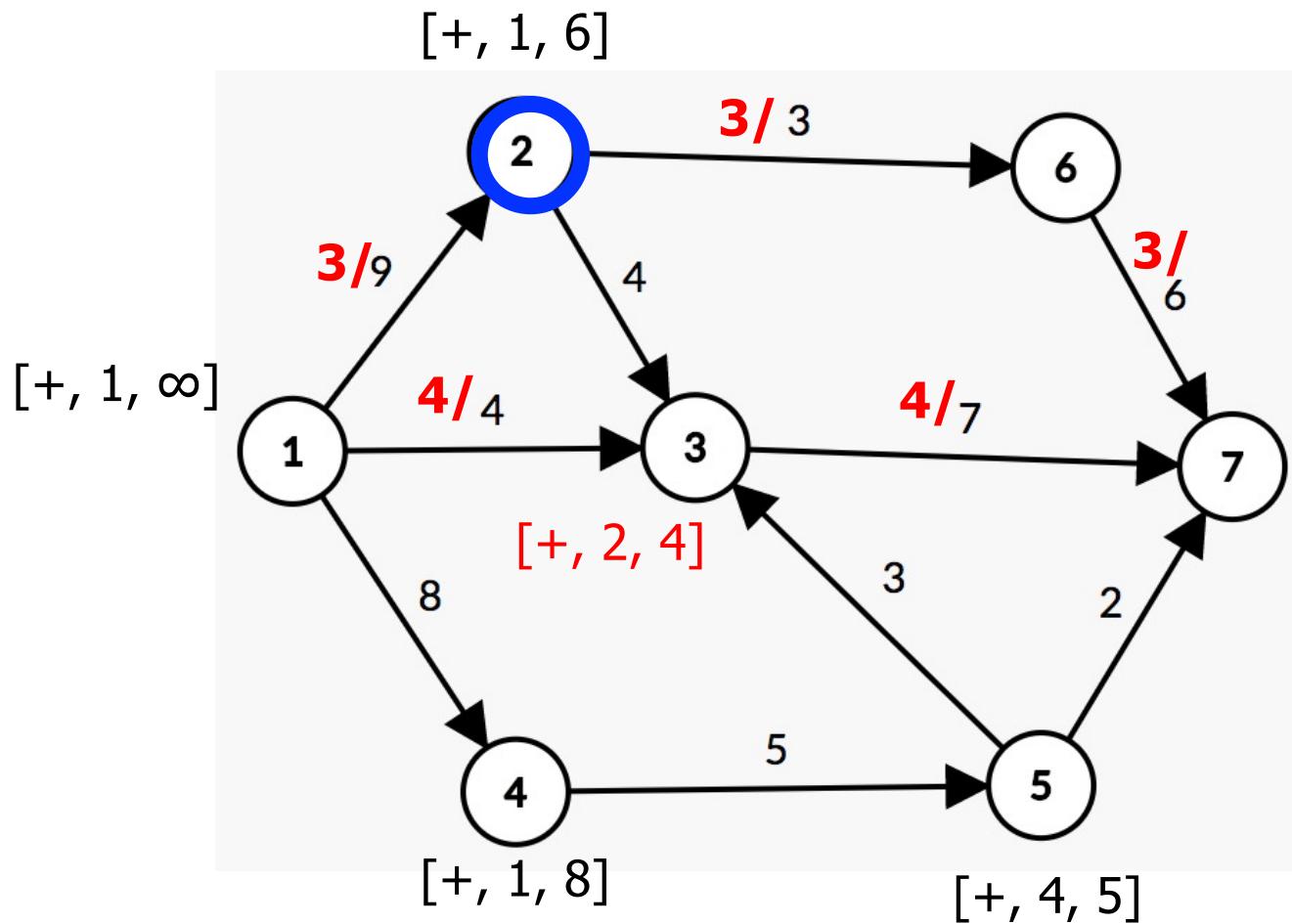
# Lắp 3

- Tìm đường tăng luồng



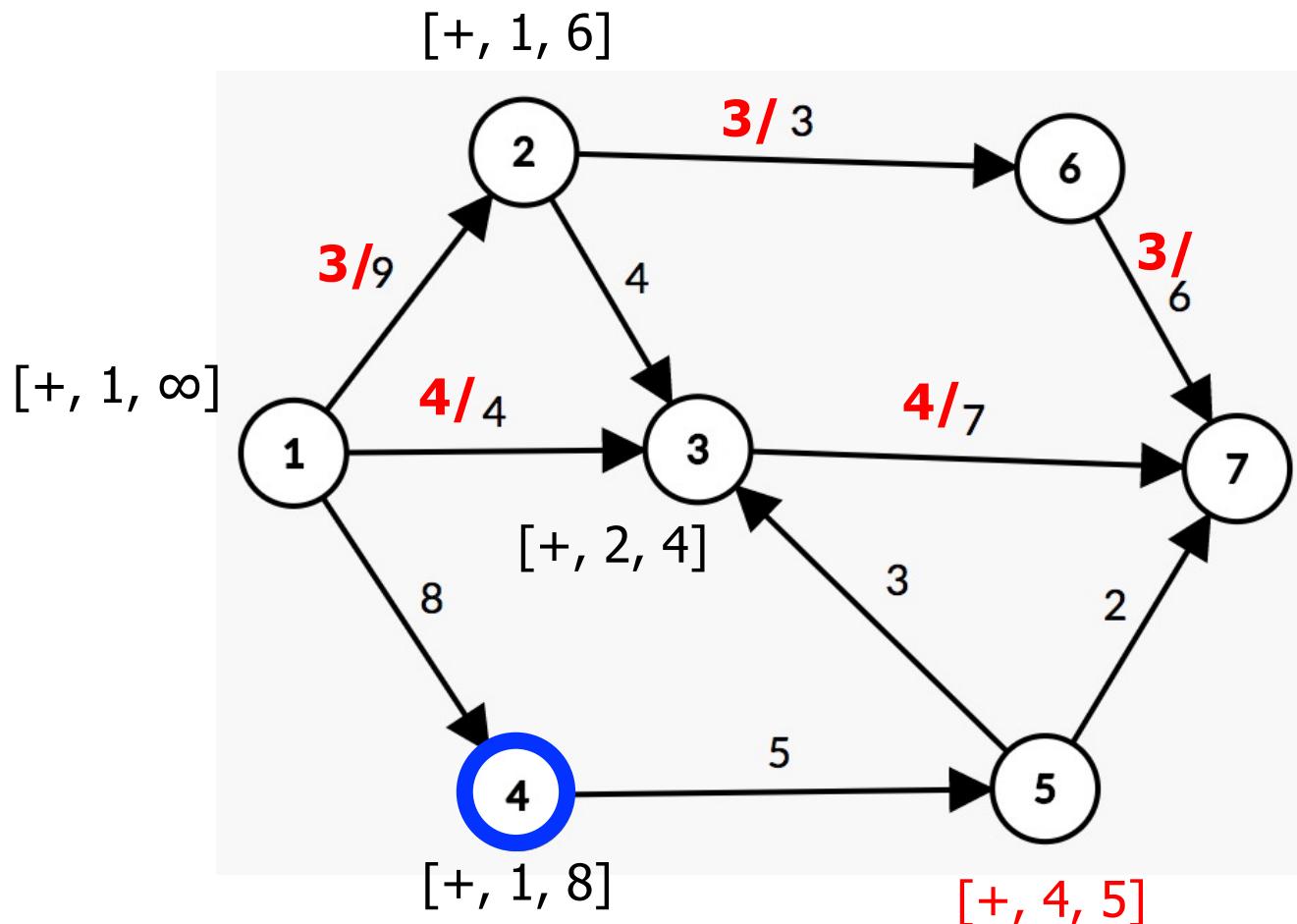
# Lắp 3

- Tìm đường tăng luồng



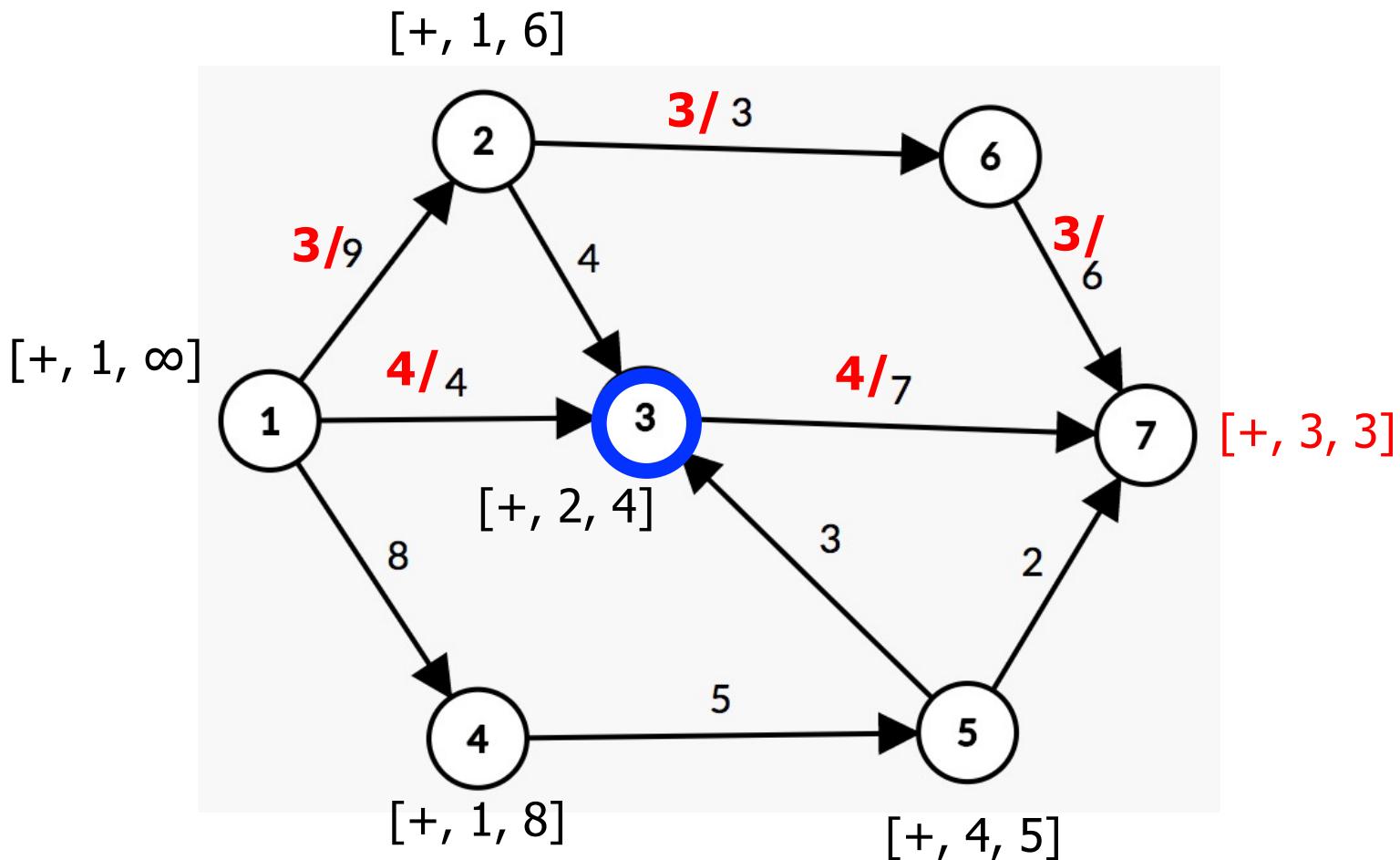
# Lắp 3

- Tìm đường tăng luồng



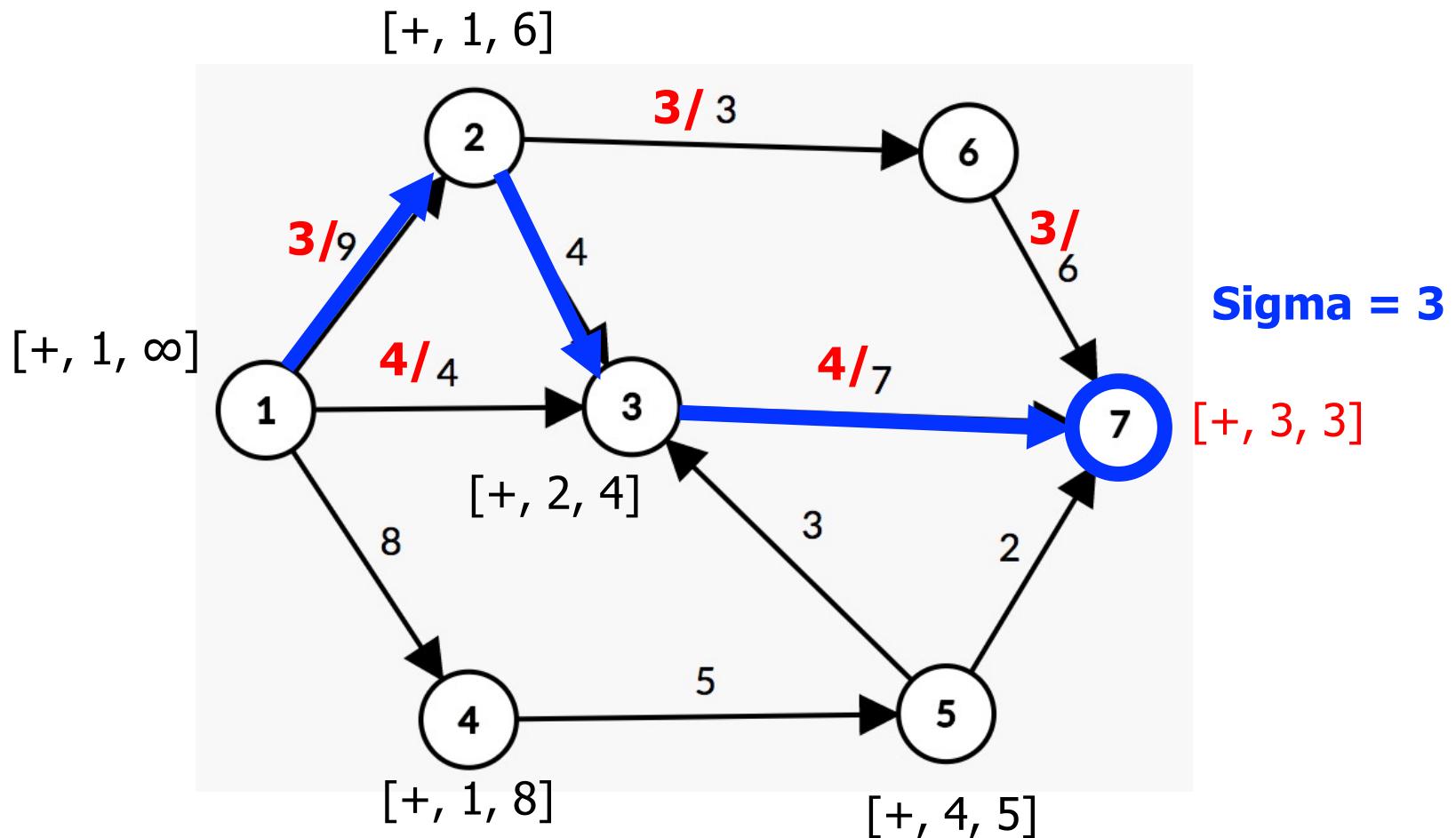
# Lắp 3

- Tìm đường tăng luồng, t có nhãn => dừng



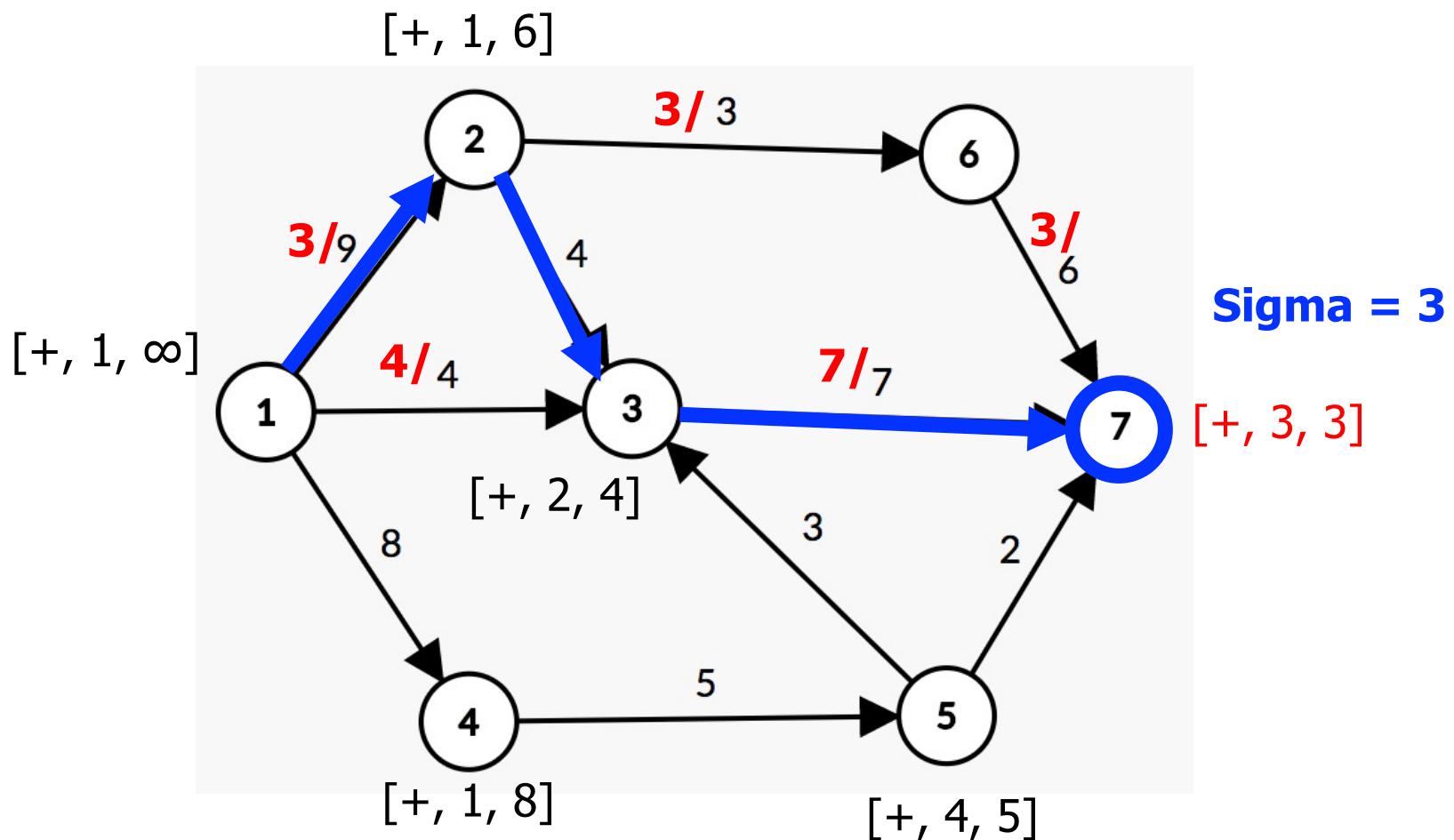
# Lắp 3

- Đường tăng luồng:  $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 7$



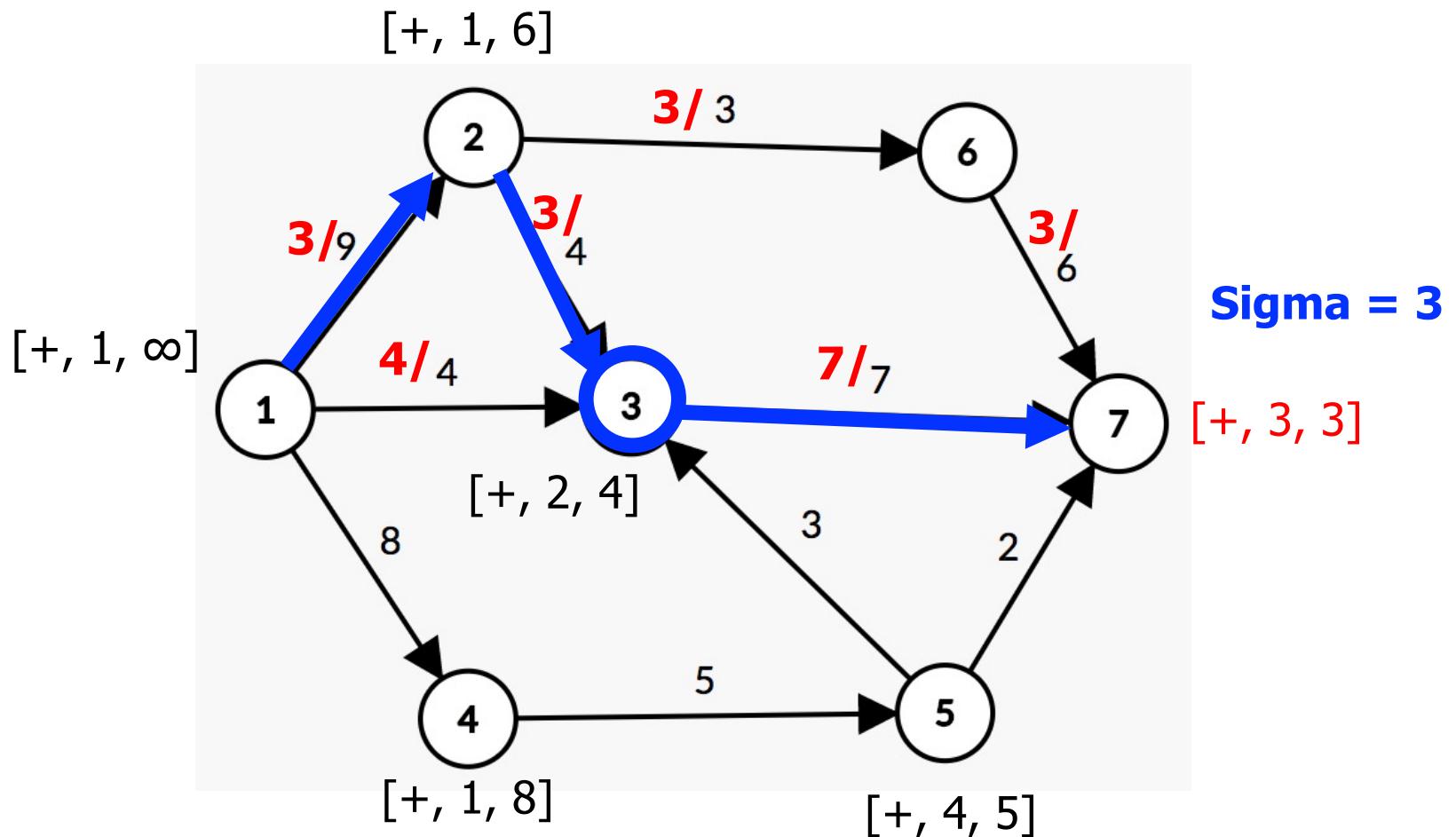
# Lắp 3

- Tăng luồng



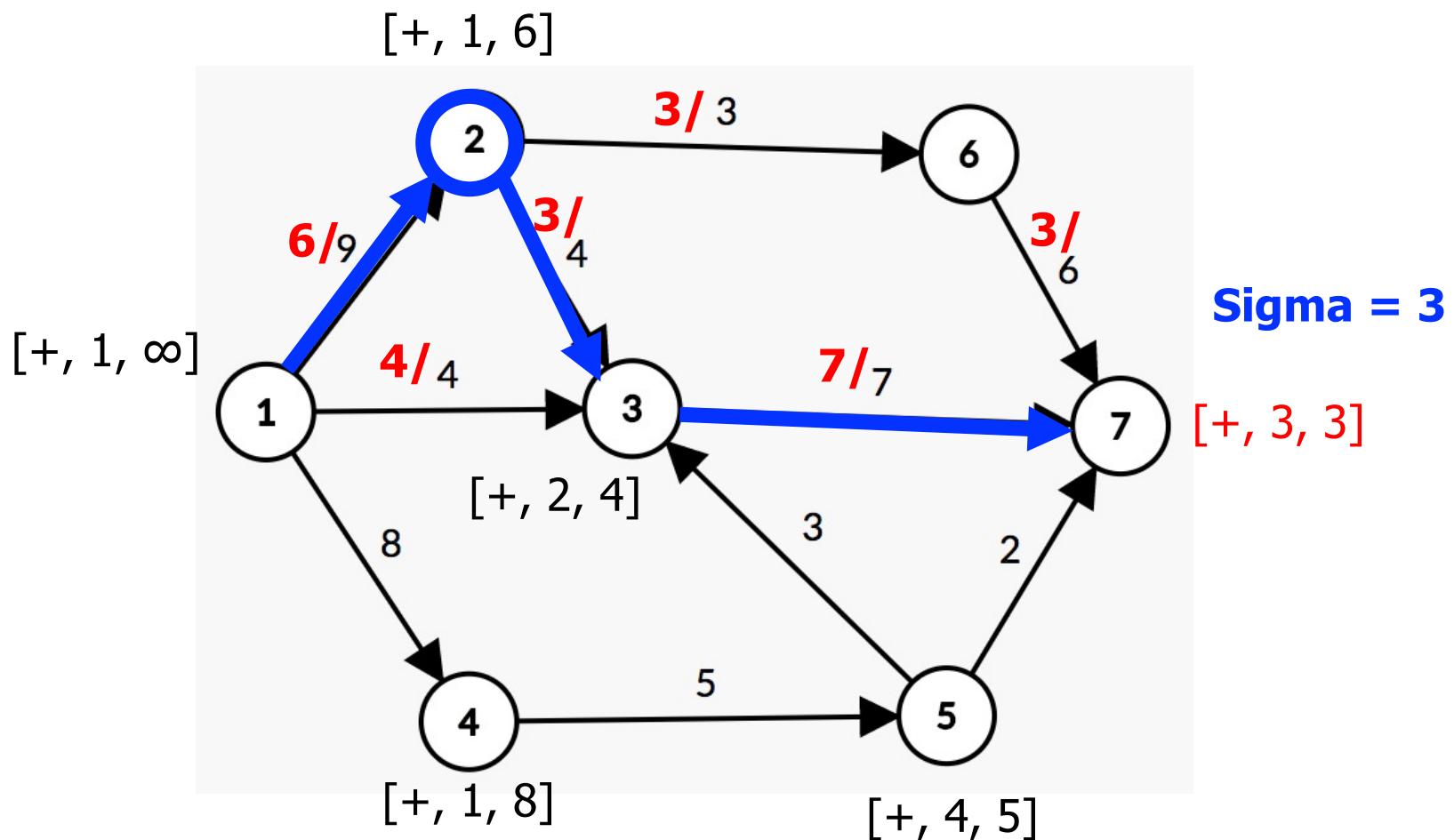
# Lắp 3

- Tăng luồng



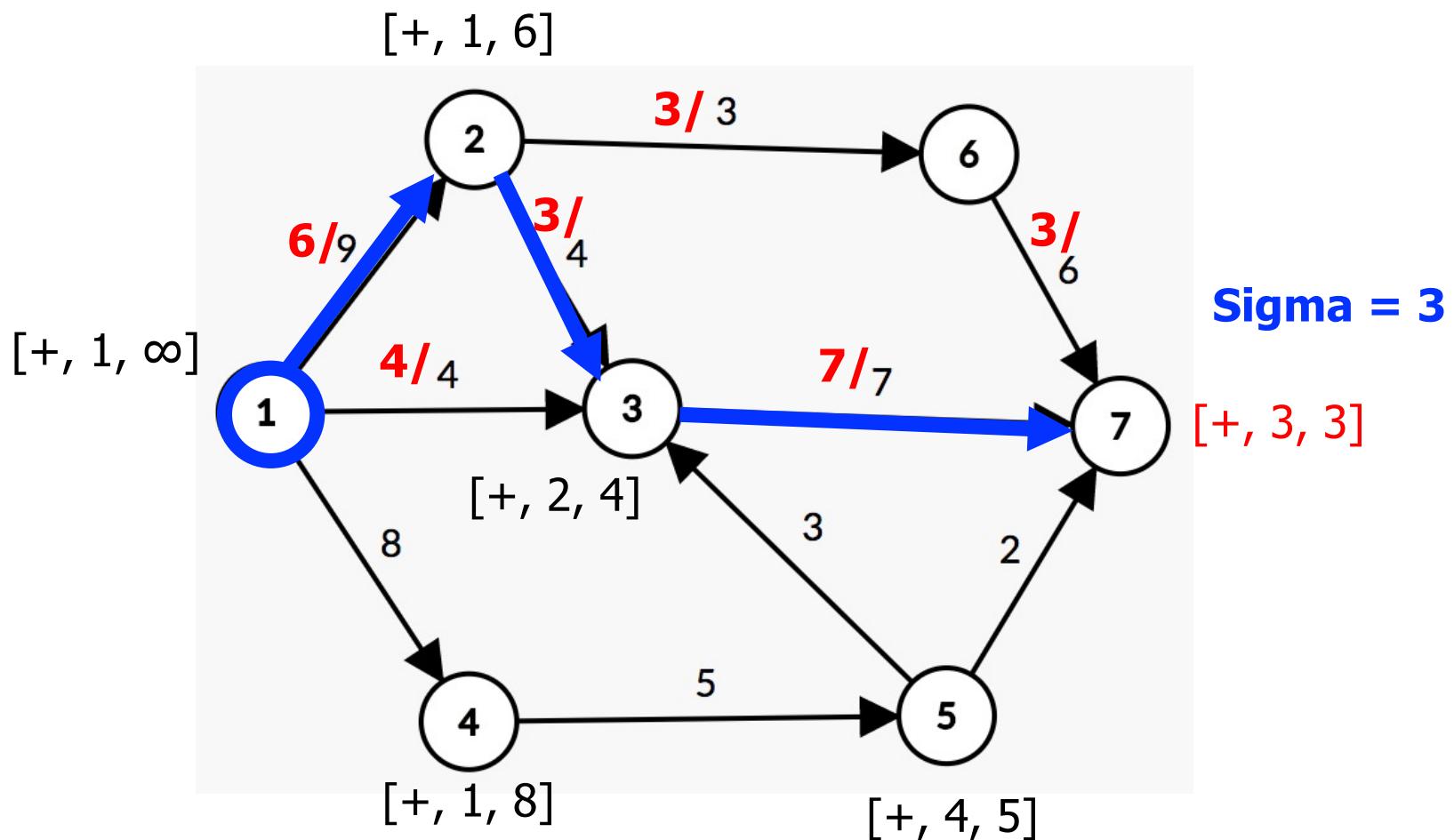
# Lắp 3

- Tăng luồng



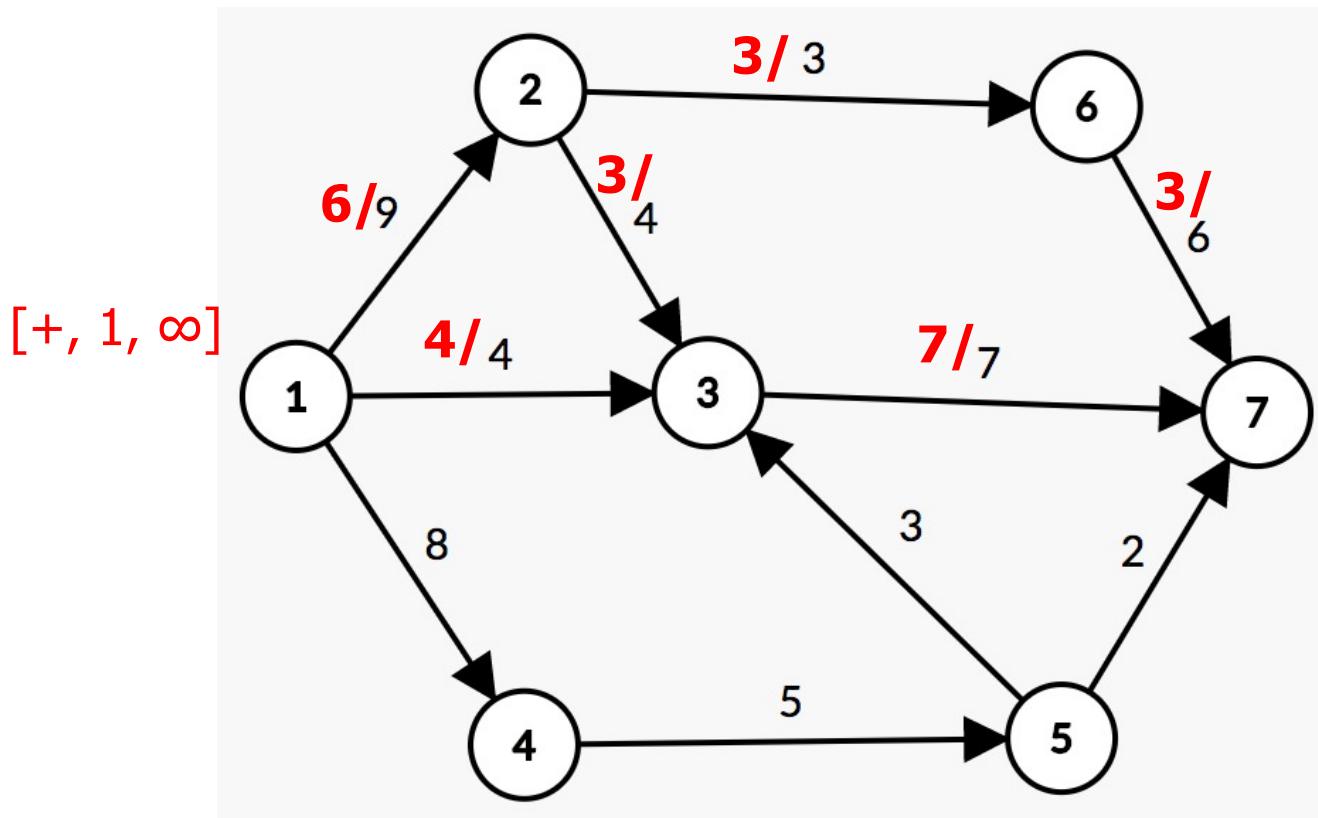
# Lắp 3

- Tăng luồng



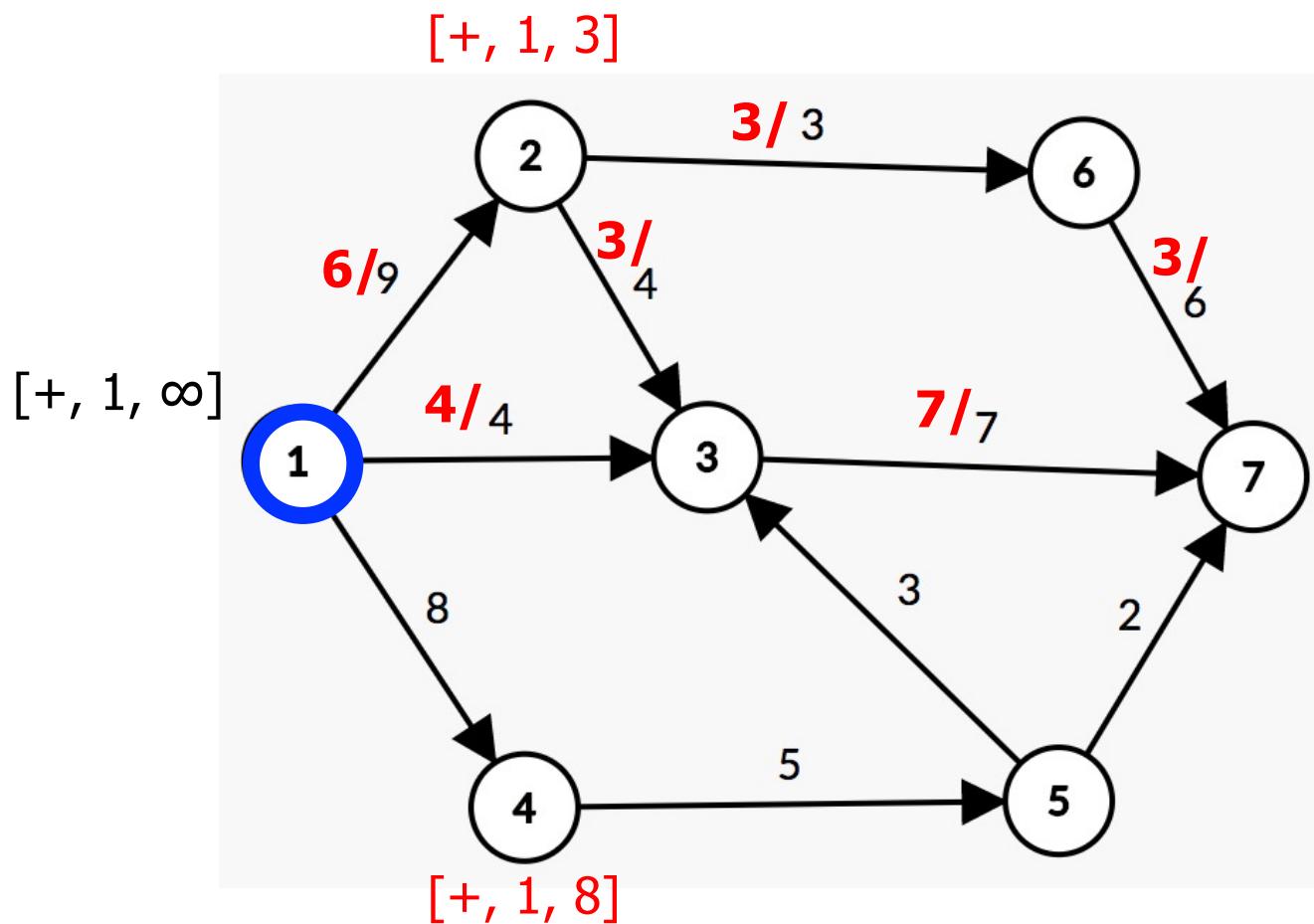
# Lắp 4

- Tìm đường tăng luồng



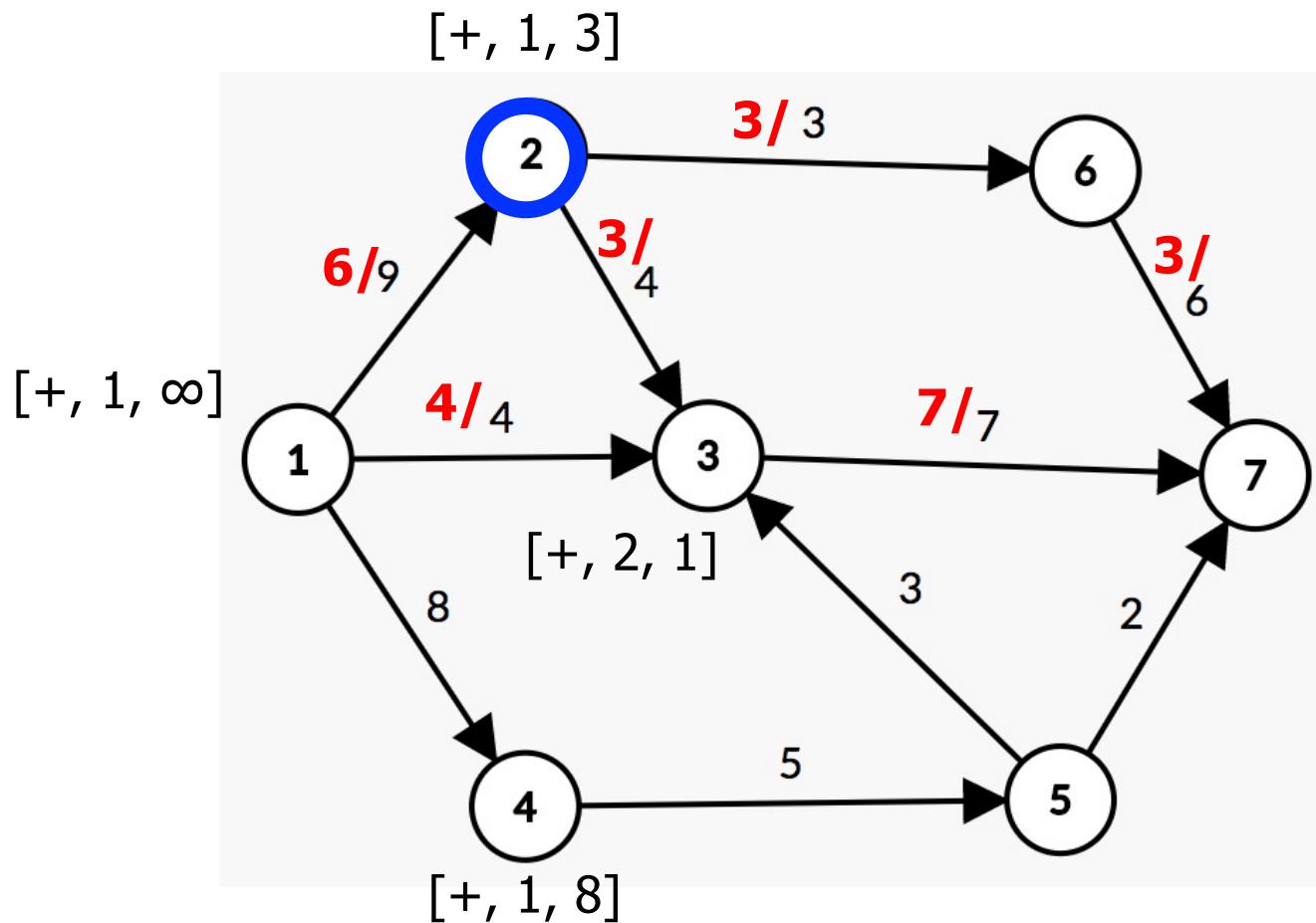
# Lắp 4

- Tìm đường tăng luồng



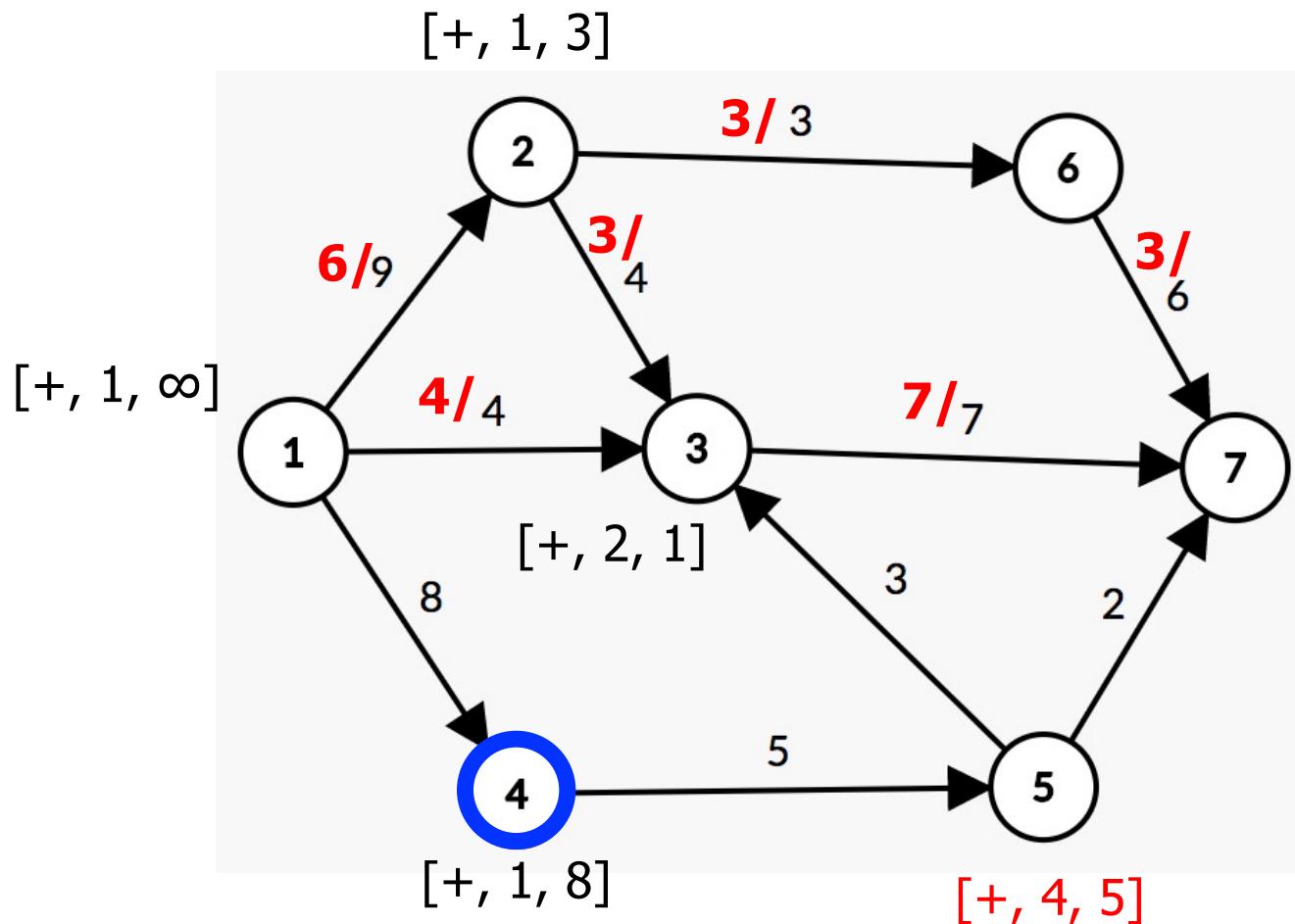
# Lắp 4

- Tìm đường tăng luồng



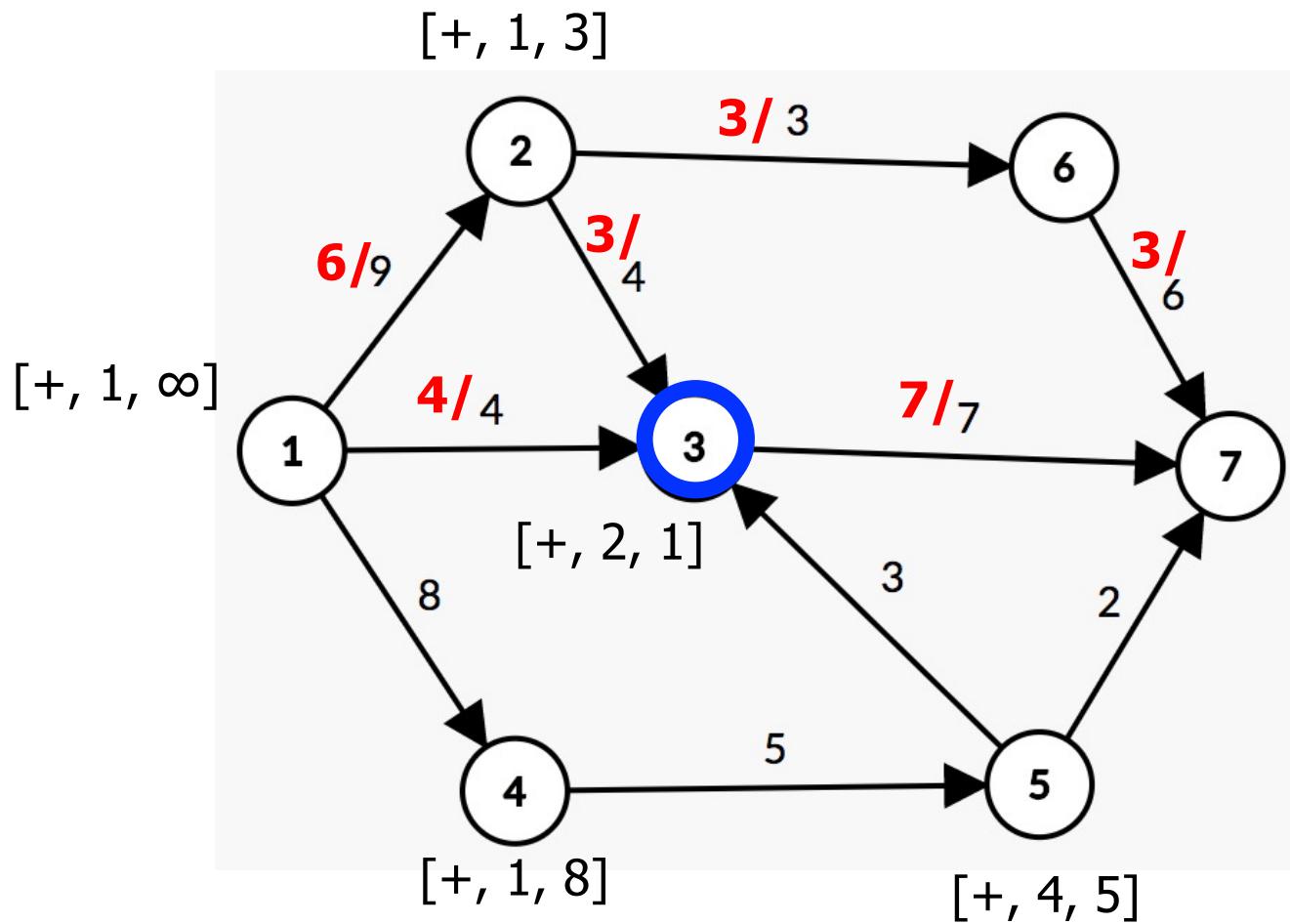
# Lắp 4

- Tìm đường tăng luồng



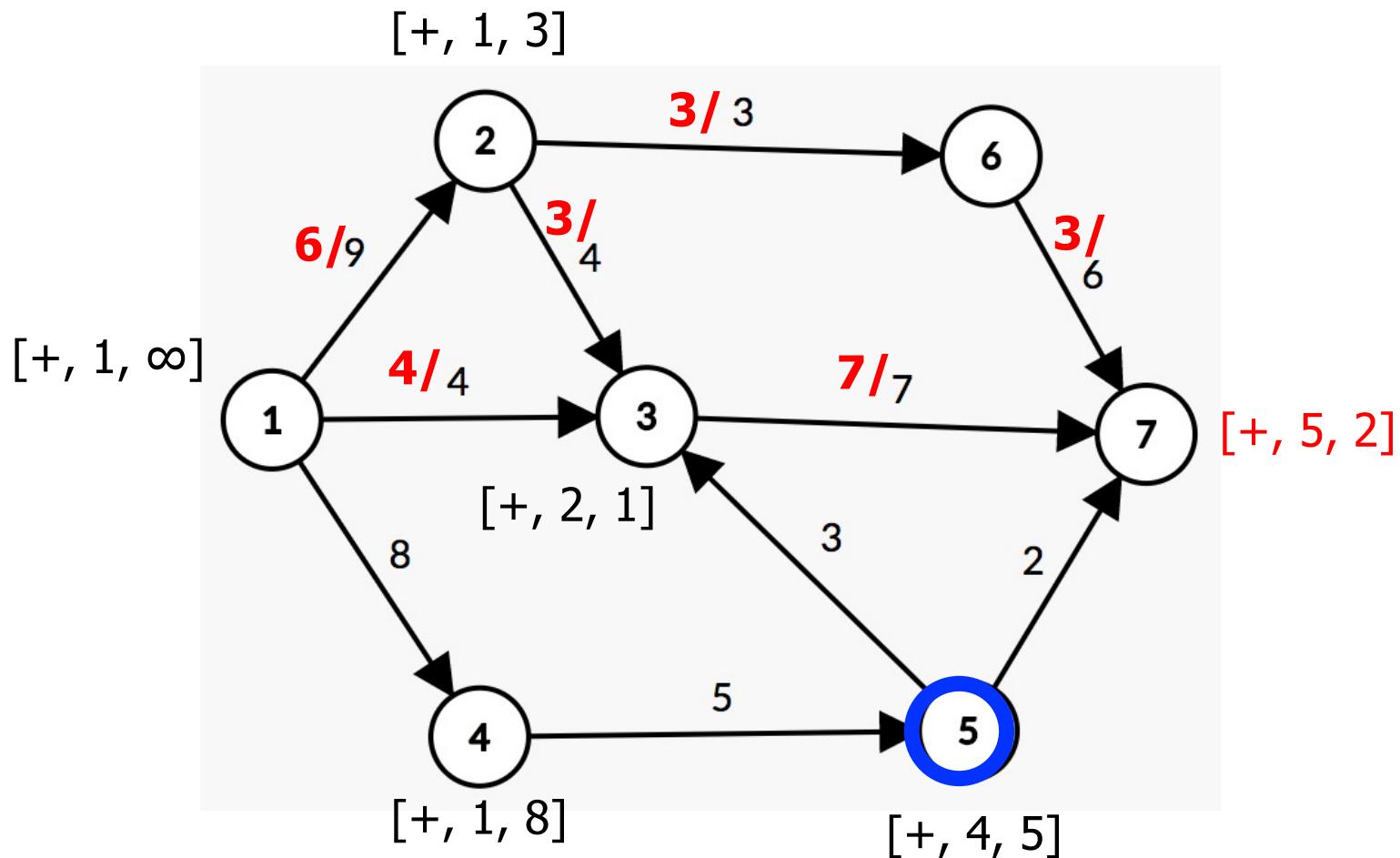
# Lắp 4

- Tìm đường tăng luồng



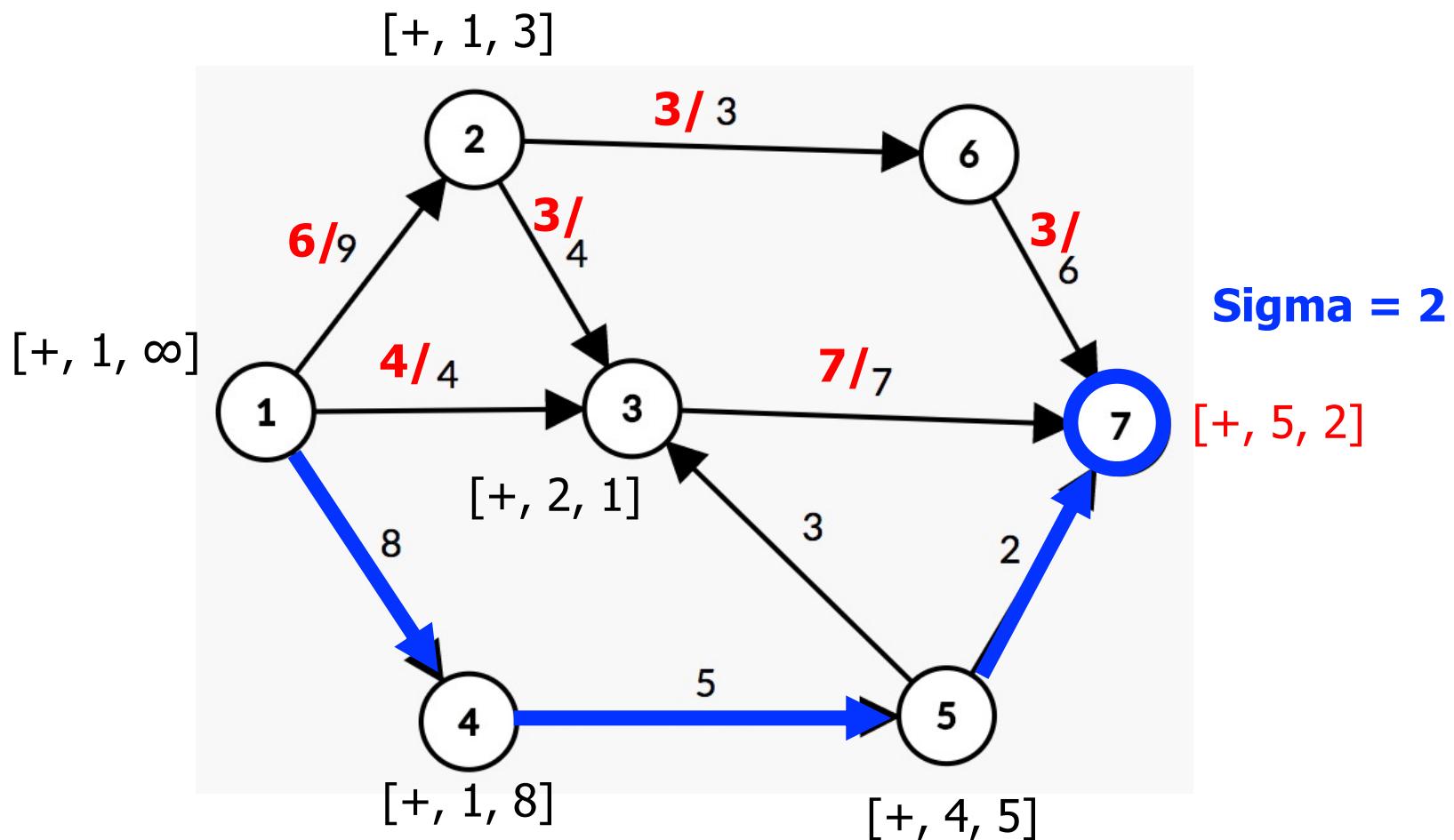
# Lắp 4

- Tìm đường tăng luồng, t có nhãn => dừng



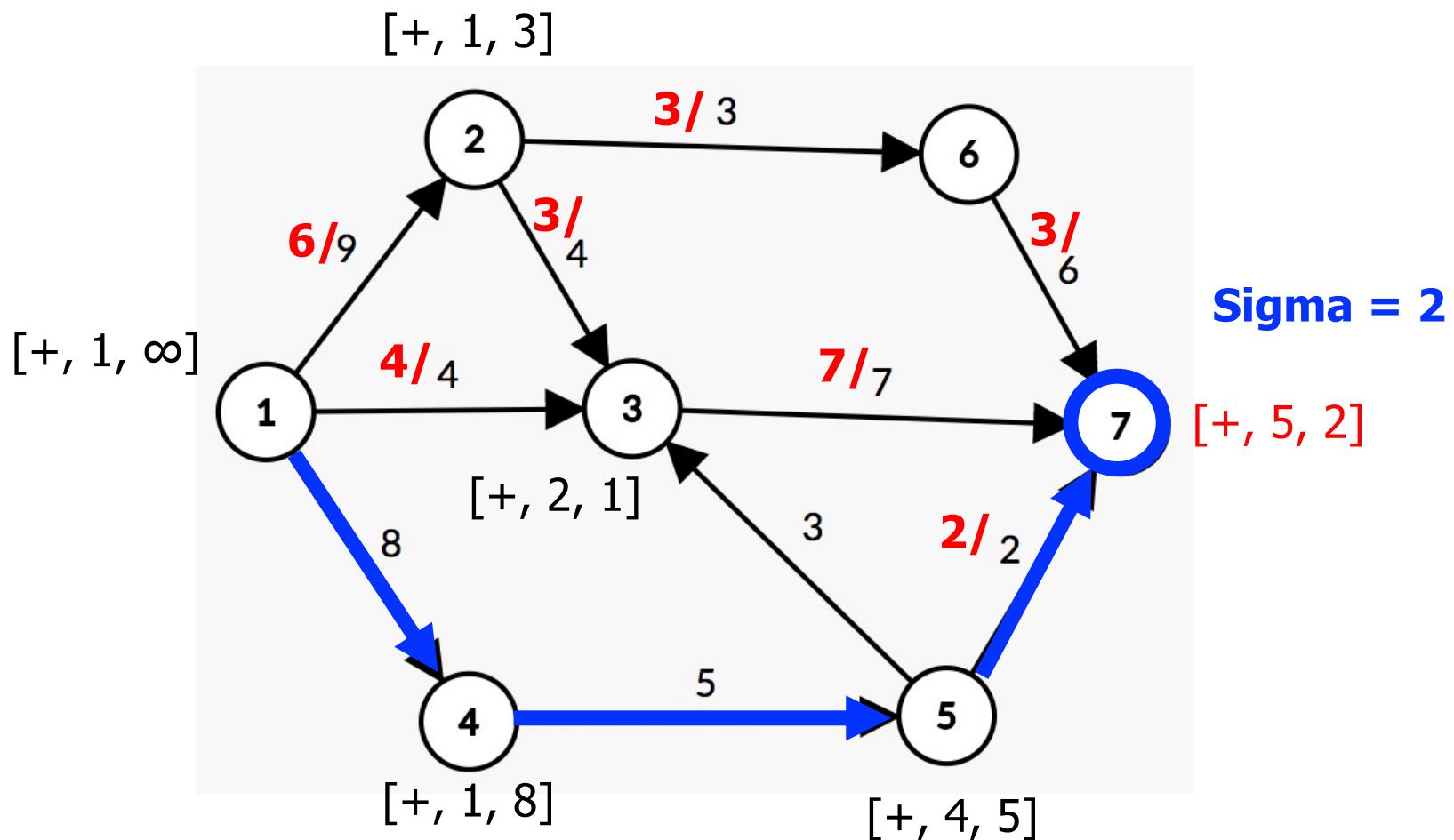
# Lắp 4

- Đường tăng luồng:  $1 \Rightarrow 4 \Rightarrow 5 \Rightarrow 7$



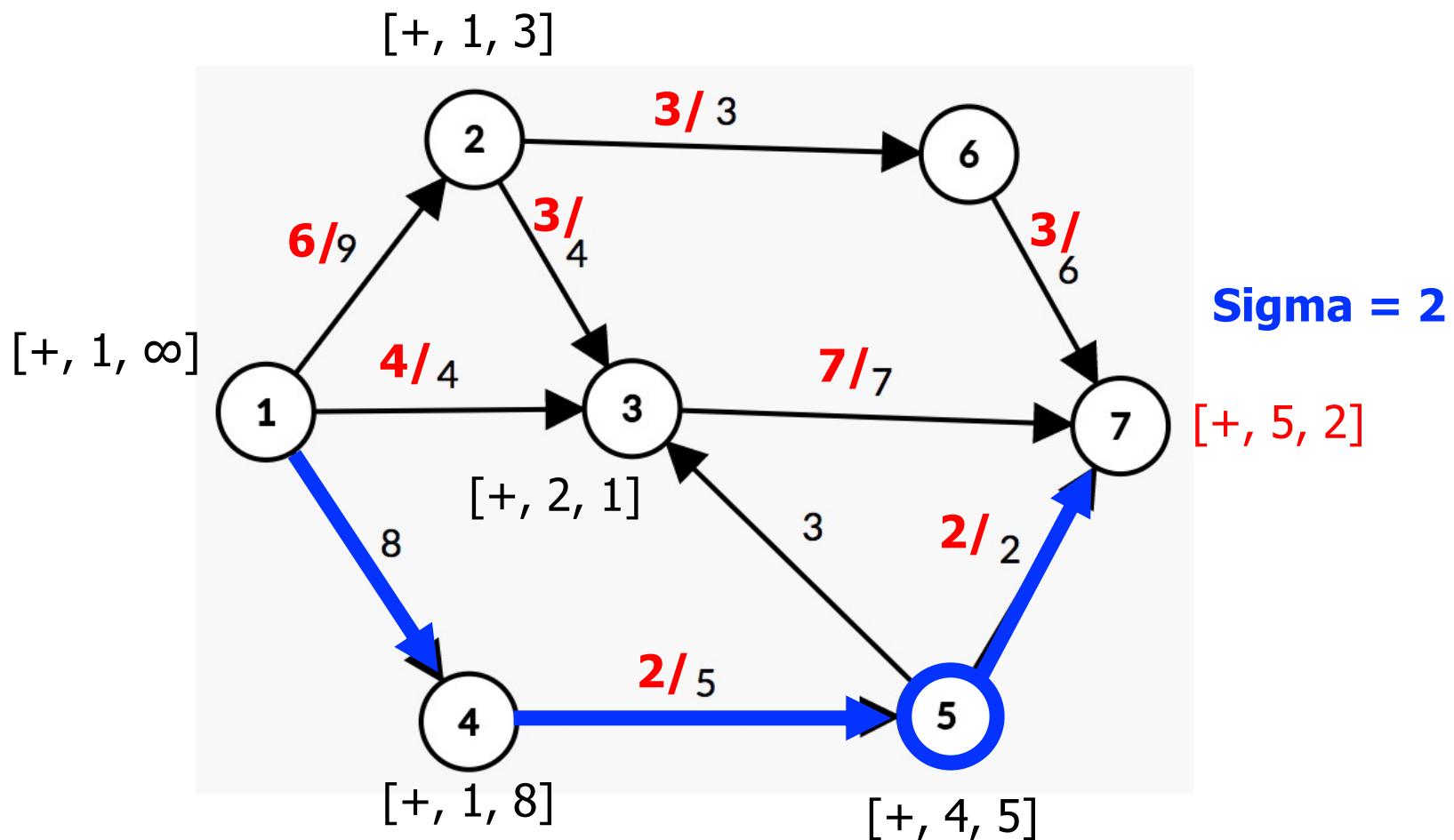
# Lắp 4

- Tăng luồng



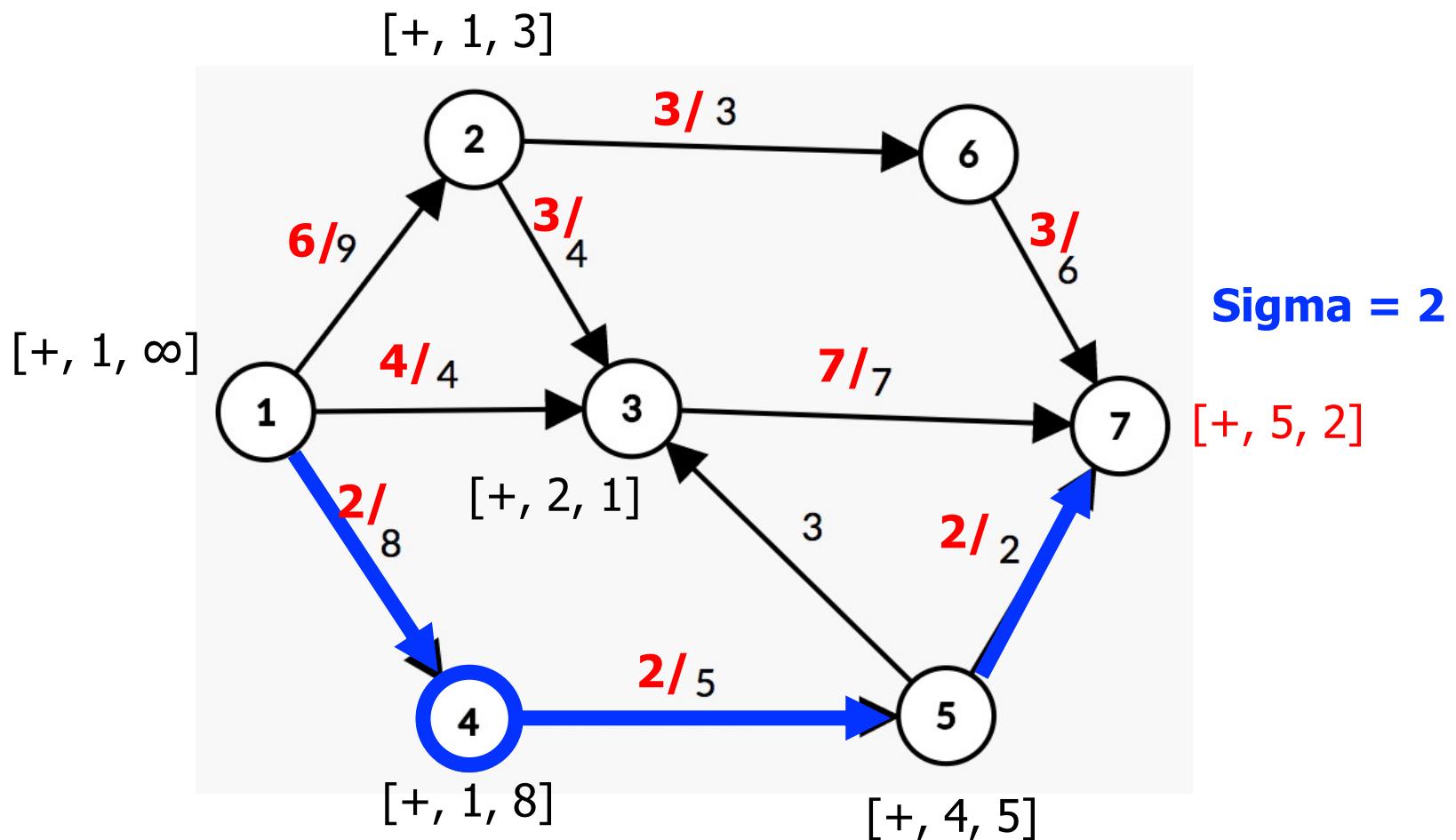
# Lắp 4

- Tăng luồng



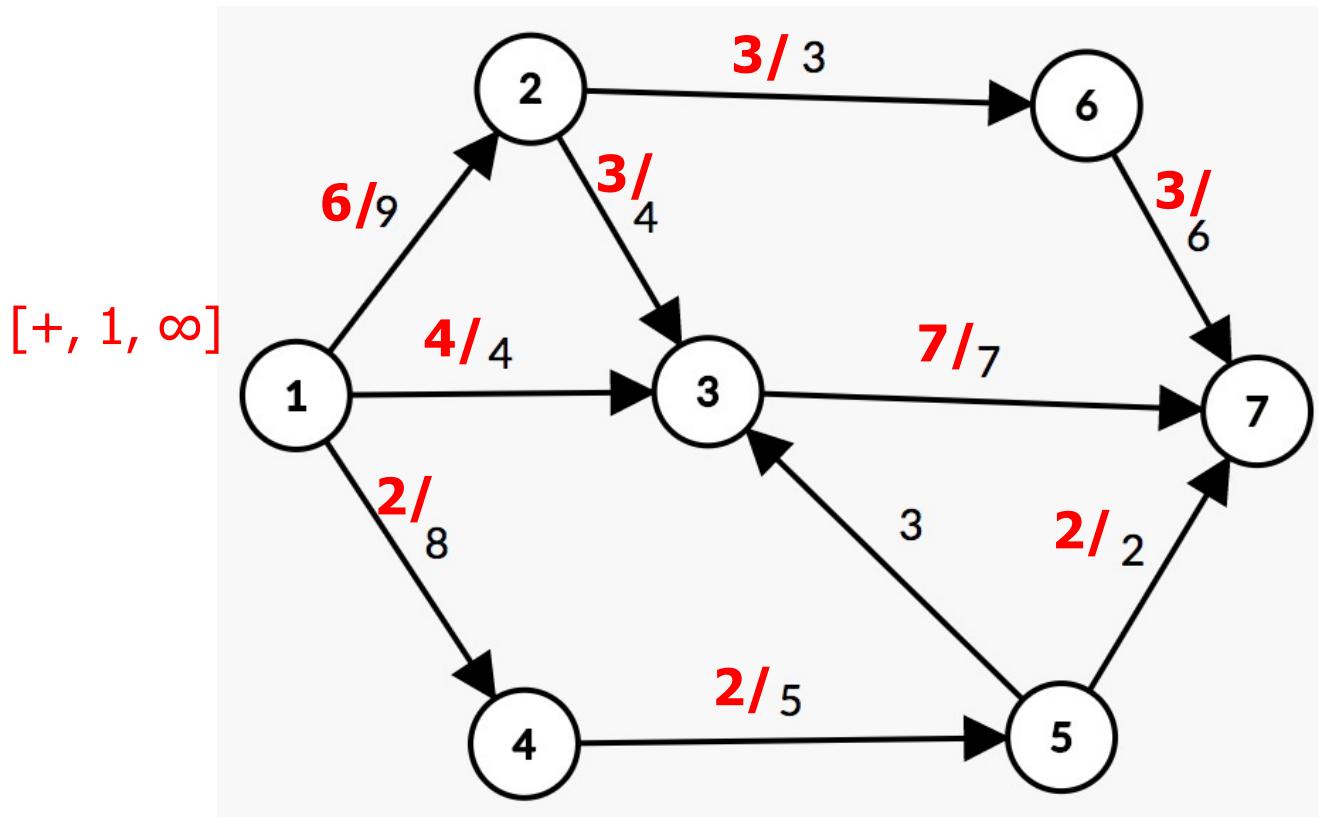
# Lắp 4

- Tăng luồng



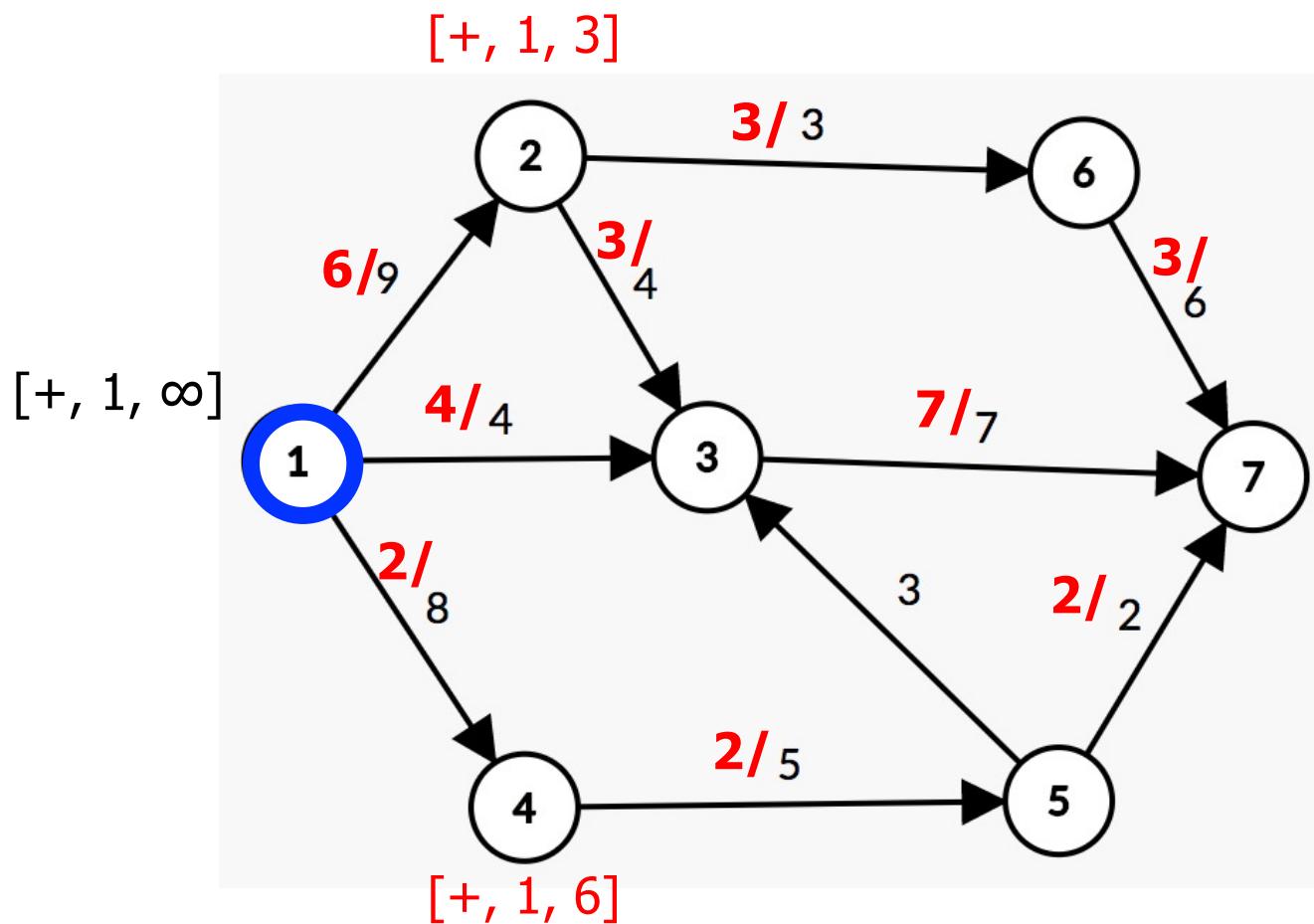
# Lắp 5

- Tìm đường tăng luồng



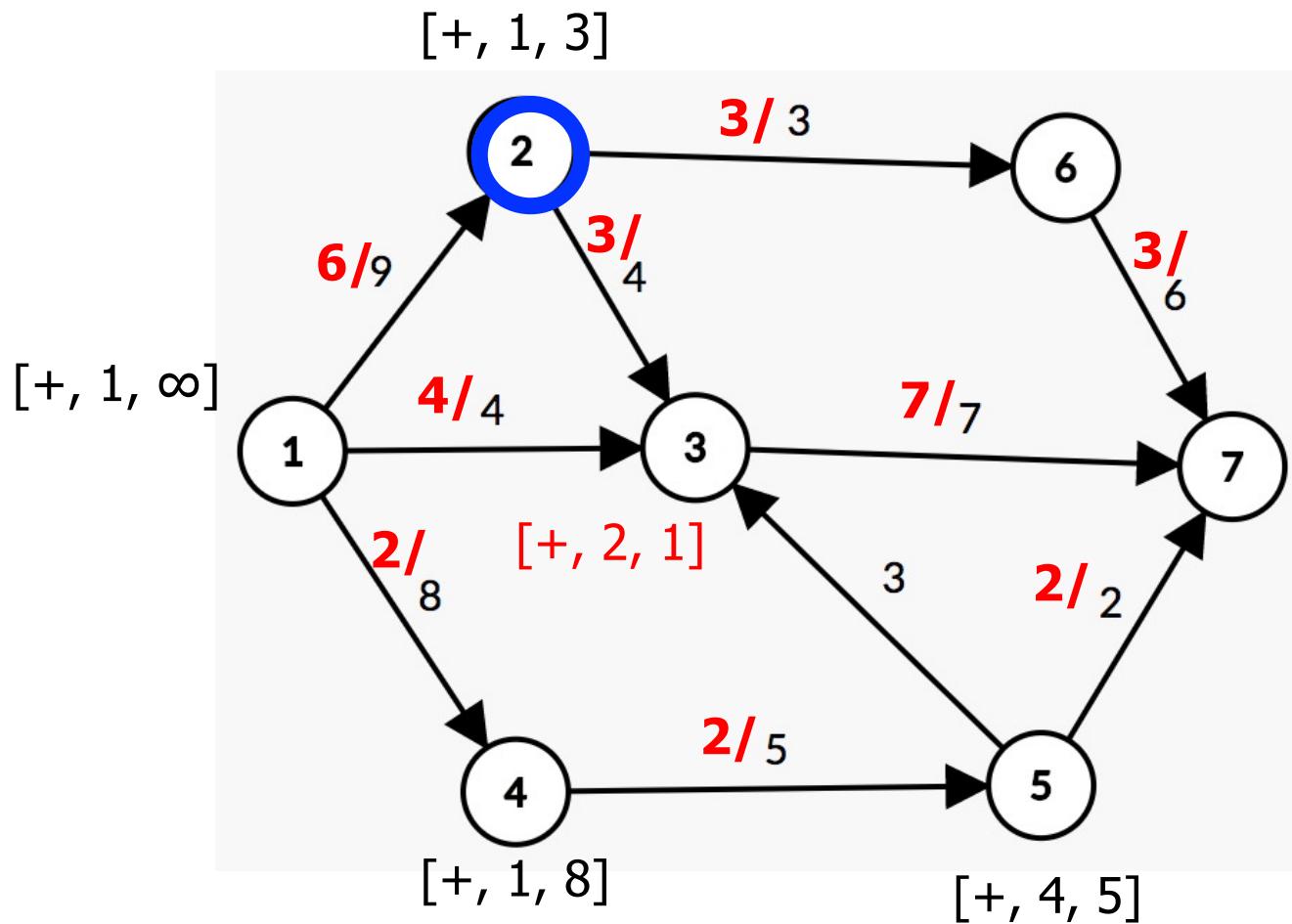
# Lăp 5

- Tìm đường tăng luồng



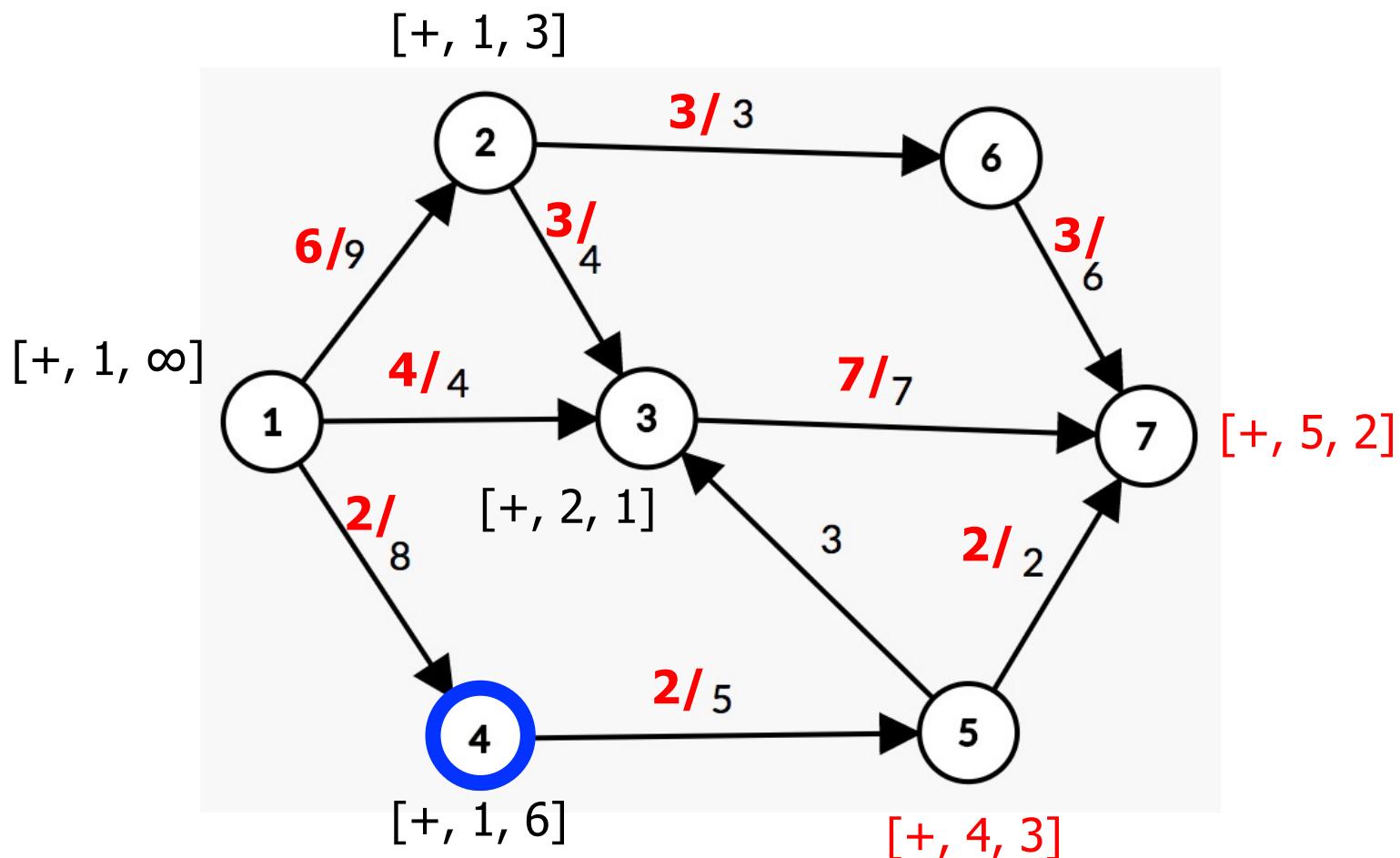
# Lắp 5

- Tìm đường tăng luồng



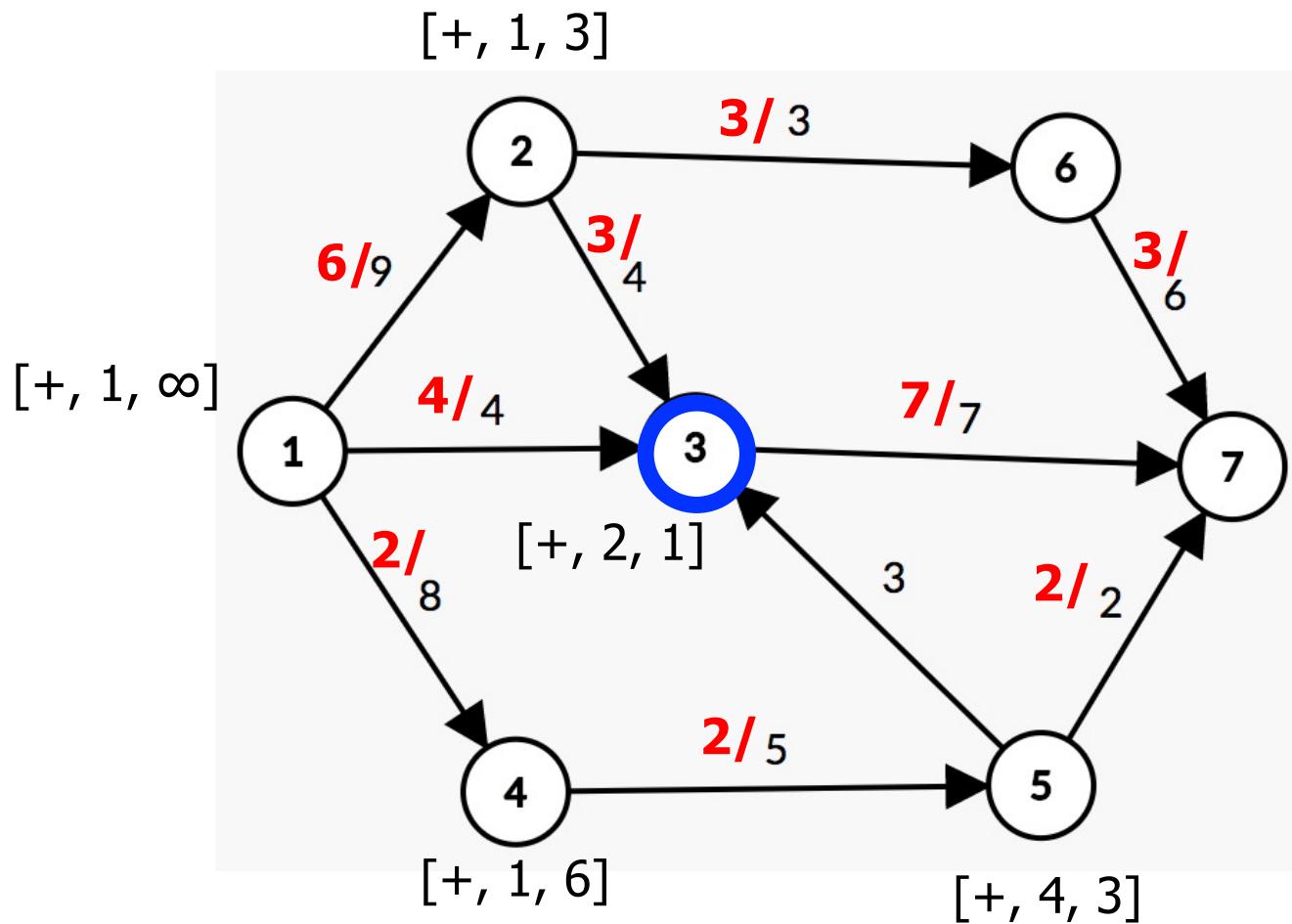
# Lắp 5

- Tìm đường tăng luồng



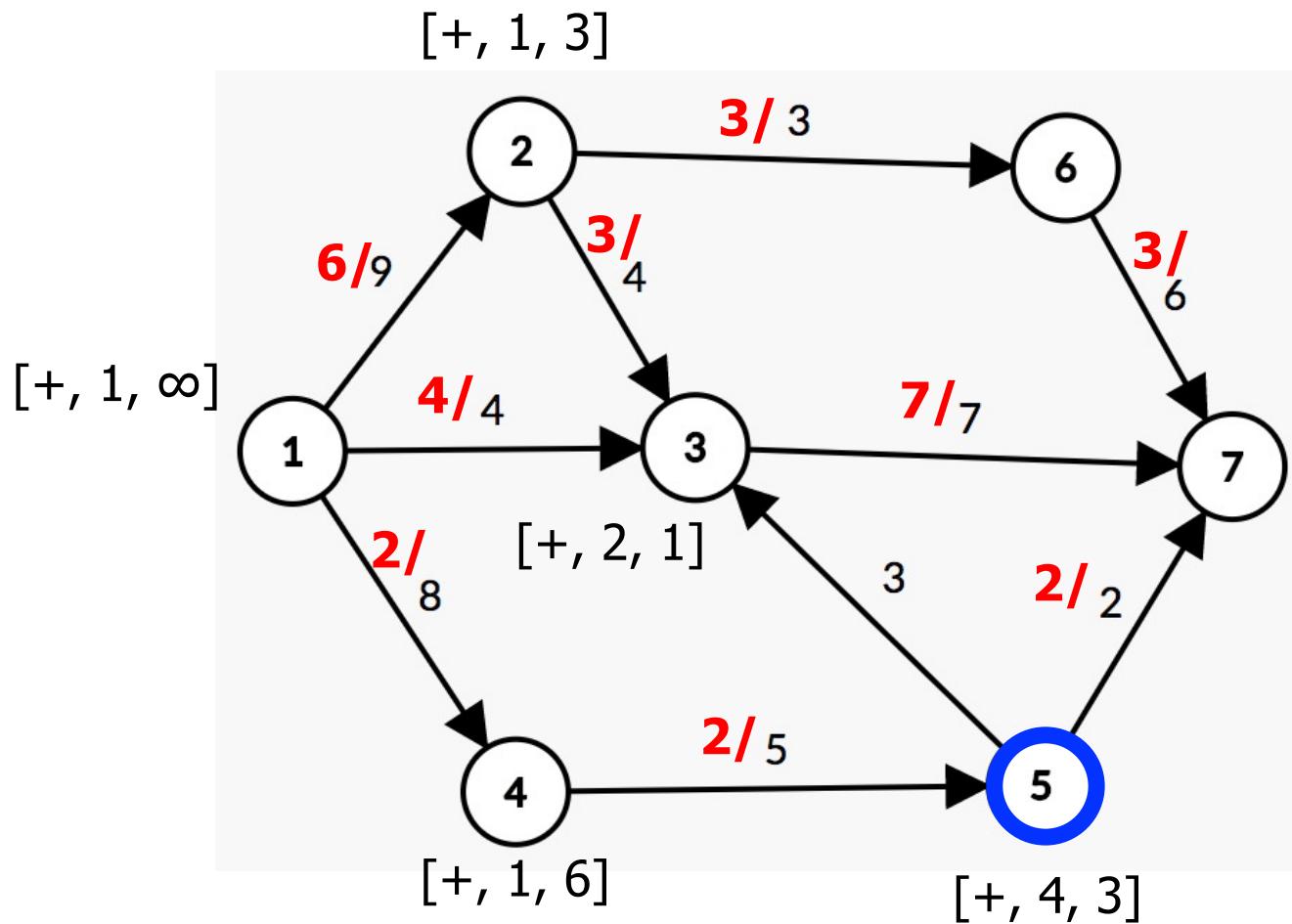
# Lăp 5

- Tìm đường tăng luồng



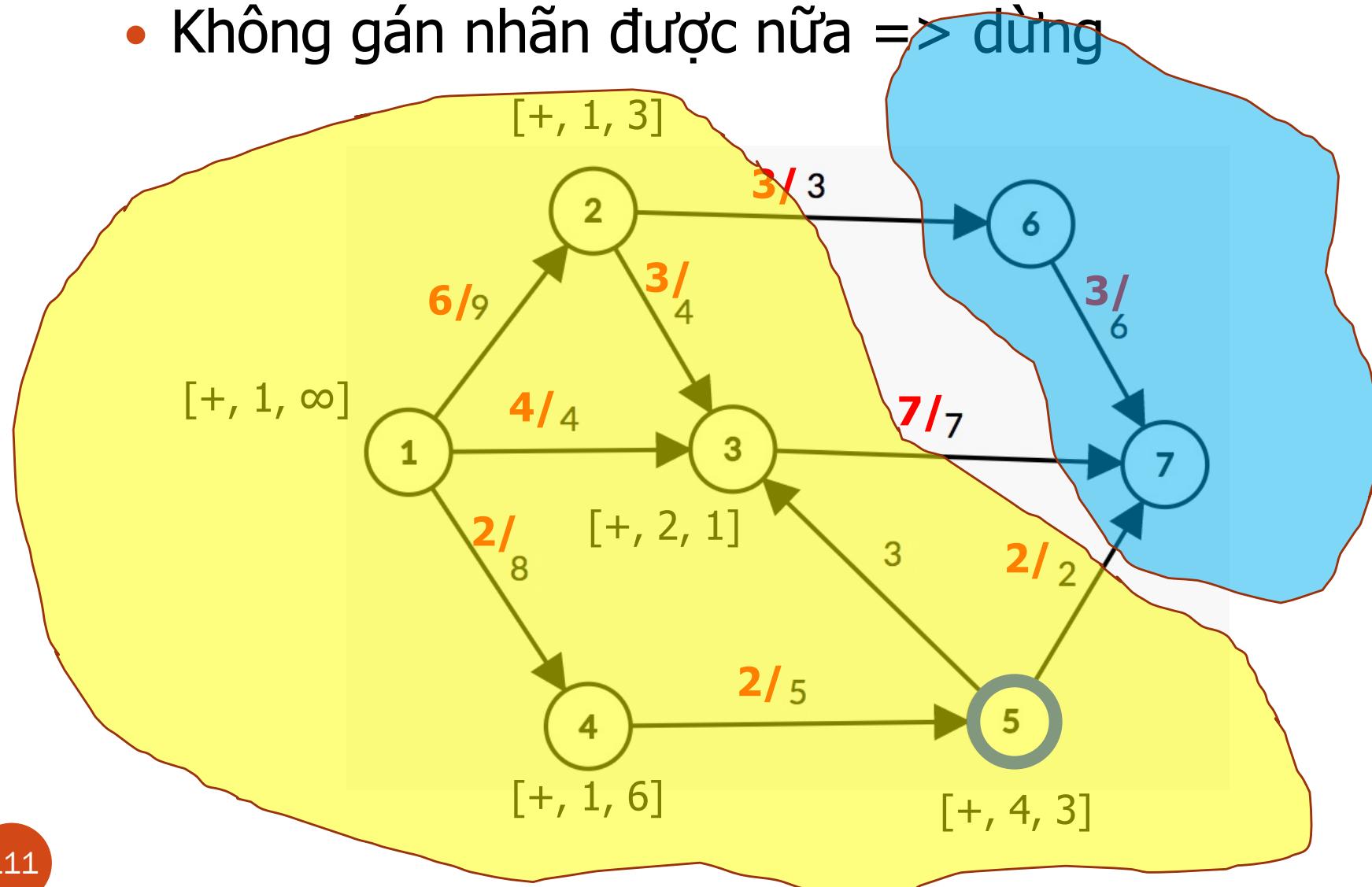
# Lặp 5

- Không gán nhãn được nữa => dừng



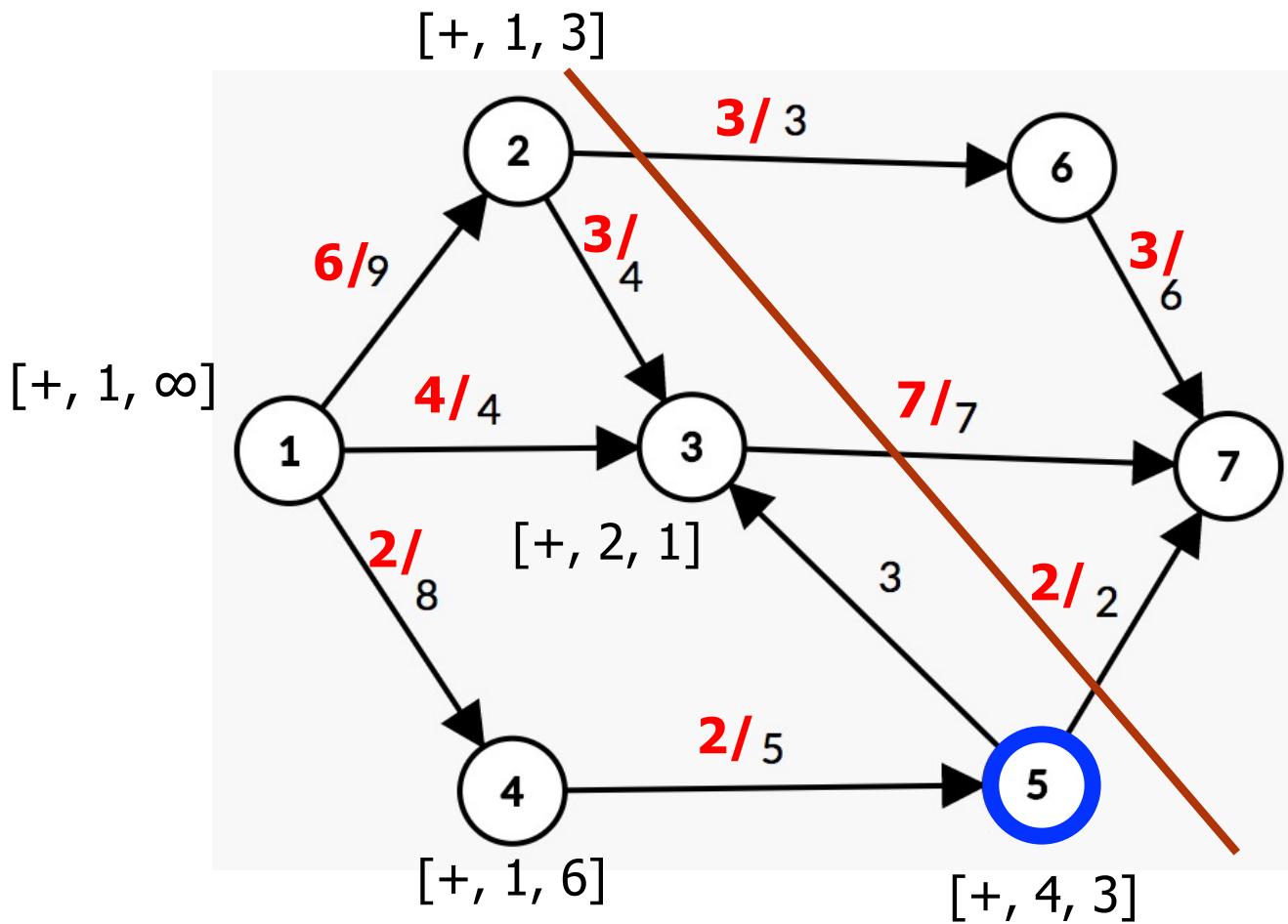
# Lặp 5

- Không gán nhãn được nữa => dừng



# Lặp 5

- Không gán nhãn được nữa => dừng



# Kết quả

- Lát cắt hẹp nhất ( $S, T$ )
  - $S = \{1, 2, 3, 4, 5\}$
  - $T = \{6, 7\}$
- Giá trị luồng cực đại:
  - $|f|_{\max} = c(S, T) = 3 + 7 + 2 = 12$

# Cài đặt phương pháp Ford-Fulkerson

- Thuật toán Ford-Fulkerson thường được gọi là **phương pháp** (method) Ford-Fulkerson thay vì **thuật toán** (algorithm)
  - Không mô tả rõ ràng cách tìm đường tăng luồng trên đồ thị tăng luồng
- Cài đặt
  - Thuật toán **Edmonds-Karp**
    - Tìm kiếm theo chiều rộng, sử dụng Queue
  - Tìm đường tăng luồng có lượng tăng luồng nhiều nhất (cải biên Dijkstra)