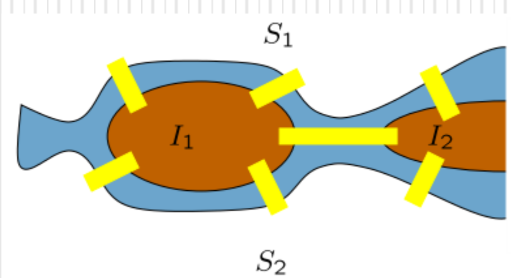
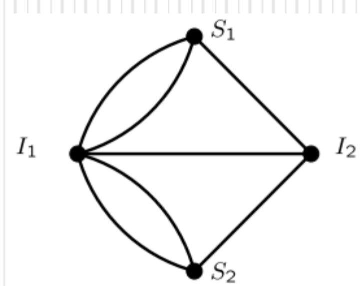
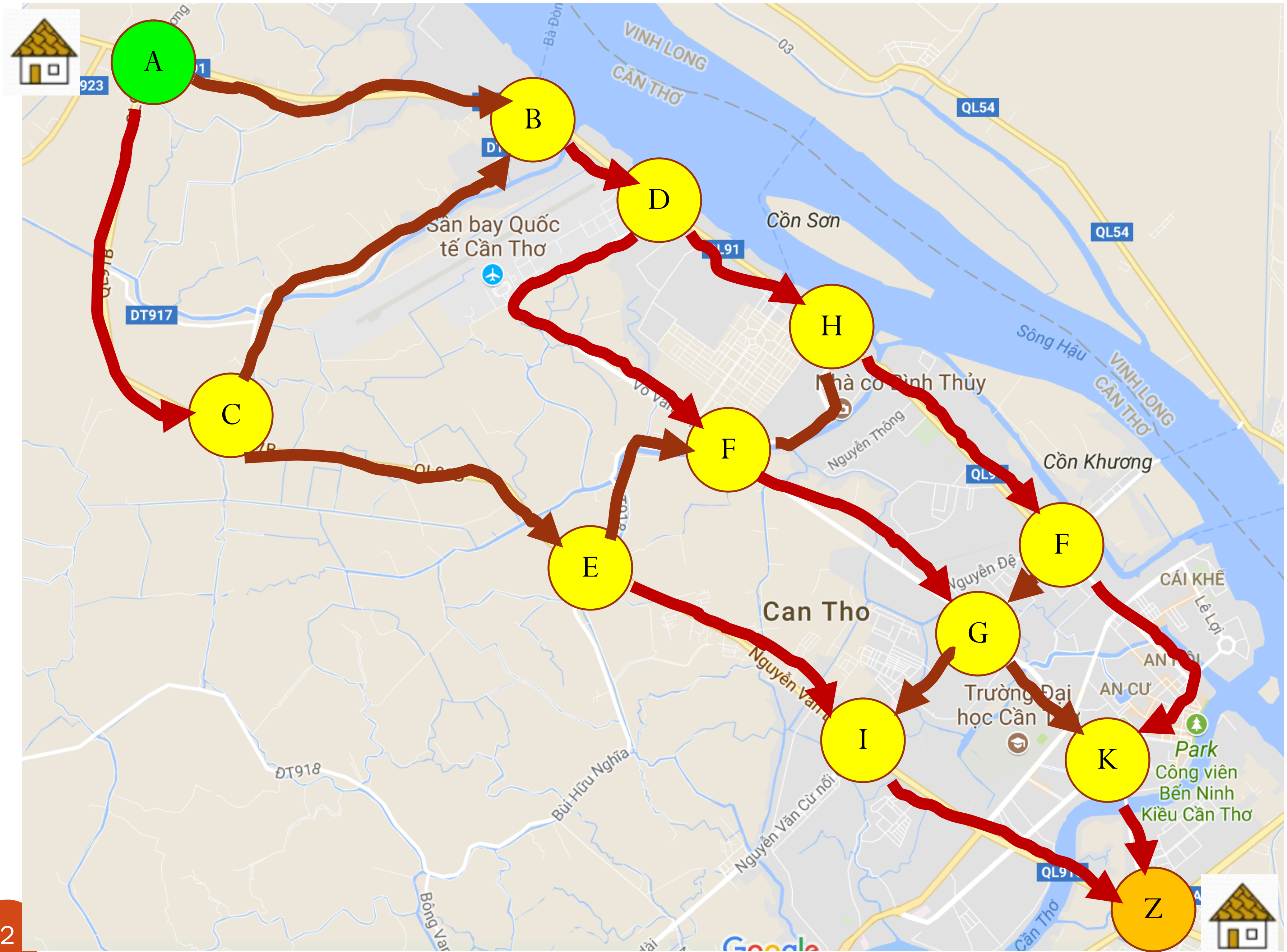
# LÝ THUYẾT ĐỒ THỊ
# Luồng cực đại trong mạng

Phạm Nguyên Hoàng
BM. Khoa học máy tính, CNTT
pnhoang@cit.ctu.edu.vn



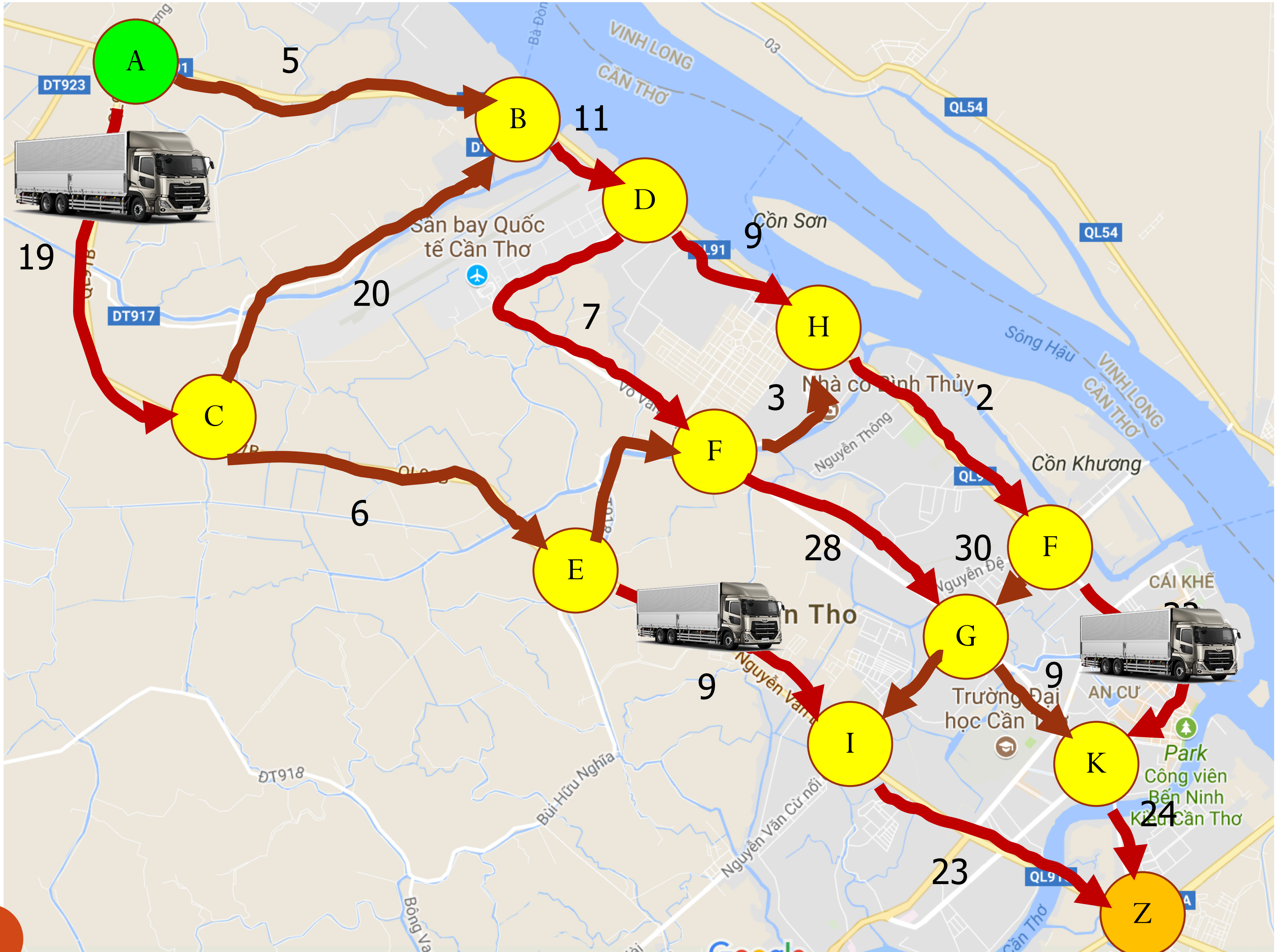



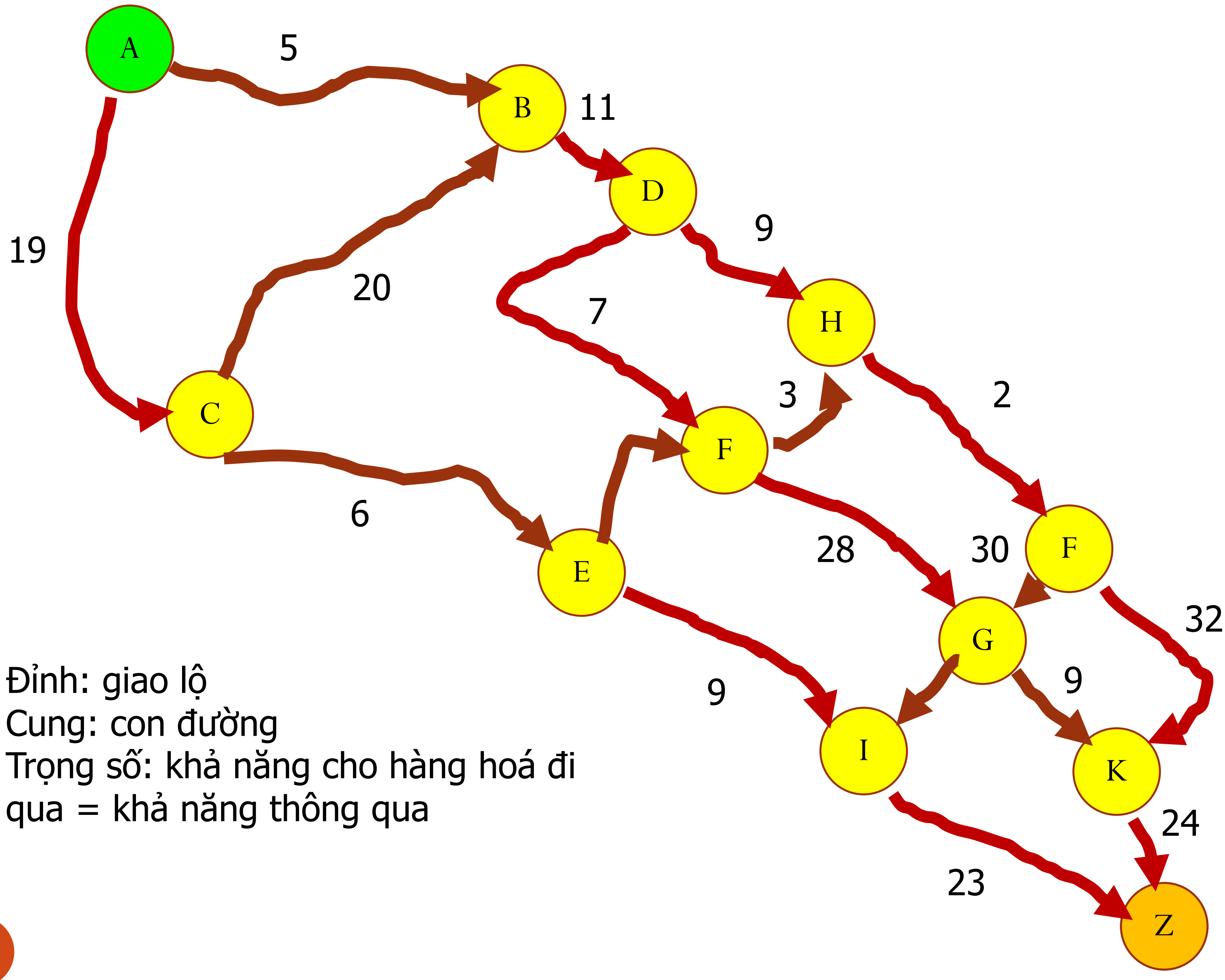

Cần Thơ, 2017


A
B
C
D
E
F
F
G
H
I
K
Z
Sân bay Quốc tế Cần Thơ
Cồn Sơn
Cồn Khương
Sông Hậu
VINH LONG
CẦN THƠ
Can Tho
Trường Đại học Cần Thơ
Công viên Bến Ninh Kiều Cần Thơ
Park
CÁI KHẾ
AN CƯ
DT917
ĐT918
QL54
QL91
Bùi Hữu Nghĩa
Nguyễn Văn Cừ nối
Nguyễn Thông
Nguyễn Đệ
Google


**Nếu biết được rằng mỗi đoạn đường có 1 khả năng cho 1 lượng hàng hoá nào đó đi qua,**

Lượng hàng hoá nhiều nhất di chuyển đi từ A đến Z là bao nhiêu ?
A
B
C
D
F
G
I
K
Z
DT923
DT917
DT918
QL54
QL91
VINH LONG
CẦN THƠ
Sân bay Quốc tế Cần Thơ
Cồn Sơn
Sông Hậu
Cồn Khương
CÁI KHẾ
AN CƯ
Trường Đại học Cần Thơ
Park
Công viên Bến Ninh Kiều Cần Thơ
Bùi Hữu Nghĩa
Nguyễn Văn Cừ nối


A
B
C
D
E
F
H
F
G
I
K
Z
5
11
19
20
7
9
3
2
6
28
30
9
9
24
23
DT923
DT917
QL54
QL54
Sân bay Quốc
tế Cần Thơ
Cồn Sơn
Sông Hậu
VINH LONG
CẦN THƠ
Cồn Khương
CÁI KHẾ
AN CƯ
Trường Đại
học Cần
Park
Công viên
Bến Ninh
Kiều Cần Thơ
ĐT918
Bùi Hữu Nghĩa
Nguyễn Văn Cừ nối
Nguyễn Thông
Google




Đỉnh: giao lộ
Cung: con đường
Trọng số: khả năng cho hàng hoá đi qua = khả năng thông qua

# Mạng (các luồng)

- Network (of flows)/Flow network
  - Mạng là 1 bộ 5: N = <V, E, s, t, c> trong đó
    - <V, E> 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) -> 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)

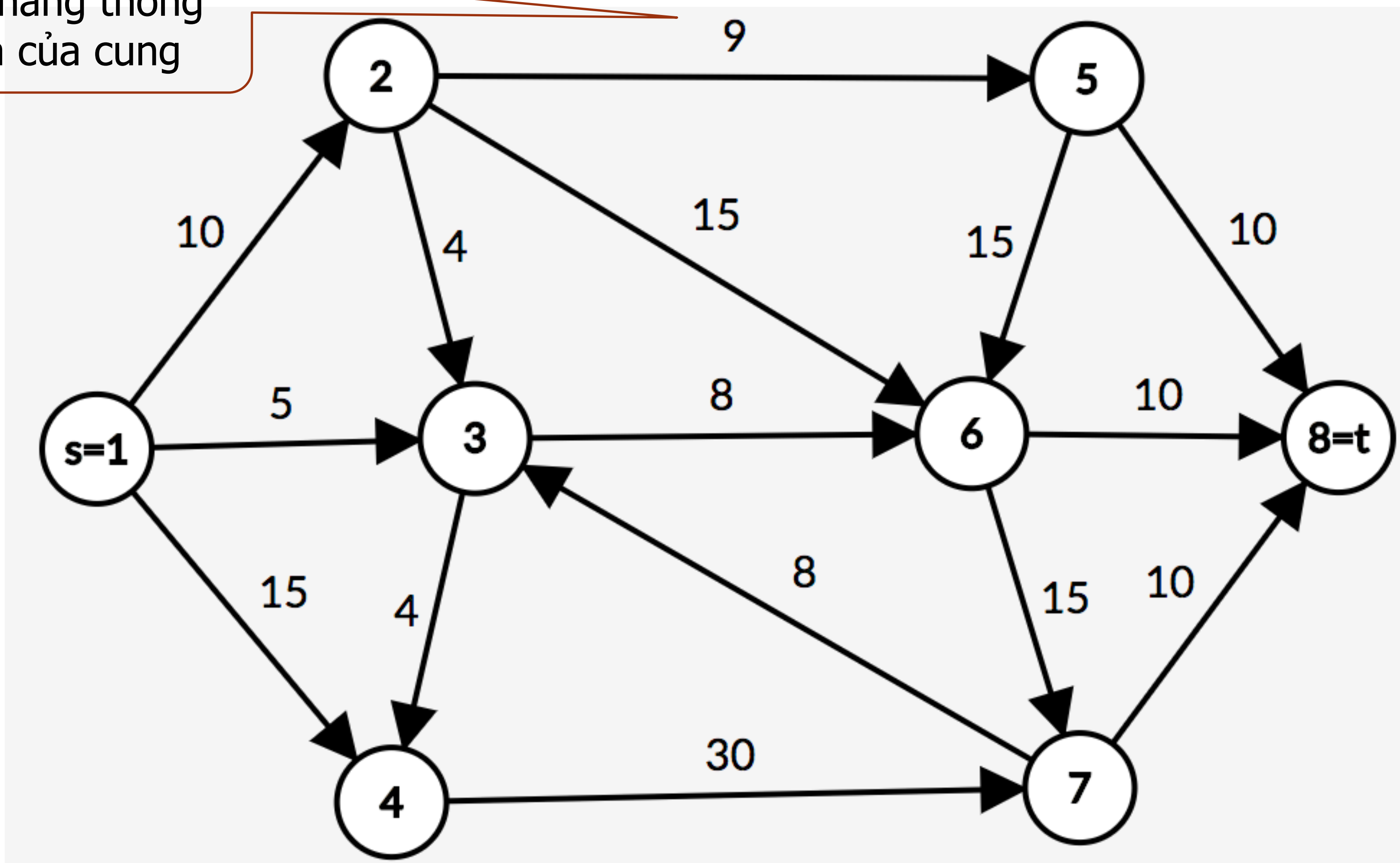
- Ví dụ: mạng các luồng giao thông: các luồng giao thông chạy trên mạng

# Luồng (trong 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) -> f(u,v)
  - thoả mãn các điều kiện:
    - Với mỗi cung: 0 <= f(u,v) <= 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 (trong mạng)

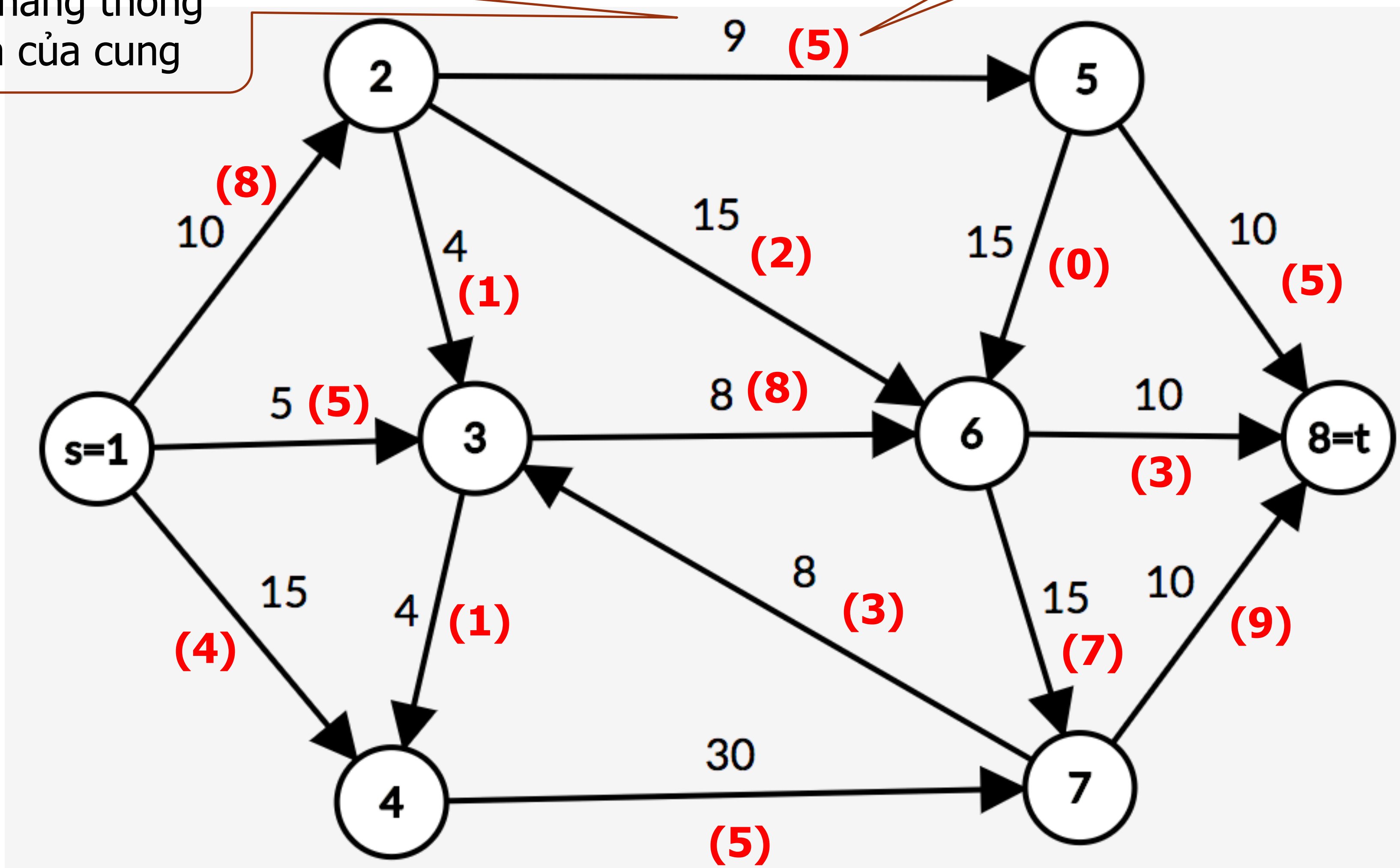
Khả năng thông qua của cung



# Luồng (trong mạng)

Luồng (qua cung)

Khả năng thông qua của cung



# Luồng (trong mạng)

- **Bài tập** kiểm tra điều kiện luồng:
  - Tính tổng luồng vào và luồng ra của từng đỉnh
  - Đỉnh 2: Tổng luồng vào: ??? Tổng luồng ra: ???
  - Đỉnh 3: Tổng luồng vào: ??? Tổng luồng ra: ???
  - Đỉnh 4: Tổng luồng vào: ??? Tổng luồng ra: ???
  - Đỉnh 5: Tổng luồng vào: ??? Tổng luồng ra: ???
  - Đỉnh 6: Tổng luồng vào: ??? Tổng luồng ra: ???
  - Đỉnh 7: Tổng luồng vào: ??? Tổng luồng ra: ???

  - Đỉnh 1: Tổng luồng ra: ???
  - Đỉnh 8: Tổng luồng vào: ???

- Luồng hợp lệ hay không, tại sao ?
- Giá trị luồng: tổng luồng ra khỏi s = ???

# Luồng (trong mạng)

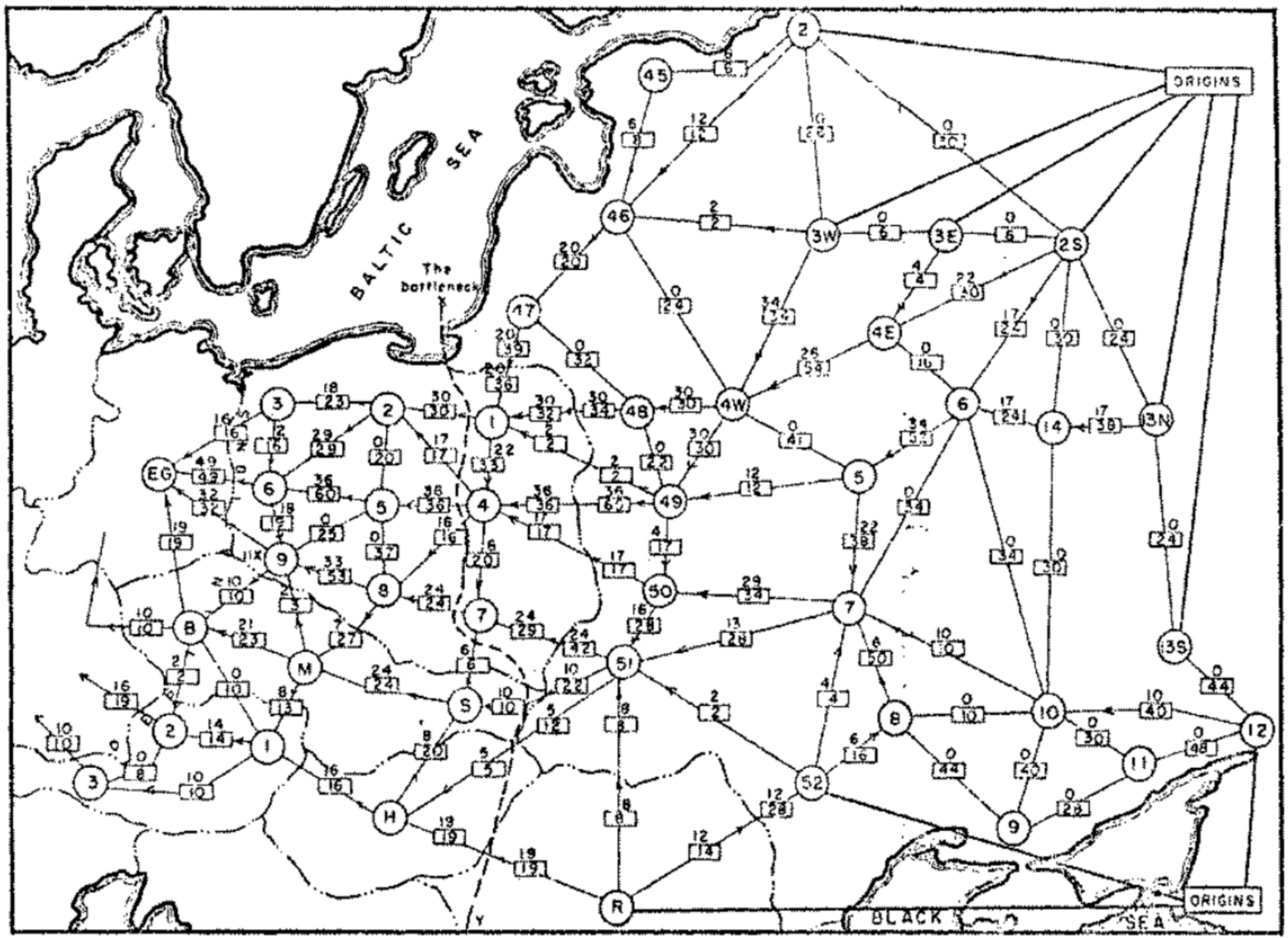
- **Kết quả** kiểm tra điều kiện luồng:
  - Tính tổng luồng vào và luồng ra của từng đỉnh
  - Đỉnh 2: Tổng luồng vào: 8 Tổng luồng ra: 8
  - Đỉnh 3: Tổng luồng vào: 9 Tổng luồng ra: 9
  - Đỉnh 4: Tổng luồng vào: 5 Tổng luồng ra: 5
  - Đỉnh 5: Tổng luồng vào: 5 Tổng luồng ra: 5
  - Đỉnh 6: Tổng luồng vào: 10 Tổng luồng ra: 10
  - Đỉnh 7: Tổng luồng vào: 12 Tổng luồng ra: 12

  - Đỉnh s=1: Tổng luồng ra: 17
  - Đỉnh t=8: Tổng luồng vào: 17
- Luồng hợp lệ
- Giá trị luồng: tổng luồng ra khỏi s = 8+5+4=17

# Luồng trong mạng (ứng dụng)

| Mạng | Đỉnh/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 sang Đông Âu



# Luồng cực đại

- Cho mạng N =<V, E, s, t, c>, 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 cát 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

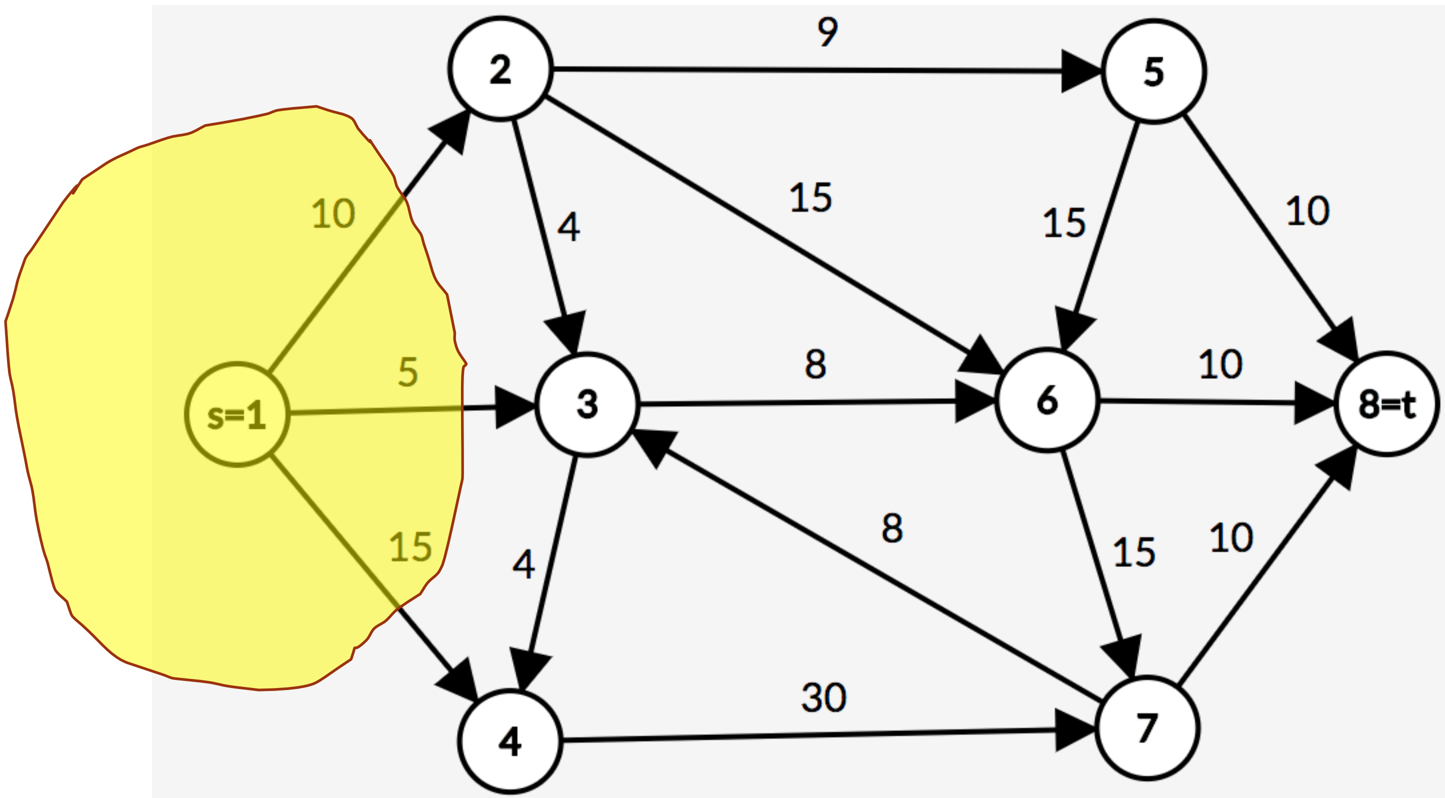
$$\{(u, v) \mid u \in S \text{ và } v \in T\}$$

- Khả năng thông qua của 1 lát cắt (tách s và t)

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

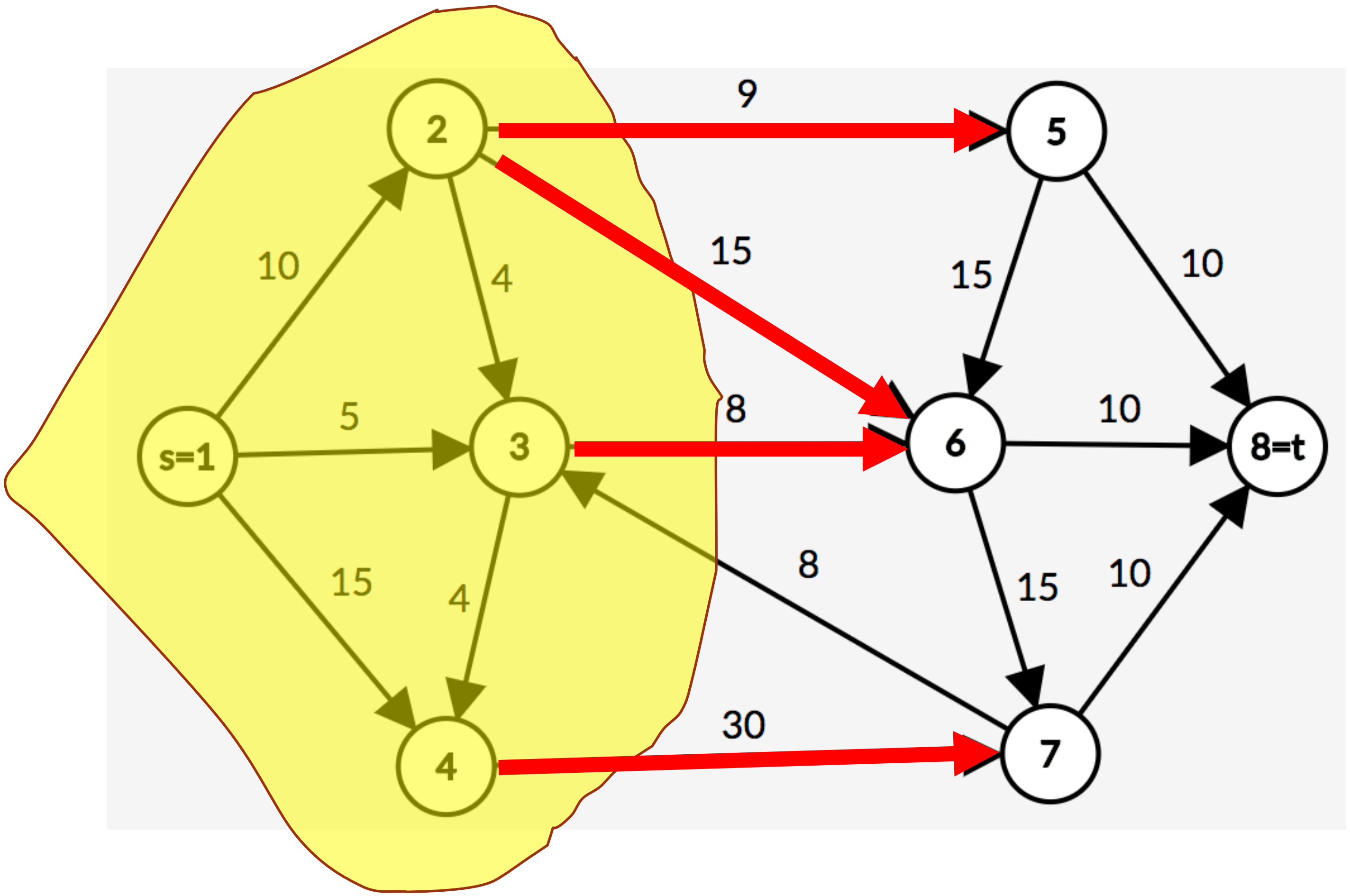
  - Tổng khả năng thông qua của các cung của lát cắt

# Lát cắt



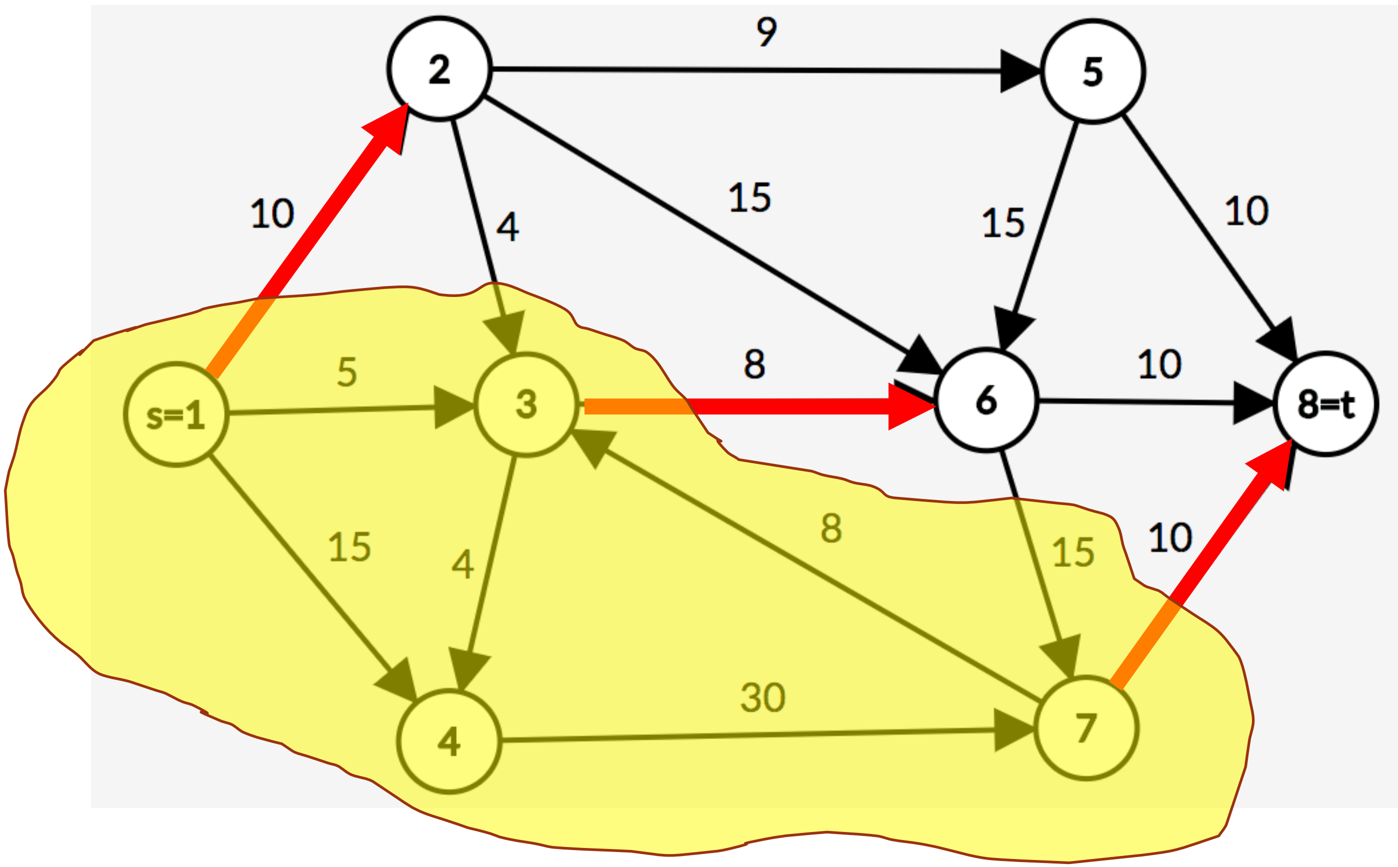

**Khả năng thông qua của lát cắt = 10+5+15 = 30**

# Lát cắt




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

# Lát cắt



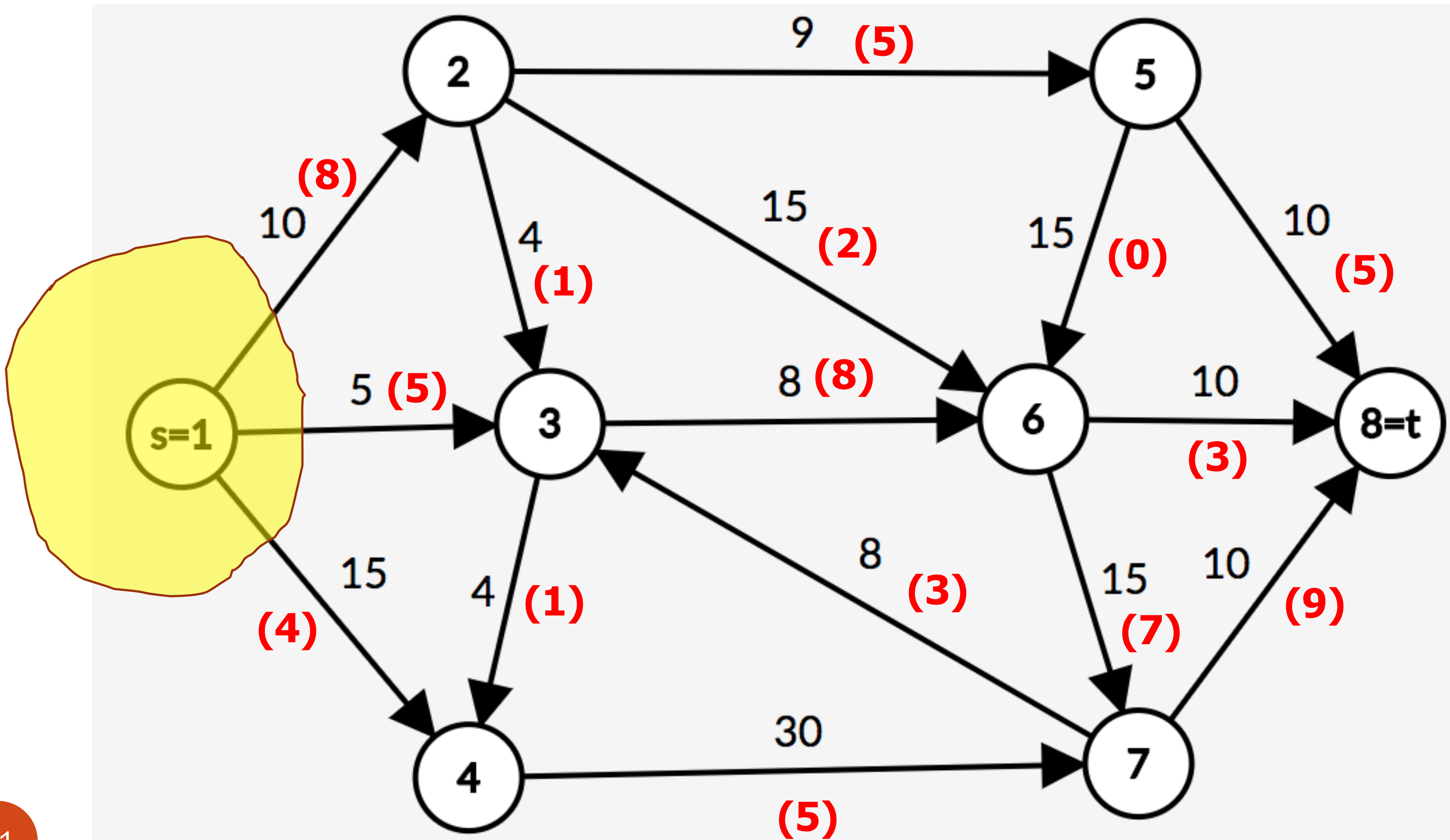

**Khả năng thông qua của lát cắt = 10+8+10 = 28**

# 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

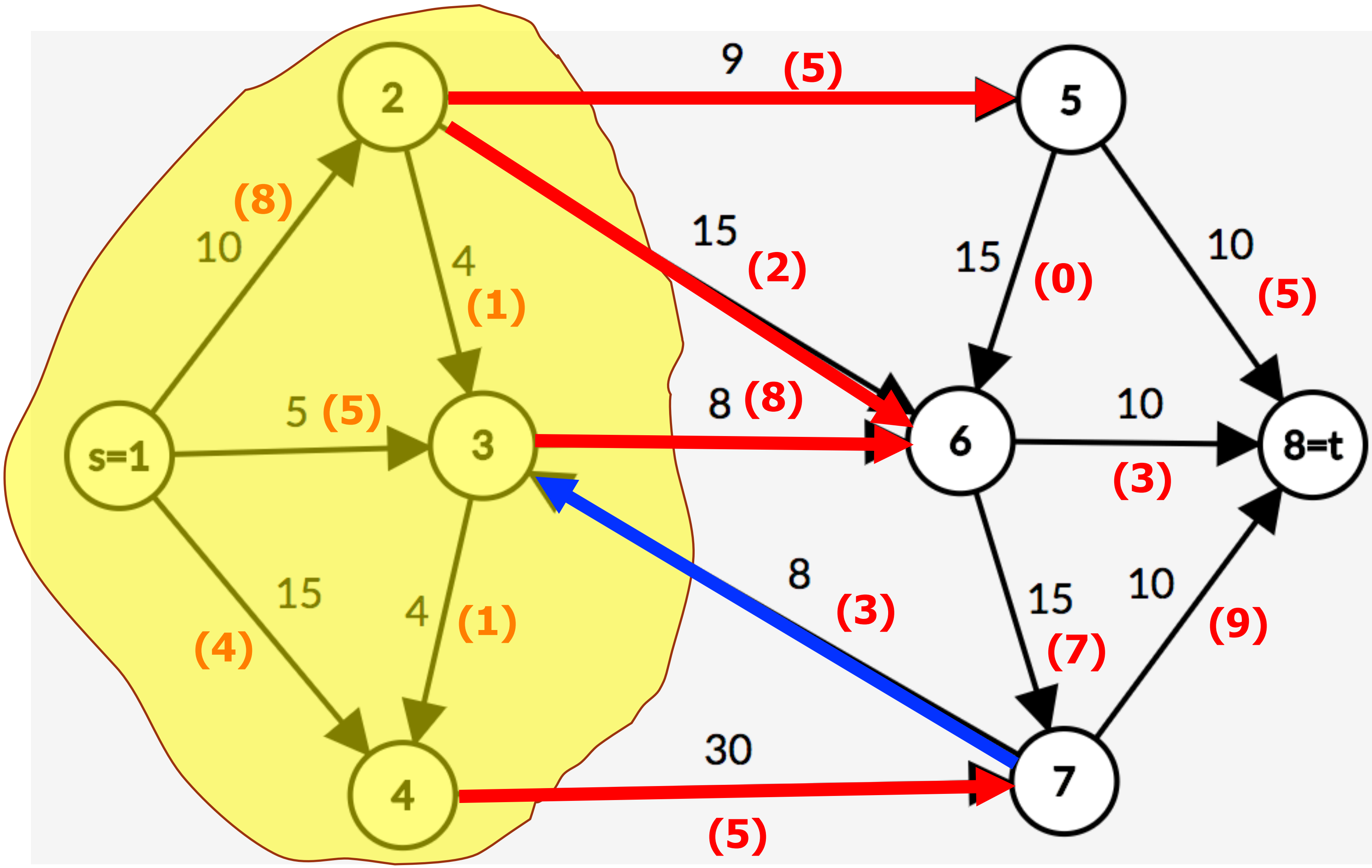
$$\sum_{e \text{ out of } S} f(e) - \sum_{e \text{ in to } S} f(e) = \sum_{e \text{ out of } s} f(e) = |f|$$

# Luồng và lát cắt



# Luồng và lát cắt

20 - 3 = 17



# Luồng và lát cắt

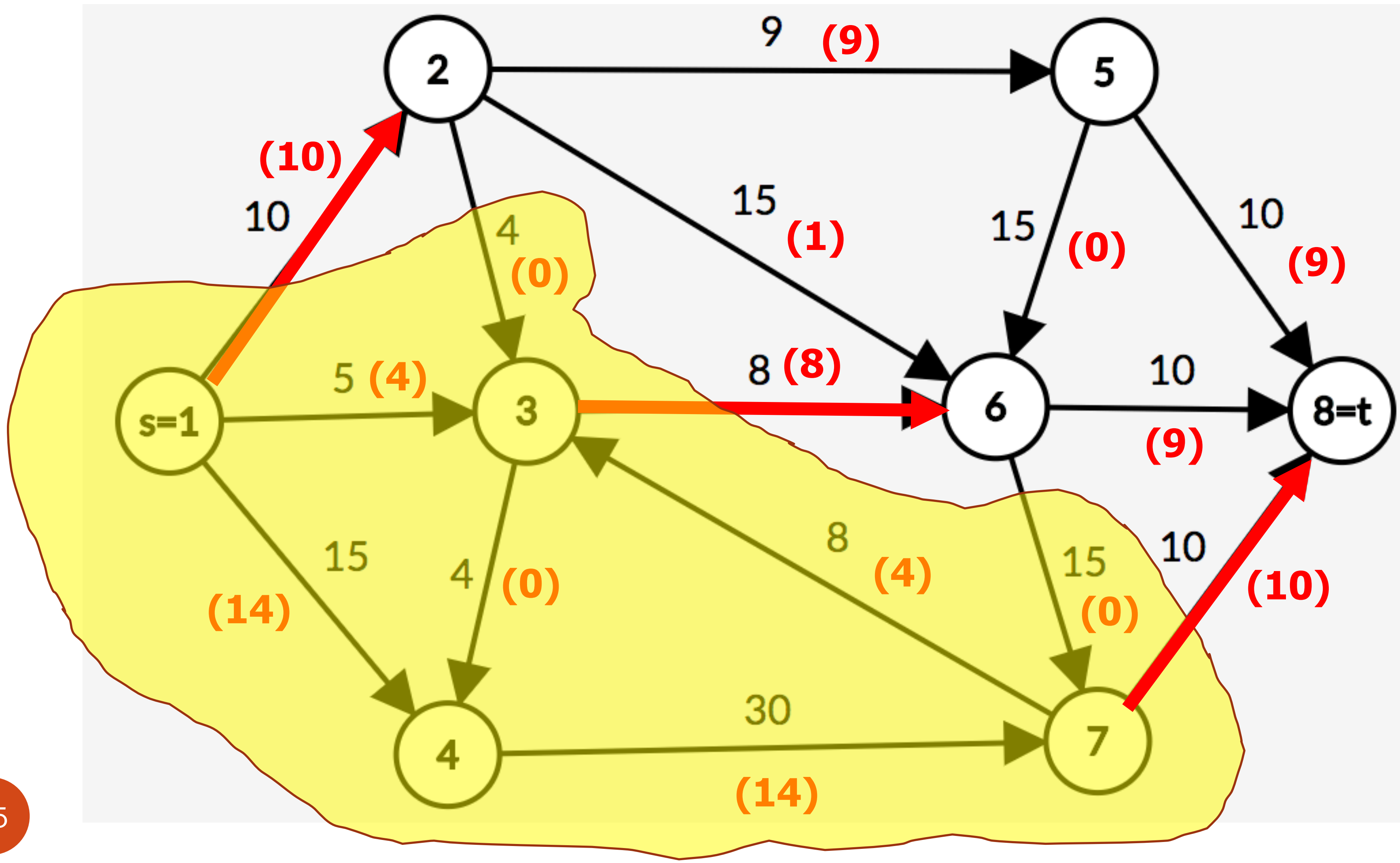
- **Bổ đề 2**
- Nếu gọi:
  - f là luồng đi qua mạng N
  - (S, T) là 1 lát cắt tách s và t
- thì
  - Giá trị luồng không vượt quá khả năng thông qua của lát cắt

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

# 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

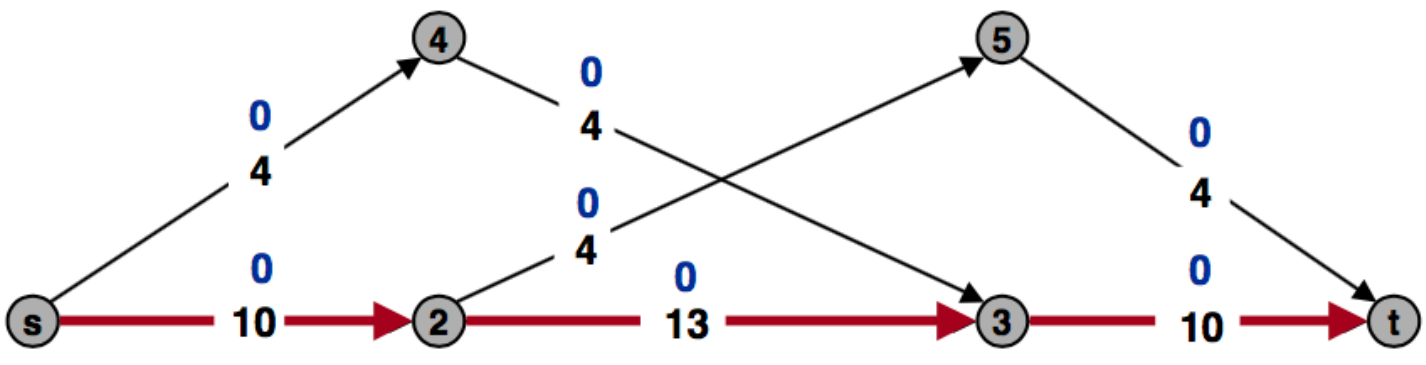
- Định lý luồng cực đại lát cắt hẹp nhất (Ford-Fulkerson, 1956)
  - Giá trị luồng cực đại = khả năng thông qua của lát cắt hẹp nhất.

# Giải thuật Ford-Fulkerson

- Ý tưởng:
  - Tìm lát cắt hẹp nhất bằng cách tăng luồng đi qua mạng.
  - 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.

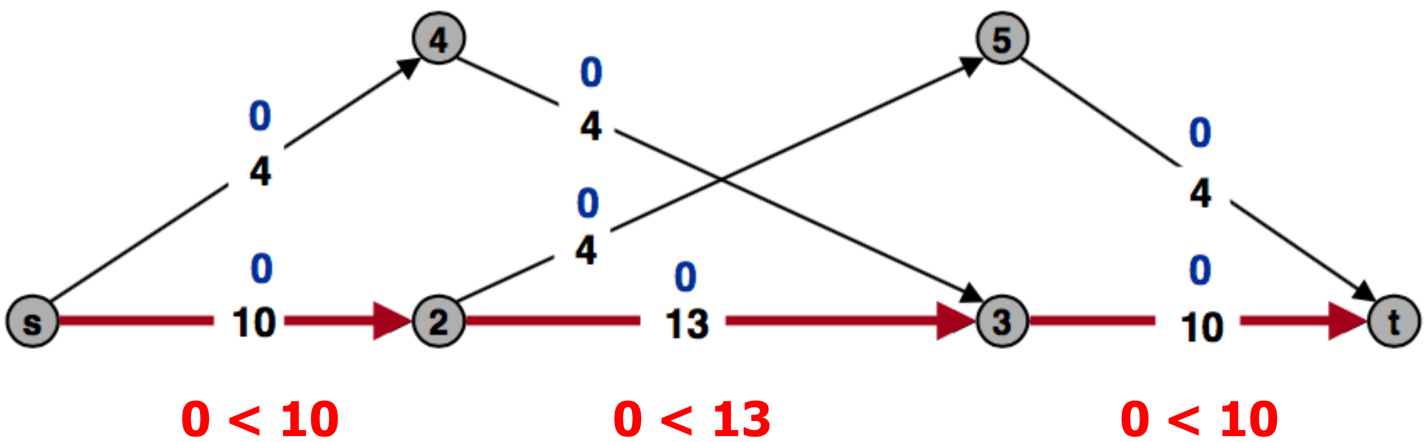
# Giải thuật Ford-Fulkerson

- 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**



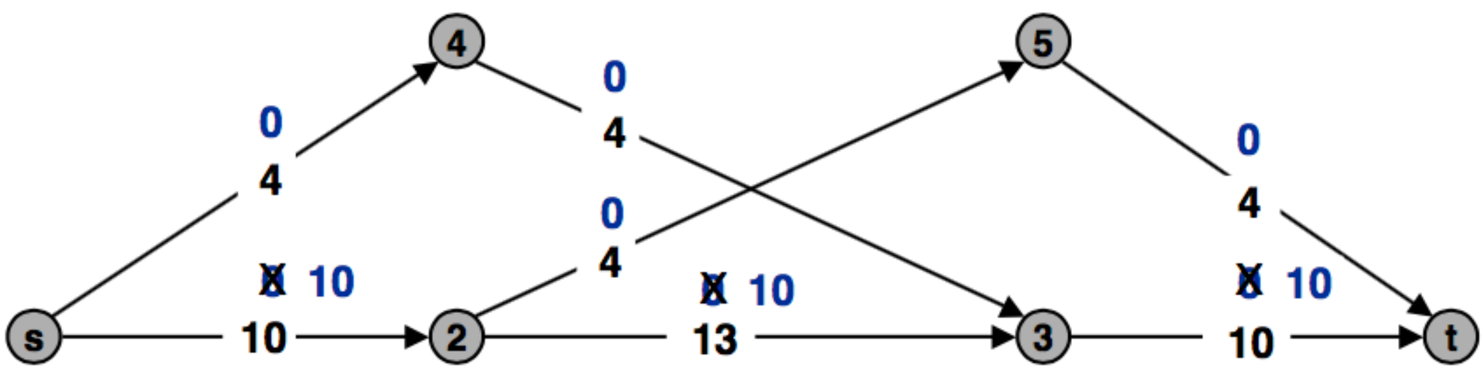
# Giải thuật Ford-Fulkerson

- 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**



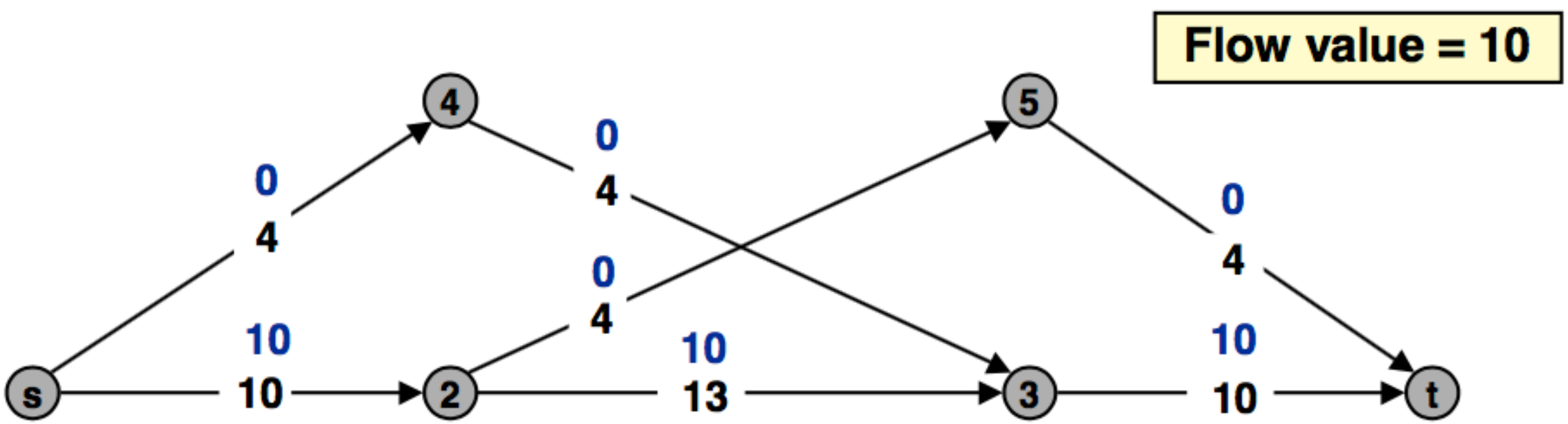
# Giải thuật Ford-Fulkerson

- 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**



# Giải thuật Ford-Fulkerson

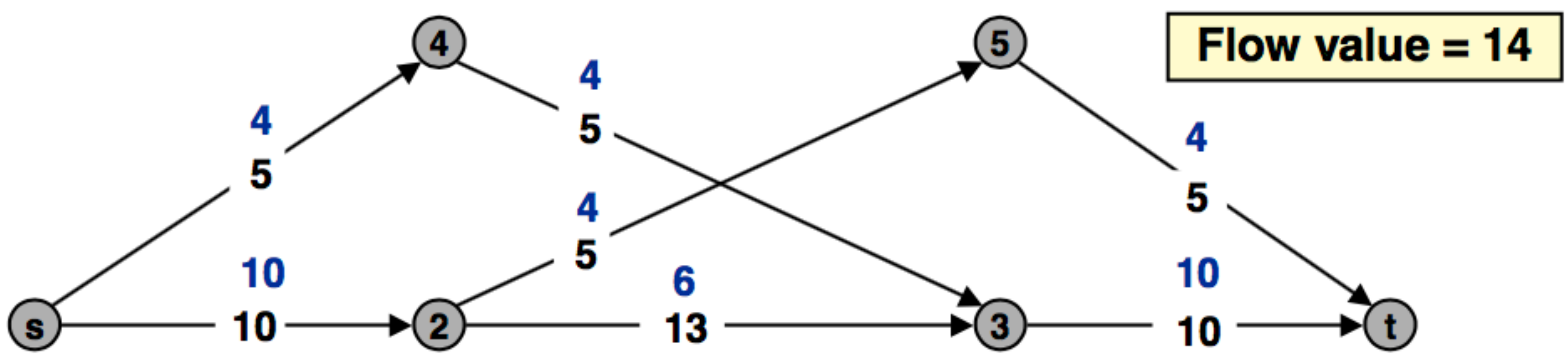
- Tìm đường tăng luồng:
  - Tìm đường đi từ s đến t sao cho có thể tăng thêm luồng trên đường đi đó.
    - **QS1: Tìm đường đi từ s đến t sao cho c(e) > f(e) với tất cả các cung trên đường đi**




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

# Giải thuật Ford-Fulkerson

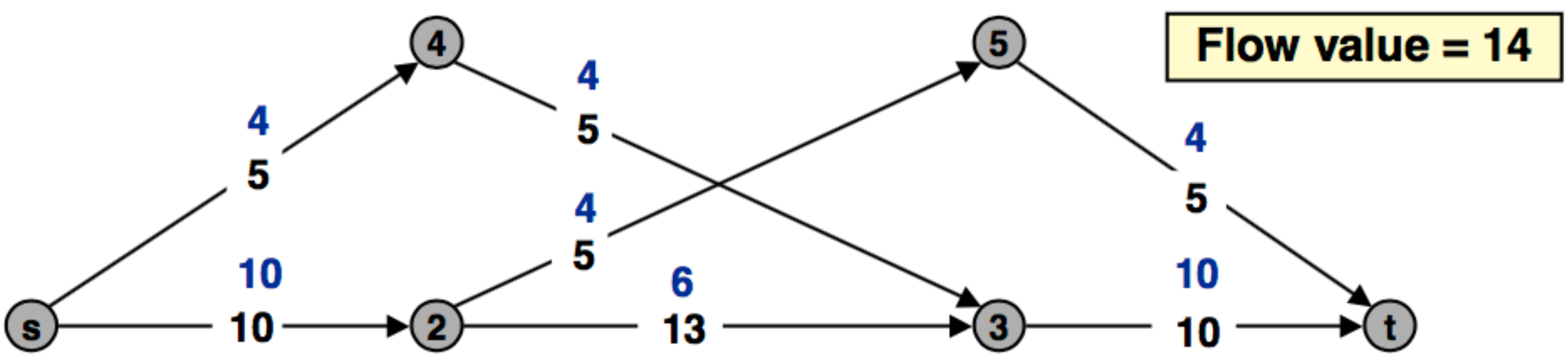
- Tìm đường tăng luồng:
  - Tìm đường đi từ s đến t sao cho có thể tăng thêm luồng trên đường đi đó.
    - **QS1: Tìm đường đi từ s đến t sao cho c(e) > f(e) với tất cả các cung trên đường đi**




**QS1: thất bại !!!**

# Giải thuật Ford-Fulkerson

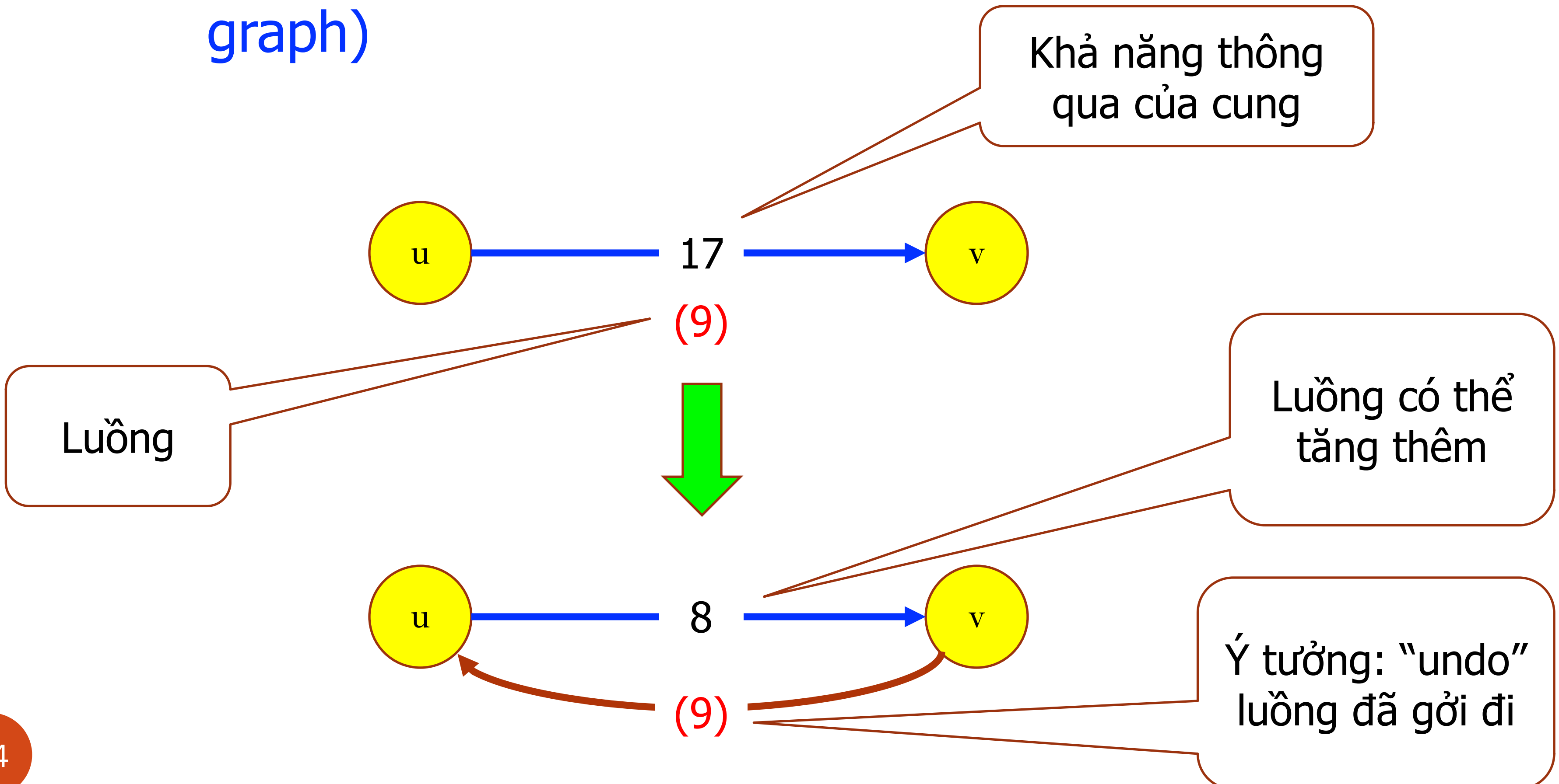
- Tìm đường tăng luồng:
  - Tìm đường đi từ s đến t sao cho có thể tăng thêm luồng trên đường đi đó.
    - **QS1: Tìm đường đi từ s đến t sao cho c(e) > f(e) với tất cả các cung trên đường đi**




**Cần tìm cách khác để tăng luồng**

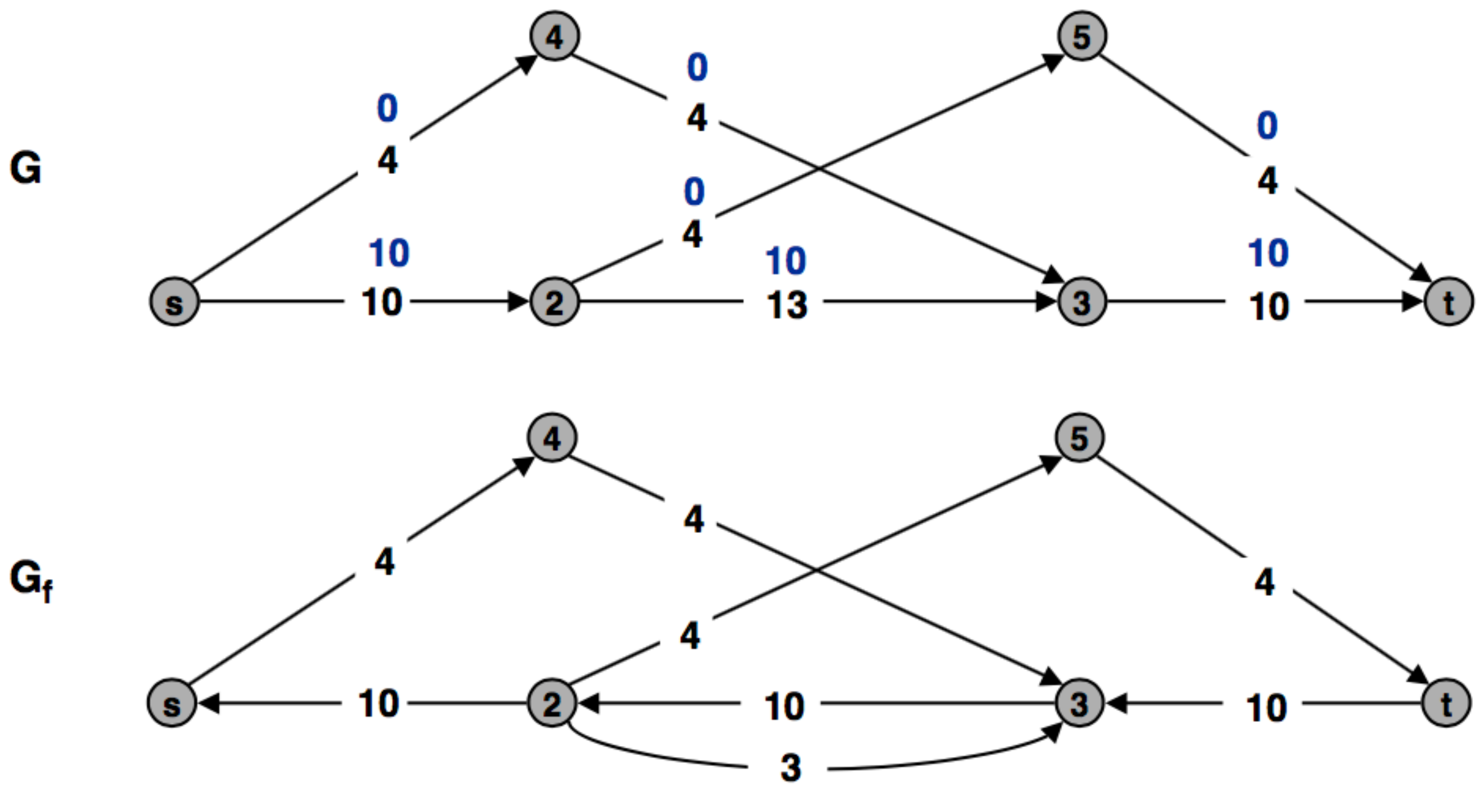
# Giải thuật Ford-Fulkerson

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



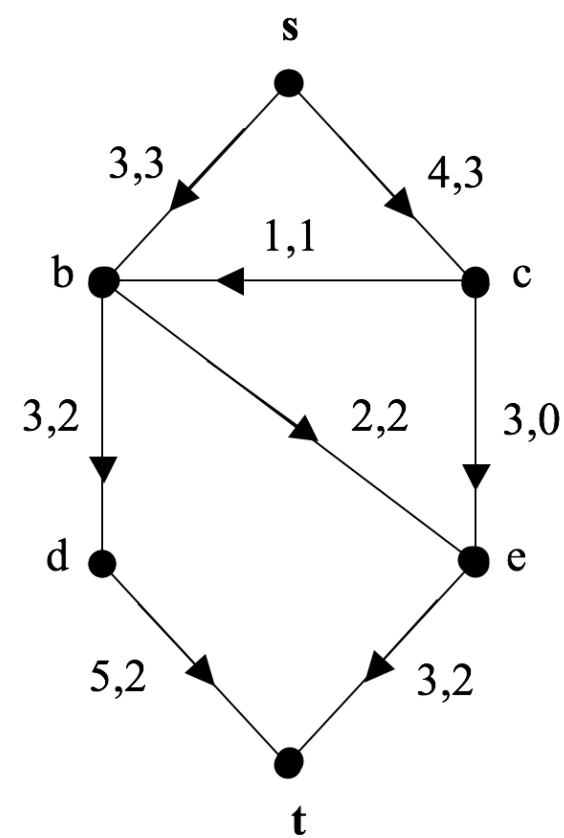
# Giải thuật Ford-Fulkerson

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



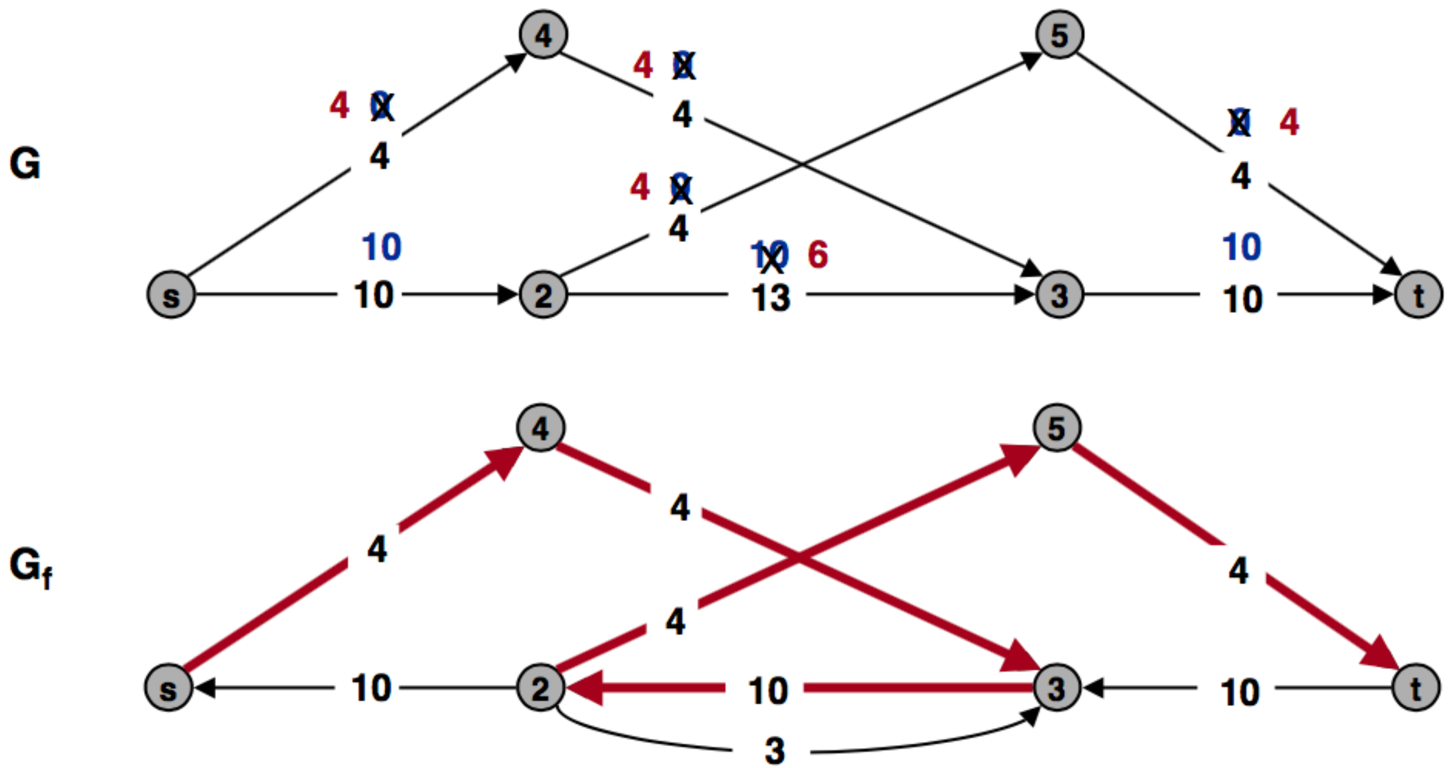
# Giải thuật Ford-Fulkerson

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



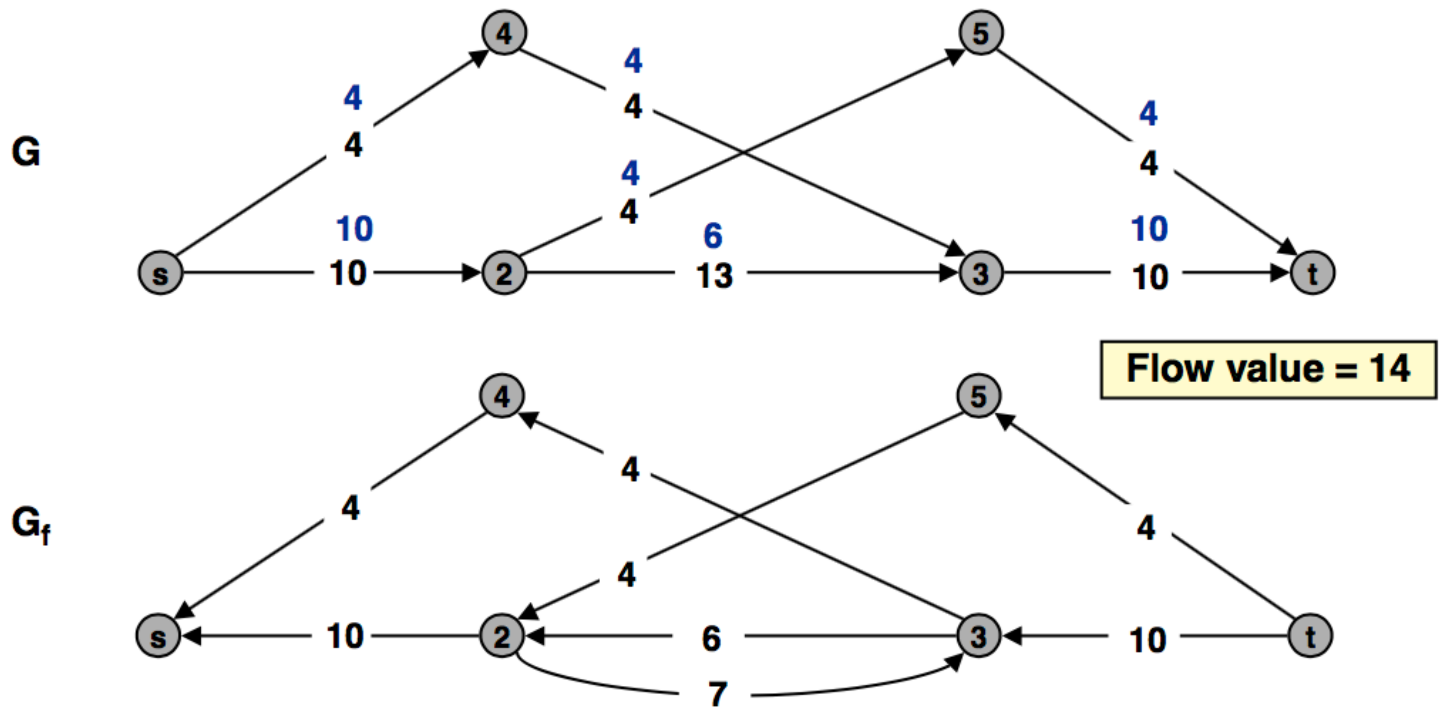
# Đường tăng luồng

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



# Đường tăng luồng

- Đường tăng luồng = đường đi từ s đến t trên **đồ thị tăng luồng**
- Luồng cực đại ⇔ không còn đường tăng luồng



# Giải thuật Ford-Fulkerson

- **Đánh dấu** các đỉnh để tìm đường tăng luồng trên đồ thị tăng luồng
  - Không cần xây dựng tường minh đồ thị tăng luồng
  - Sử dụng **duyệt theo chiều sâu** để tìm đường đi từ s đến t.
- Mỗi đỉnh u được dánh 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

# Giải thuật 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
  - **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

# Giải thuật 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 stack

# Giải thuật Ford-Fulkerson

- **Tìm đường tăng luồng từ s đến t dùng Stack**

  1. Đánh dấu s: (+, s, ∞)
  2. Push s vào Stack
  3. while (stack không rỗng) {

     a. u = top(Stack); pop(Stack);

     b. for (các đỉnh kề v của u và có **f(u, v) < c(u, v)**)

        Đánh dấu v: (+, u, min{**σ[u]**, c(u, v) – f(u, v)})

        Push v vào Stack

     c. for (các đỉnh x có cung đi đến u có **f(x, u) > 0**)

        Đánh dấu x: (-, u, min{**σ[u]**, f(x, u)})

        push x vào Stack

     d. Nếu t được đánh dấu => thoát vòng lặp while

     }

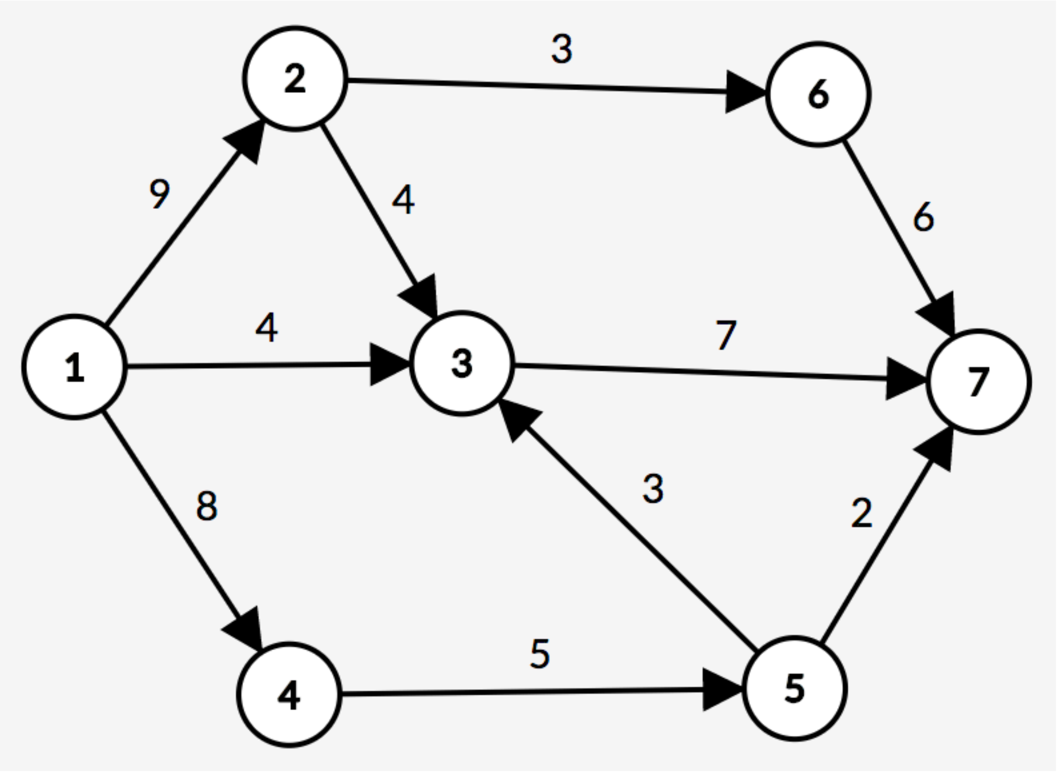
# Giải thuật Ford-Fulkerson

- **Tăng luồng theo đường tăng luồng**

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

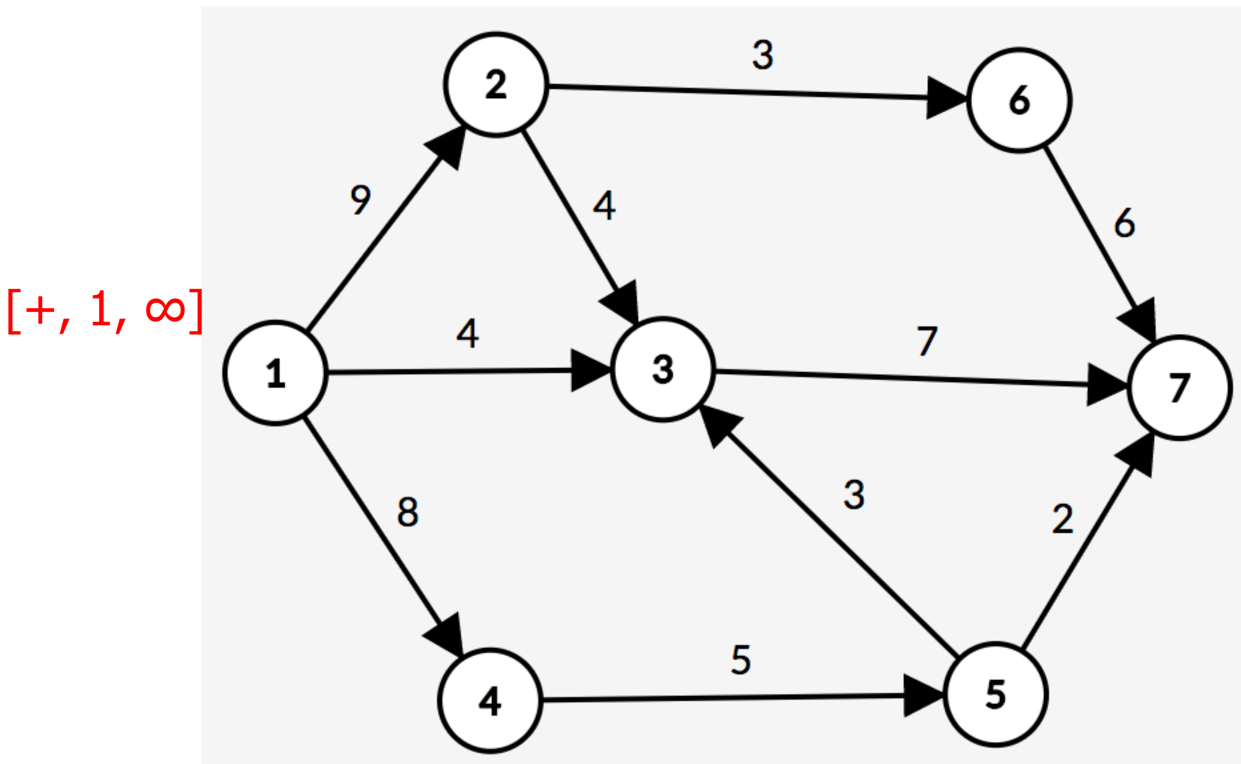
# Ví dụ

- Tìm luồng cực đại và lát cắt hẹp nhất



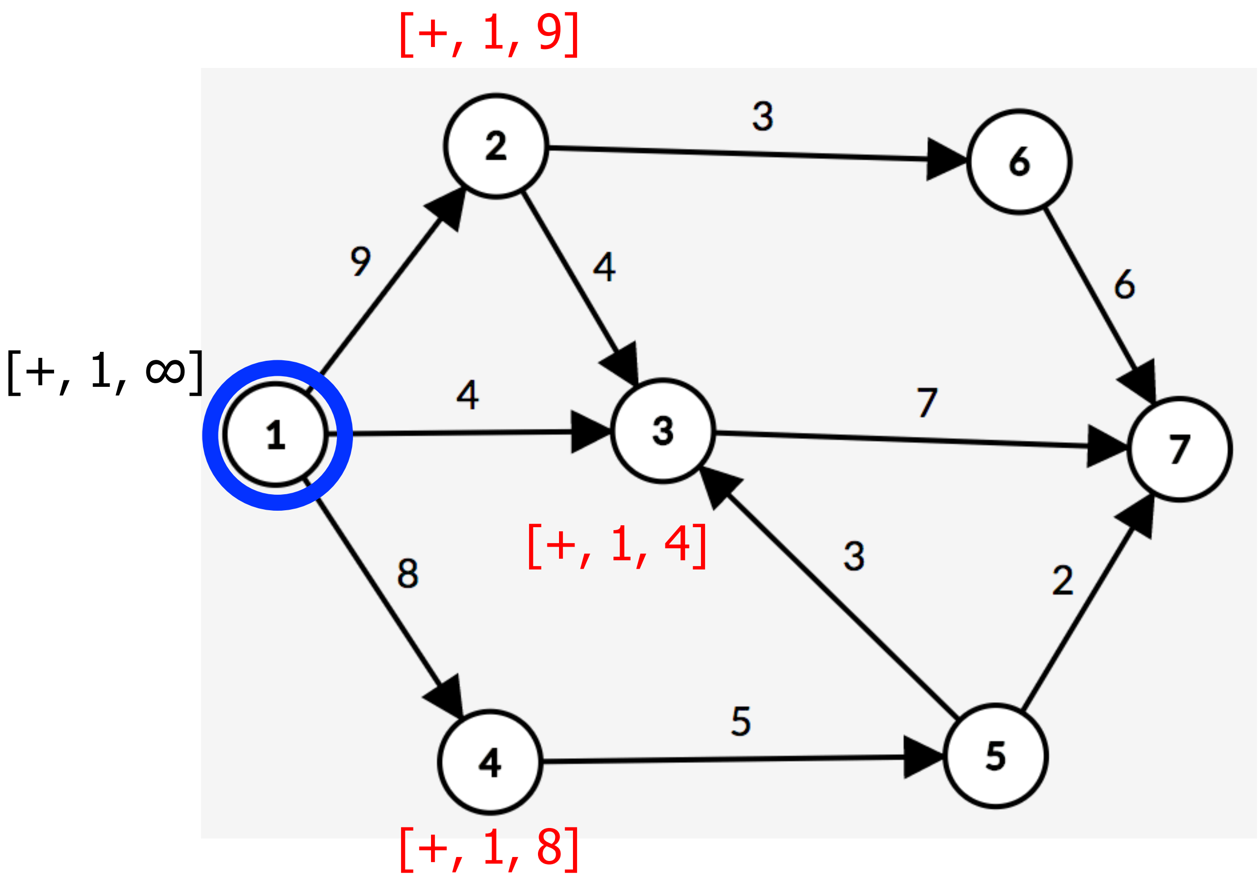
# Lặp 1

- Tìm đường tăng luồng



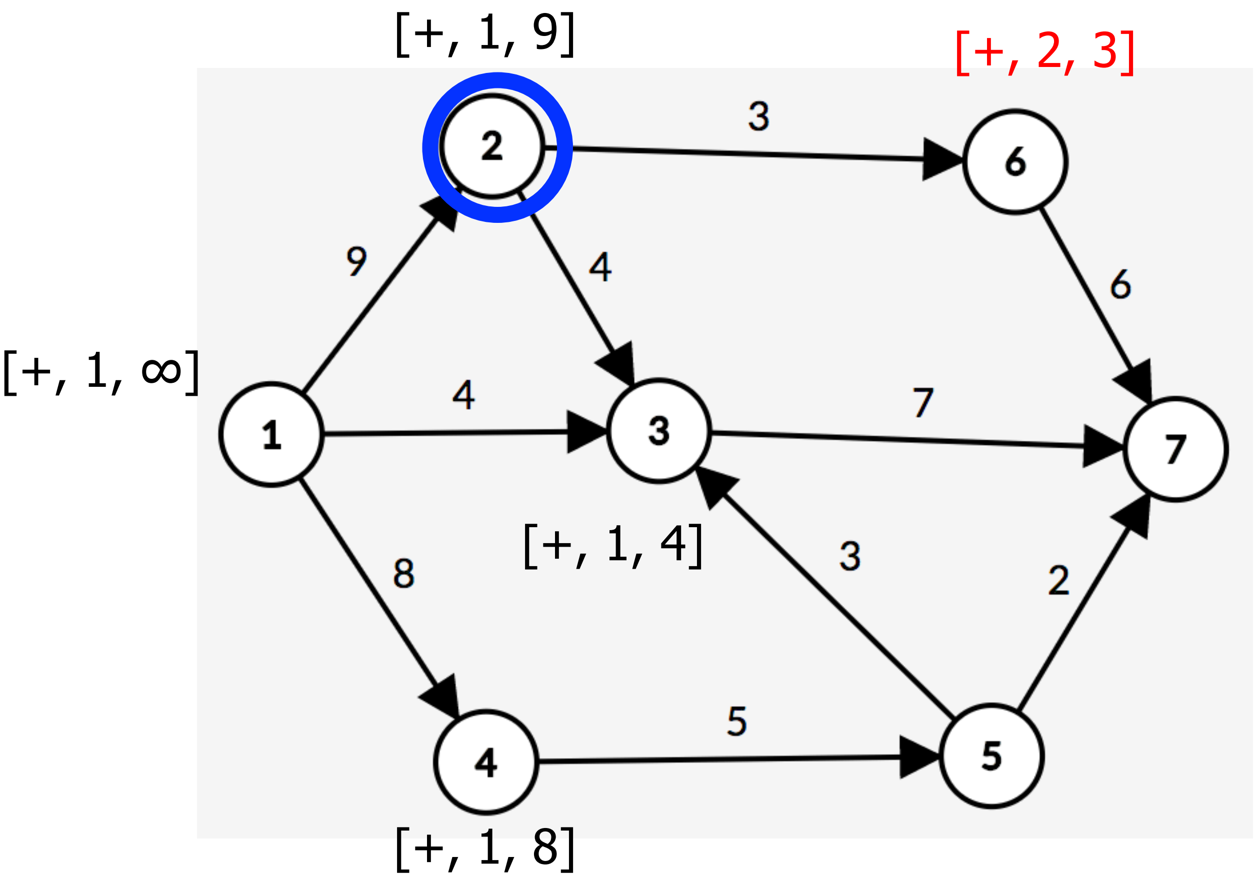
# Lặp 1

- Tìm đường tăng luồng



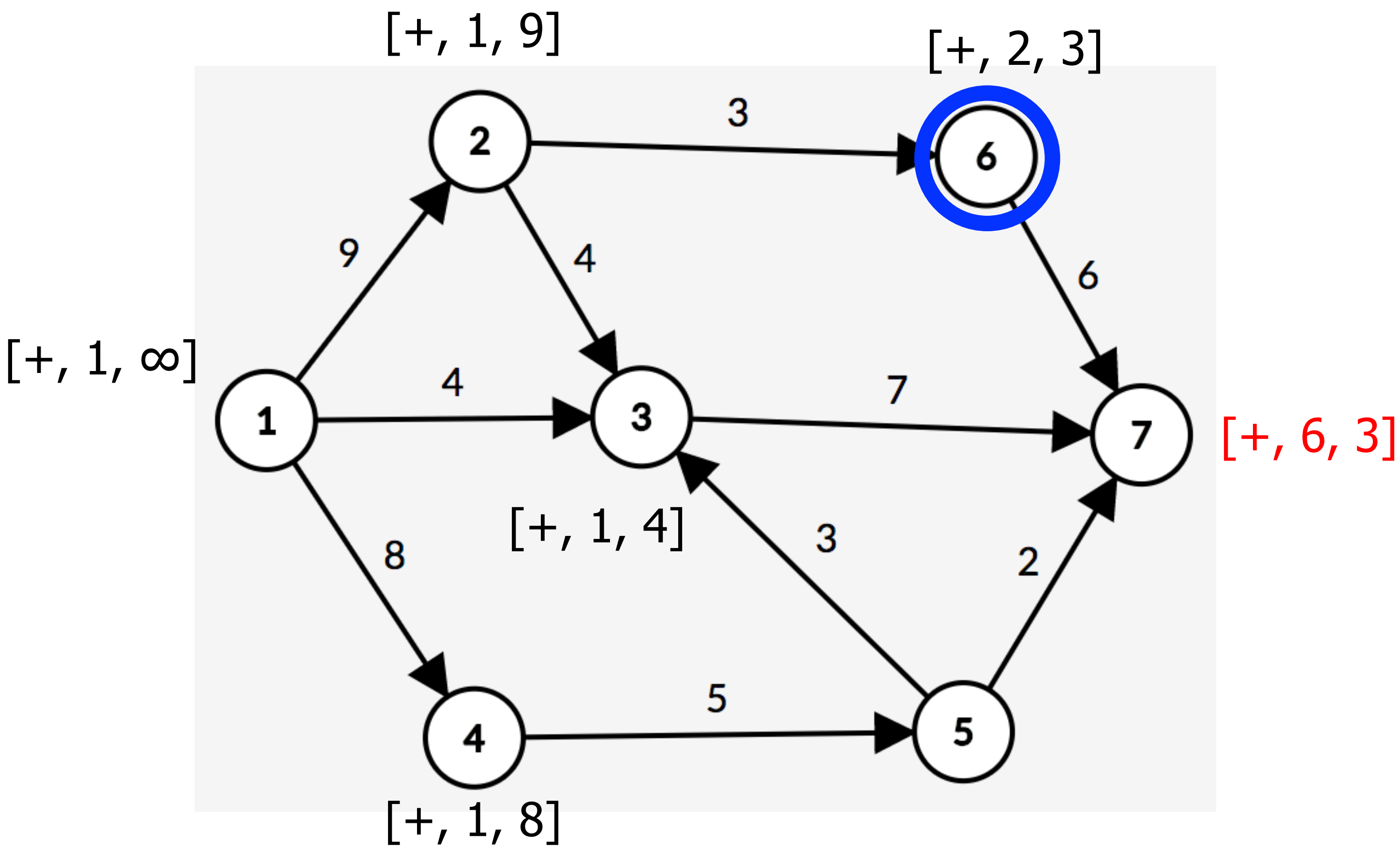
# Lặp 1

- Tìm đường tăng luồng



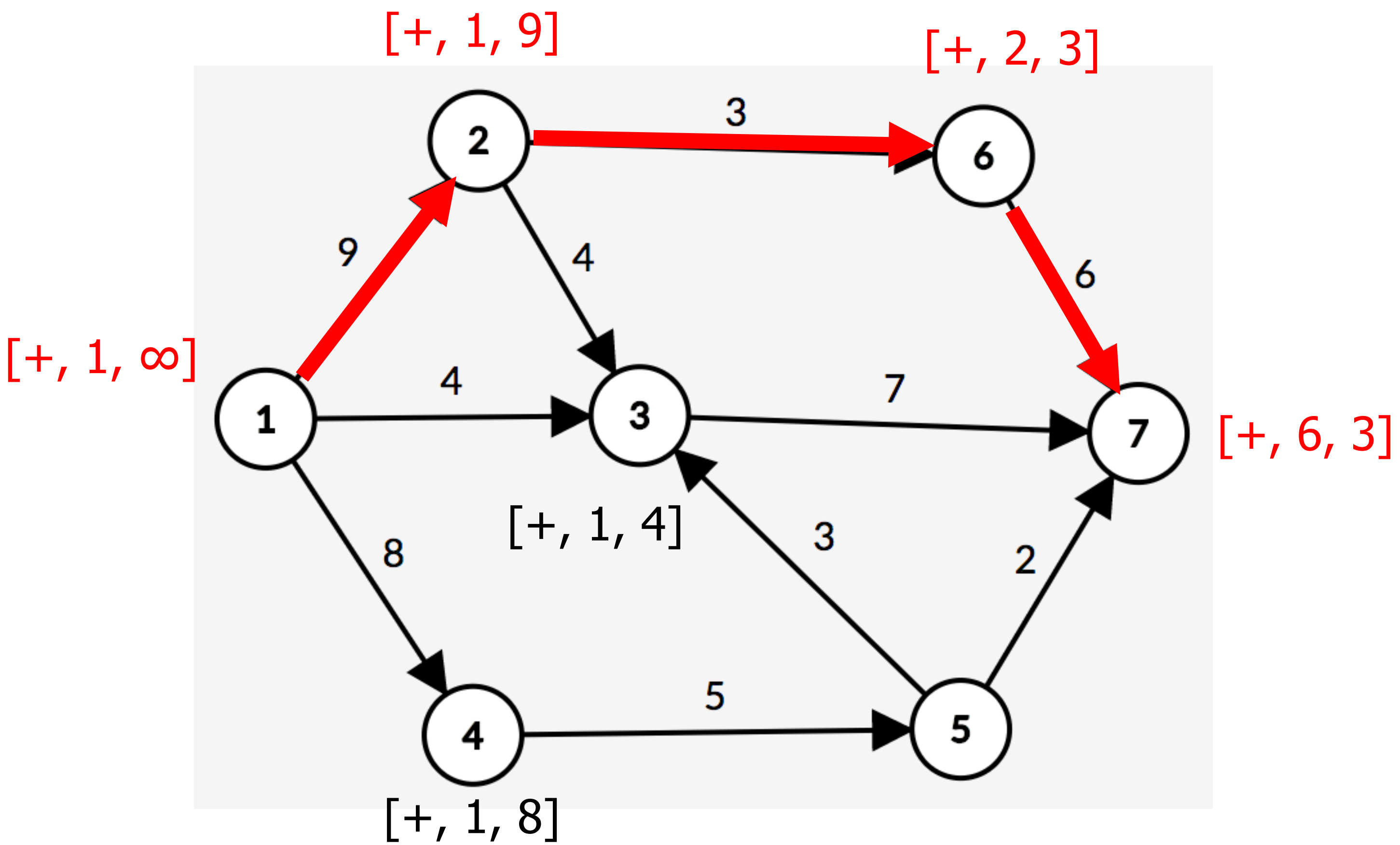
# Lặp 1

- Tìm đường tăng luồng



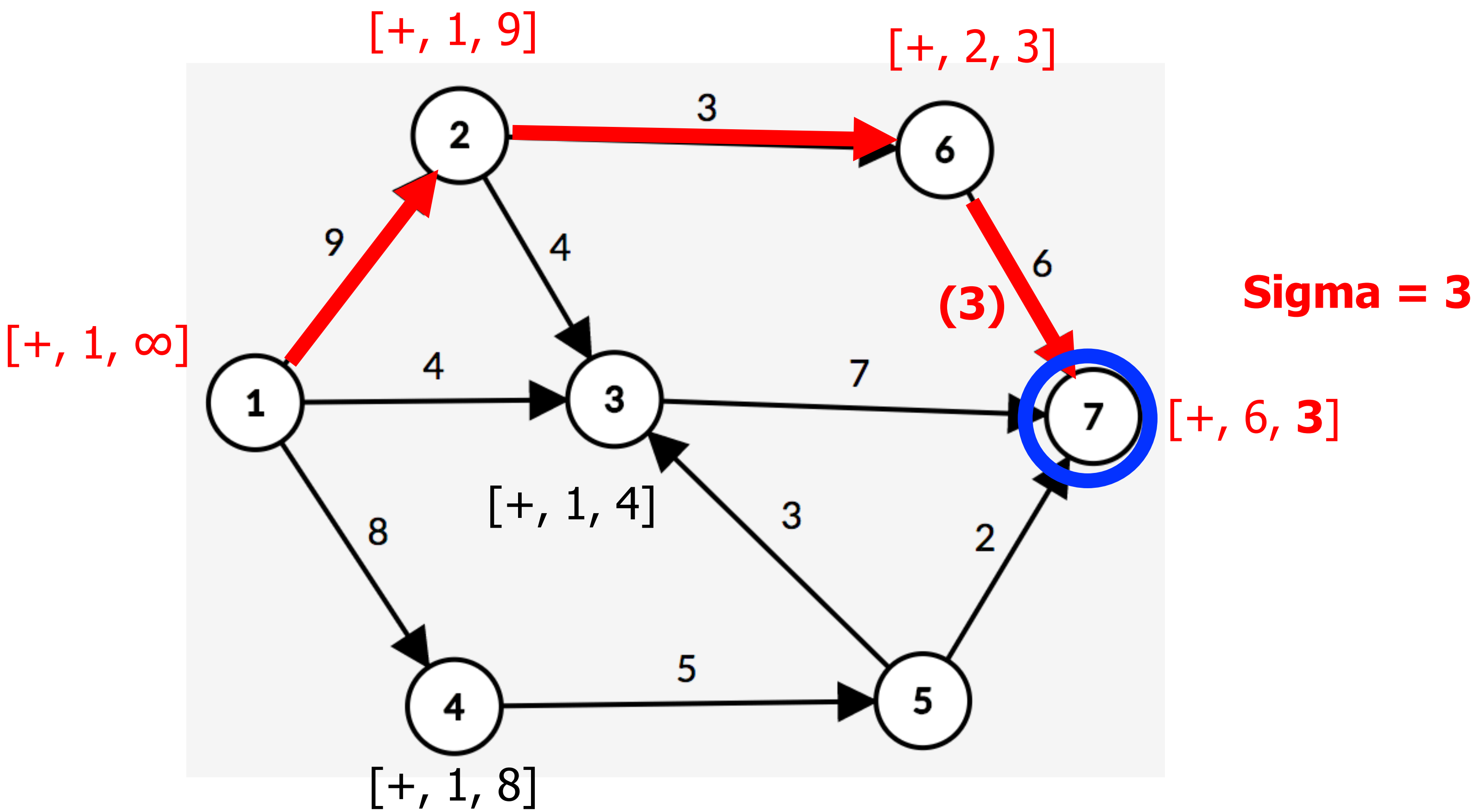
# Lặp 1

- Tìm đường tăng luồng



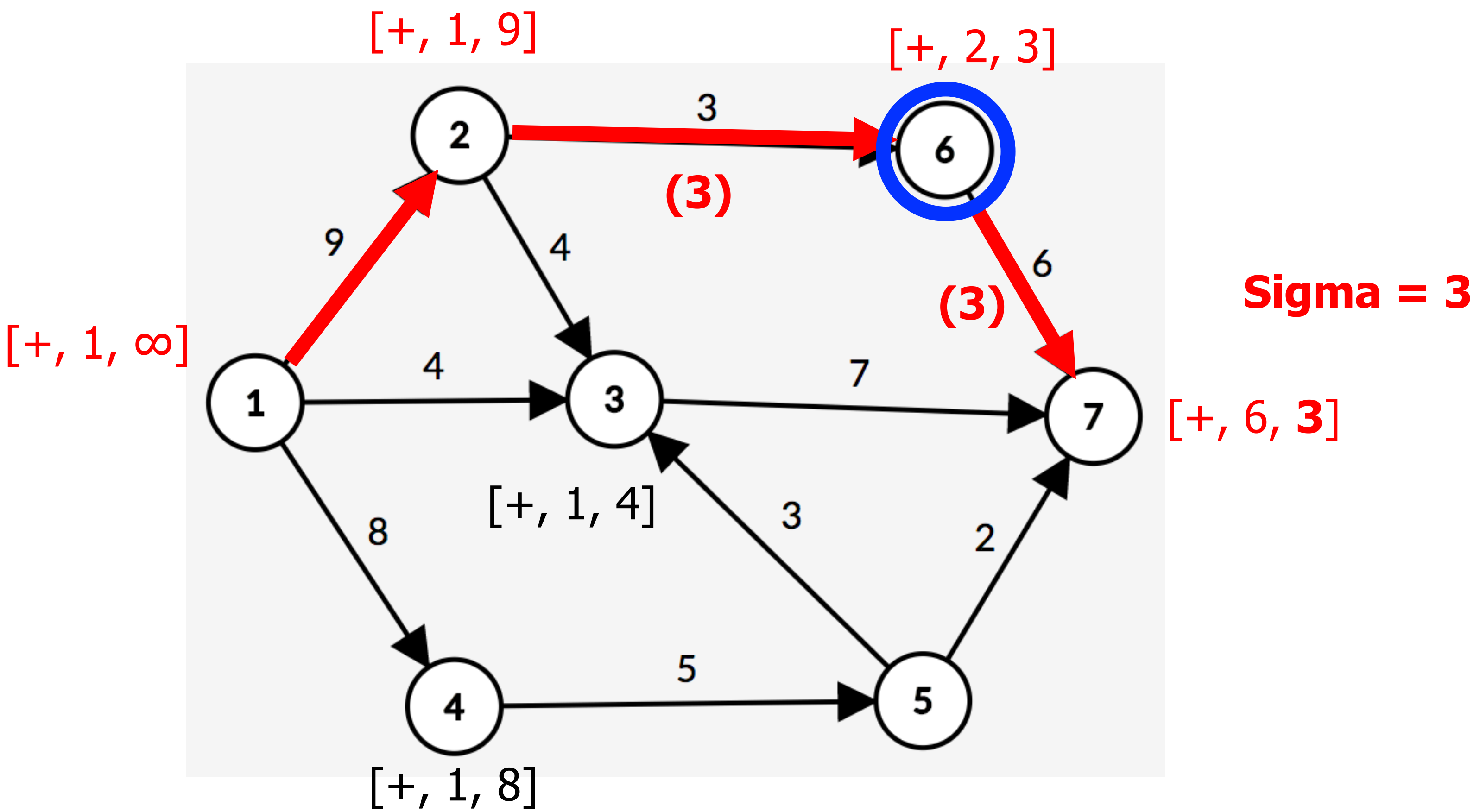
# Lặp 1

- Tăng luồng



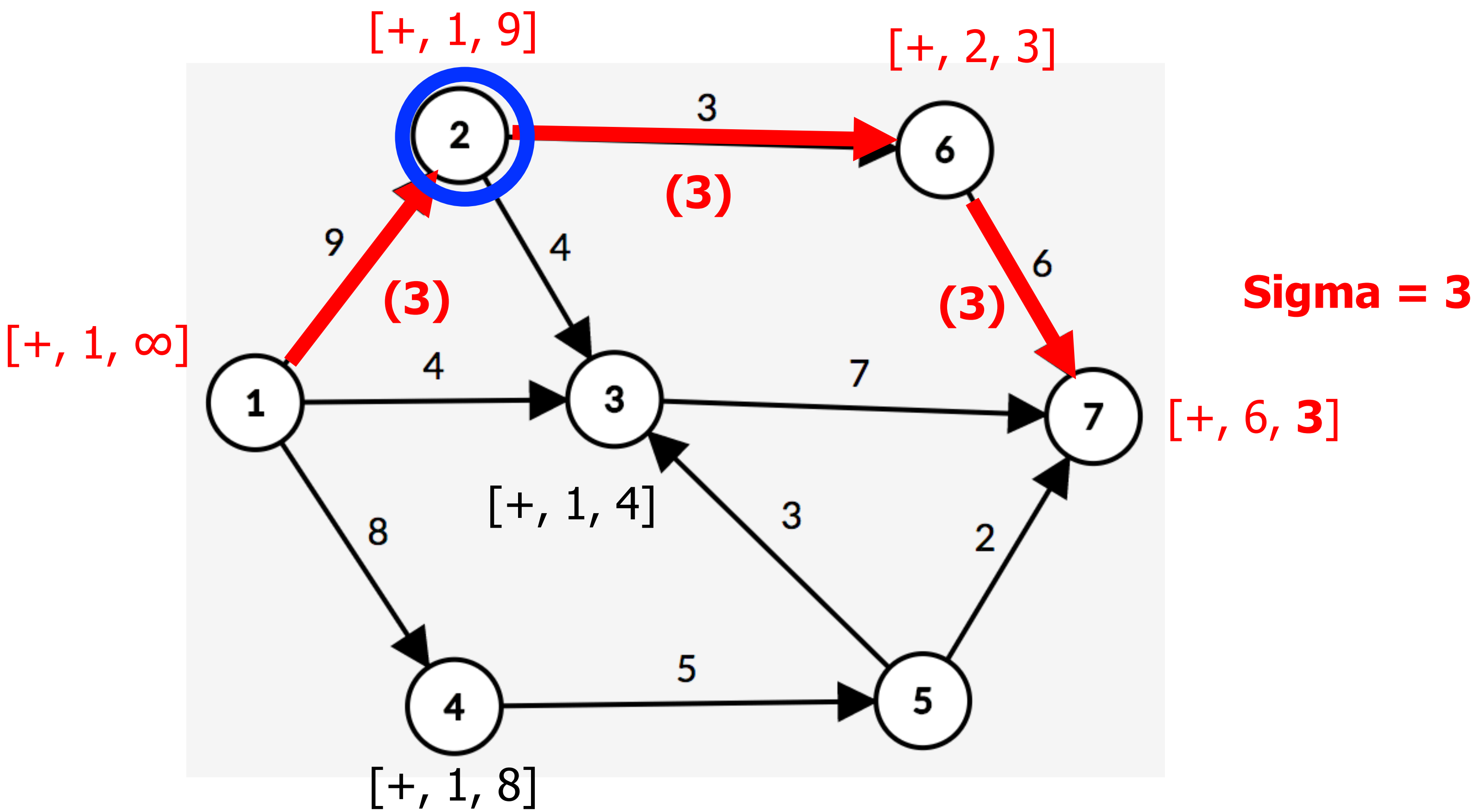
# Lặp 1

- Tăng luồng



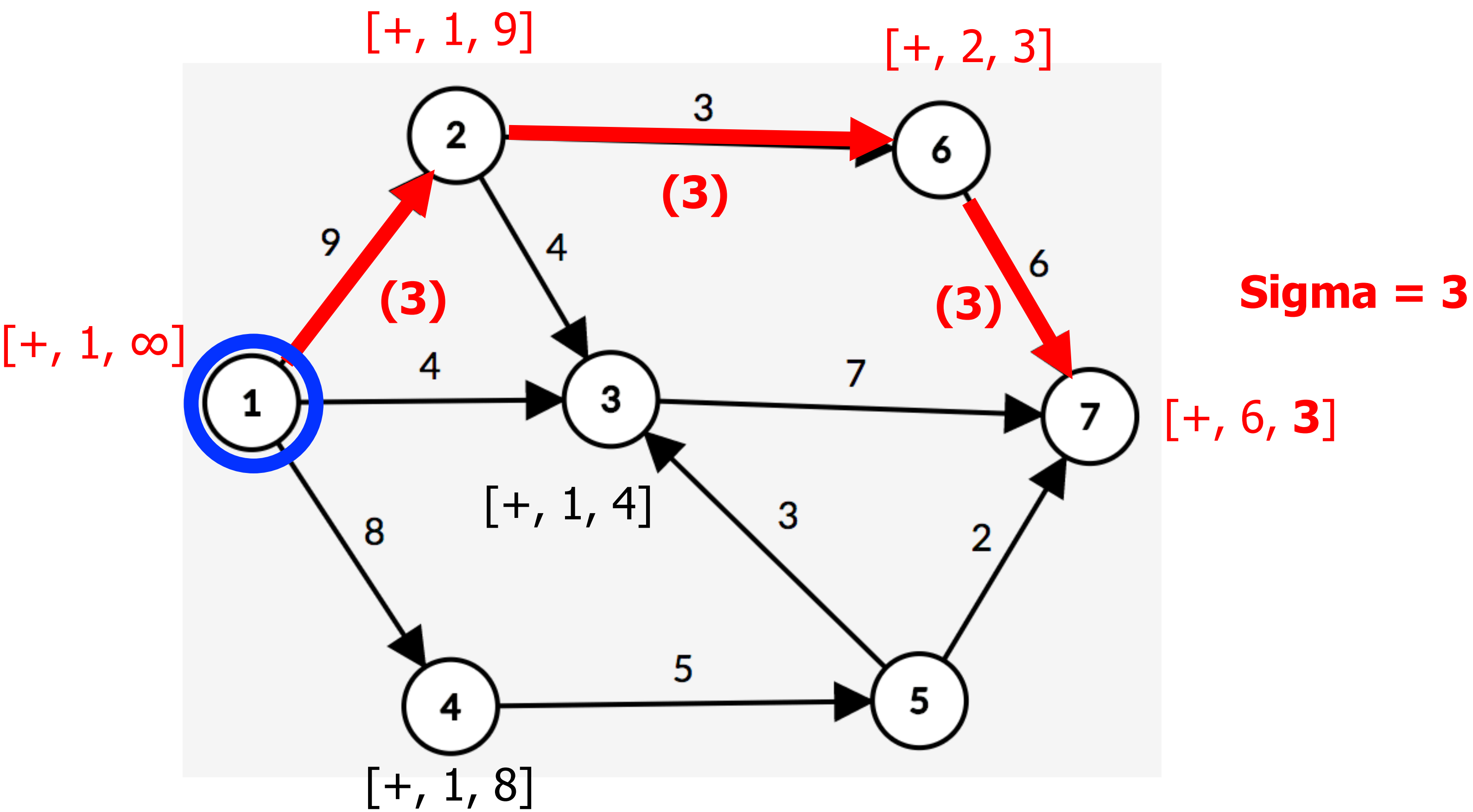
# Lặp 1

- Tăng luồng



# Lặp 1

- Tăng luồng



# Mẹo

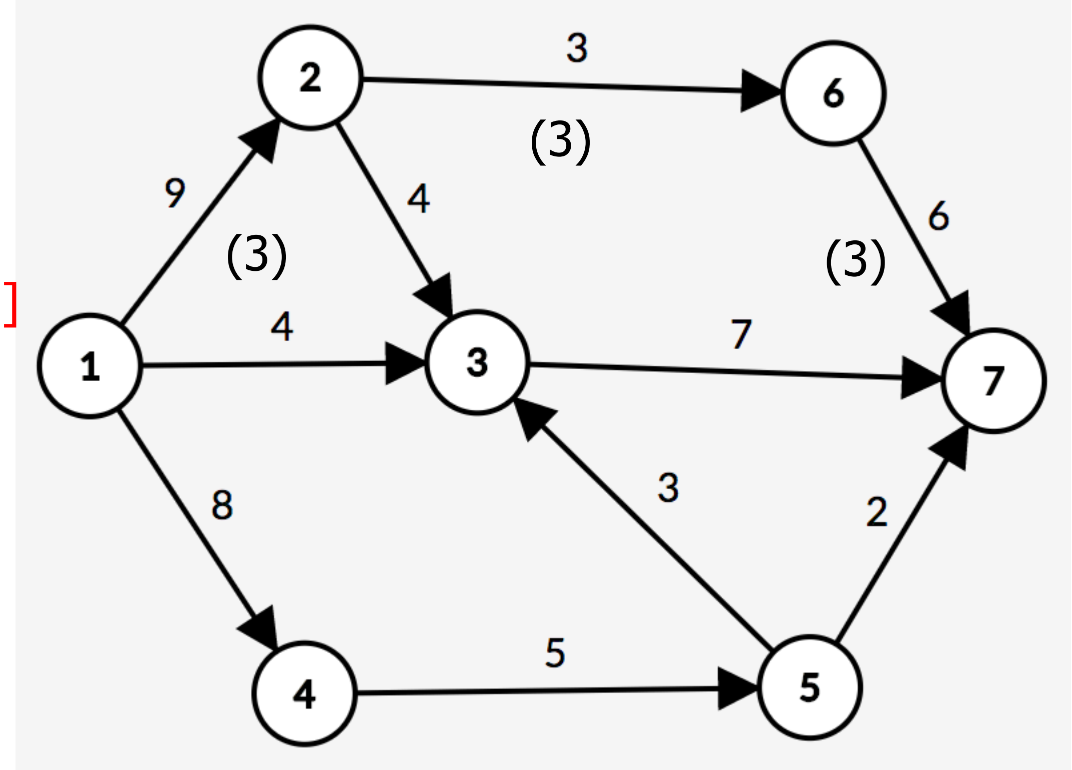
- Tất cả các cung trên đường tăng luồng sẽ được tăng/giảm 1 lượng giống nhau = **σ[t]**

- Trình bày:
  - Lặp 1:
    - Đánh dấu trên đồ thị gốc
    - Vẽ đồ thị mới và ghi kết quả tăng luồng trên đồ thị mới
  - Lặp 2:
    - Tiếp tục đánh dấu trên đồ thị mới
    - Vẽ đồ thị mới nữa và ghi kết quả tăng luồng
  - ...

# Lặp 2

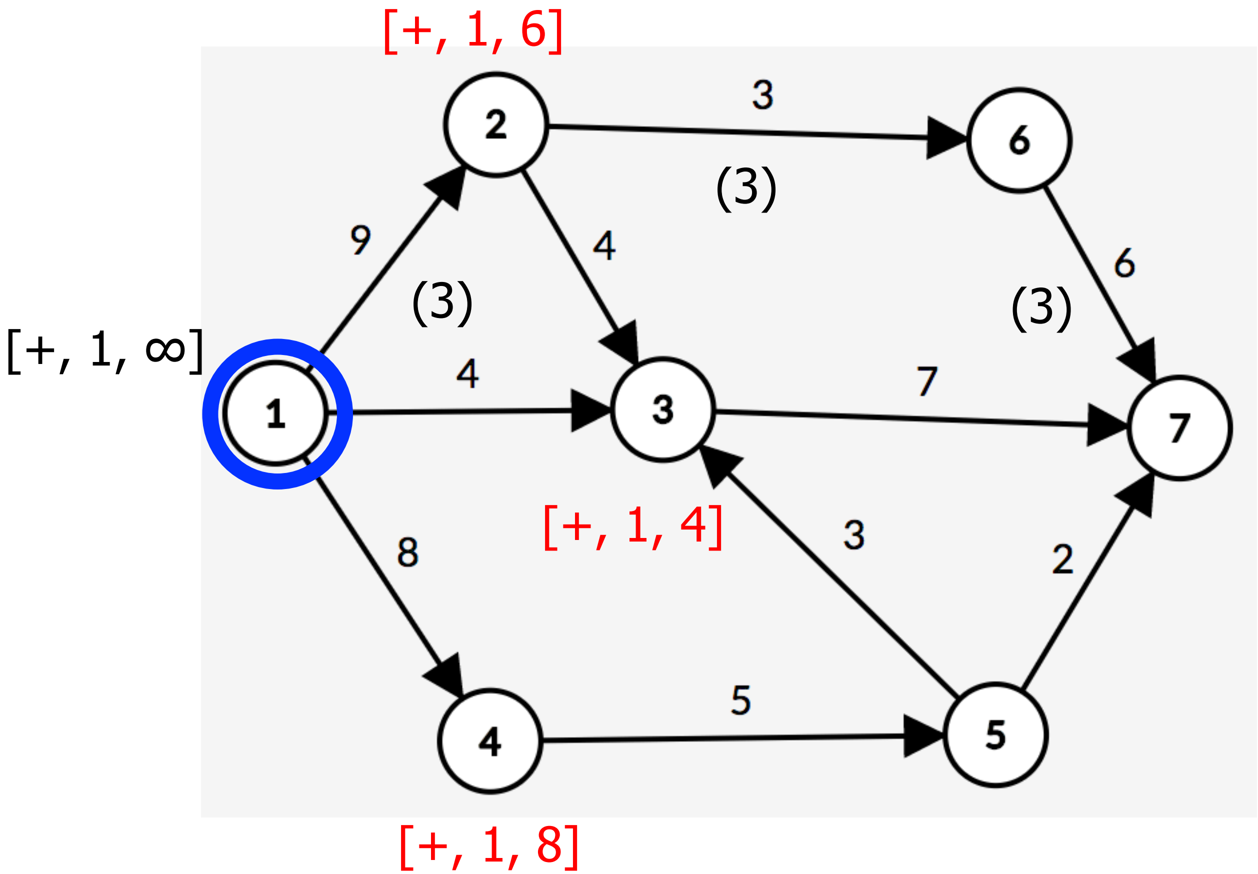
- Tìm đường tăng luồng

[+, 1, ∞]



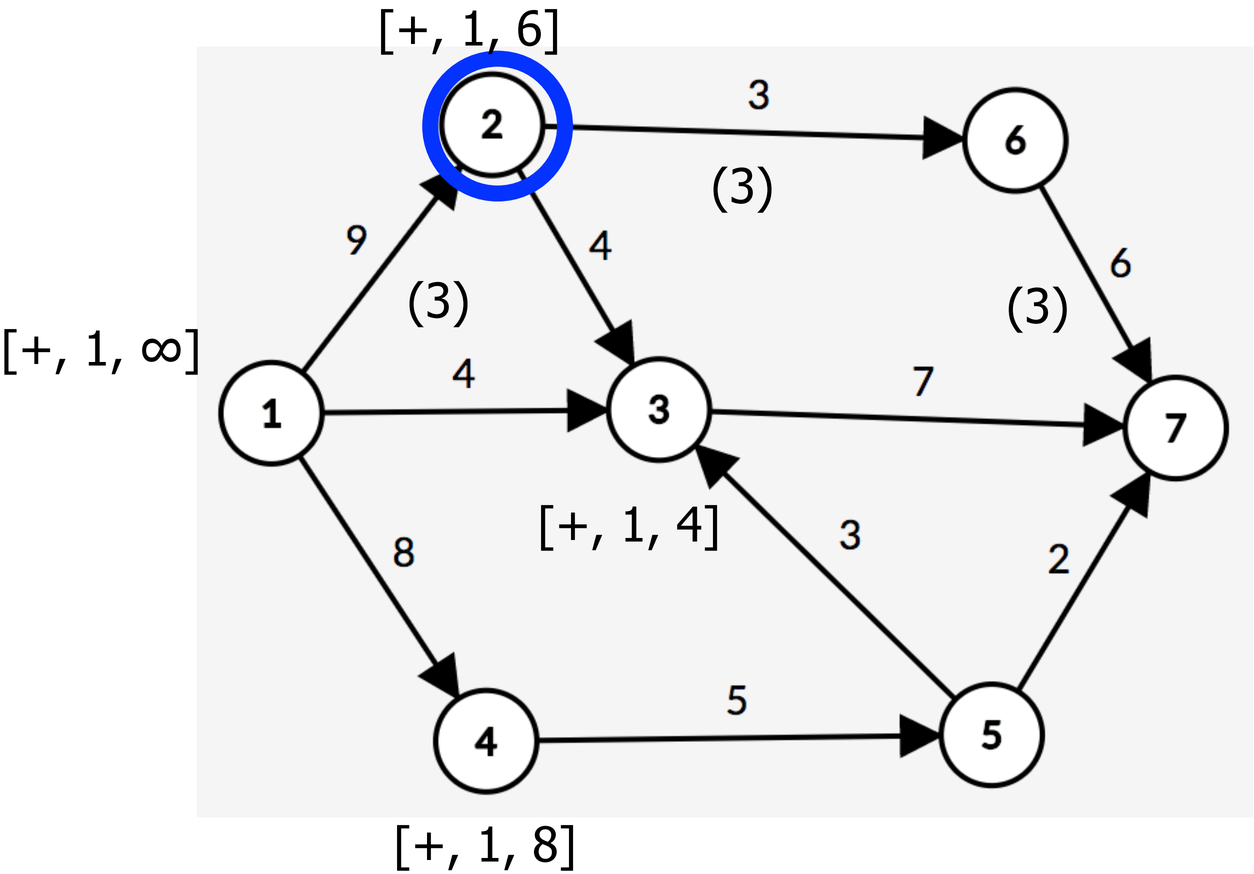
# Lặp 2

- Tìm đường tăng luồng



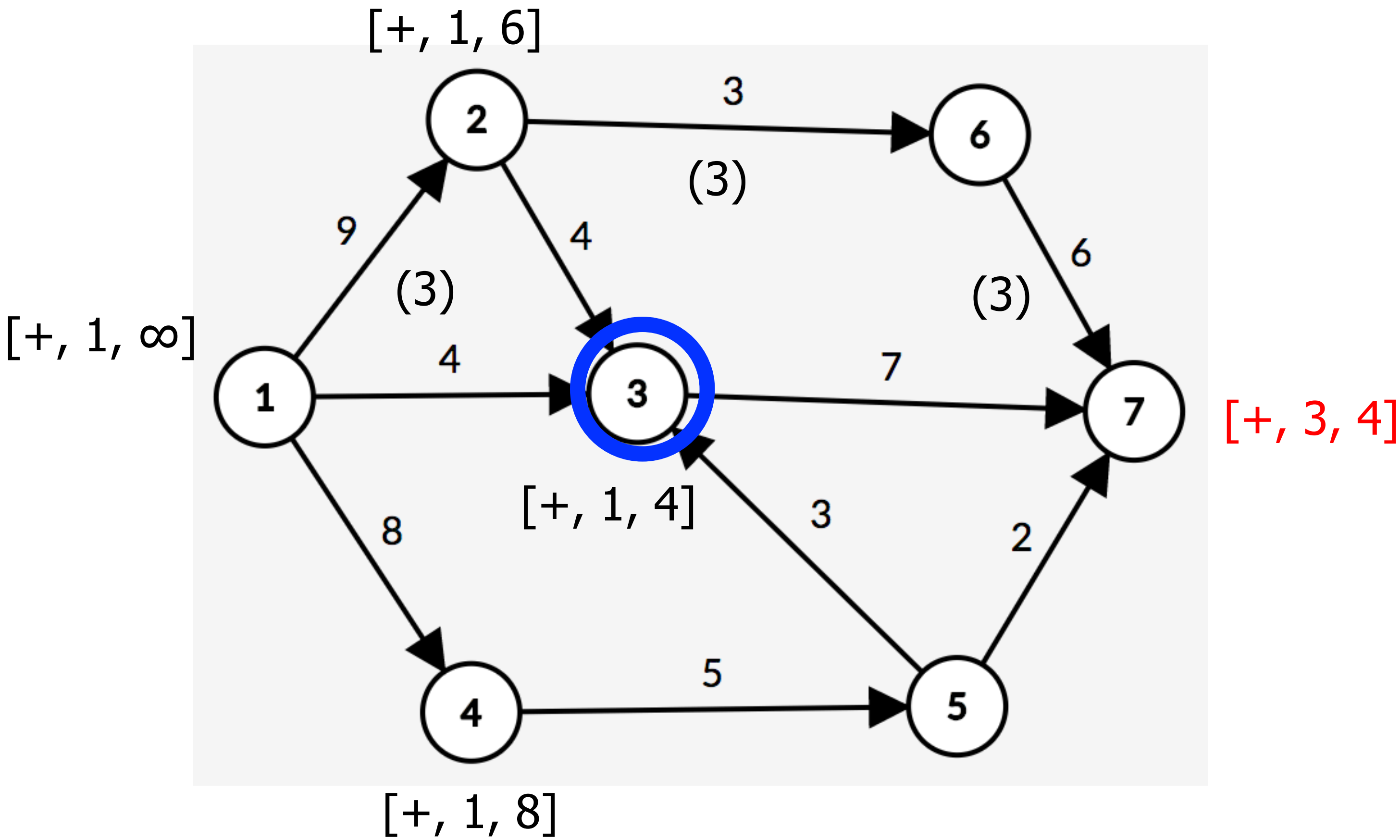
# Lặp 2

- Tìm đường tăng luồng



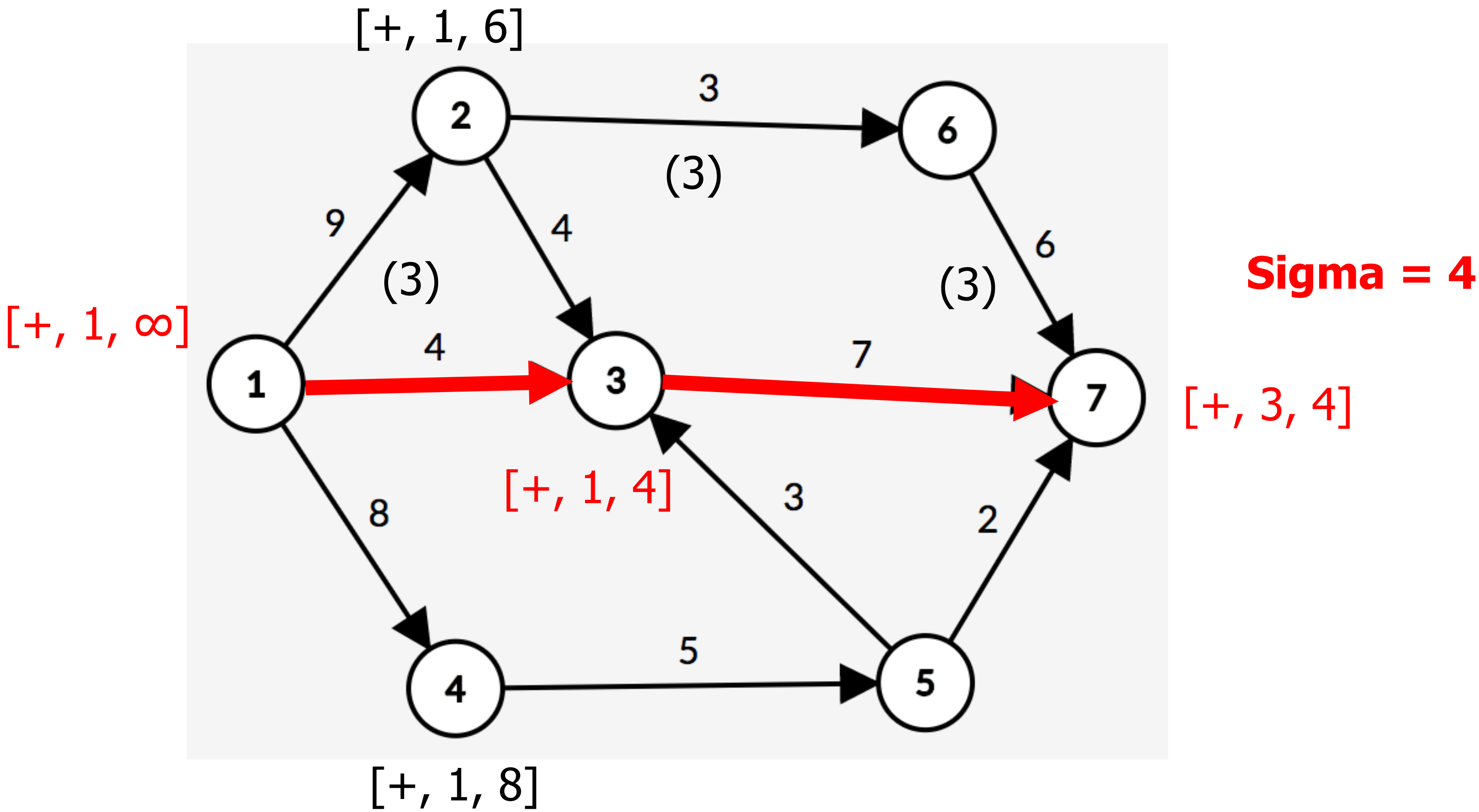
# Lặp 2

- Tìm đường tăng luồng



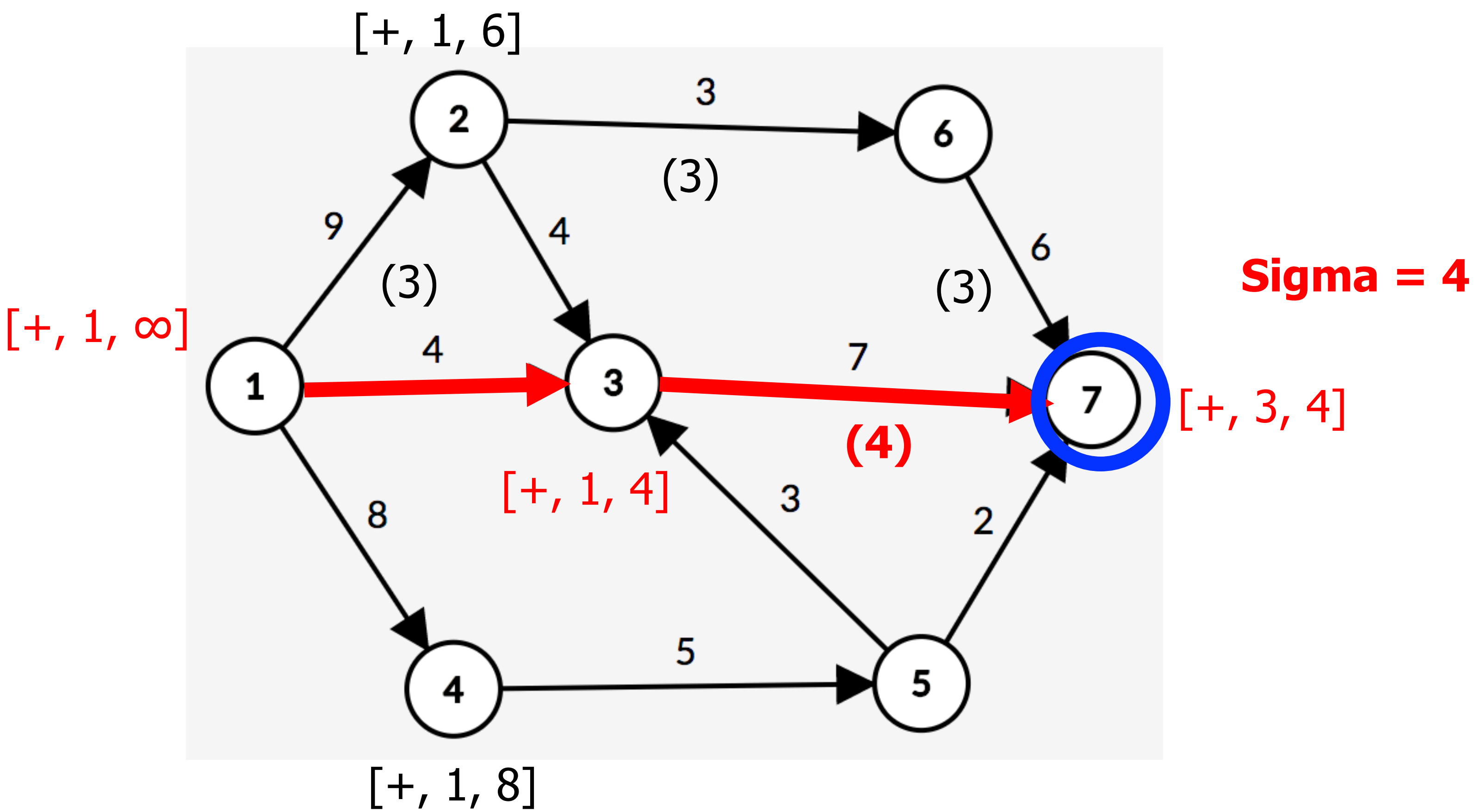
# Lặp 2

- Tìm đường tăng luồng



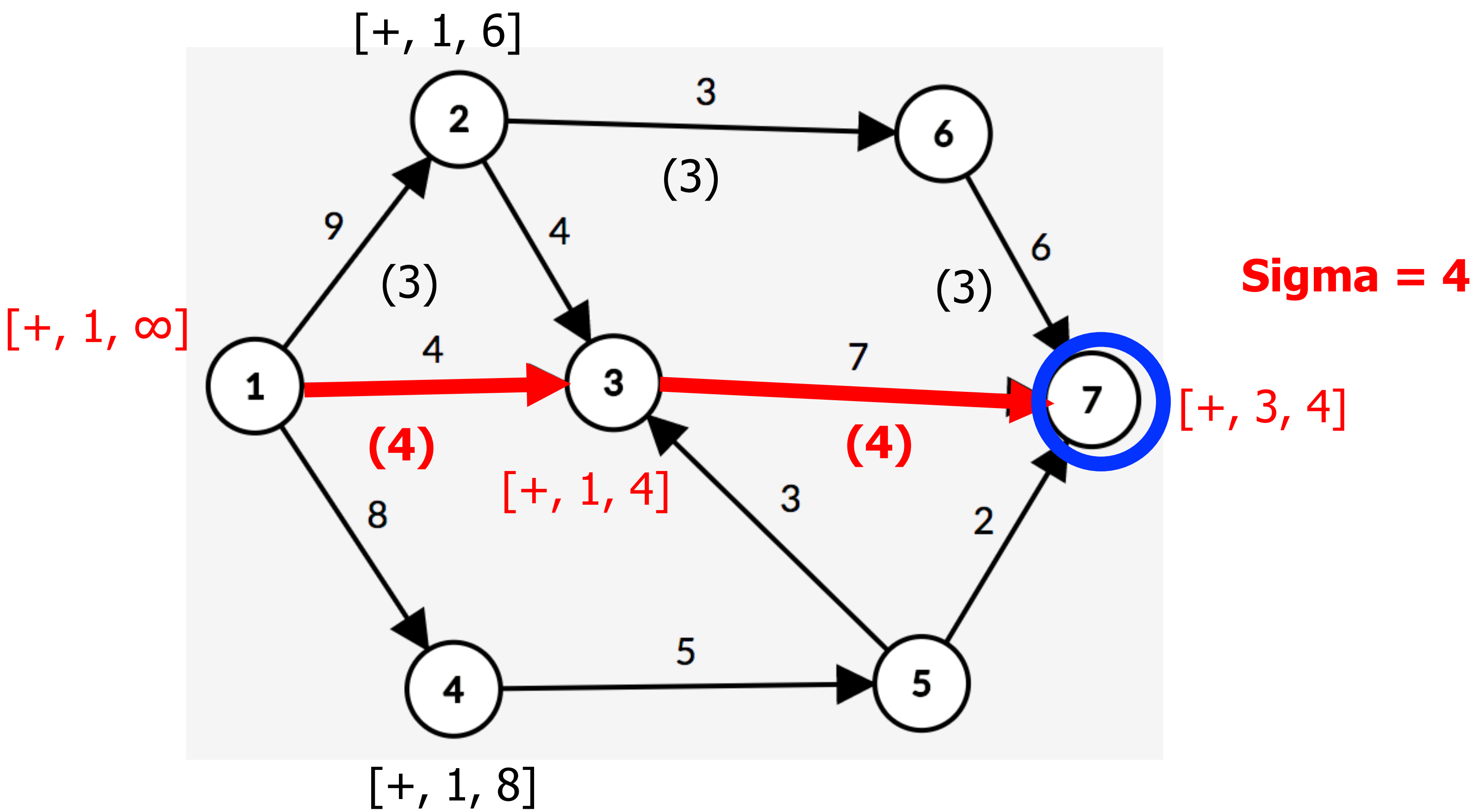
# Lặp 2

- Tăng luồng



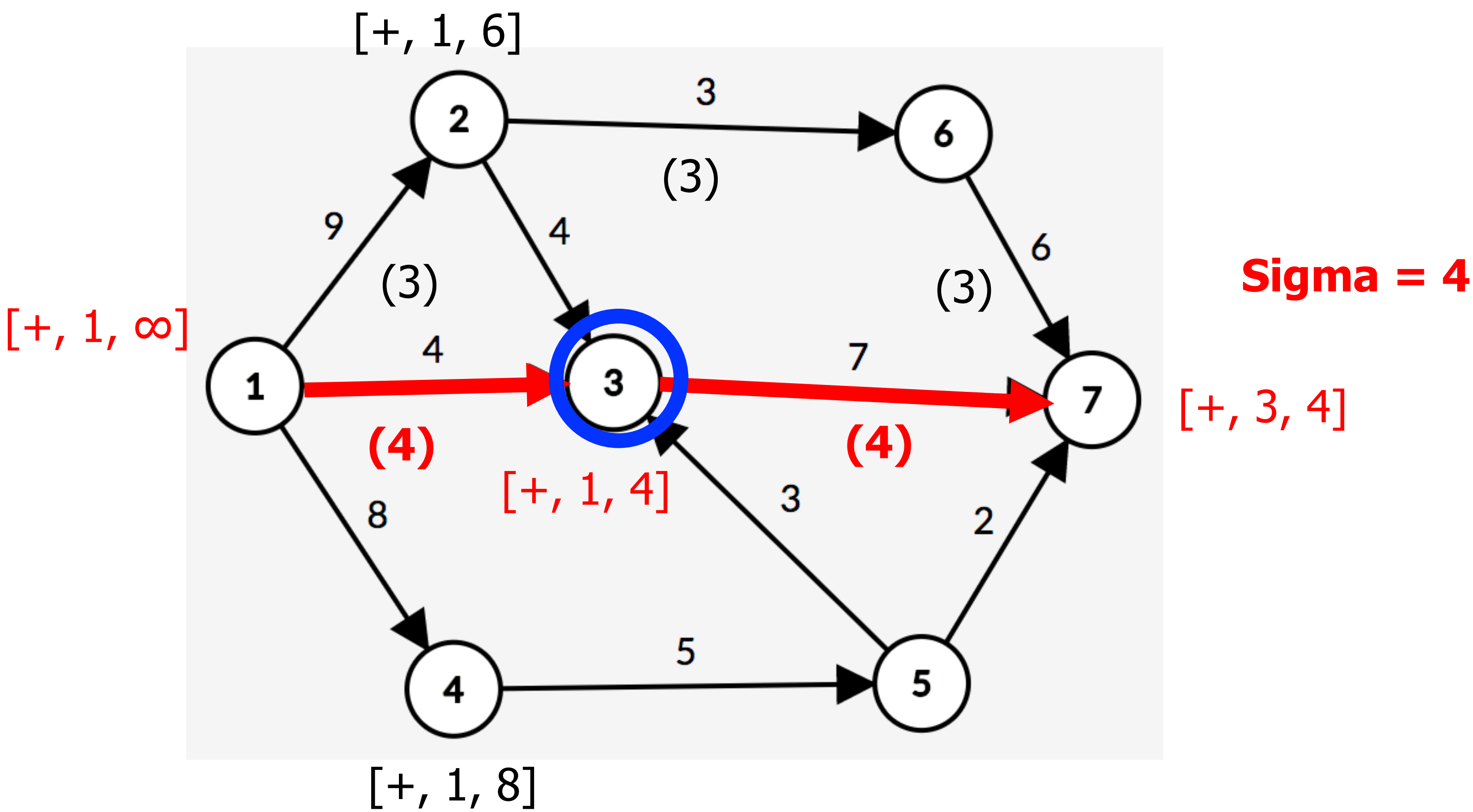
# Lặp 2

- Tăng luồng



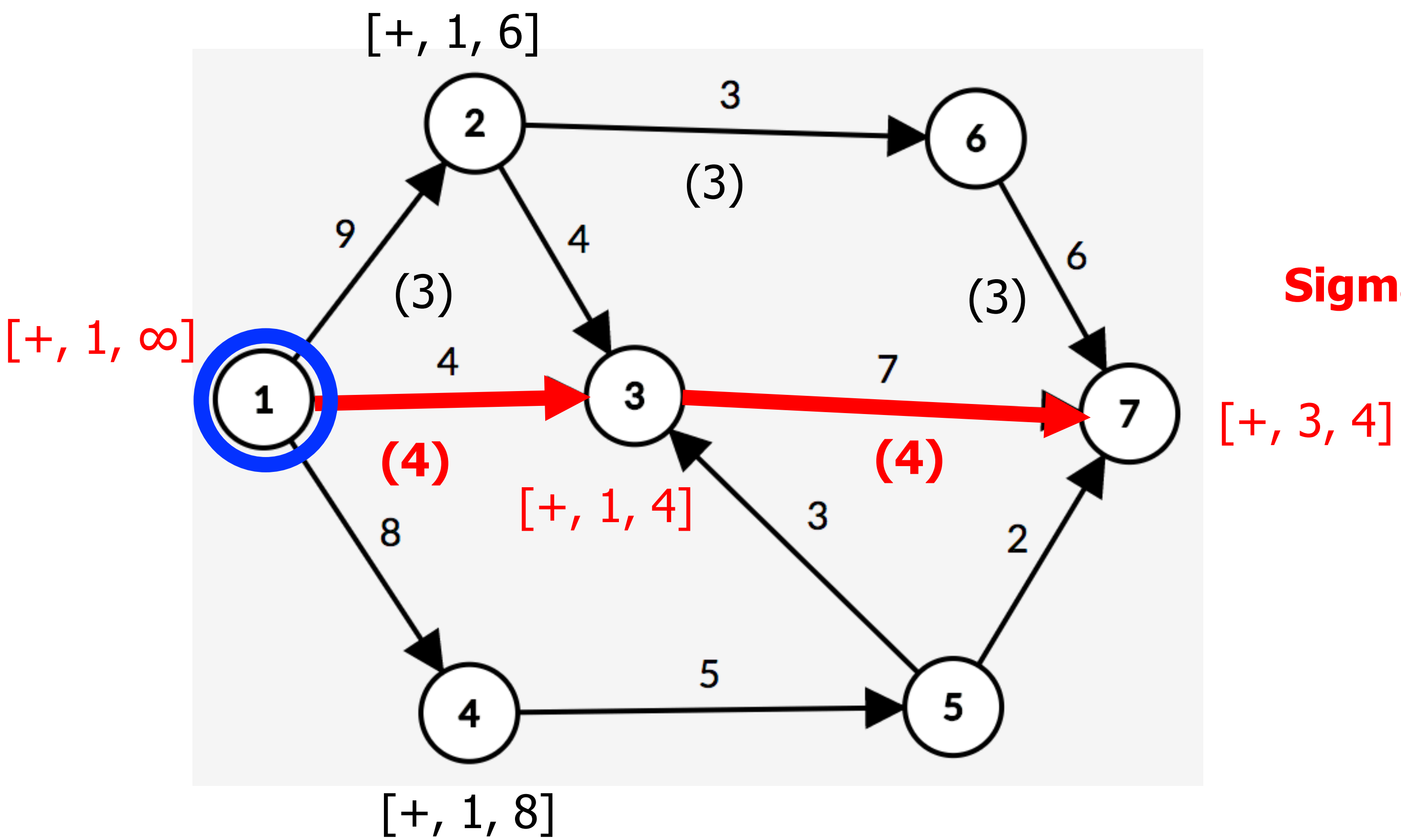
# Lặp 2

- Tăng luồng



# Lặp 2

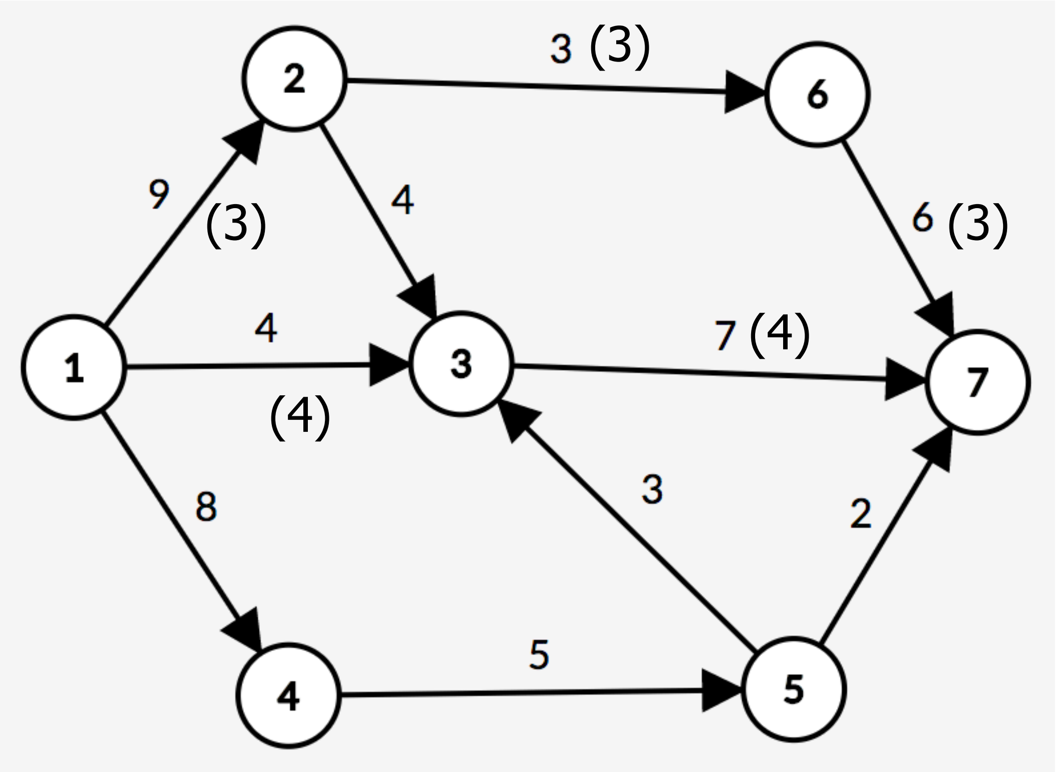
- Tăng luồng




Sigma = 4

# Lặp 3

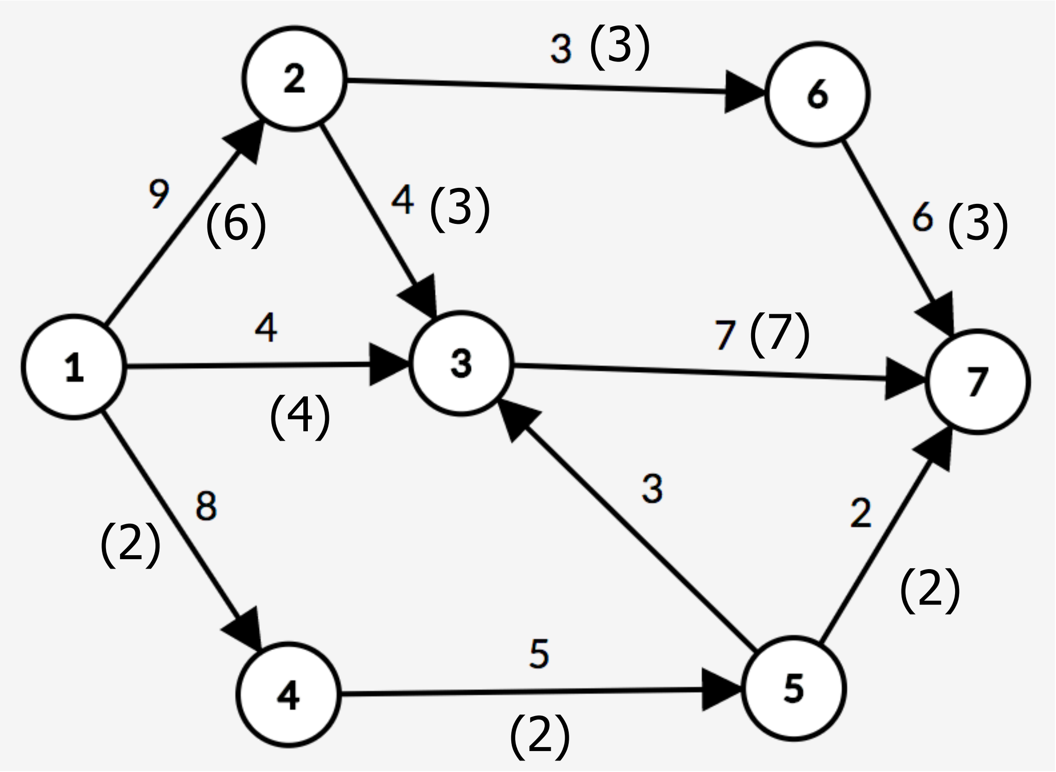
- Tìm đường tăng luồng



- ...

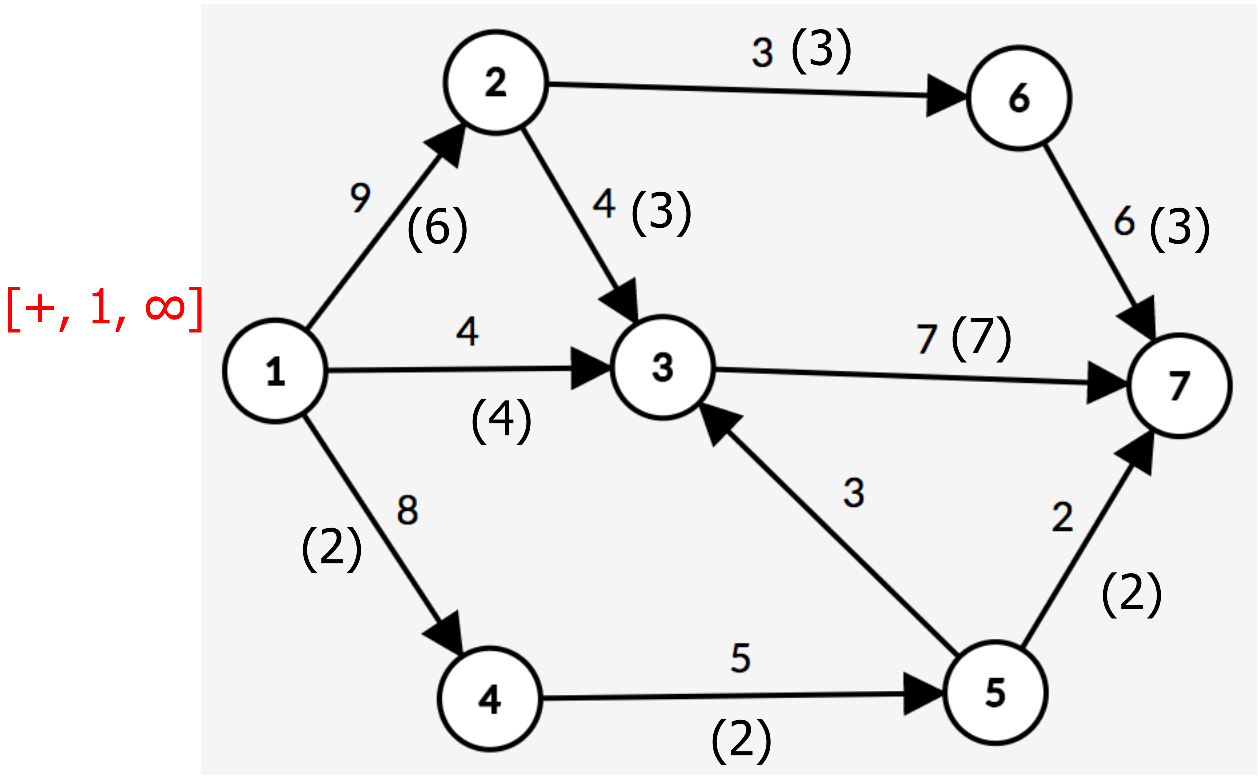
# Lặp 5

- Tìm đường tăng luồng



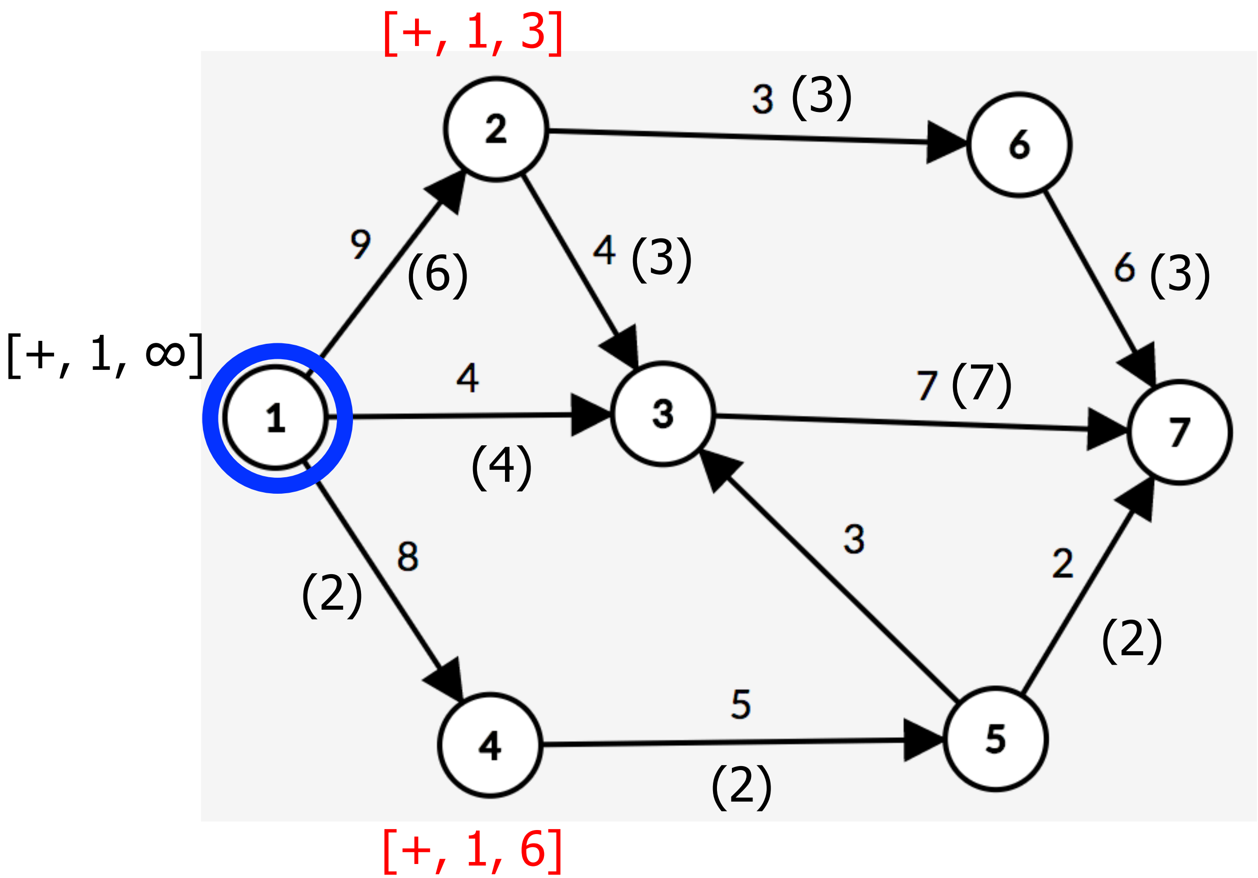
# Lặp 5

- Tìm đường tăng luồng



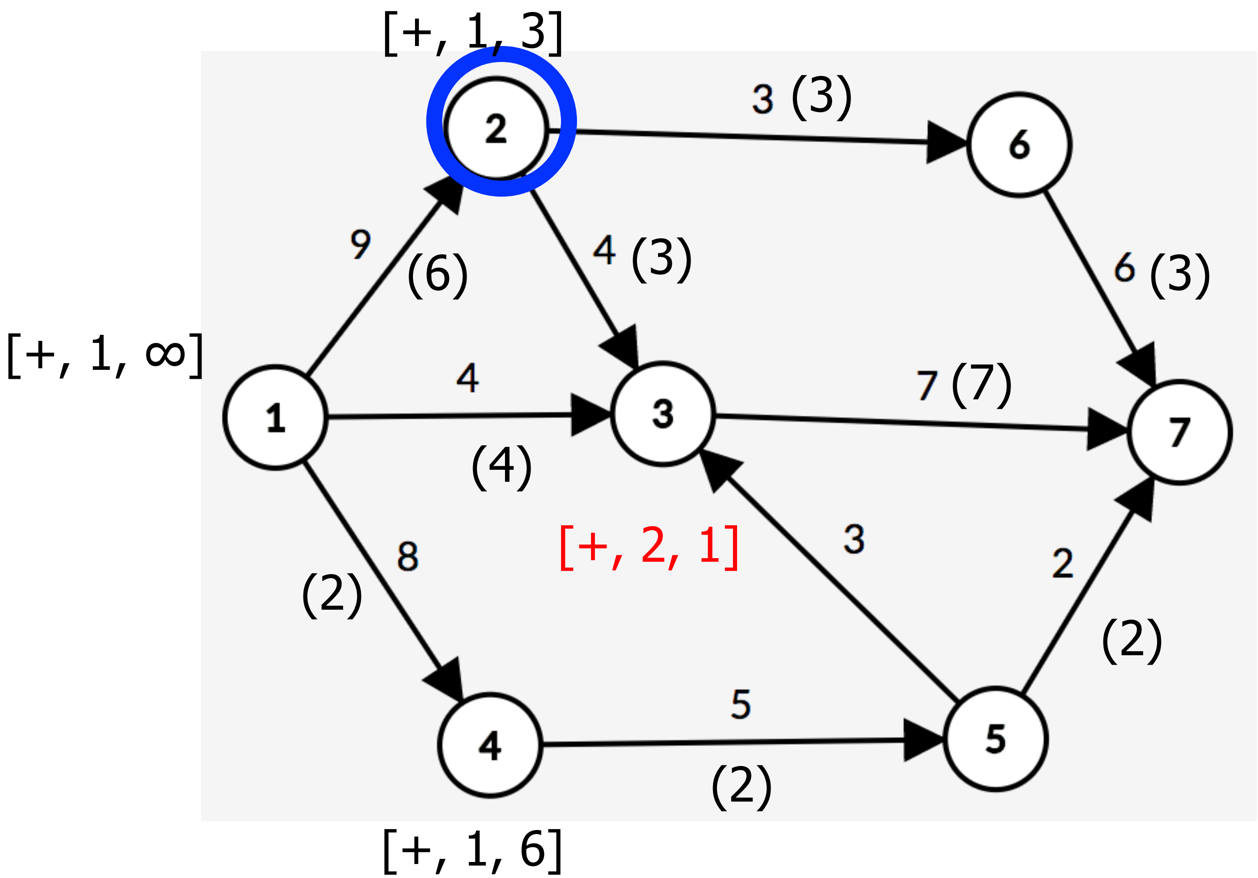
# Lặp 5

- Tìm đường tăng luồng



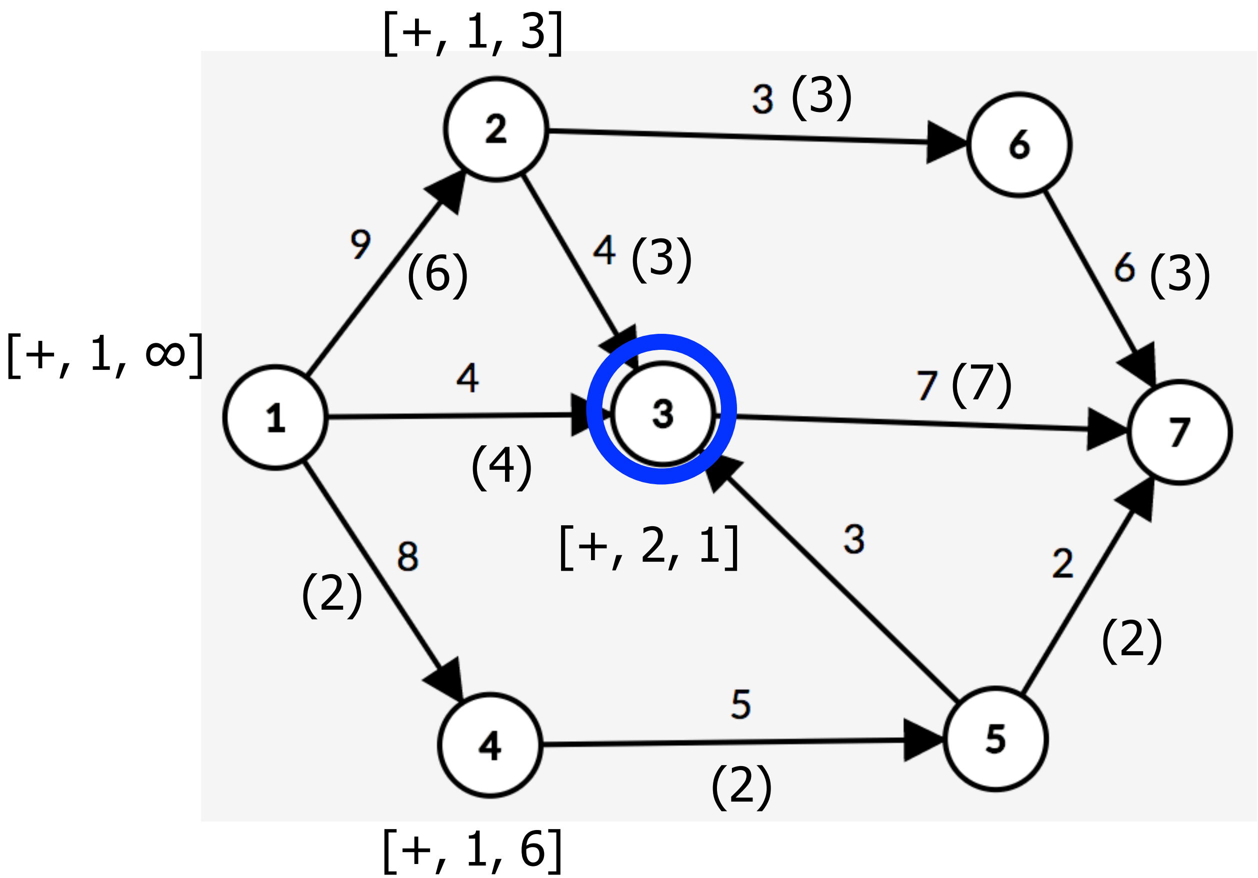
# Lặp 5

- Tìm đường tăng luồng



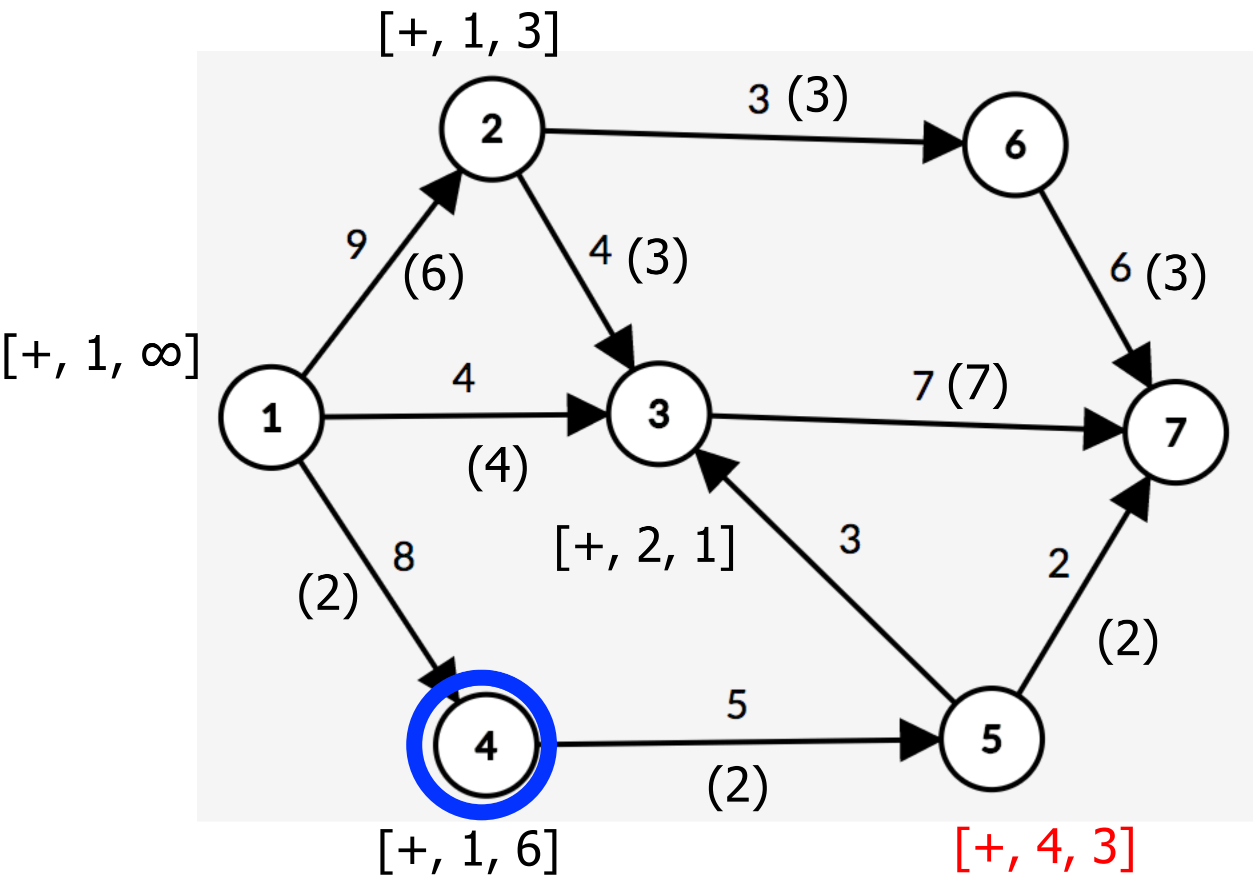
# Lặp 5

- Tìm đường tăng luồng



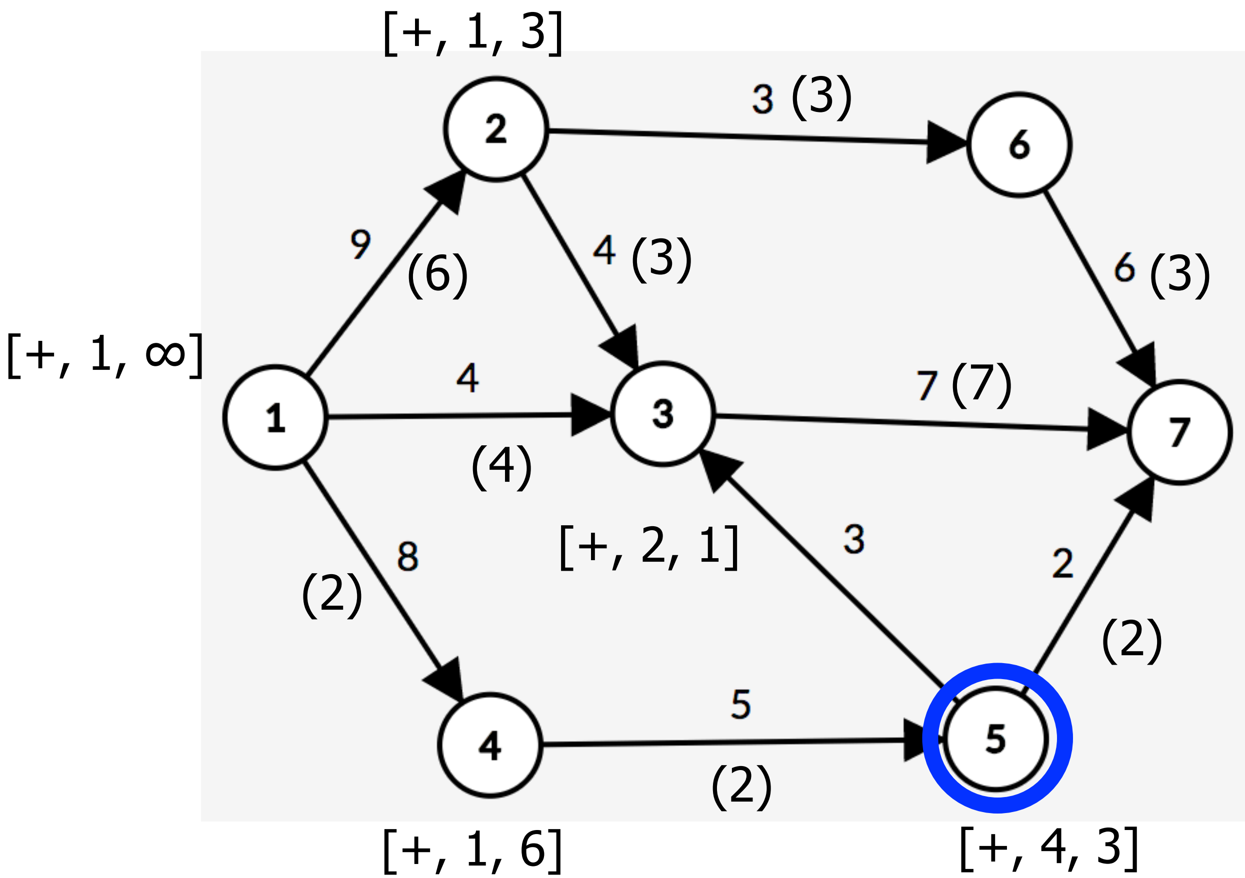
# Lặp 5

- Tìm đường tăng luồng



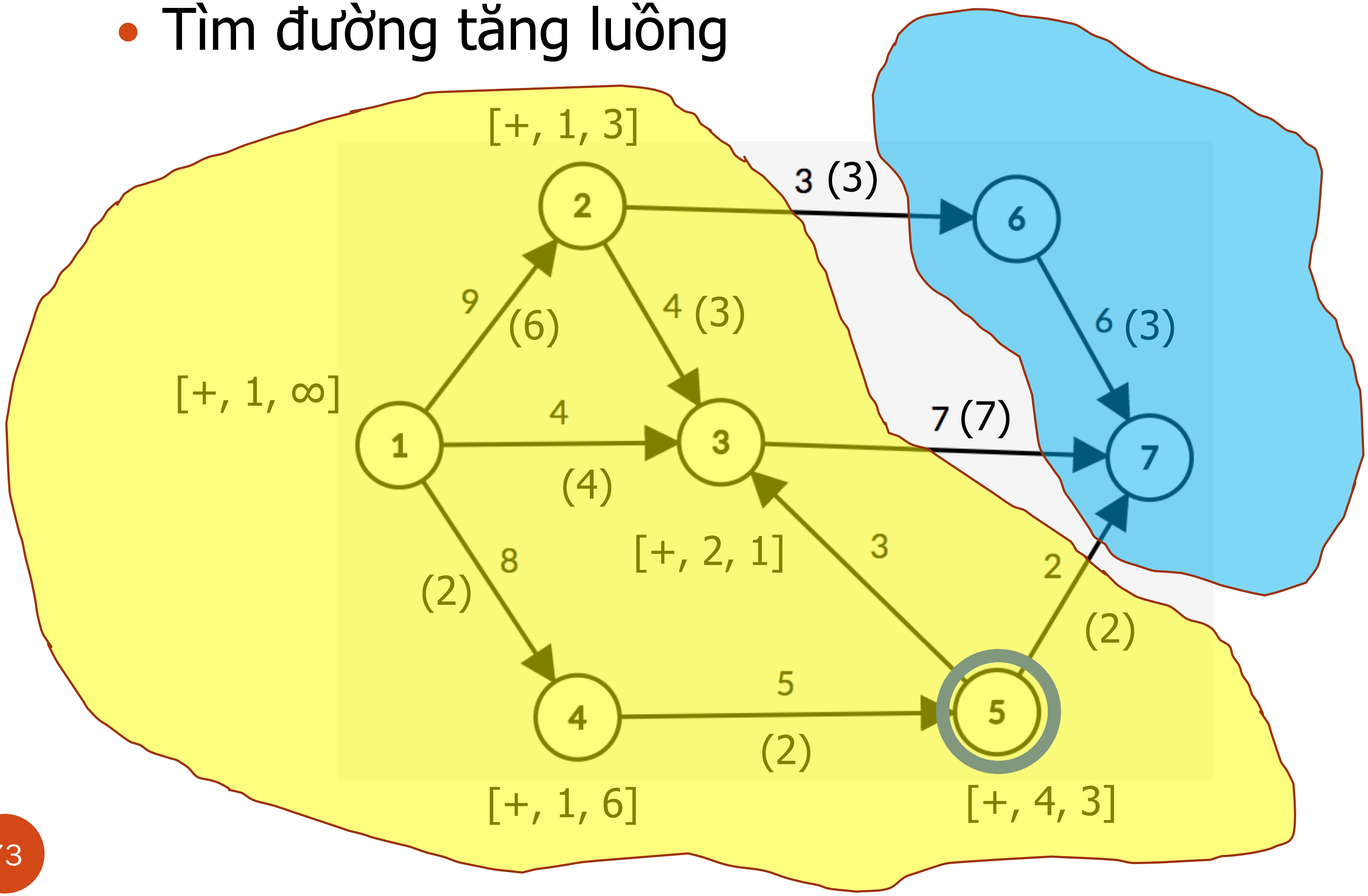
# Lặp 5

- Tìm đường tăng luồng



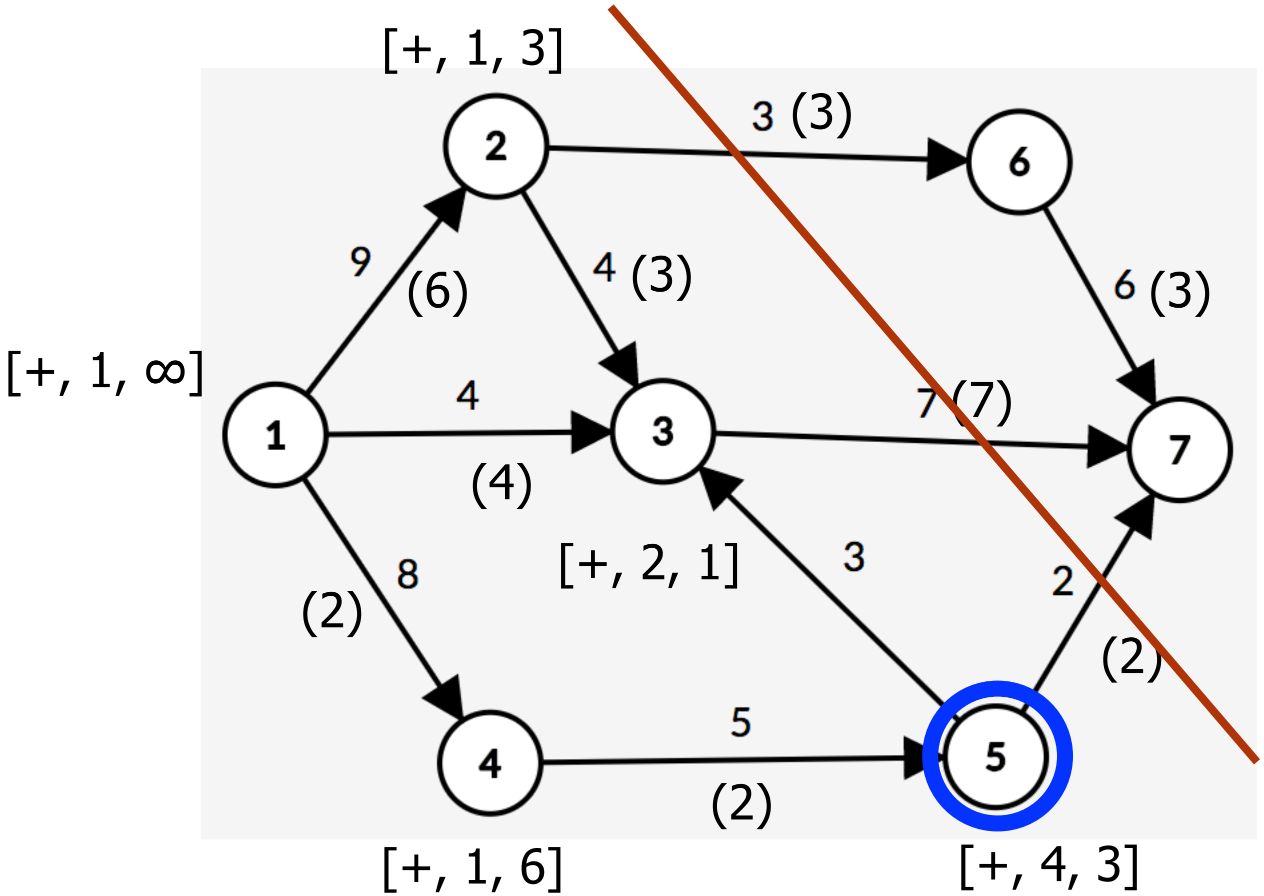
# Lặp 5

- Tìm đường tăng luồng



# Lặp 5

- Tìm đường tăng luồng



# Kết quả

- Lát cắt hẹp nhất (S, T)
  - S = {1, 2, 3, 4, 5}
  - T = {6, 7}

- Giá trị luồng cực đại:
  - $|f|_{max} = c(S, T) = 3 + 7 + 2 = 12$

# Cải tiến/biến thể của giai thuật Ford-Fulkerson

- Giải thuật Ford-Fulkerson thường được gọi là **phương pháp** (method) Ford-Fulkerson thay vì **giải thuật** (algorithm)
  - Không mô tả rõ ràng cách tìm đường tăng luồng trên đồ thị tăng luồng

- Biến thể:
  - Giải thuật **Edmonds-Karp**
    - Tìm kiếm theo chiều rộng, sử dụng Queue thay vì Stack
  - Tìm đường tăng luồng có lượng tăng luồng nhiều nhất (cải biên Dijkstra)

# Bài tập

- Cho đồ thị G như hình. Hãy vận dụng giải thuật Ford-Fulkerson để tìm luồng cực đại và lát cắt hẹp nhất