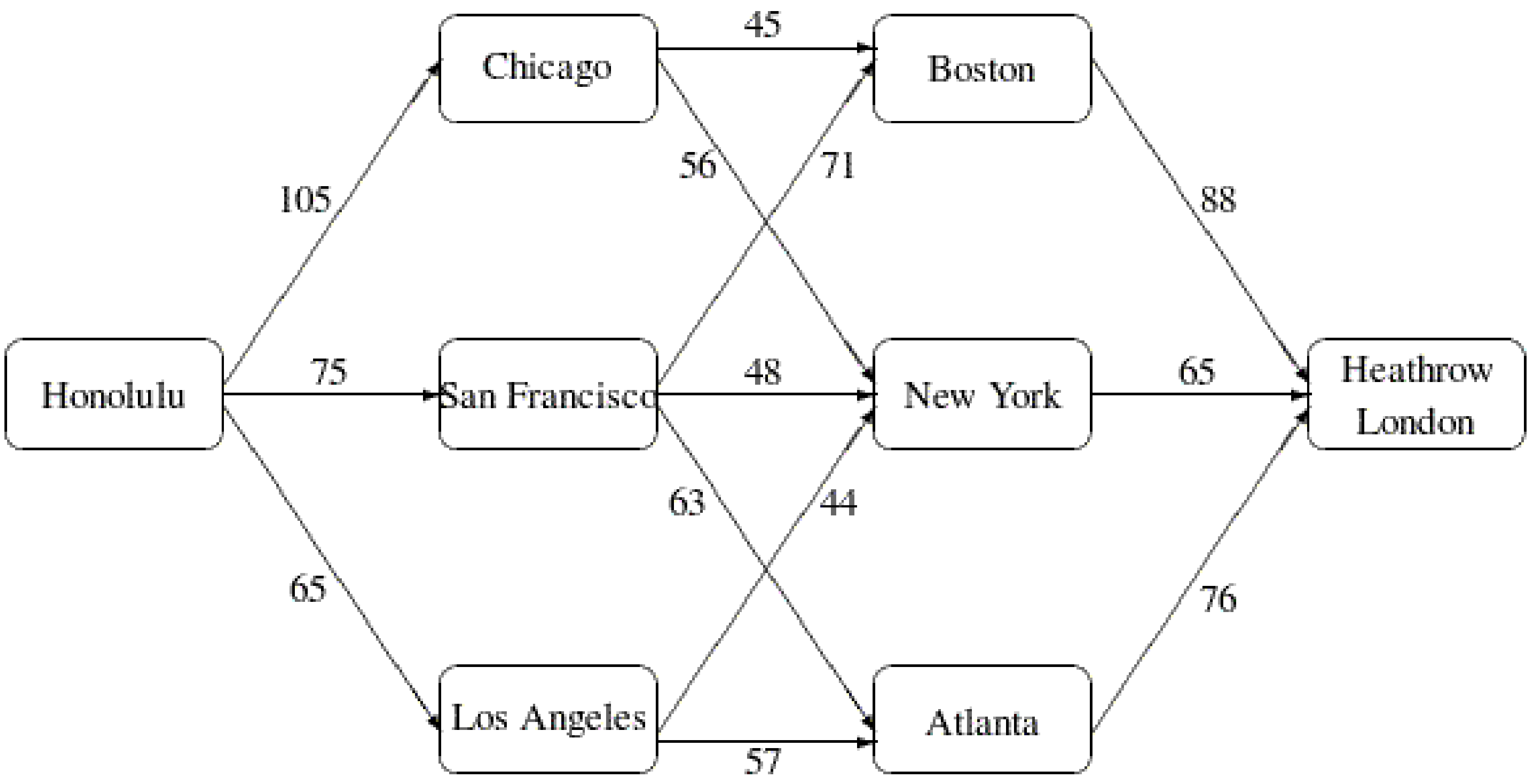
CHƯƠNG 8

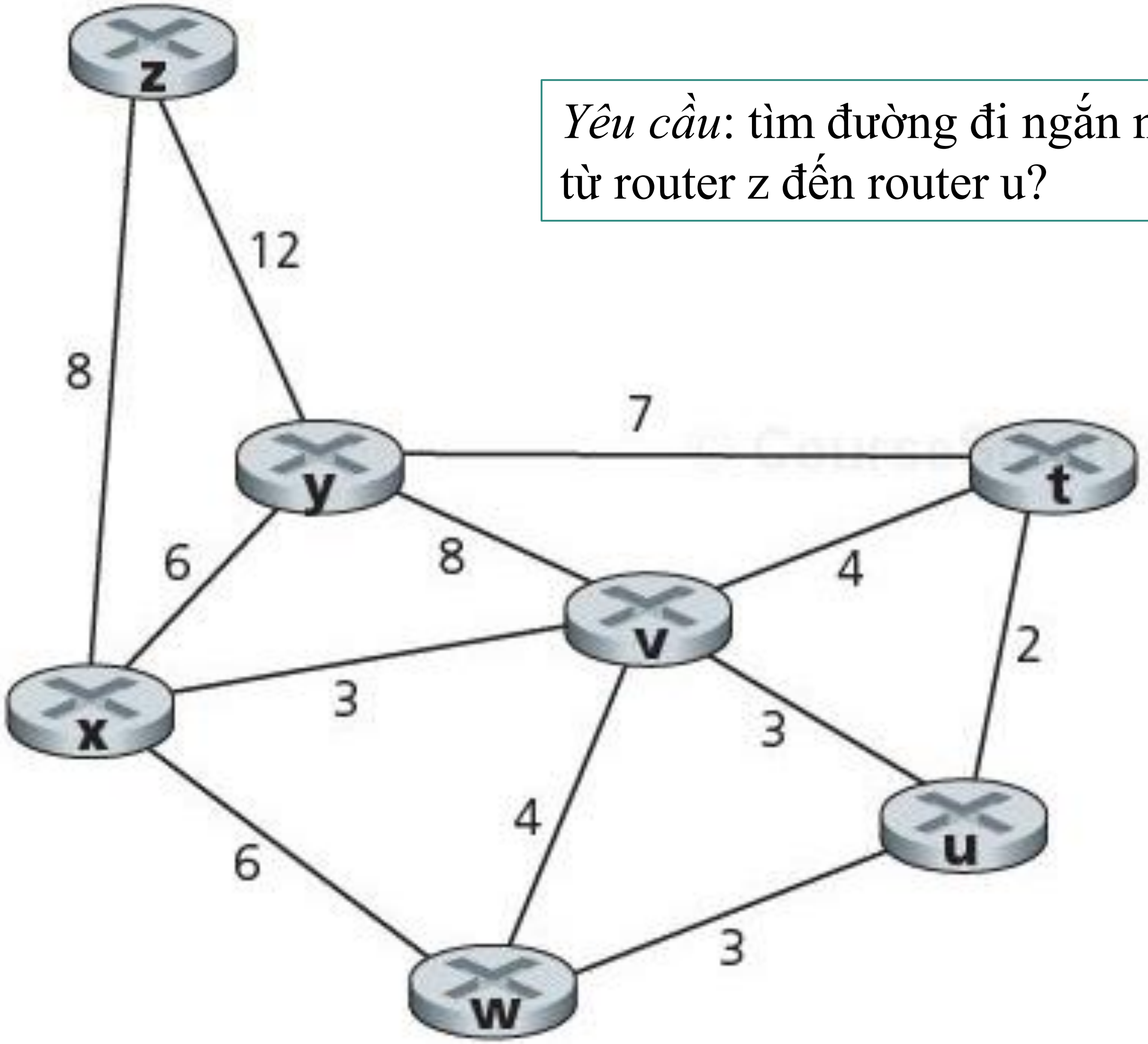
TÌM ĐƯỜNG ĐI NGẮN NHẤT VÀ LUỒNG CỰC ĐẠI TRÊN MẠNG

Bài toán đường đi ngắn nhất (1/5) (SPP – Shortest path problem)



Yêu cầu: tìm đường đi ngắn nhất từ Chicago tới Atlanta?

Bài toán đường đi ngắn nhất (2/5)



Yêu cầu: tìm đường đi ngắn nhất để truyền gói tin từ router z đến router u?

Bài toán đường đi ngắn nhất (3/5)

Phát biểu

- Cho đồ thị có hướng, có trọng số G(V,E,C)

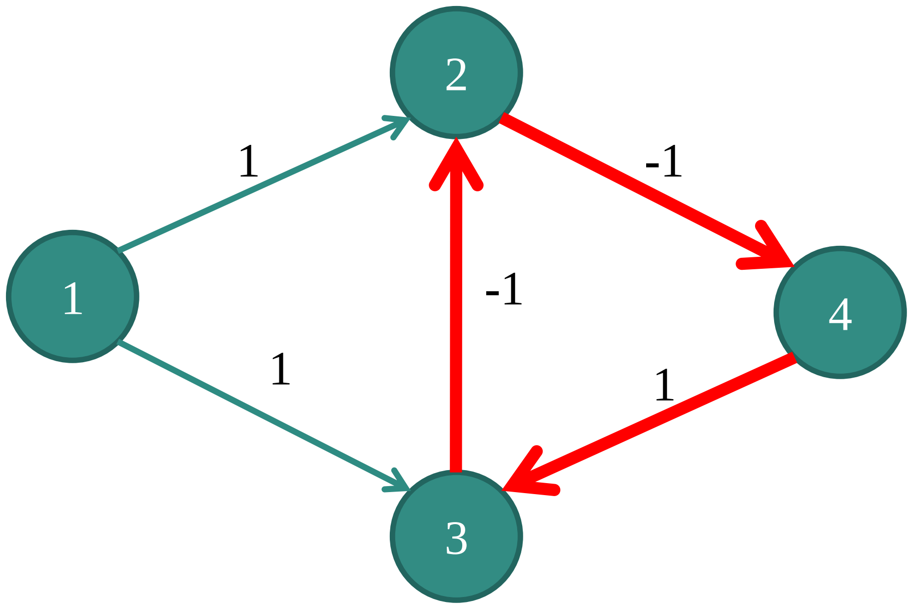
Yêu cầu

- Tìm đường đi ngắn nhất giữa 2 đỉnh bất kỳ
- *Tìm đường đi ngắn nhất từ một đỉnh đến mọi đỉnh còn lại của đồ thị*
- Tìm đường đi ngắn nhất giữa mọi cặp đỉnh của đồ thị

Bài toán đường đi ngắn nhất (4/5)

Điều kiện tồn tại đường đi ngắn nhất

- Đồ thị không chứa chu trình âm
- Chu trình âm là chu trình có tổng trọng số âm.



Bài toán đường đi ngắn nhất (5/5)

- **Đầu vào:**
  - Đồ thị có hướng, có trọng số G(V,E,C).
  - [G không chứa chu trình âm].
  - Đỉnh xuất phát s.
- **Đầu ra:**
  - Đường đi ngắn nhất từ s đến các đỉnh còn lại của G.
  - Đường đi ngắn nhất là đường đi có tổng trọng số nhỏ nhất.

Thuật toán Dijkstra (1/12)

- **Đặc điểm:**
  - Chỉ áp dụng với đồ thị có trọng số không âm.
  - Tìm đường đi ngắn nhất từ 1 đỉnh xuất phát đến mọi đỉnh còn lại của đồ thị.
- **Ý tưởng: sử dụng nguyên lý tham lam (greedy)**
  - Tại mỗi bước luôn chọn đường đi ngắn nhất có thể.
  - Đường đi ngắn nhất tới đỉnh chưa xét được xây dựng từ đường đi ngắn nhất qua các đỉnh đã được xét.

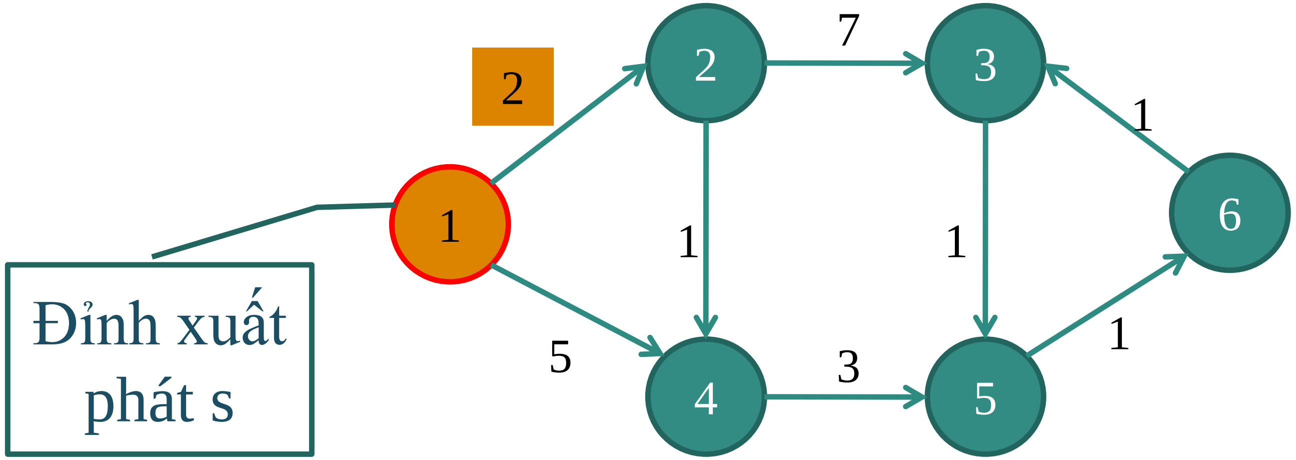
Thuật toán Dijkstra (2/12)

Dijkstra(V,E,C,s){

- Với mỗi đỉnh v thuộc V gán:
  - distance(v) = vô cùng; previous(v) = s;
- Distance(s)=0; Queue = V \ {s}; SP = s;
- While(Queue khác rỗng){
  - **Lấy x là đỉnh có distance(x) = Min {distance(Queue)}**
  - Nạp x vào SP
  - Với mỗi đỉnh y thuộc Queue và kề với x,
  - nếu distance(y) < distance(x)+c(x,y){
    - Cập nhật distance(y) = distance(x)+c(x,y);
    - Cập nhật previous(y) = x;
  - }
- }
- Return SP;

}

Thuật toán Dijkstra (3/12)

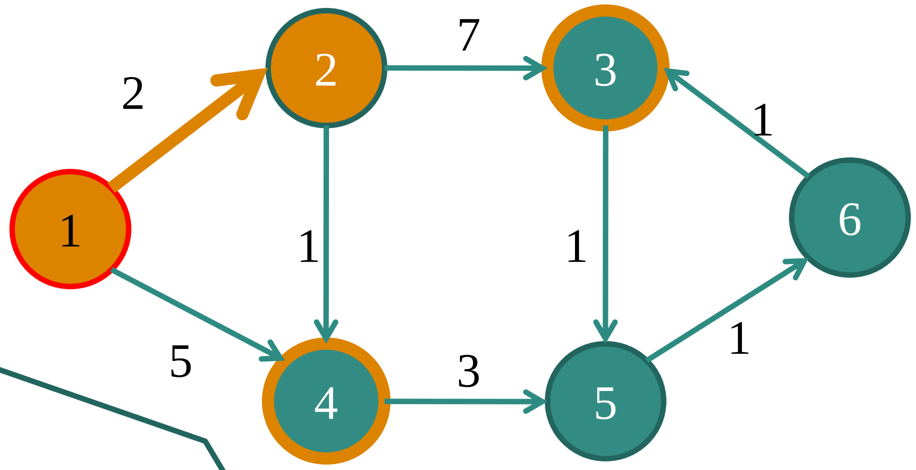


Đỉnh =>	1	2	3	4	5	6
0	0,1	2,1	∞,1	5,1	∞,1	∞,1

Distance(2)=2

Previous(2)=1

Thuật toán Dijkstra (4/12)

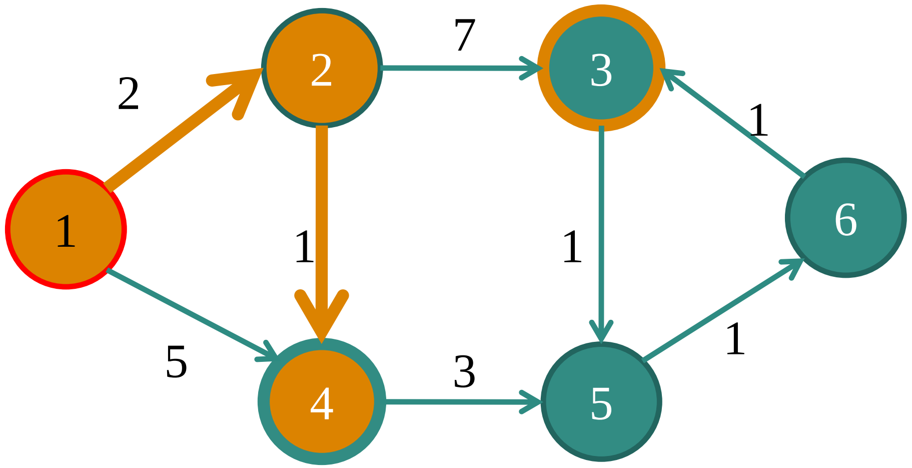


Chọn đỉnh có distance nhỏ nhất

	1	2	3	4	5	6
0	0,1	2,1	∞,1	5,1	∞,1	∞,1
1			9,2	3,2	---	---

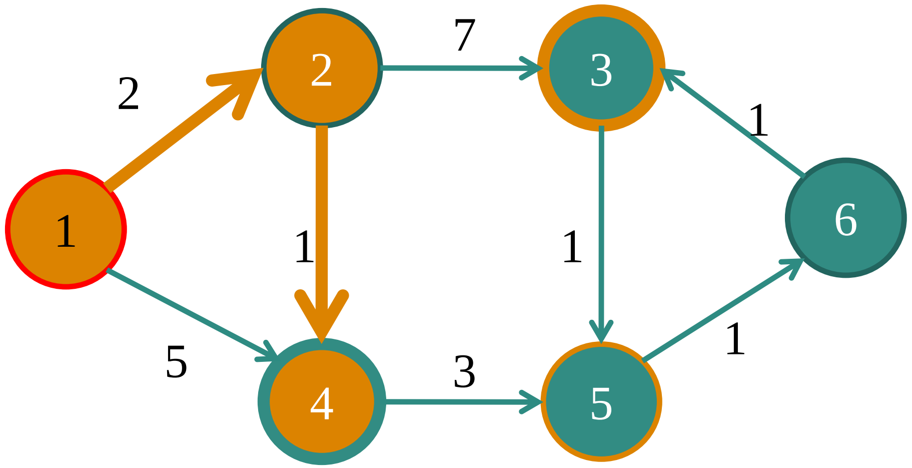
Cập nhật lại distance và previous của các đỉnh kề với đỉnh mới được chọn

Thuật toán Dijkstra (5/12)



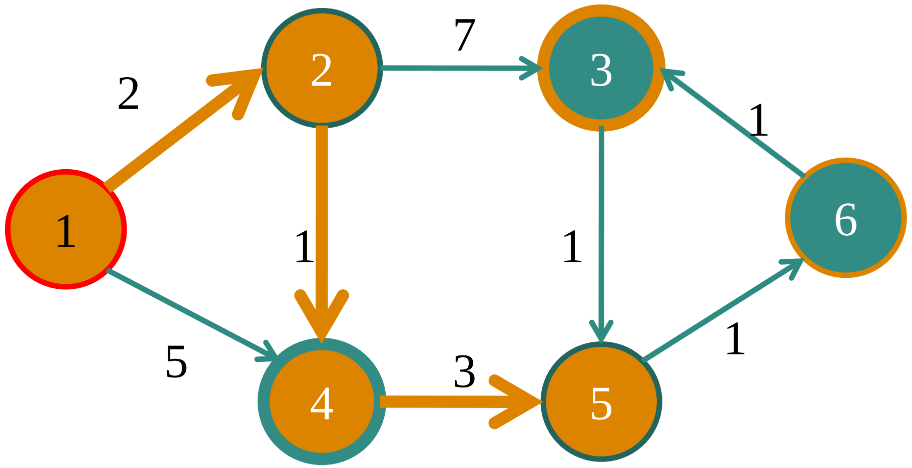
	1	2	3	4	5	6
0	0,1	2,1	∞,1	5,1	∞,1	∞,1
1			9,2	3,2	---	---

Thuật toán Dijkstra (6/12)



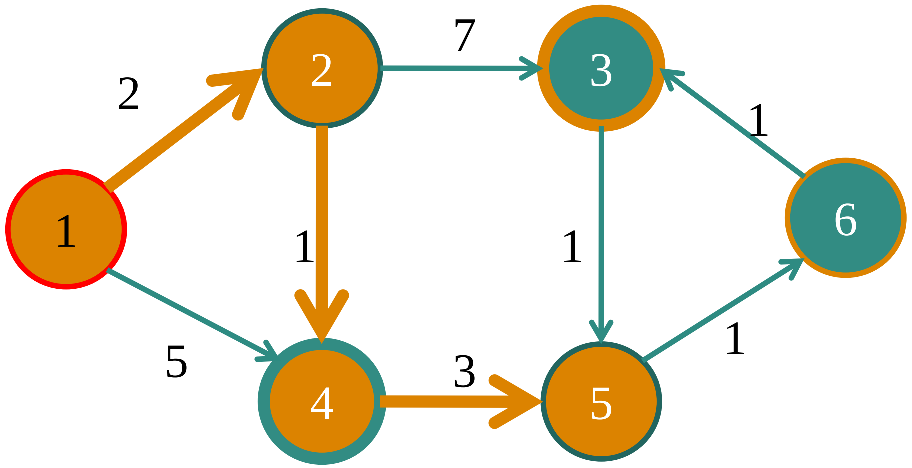
	1	2	3	4	5	6
0	0,1	2,1	∞,1	5,1	∞,1	∞,1
1			9,2	3,2	---	---
2			9,2		6,4	---

Thuật toán Dijkstra (7/12)



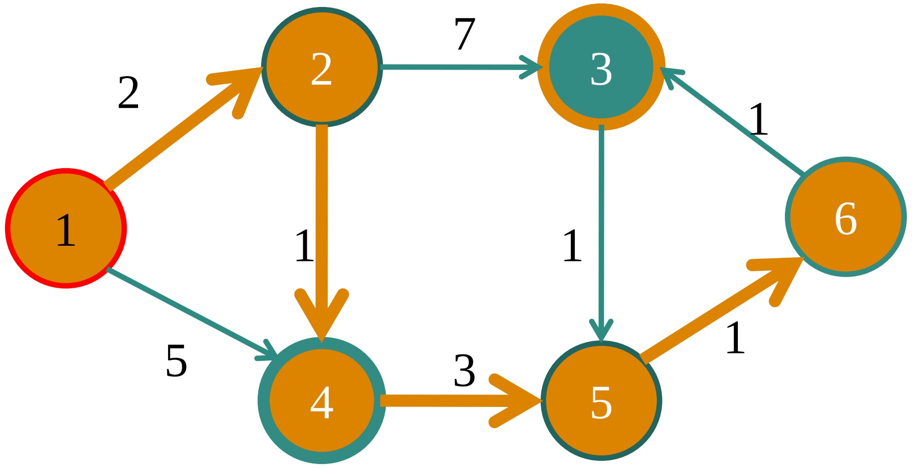
	1	2	3	4	5	6
0	0,1	2,1	∞,1	5,1	∞,1	∞,1
1			9,2	3,2	---	---
2			9,2		6,4	---

Thuật toán Dijkstra (8/12)



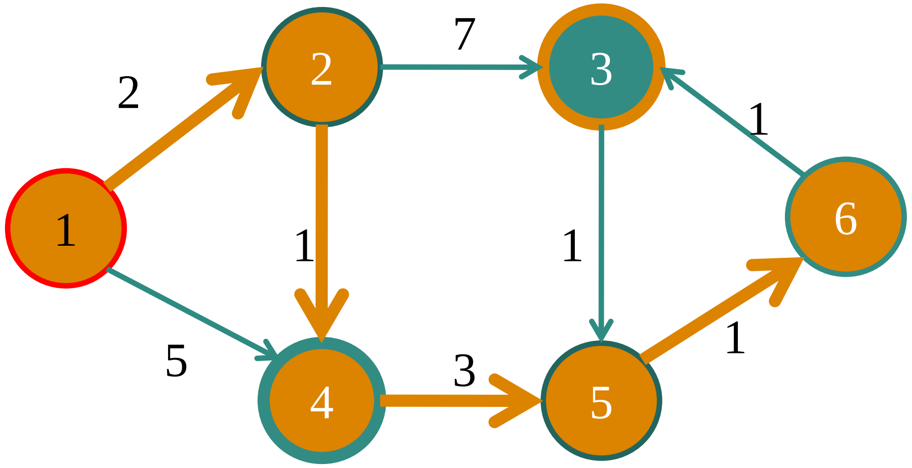
	1	2	3	4	5	6
0	0,1	2,1	∞,1	5,1	∞,1	∞,1
1			9,2	3,2	---	---
2			9,2		6,4	---
3			9,2			7,5

Thuật toán Dijkstra (9/12)



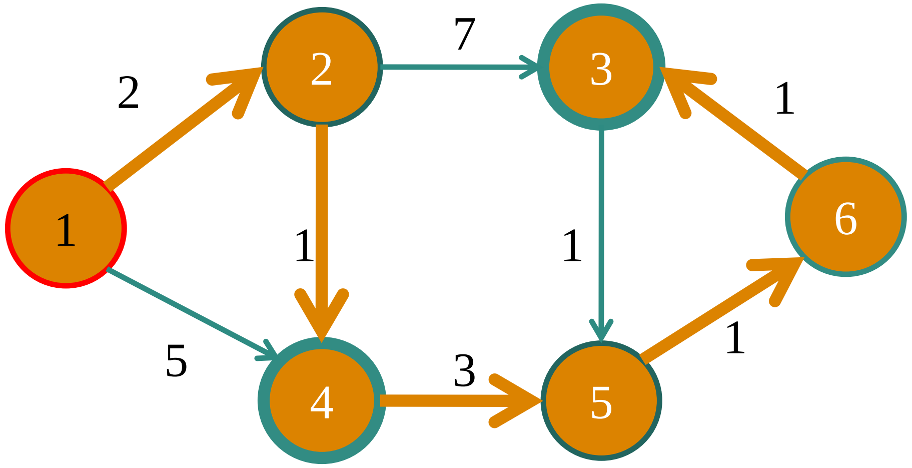
	1	2	3	4	5	6
0	0,1	2,1	∞,1	5,1	∞,1	∞,1
1			9,2	3,2	---	---
2			9,2		6,4	---
3			9,2			7,5

Thuật toán Dijkstra (10/12)



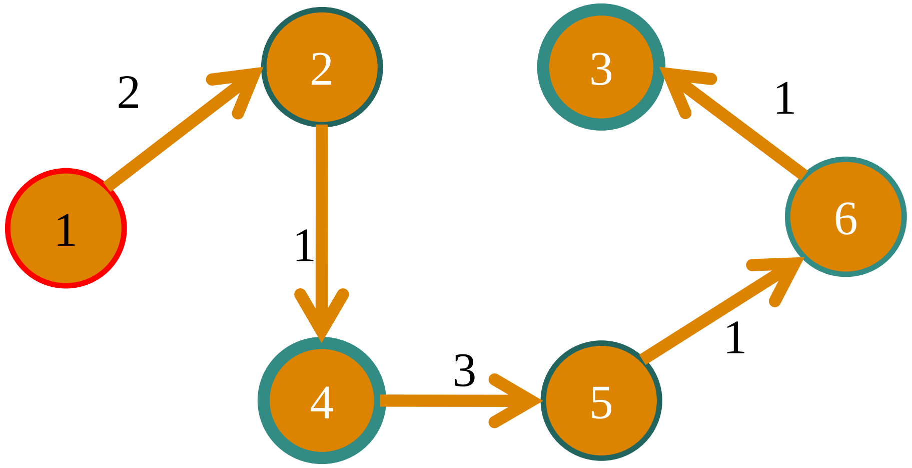
	1	2	3	4	5	6
0	0,1	2,1	∞,1	5,1	∞,1	∞,1
1			9,2	3,2	---	---
2			9,2		6,4	---
3			9,2			7,5
			8,6			

Thuật toán Dijkstra (11/12)



	1	2	3	4	5	6
0	0,1	2,1	∞,1	5,1	∞,1	∞,1
1			9,2	3,2	---	---
2			9,2		6,4	---
3			9,2			7,5
4			8,6			

Thuật toán Dijkstra (12/12)



	1	2	3	4	5	6
0	0,1	2,1	∞,1	5,1	∞,1	∞,1
1			9,2	3,2	---	---
2			9,2		6,4	---
3			9,2			7,5
4			8,6			
Kết luận	0,1	2,1	8,6	3,2	6,4	7,5

Thuật toán Bellman – Ford (1/11)

Đầu vào:

- Đồ thị có hướng, có trọng số G(V,E,C,s).
- Trong đó s là đỉnh xuất phát, C là trọng số.

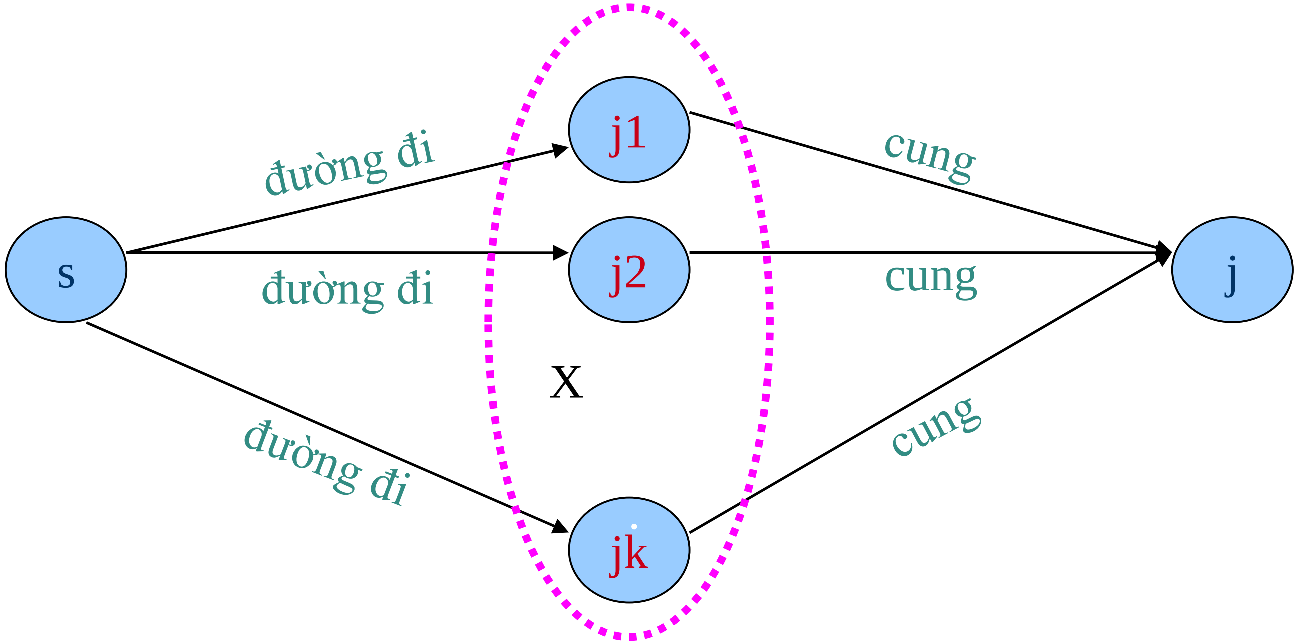
Đầu ra:

- Đường đi ngắn nhất từ s đến mọi đỉnh còn lại.
- Hoặc đồ thị chứa chu trình âm.

Thuật toán Bellman – Ford (2/11)

- **Khởi tạo**
  - Distance(s) = 0
  - Với mọi v thuộc V \ {s}: distance(v)=∞, previous(v)=s
- **Với k = 1 đến |V| - 1 làm:**
  - Với mỗi cung (u,v) thuộc E:
    - Nếu distance(v) < distance(u) + c(u,v) thì:
      - distance(v) = distance(u) + c(u,v)
      - Cập nhật previous(v) = u
- **Với mỗi cạnh (u,v) thuộc E:**
  - Nếu distance(v) > distance(u) + c(u,v) thì đồ thị chứa chu trình âm.

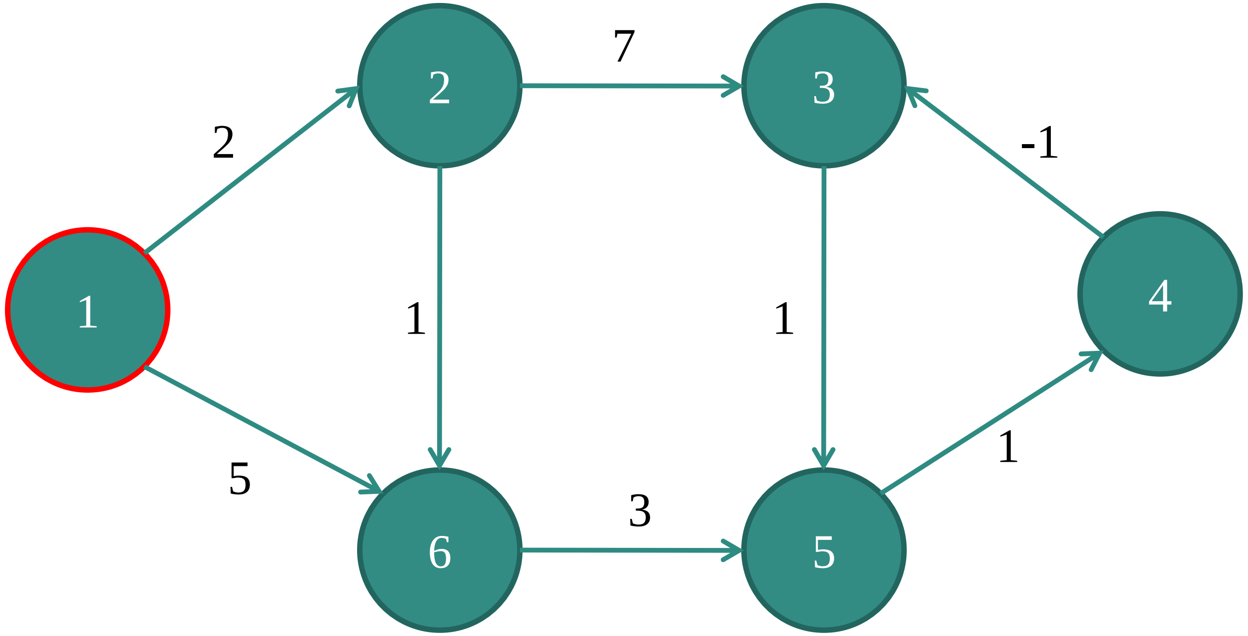
Thuật toán Bellman – Ford (3/11)



Nguyên lý Bellman

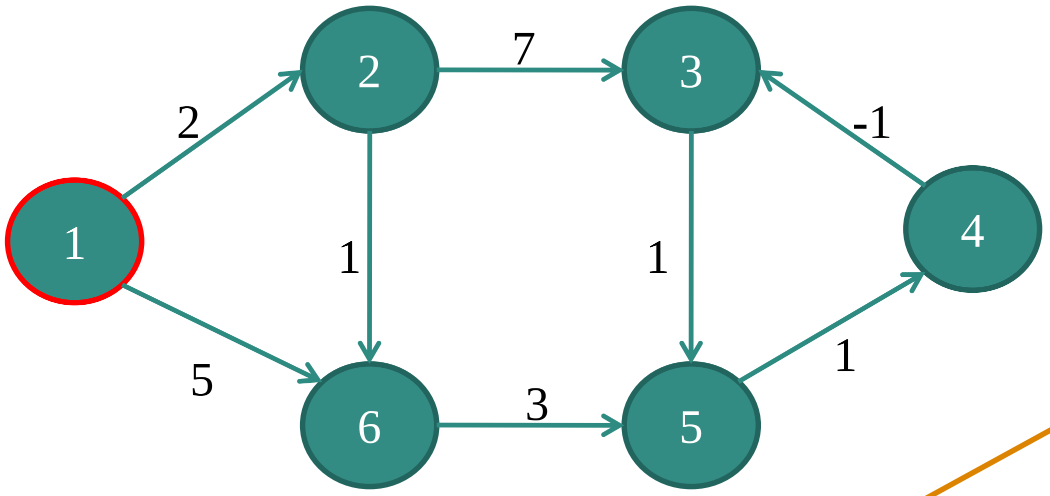
$$\text{distance}(\text{j}) = \underset{j_k \in X}{Min}\left\{\text{distance}(j_k) + c(j_k, j)\right\}$$

Thuật toán Bellman – Ford (4/11)



Đỉnh =>	1	2	3	4	5	6
0	0,1	∞,1	∞,1,	∞,1	∞,1	∞,1

Thuật toán Bellman – Ford (5/11)



Xét j = 2:
X = {1}
Distance(j)=Min{∞,0+2}=2
Previous(j)=1

Đỉnh =>	1	2	3	4	5	6
0	0,1	∞,1	∞,1,	∞,1	∞,1	∞,1
1		2,1				

Xét j = 3:
X = {2}
Distance(3)=Min{∞, ∞}= ∞
=> Không cập nhật distance(3) và Previous(3)

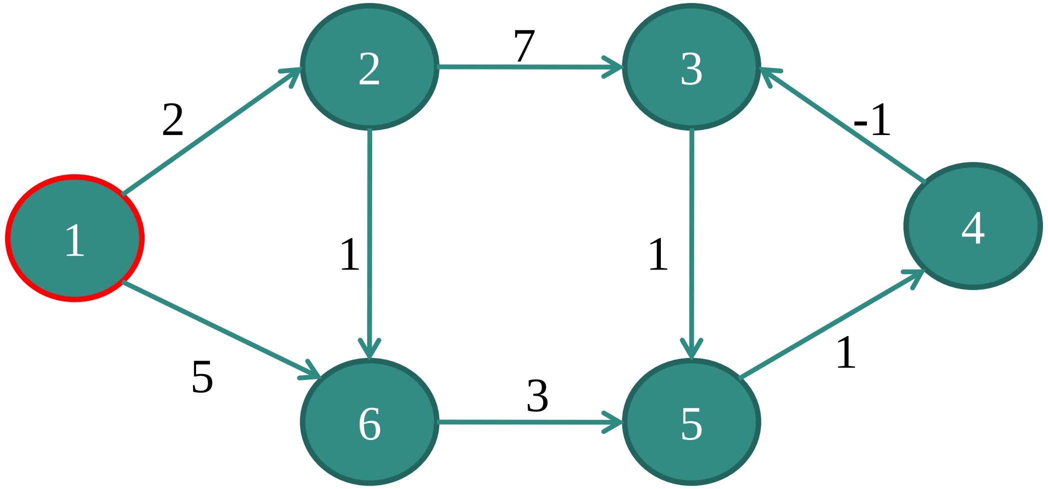
Thuật toán Bellman – Ford (6/11)

❖ Giải thích:

- Sở dĩ chúng ta vẫn chọn distance(3) = Min{∞, ∞} trong đó ∞ là gí trị "cũ" của distance(2) [*trong khi giá trị "mới" của distance(2) là 2*] vì trong trường hợp tổng quát đỉnh *j* có thể được chọn tùy ý – tức là đỉnh 3 có thể được xét trước đỉnh 2!

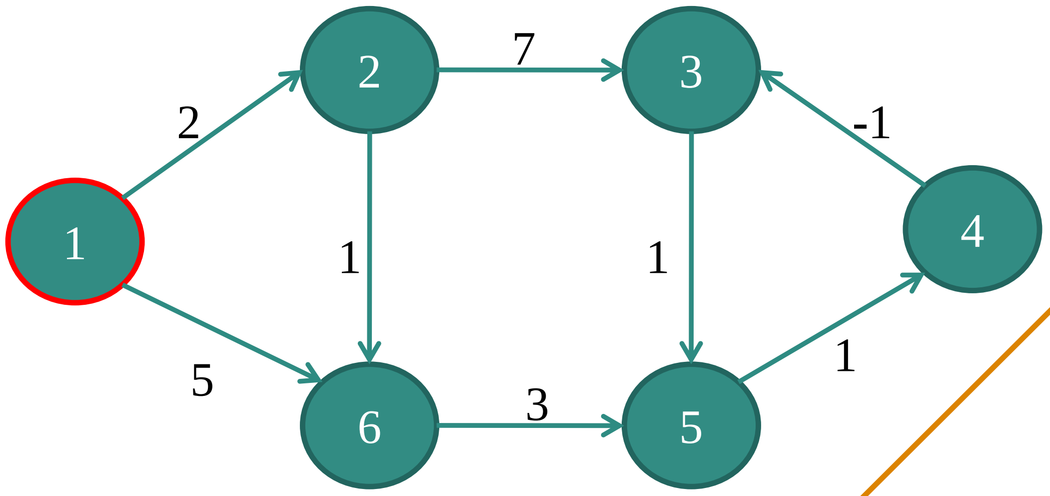
Thuật toán Bellman – Ford (7/11)

❖ Tương tự với j = 4, 5 và 6 ta được:



Đỉnh =>	1	2	3	4	5	6
0	0,1	∞,1	∞,1,	∞,1	∞,1	∞,1
1		2,1	-----	-----	-----	5,1

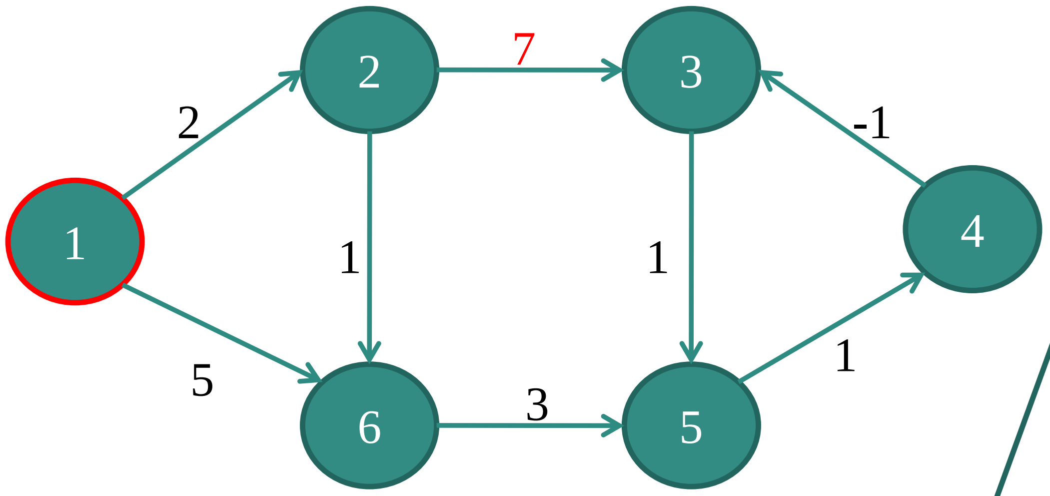
Thuật toán Bellman – Ford (8/11)



Quá trình lặp lại:
Xét j = 2:
X = {1} => Không cập nhật distance(2) và previous(2)

Đỉnh =>	1	2	3	4	5	6
0	0,1	∞,1	∞,1,	∞,1	∞,1	∞,1
1		2,1	-----	-----	-----	5,1
2		2,1				

Thuật toán Bellman – Ford (9/11)

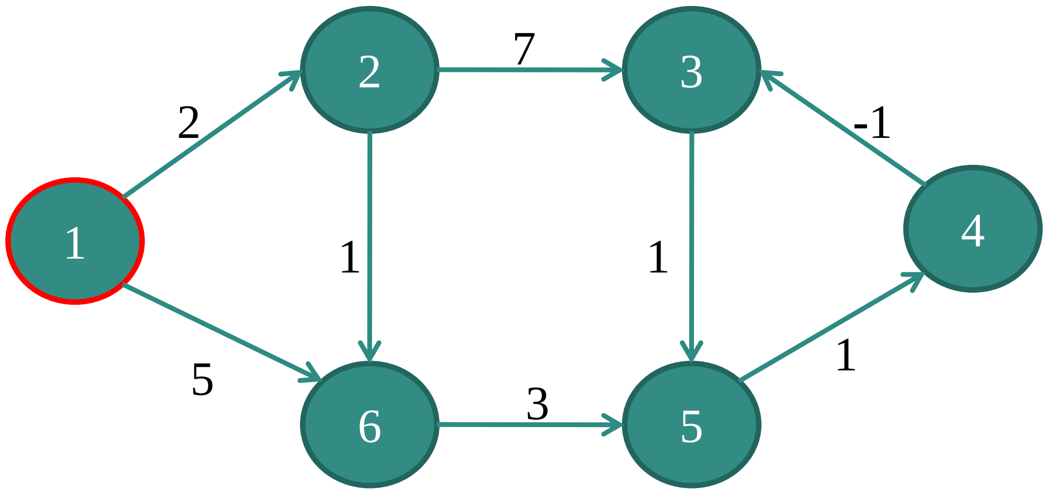


Quá trình lặp lại:
Xét j = 3:
X = {2} =>
distance(3) = min{∞,2+7}=9
Cập nhật previous(3) = 2

∞

Đỉnh =>	1	2	3	4	5	6
0	0,1	∞,1	∞,1,	∞,1	∞,1	∞,1
1		2,1	-----	-----	-----	5,1
2		2,1	9,2			

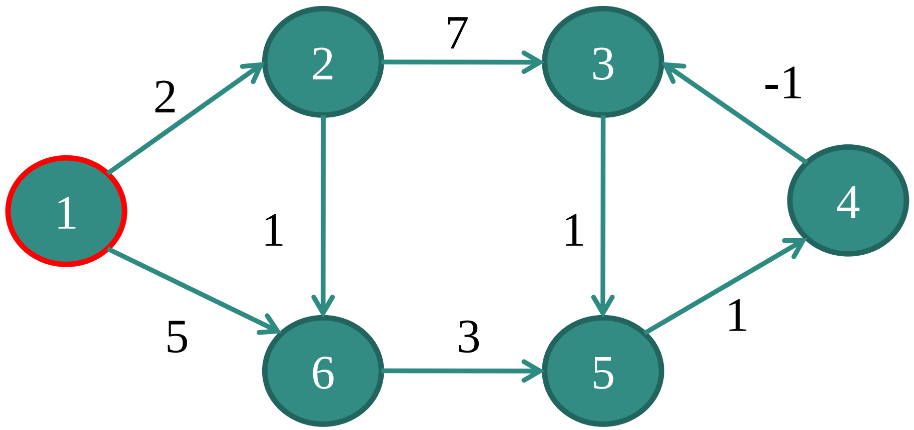
Thuật toán Bellman – Ford (10/11)



Tương tự với j = 4, 5 và 6:

Đỉnh =>	1	2	3	4	5	6
0	0,1	∞,1	∞,1,	∞,1	∞,1	∞,1
1		2,1	-----	-----	-----	5,1
2		2,1	9,2	-----	8,6	3,2

Thuật toán Bellman – Ford (11/11)



Quá trình lặp lại cho đến khi:
- Trong bảng có 2 dòng liên tiếp trùng nhau!
- Hoặc đã đủ n dòng!
(????)

Đỉnh =>	1	2	3	4	5	6
0	0,1	∞,1	∞,1,	∞,1	∞,1	∞,1
1		2,1	-----	-----	-----	5,1
2		2,1	9,2	-----	8,6	3,2
3		2,1	9,2	9,5	6,6	3,2
4		2,1	8,4	7,5	6,6	3,2
5		2,1	6,4	7,5	6,6	3,2
6		2,1	6,4	7,5	6,6	3,2

Thảo luận & bài tập (1/3)

❖ Về thuật toán Bellman-Ford:

1. Thuật toán cho kết quả là đường đi ngắn nhất ?
2. Khi nào thuật toán dừng?
3. Thuật toán có thể phát hiện chu trình âm?

Thảo luận & bài tập (2/3)

❖ **Giải thích:**

- Đường đi ngắn nhất (nếu có) giữa 2 đỉnh bất kỳ chứa nhiều nhất n -1 cung, với n là số đỉnh của đồ thị.
- Tại bước lặp thứ k trong thuật toán, mọi đường đi ngắn nhất qua không quá k cung được xác định.
- Sau |V|-1 bước lặp, tất cả các cung (u,v) đều thỏa điều kiện: distance(v) ≤ distance(u) + c(u,v). Tức là đường đi ngắn nhất từ s đến v đã được xác định.
- Nếu (ở bước kế tiếp – bước thứ |V|) tồn tại cung (u,v) nào đó có distance(v) > distance(u) + c(u,v) tức là đồ thị có chu trình âm.
- Trong thực tế, có thể dừng thuật toán ngay khi xuất hiện 2 dòng trùng nhau trong bảng biểu diễn các giá trị distance.

Thảo luận & bài tập (3/3)

❖ Các vấn đề khác:

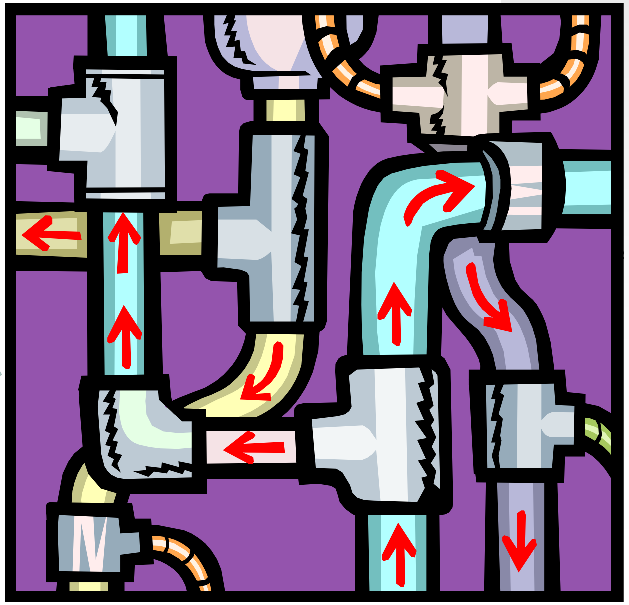
Vấn đề	Dijkstra	Bellman-Ford
Độ phức tạp tính toán?	?	?
Biểu diễn đồ thị bằng phương pháp nào sẽ phù hợp với thuật toán?		
Giải quyết bài toán trên đồ thị vô hướng?		
Tìm đường đi ngắn nhất đi qua (các) cạnh / cung cho trước?		
Cài đặt thuật toán trên máy tính		

Bài toán luồng cực đại (1/1)
(Max flow problem)

Cho trước một mạng giao thông kết nối các thành phố.

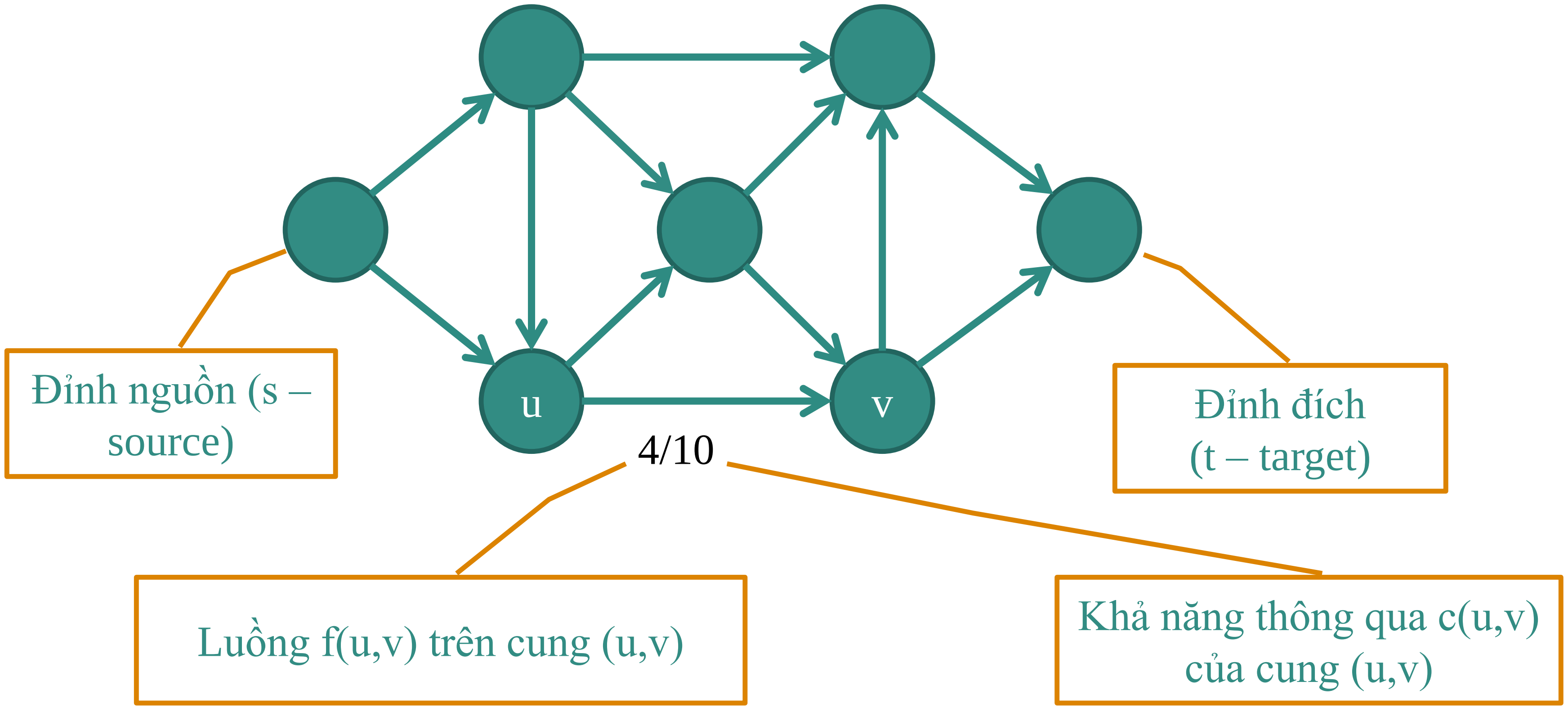
Làm thế nào để khai thác tối đa công suất vận chuyển của nó?

Trên một mạng máy tính cho trước, làm sao để truyền dữ liệu với tốc độ cao nhất giữa 2 nút mạng?



Bài toán luồng cực đại trên mạng

Mạng và luồng trên mạng (1/6)



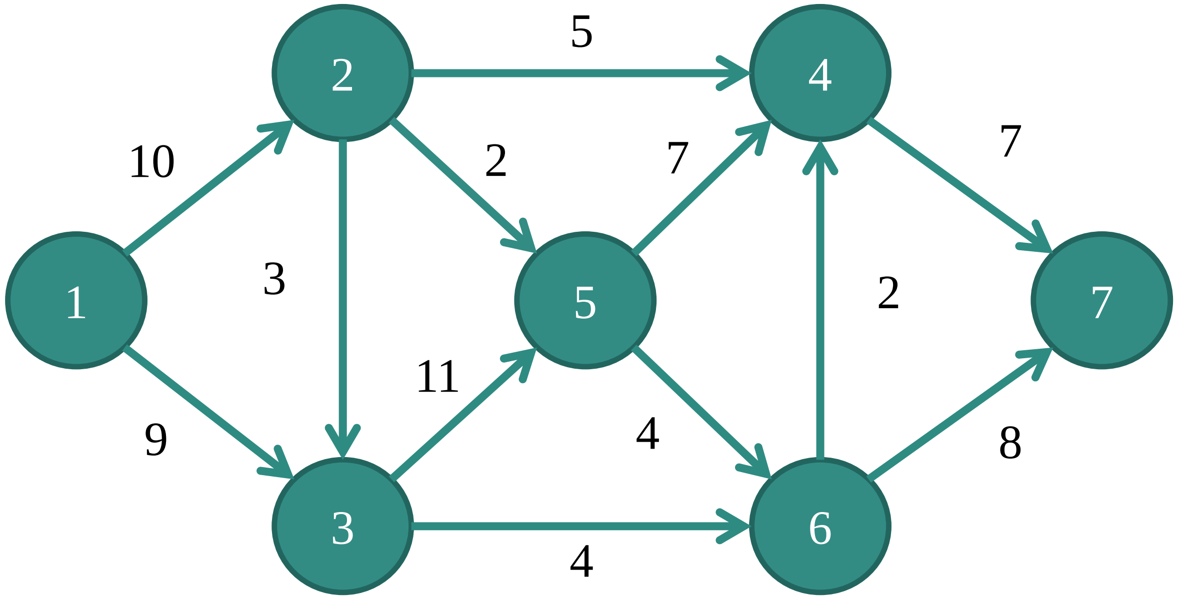
Mạng và luồng trên mạng (2/6)

- **Đỉnh nguồn:**
  - Là đỉnh chỉ có các cung đi ra
- **Đỉnh đích:**
  - Là đỉnh chỉ có các cung đi vào

Mạng và luồng trên mạng (3/6)

❖ Định nghĩa mạng:

- Là đồ thị có hướng, có trọng số
- Tồn tại đỉnh nguồn *s* và đỉnh đích *t*.



Mạng và luồng trên mạng (4/6)

❖ Định nghĩa luồng:

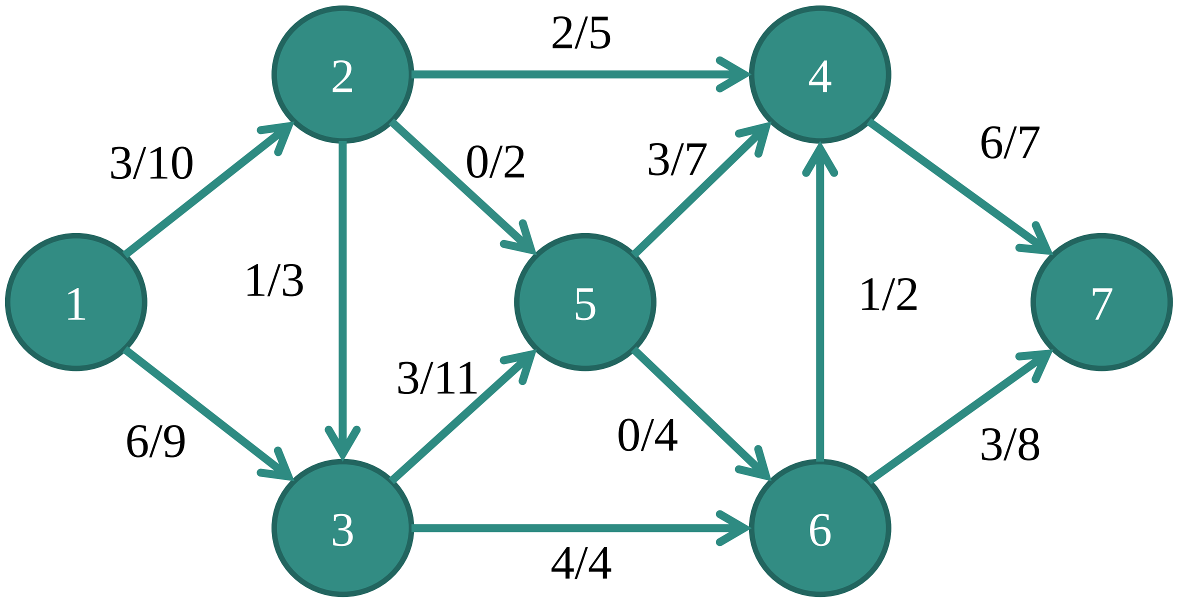
- Giả sử có mạng *G(V,E,s,t)*
- Ánh xạ *f* : V × V → R (tập số thực) thỏa điều kiện:
  - Cân bằng luồng, với mọi đỉnh v thuộc *V \ {s,t}*:

$$\sum_{x \in V} f(v,x) = \sum_{y \in V} f(y,v)$$

  - Giới hạn luồng trên cung: $0 \le f(u,v) \le c(u,v)$
- Khi đó *f* được gọi là 1 luồng trên mạng G,
- Và giá trị luồng *f* được xác định là:

$$val(f) = \sum_{x \in V} f(s,x) = \sum_{y \in V} f(y,t)$$

Mạng và luồng trên mạng (5/6)



Biểu diễn mạng G và luồng *f* với giá trị luồng là: Val(*f*) = 9

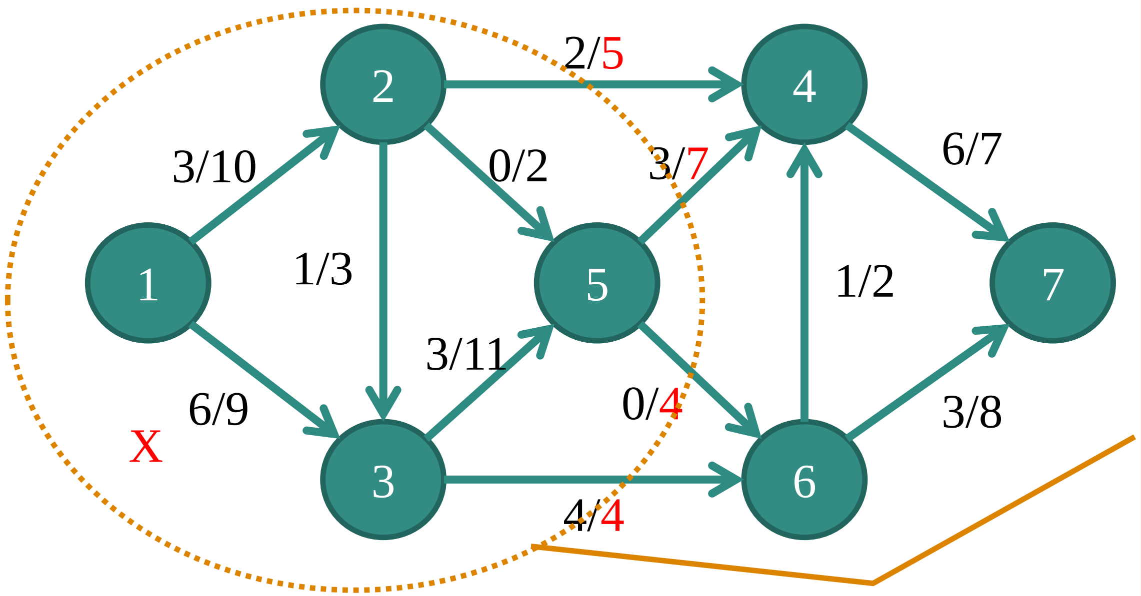
Mạng và luồng trên mạng (6/6)

❖ Bài toán luồng cực đại

- Đầu vào: Mạng $G(V,E,s,t)$
- Đầu ra: Luồng f trên G sao cho: val(f) → max

Lát cắt và đường tăng luồng (1/9)

❖ Định nghĩa lát cắt:



Lát cắt (X,X') là phân hoạch của tập V thỏa mãn:

- $s \in X$
- $X' = V \setminus X$
- $t \in X'$

C(X,X') = 5+7+4+4

❖ Khả năng thông của của lát cắt:

$$C(X, X') = \sum_{u \in X, v \in X'} f(u, v)$$

Lát cắt và đường tăng luồng (2/9)

- **Định lý max-flow min-cut:**
  - Trên một mạng bất kỳ, giá trị của luồng cực đại bằng khả năng thông qua của lát cắt cực tiểu.
  - Định lý được Ford và Fulkerson chứng minh
    - Năm 1954 với đồ thị vô hướng,
    - Năm 1955 với đồ thị có hướng.

Lát cắt và đường tăng luồng (3/9)

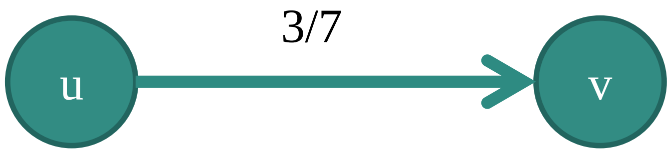
❖ Đồ thị tăng luồng:

- Giả sử có mạng G(V,E,C,s,t) với luồng *f*.
- Đồ thị tăng luồng G'(V,E',W) được xây dựng trên G và *f* với 3 trường hợp sau:

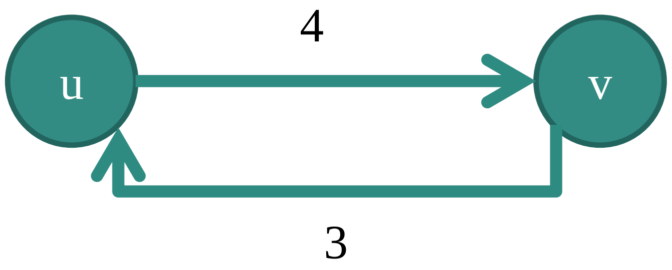
Lát cắt và đường tăng luồng (4/9)

❖ **Đồ thị tăng luồng:**

- Trường hợp 1: $f(u,v) < c(u,v)$



Cung (u,v) trên mạng G(V,E)



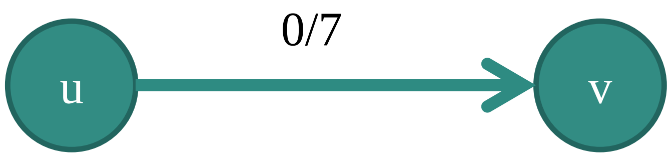
Hình thành 2 cung trên đồ thị tăng luồng G'(V,E',W):

- (u,v) với $w(u,v) = c(u,v)-f(u,v)$
- (v,u) với $w(v,u) = f(u,v)$

Lát cắt và đường tăng luồng (5/9)

❖ Đồ thị tăng luồng:

- Trường hợp 2: $f(u,v) = 0$

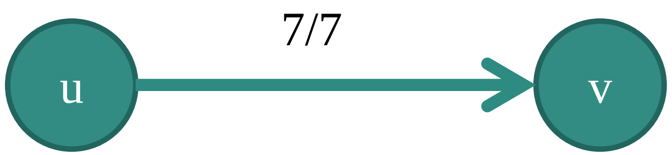


Cung (u,v) trên mạng G(V,E)

Hình thành 1 cung trên đồ thị tăng luồng G’(V,E’,W): w(u,v) = c(u,v)

Lát cắt và đường tăng luồng (6/9)

- **Đồ thị tăng luồng:**
  - Trường hợp 3: $f(u,v) = c(u,v)$

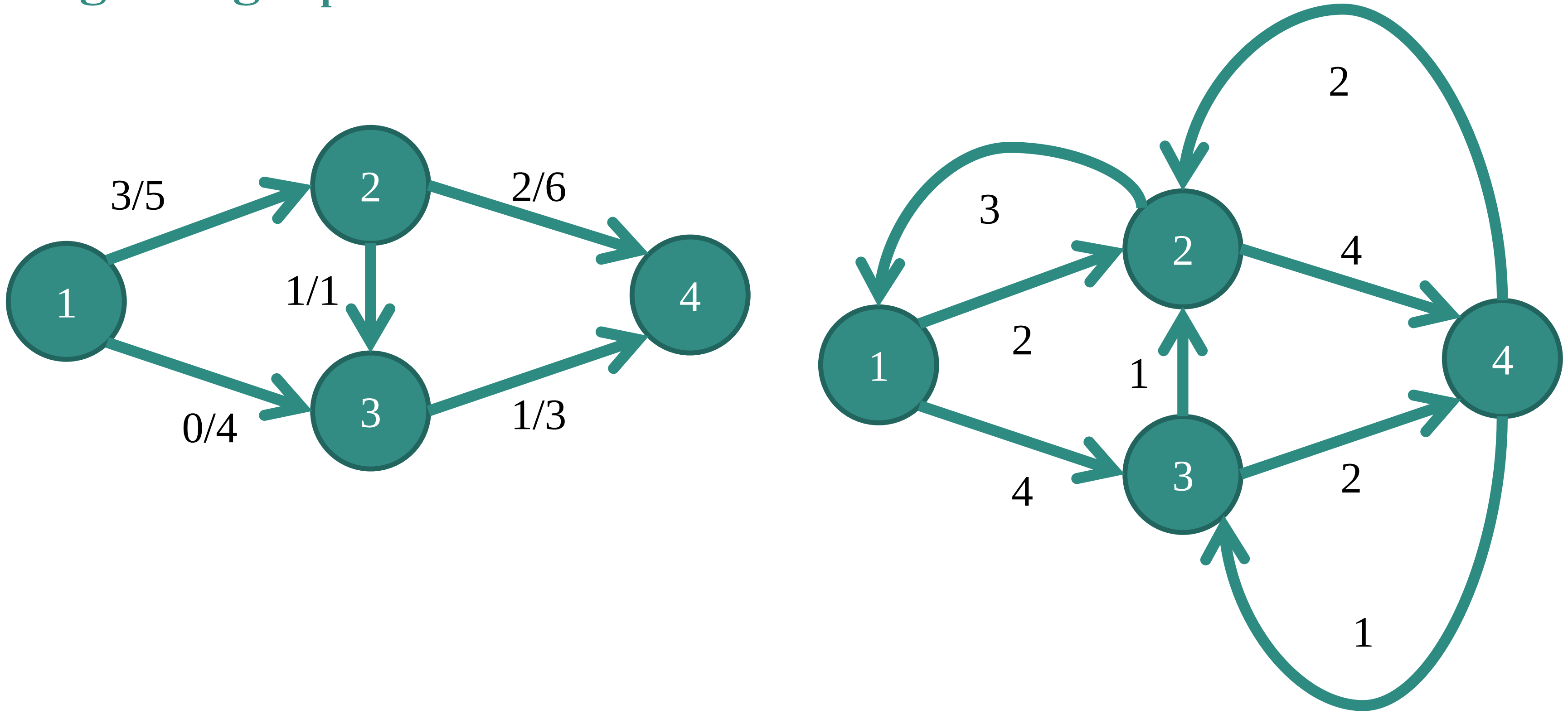


Cung (u,v) trên mạng G(V,E)

Hình thành 1 cung trên đồ thị tăng luồng G'(V,E',W):
$w(v,u) = f(u,v) = c(u,v)$

Lát cắt và đường tăng luồng (7/9)

❖ **Đồ thị tăng luồng G_f:**

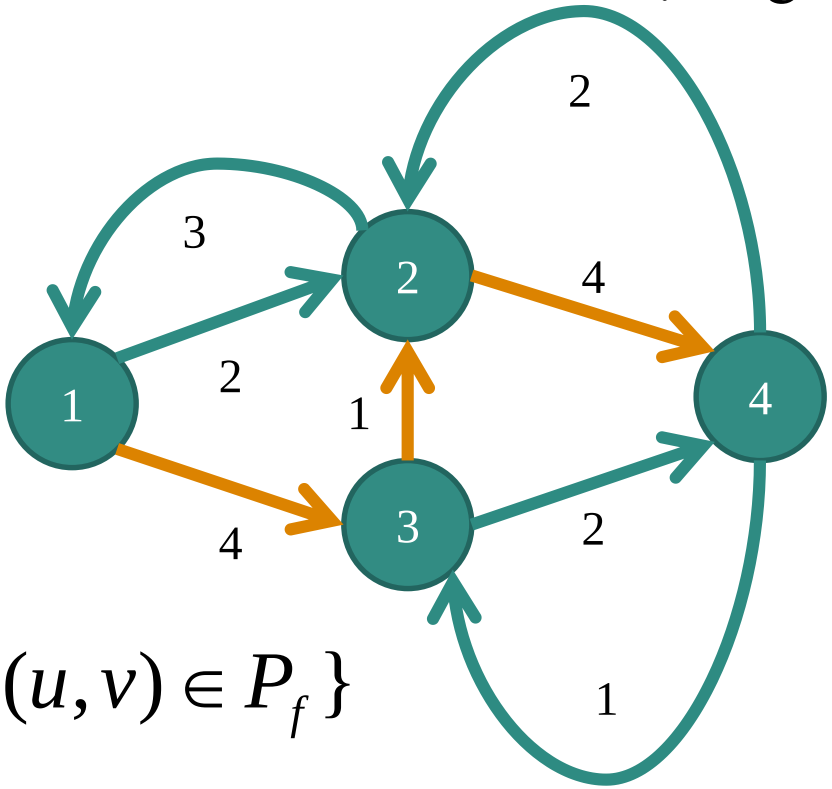


Mạng G và đồ thị tăng luồng Gf tương ứng với luồng f.

Lát cắt và đường tăng luồng (8/9)

❖ **Đường tăng luồng P_f:**

- Là đường đi từ nguồn s đến đích t trên đồ thị tăng luồng.



$$d = \min\{w(u,v) : (u,v) \in P_f\}$$

- 1, 3, 2, 4 là một đường tăng luồng với trọng số nhỏ nhất d = 1
- Đường tăng luồng là cơ sở để tìm luồng cực đại f*

Lát cắt và đường tăng luồng (9/9)

❖ Định lý:

- Giả sử có luồng f trên mạng G(V,E,C,s,t)
- Và Gf là đồ thị tăng luồng tương ứng.
- Gọi d là trọng số nhỏ nhất của đường tăng luồng Pf trên Gf.
- Đặt:
  - f'(u,v) = f(u,v) + d nếu (u,v) thuộc E.
  - f'(u,v) = f(u,v) – d nếu (u,v) *không* thuộc E.
- Khi đó:
  - f' là một luồng mới trên G
  - Và val(f') = val(f) + d

Thuật toán Ford-Fulkerson (1/7)

❖ Định lý:

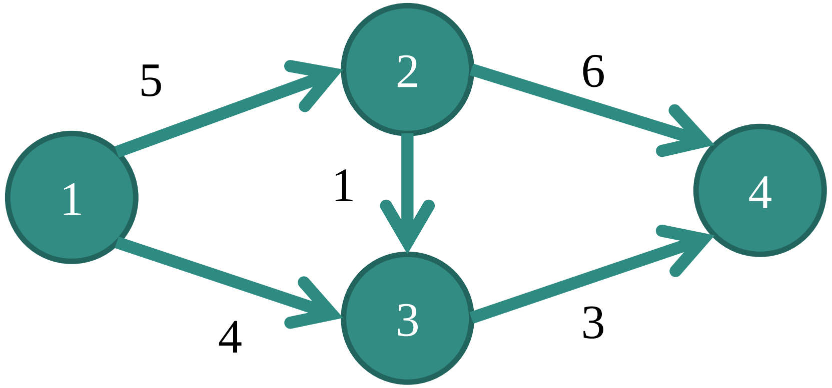
- Giả sử f là một luồng trên mạng G, khi đó các phát biểu sau là tương đương:
  - f là luồng cực đại.
  - Gf không tồn tại đường tăng luồng.
  - Val(f) = c(X,X’) với (X,X’) là lát cắt bất kỳ.

Thuật toán Ford-Fulkerson (2/7)

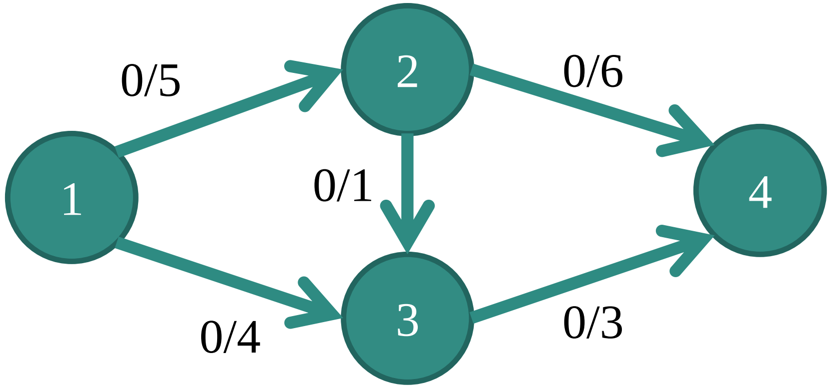
- **Ford-Fulkerson**
  - Đầu vào: mạng G(V,E,s,t)
  - Đầu ra: luồng cực đại trên G
- **begin**
  - Khởi tạo f(u,v) = 0, với mọi (u,v) thuộc E.
  - Trong khi còn tồn tại đường tăng luồng Pf trên Gf:
    - Tăng luồng f = f + d, với d là trọng số nhỏ nhất trên Pf.
  - Return f.
- **end**

Thuật toán Ford-Fulkerson (3/7)

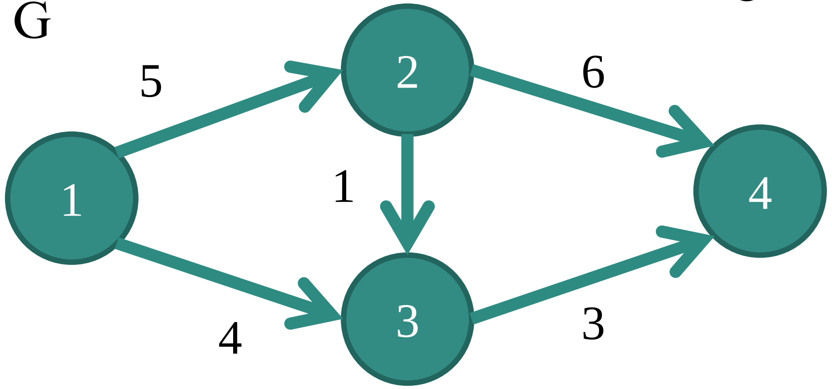
❖ Ví dụ: bước 1, khởi tạo $f = 0$



Mạng G



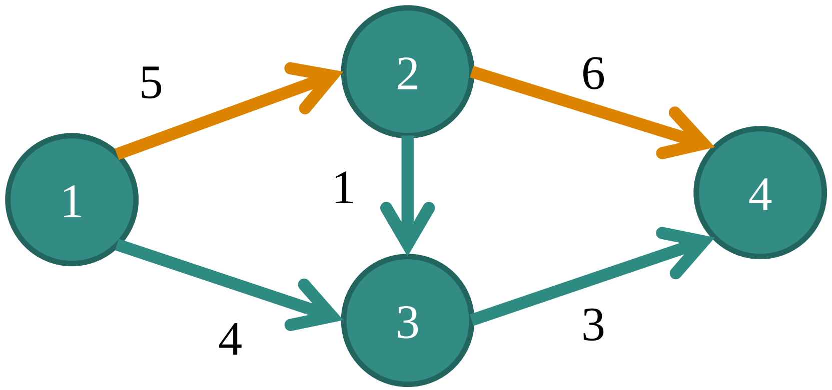
Mạng G với luồng 0



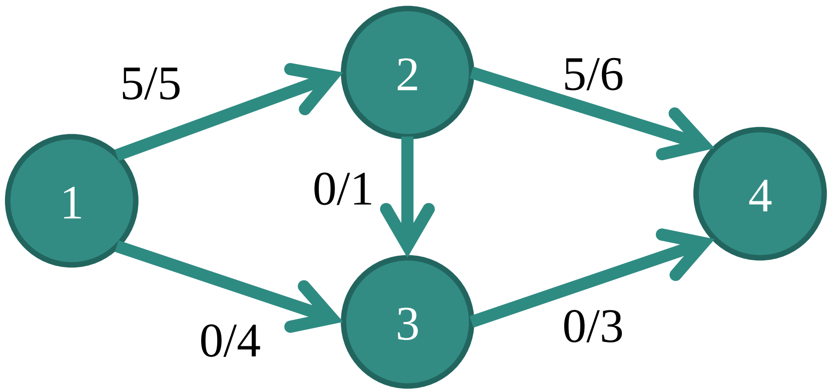
Đồ thị tăng luồng *Gf*

Thuật toán Ford-Fulkerson (4/7)

❖ Ví dụ: bước 2, lặp lại quá trình tăng luồng



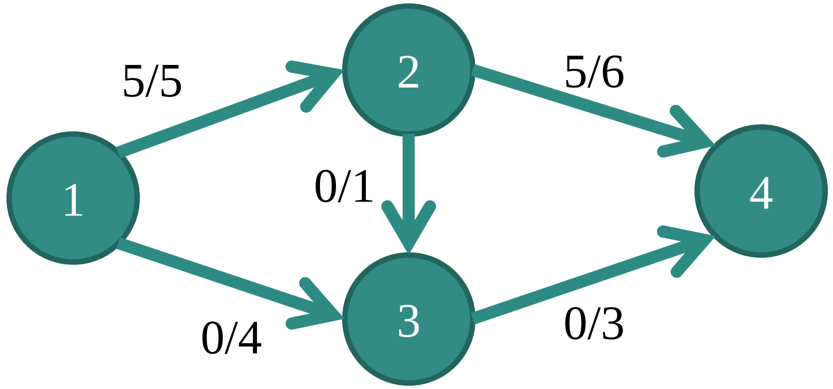
Đồ thị tăng luồng *Gf*



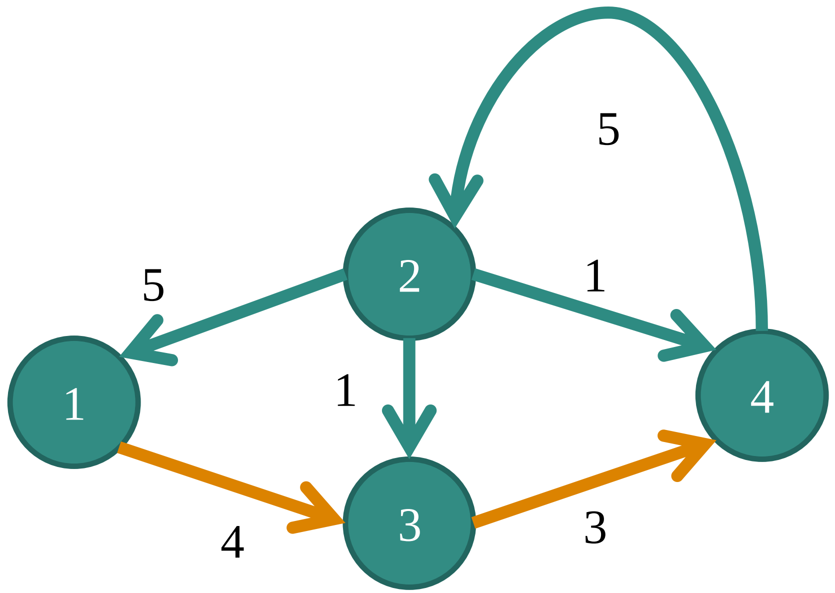
Mạng G với luồng 0

Thuật toán Ford-Fulkerson (5/7)

❖ Ví dụ: bước 2, lặp lại quá trình tăng luồng



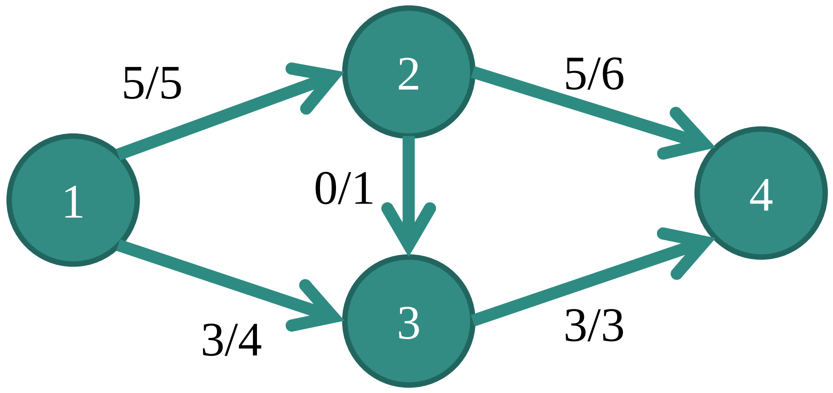
Mạng G với luồng 0



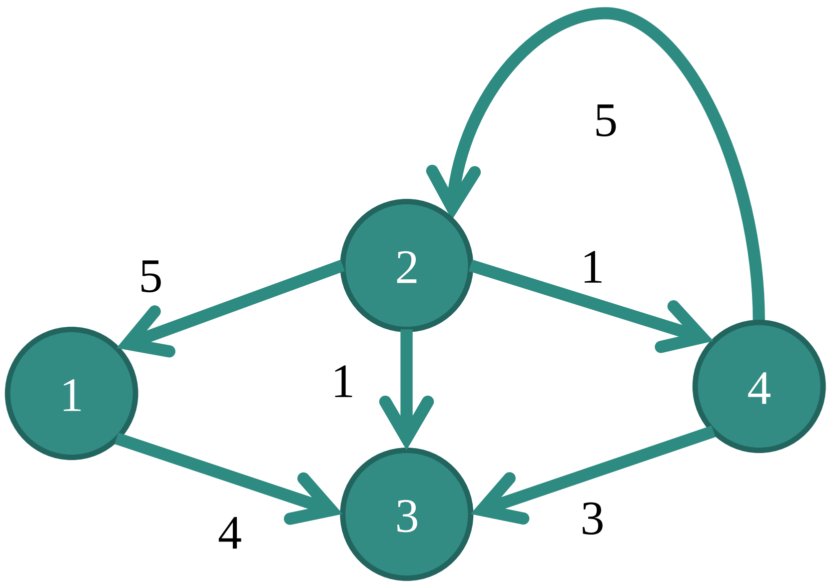
Đồ thị tăng luồng *Gf*

Thuật toán Ford-Fulkerson (6/7)

❖ Ví dụ: bước 2, lặp lại quá trình tăng luồng



Mạng G với luồng 0

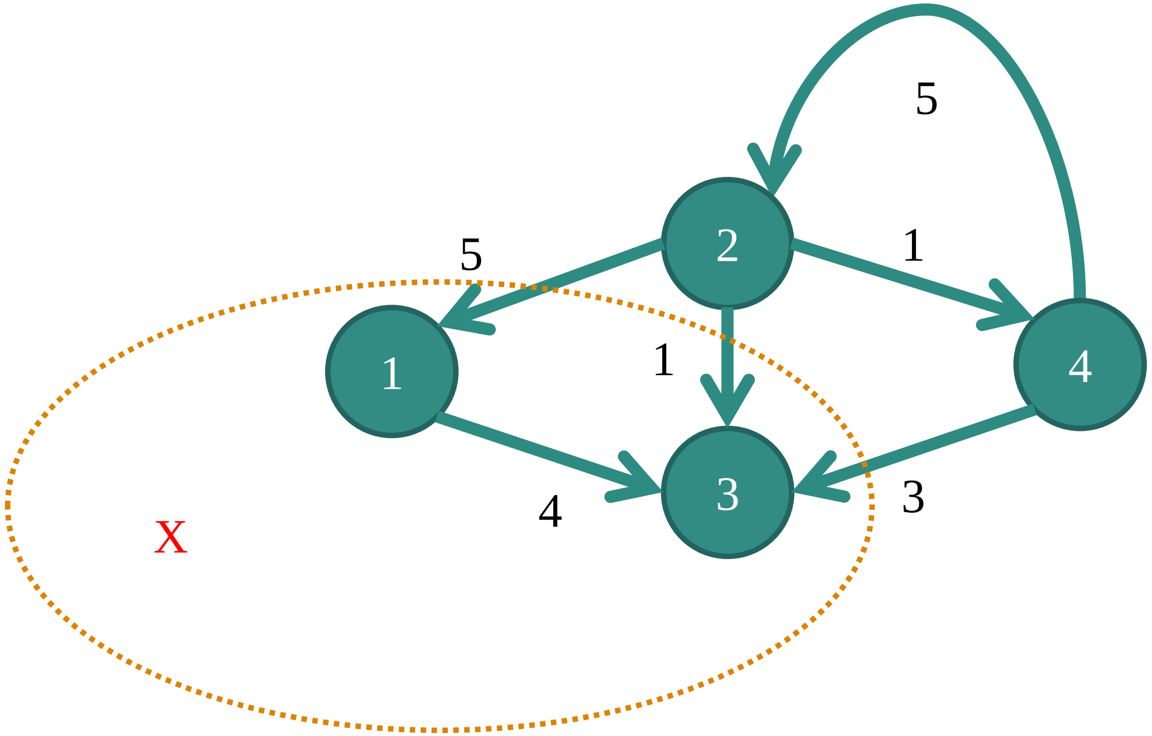


Đồ thị tăng luồng *Gf*

- Trên *Gf* không tồn tại đường tăng luồng, thuật toán kết thúc
- Giá trị luồng cực đại val(f) = 5 + 3

Thuật toán Ford-Fulkerson (7/7)

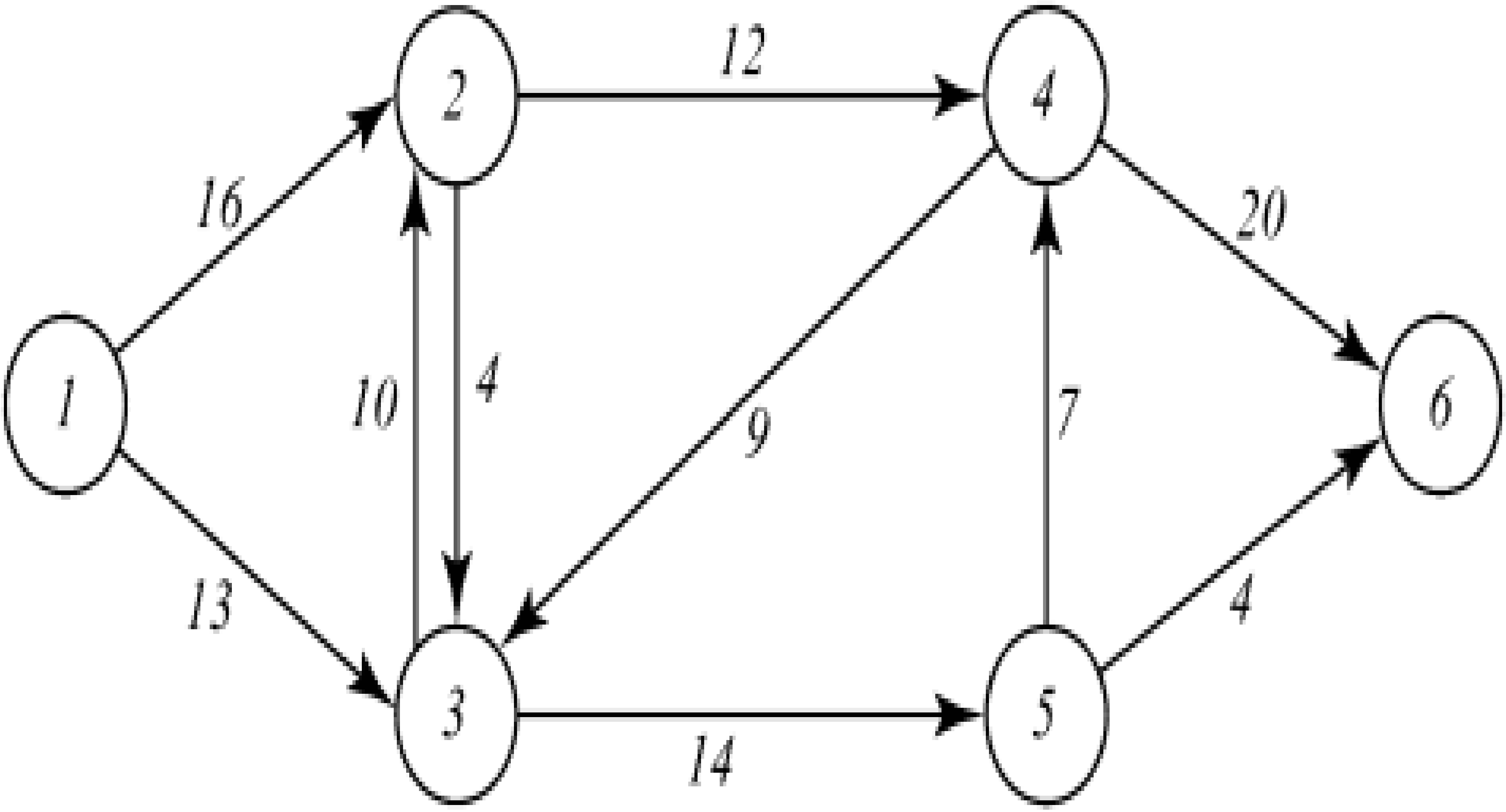
❖ **Ví dụ: bước 2, lặp lại quá trình tăng luồng**



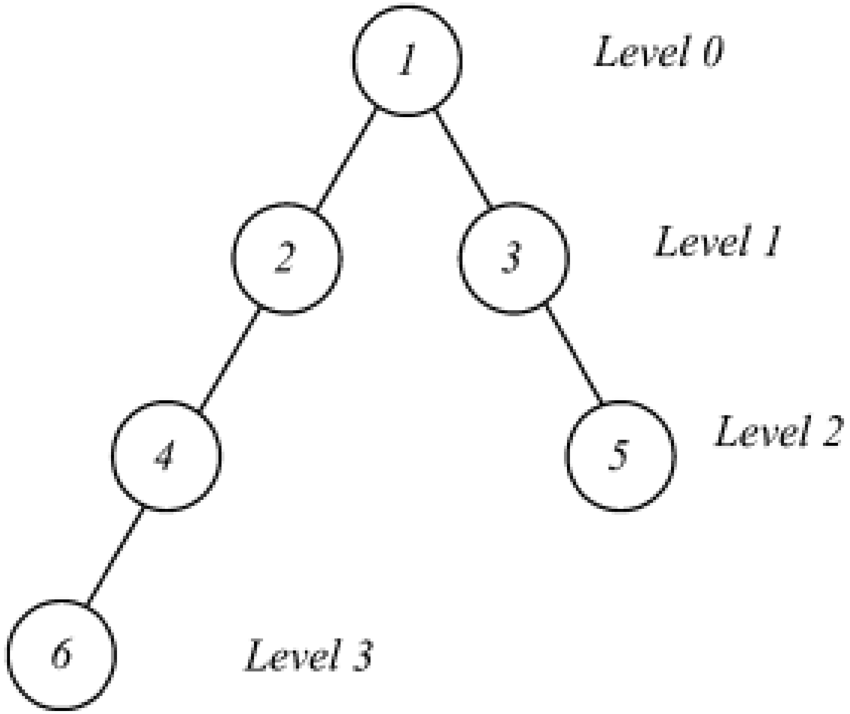
Đồ thị tăng luồng *Gf*

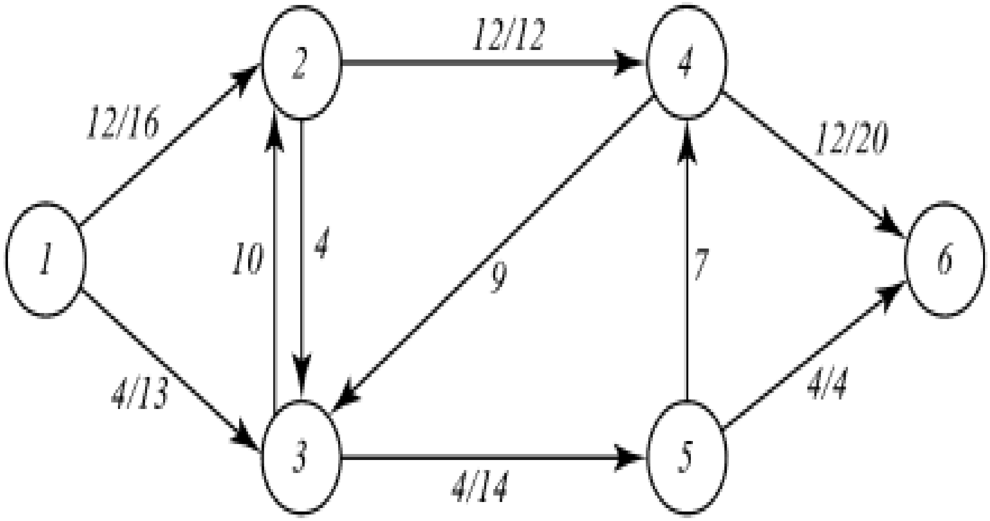
Lát cắt cực tiểu *(X,X)* với X = {1,3}, X’ = {2,4}

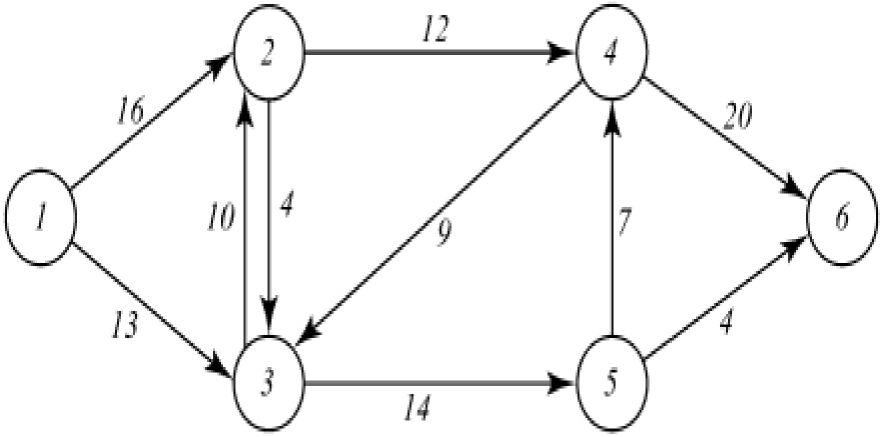
Ví dụ 1 (tự làm): Dùng thuật toán Ford-Fulkerson tìm luồng cực đại

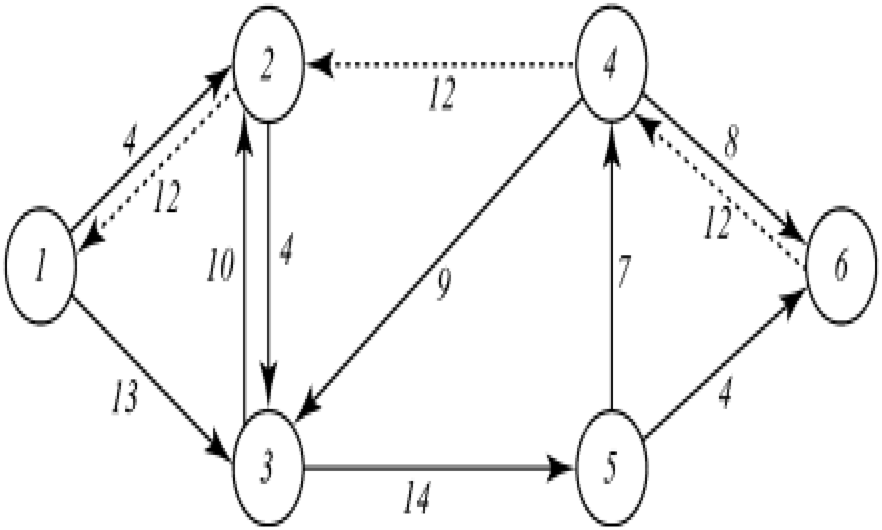


Ví dụ 1 (tự làm): Dùng thuật toán Ford-Fulkerson tìm luồng cực đại

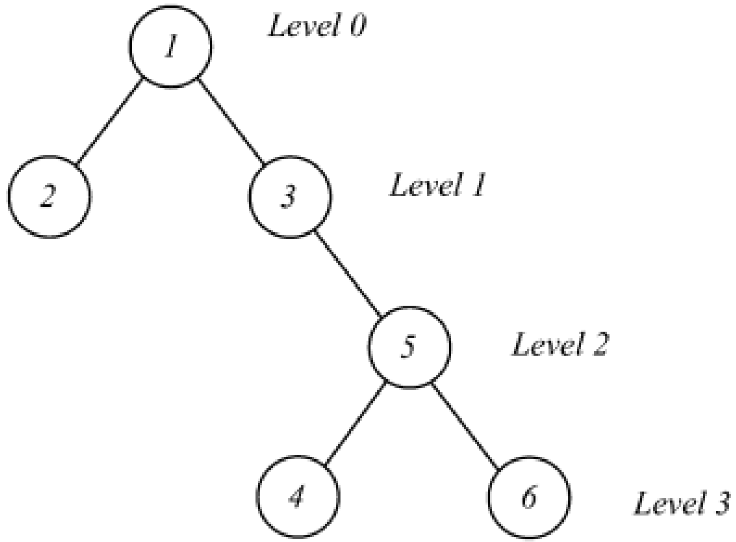


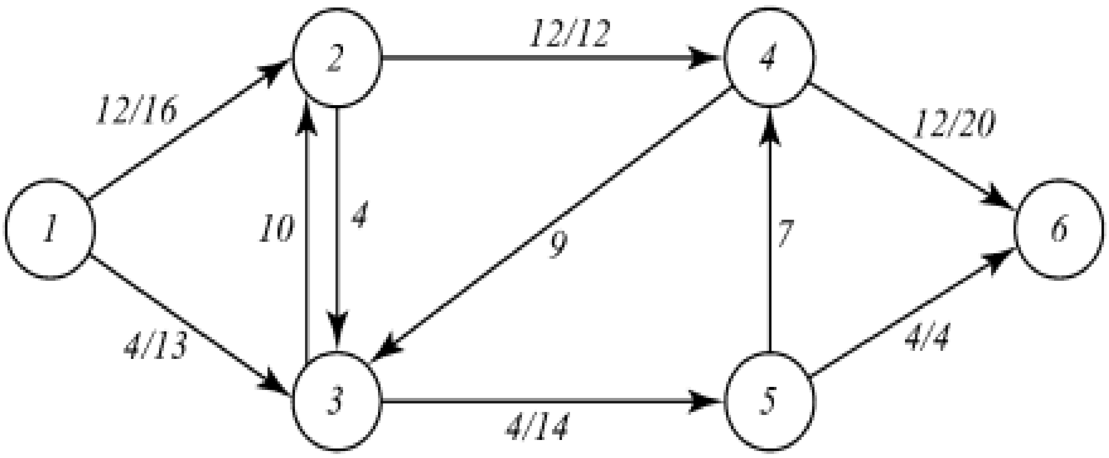


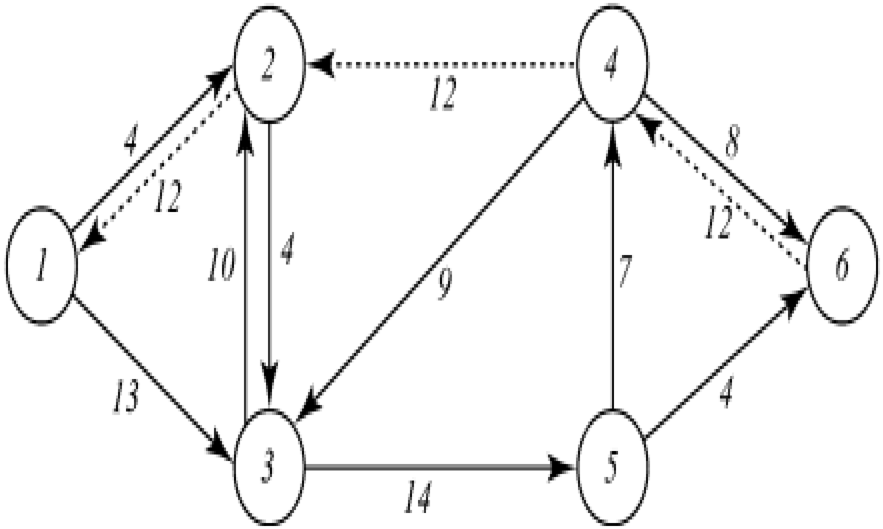


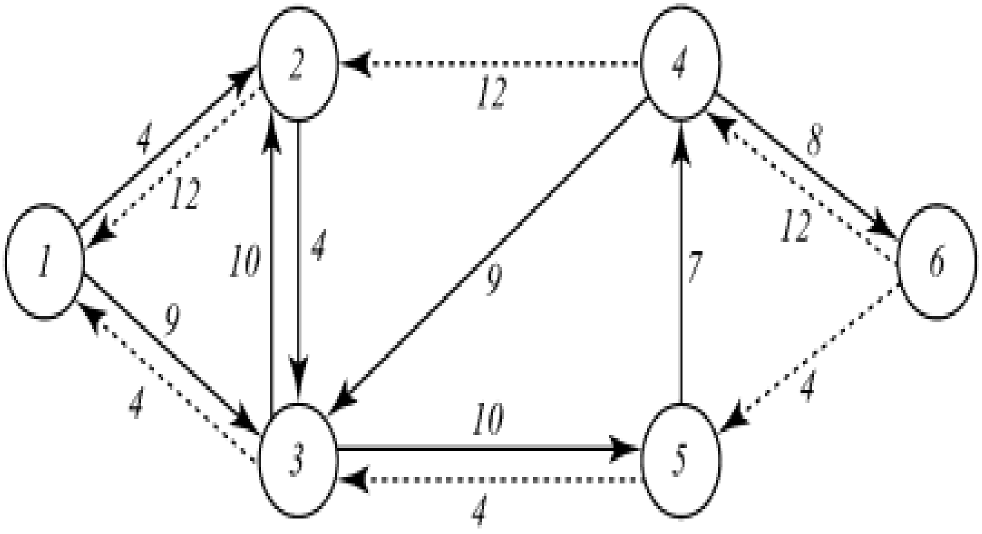


Ví dụ 1 (tự làm): Dùng thuật toán Ford-Fulkerson tìm luồng cực đại

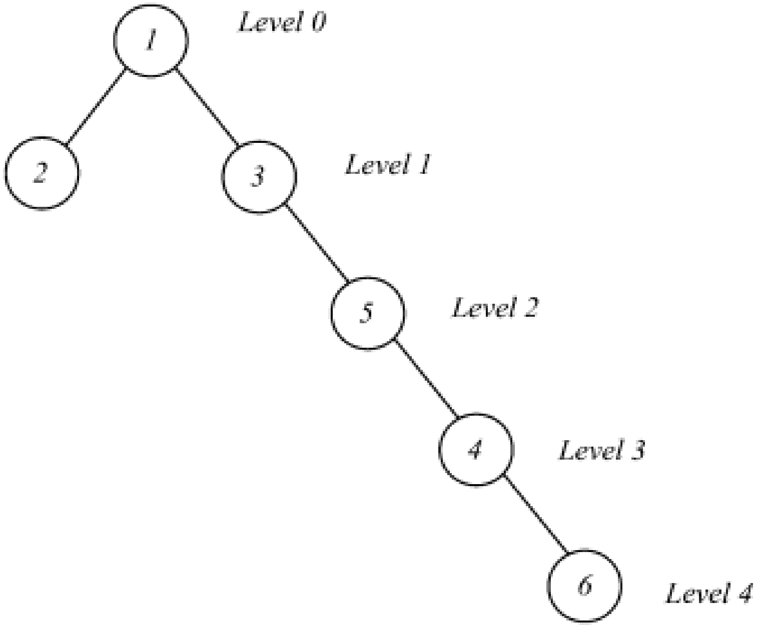


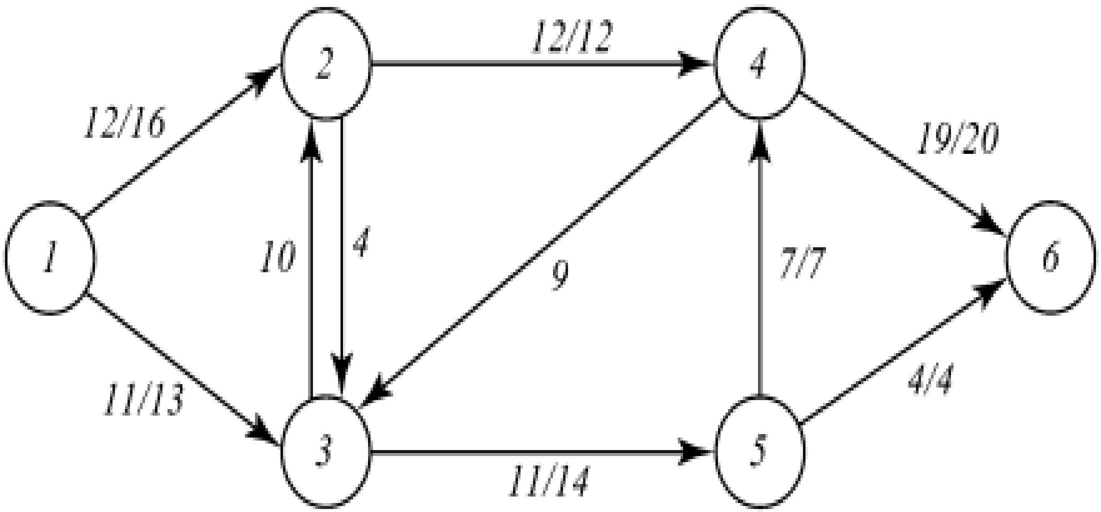


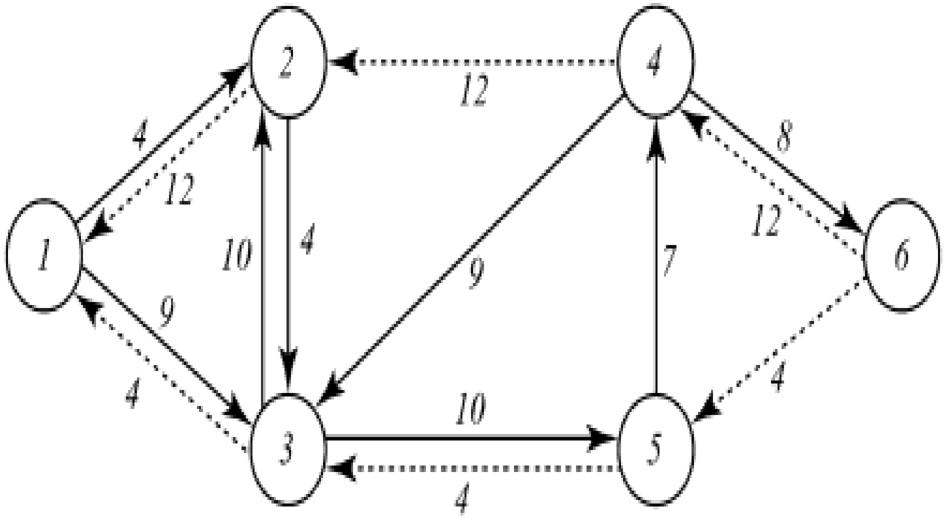


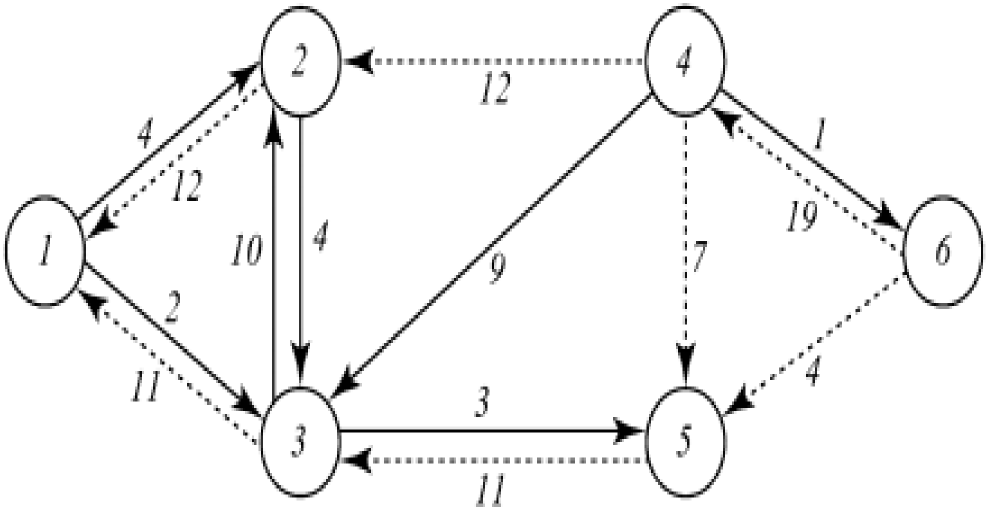


Ví dụ 1 (tự làm): Dùng thuật toán Ford-Fulkerson tìm luồng cực đại

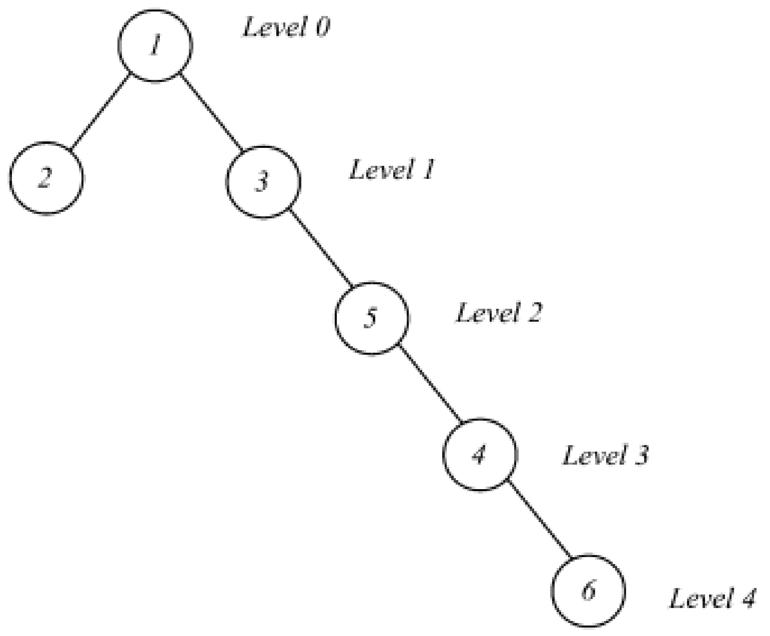


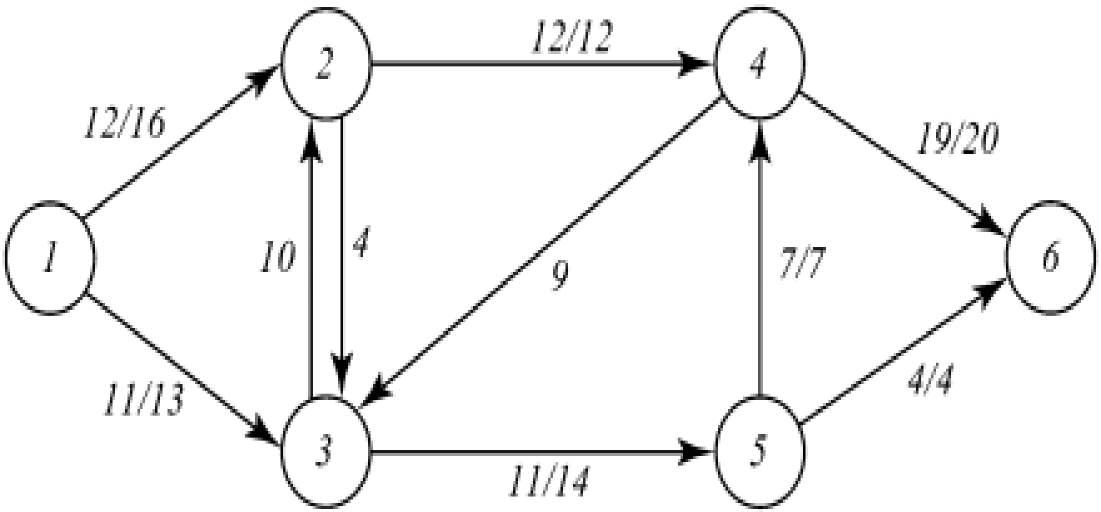


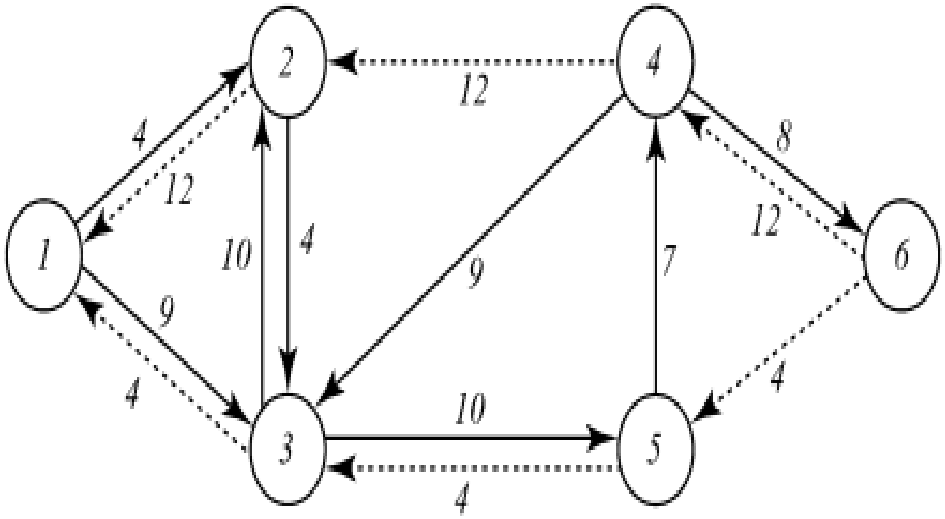


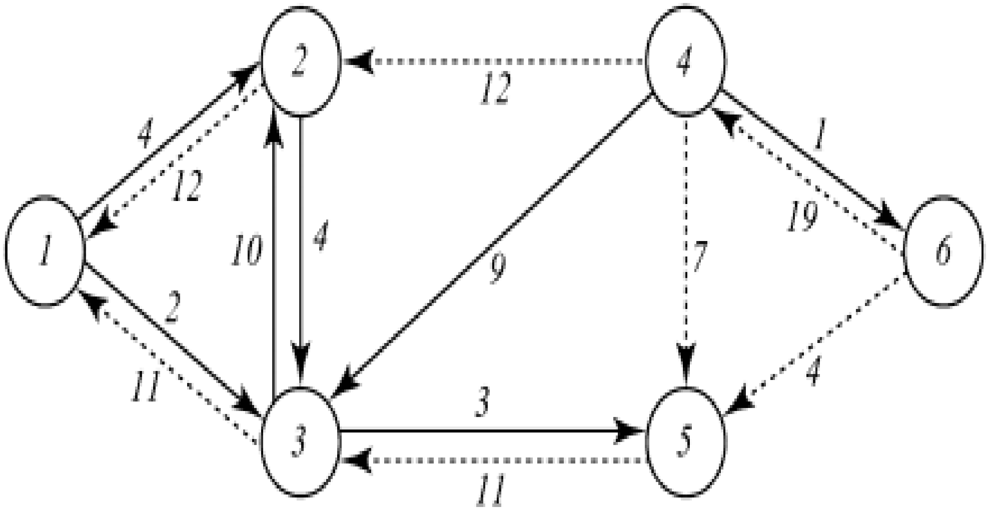


Ví dụ 1 (tự làm): Dùng thuật toán Ford-Fulkerson tìm luồng cực đại

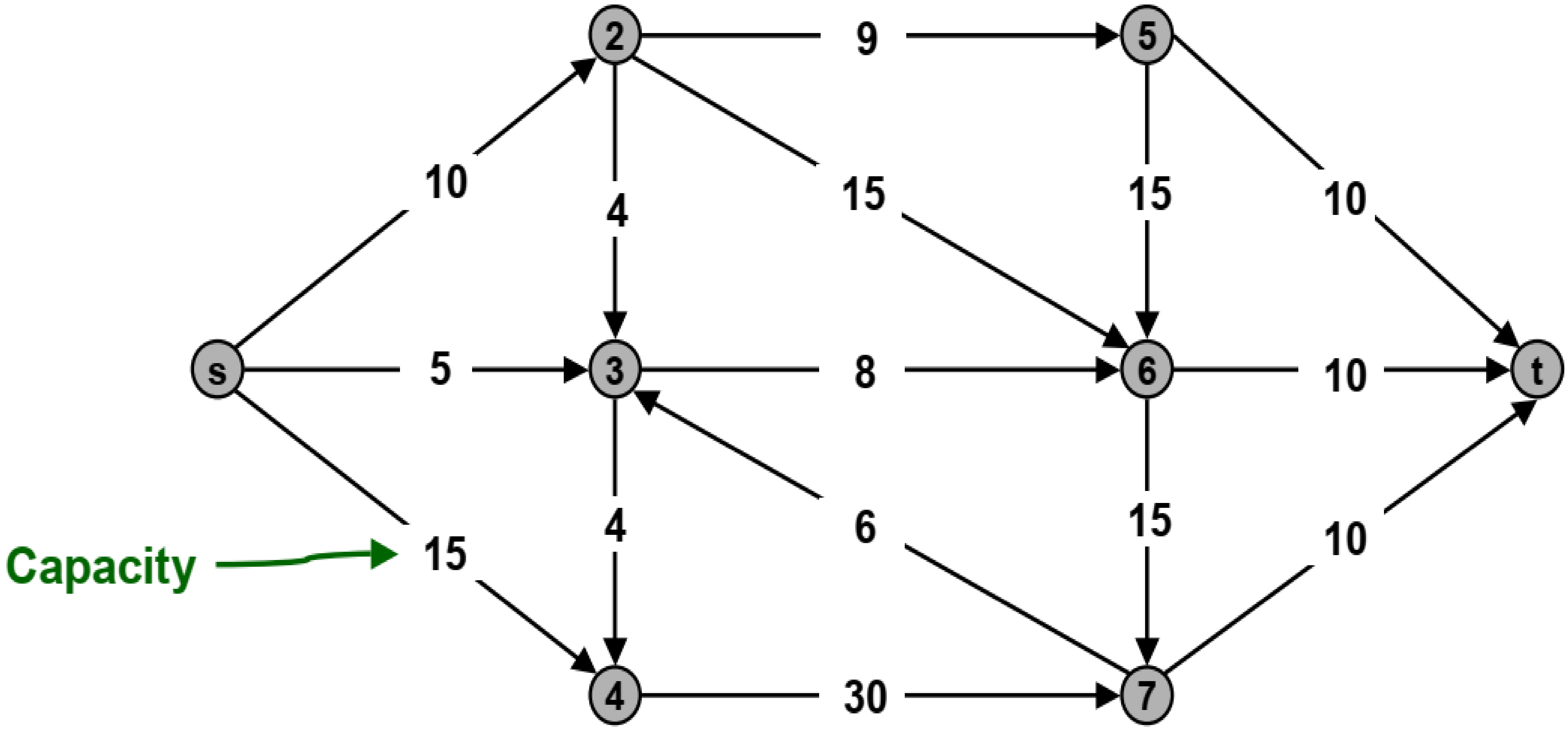




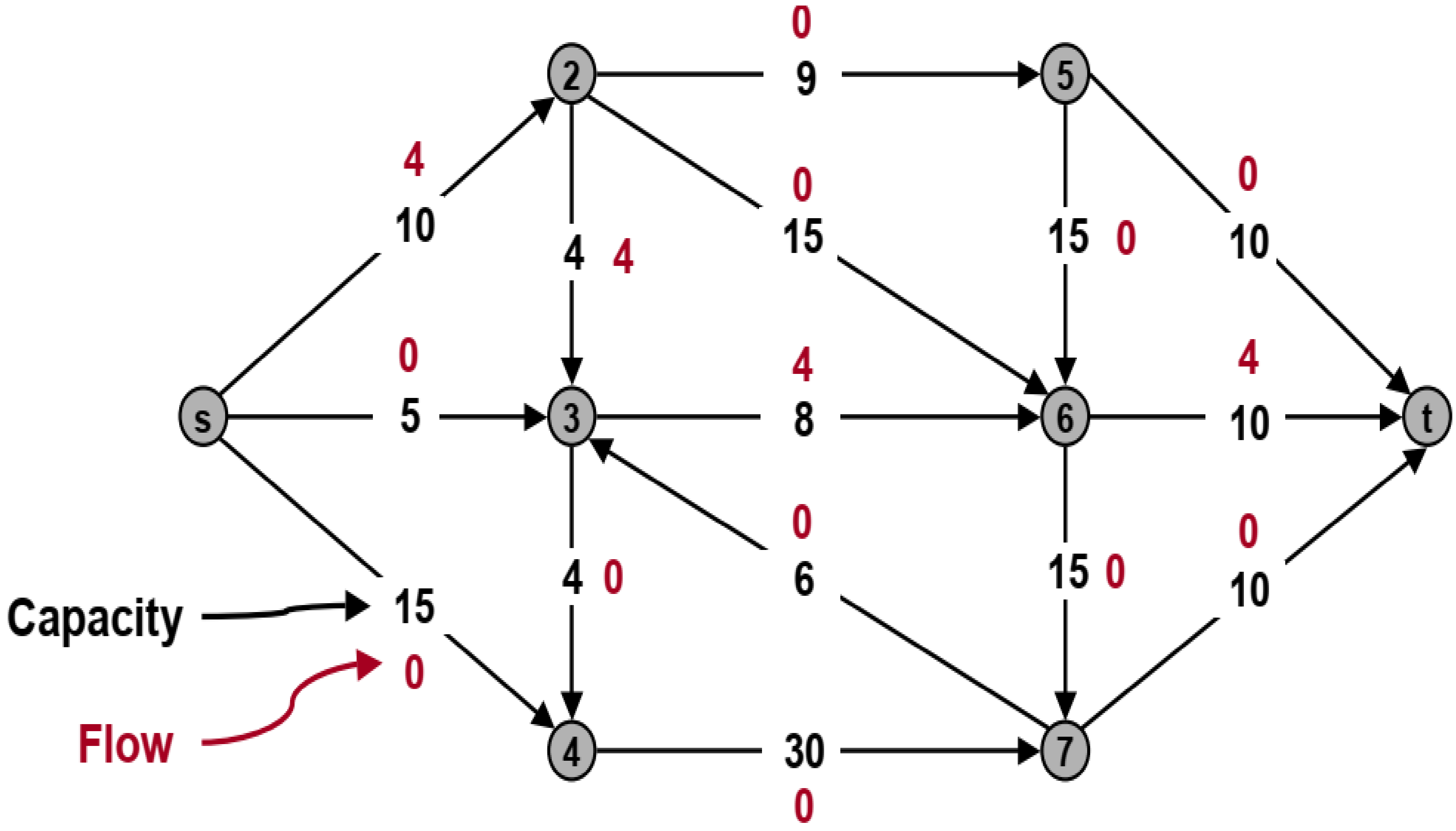




Ví dụ 2 (tự làm): Dùng thuật toán Ford-Fulkerson tìm luồng cực đại

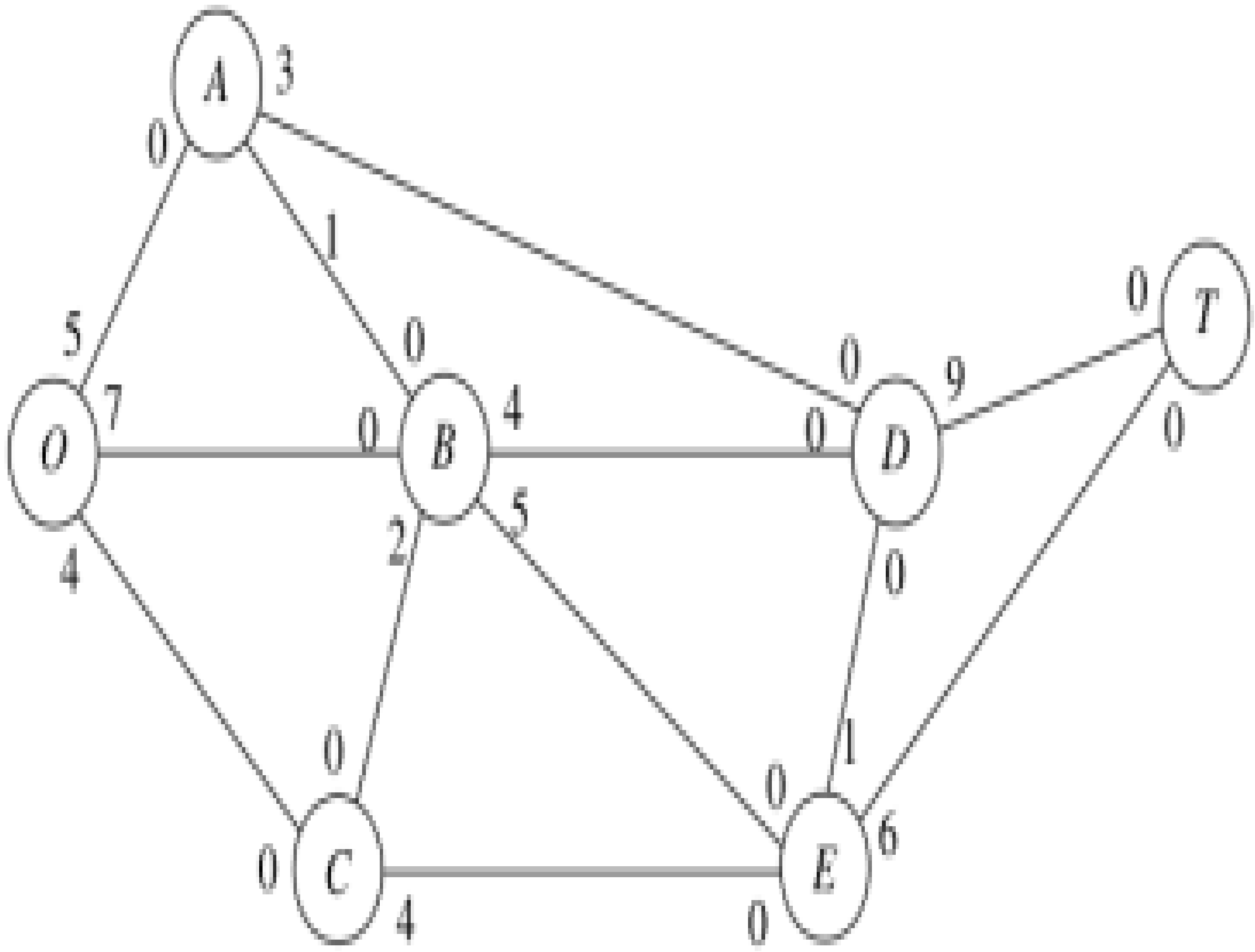


Ví dụ 2 (tự làm): Dùng thuật toán Ford-Fulkerson tìm luồng cực đại



Ví dụ 2 (tự làm): Dùng thuật toán Ford-Fulkerson tìm luồng cực đại

Step 1
Initial residual network from G



Thảo luận & Bài tập (1/1)

- Tại sao khởi tạo từ luồng $f = 0$?
- Có thể khởi tạo từ luồng tùy ý được không?
- Làm thế nào để tìm đường tăng luồng?
- Chứng minh (lại) các định lý.
- Minh họa trường hợp xấu nhất của thuật toán.
- Cài đặt thuật toán Ford-Fulkerson trên máy tính?