

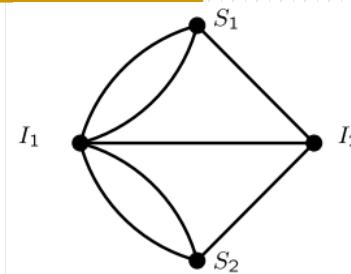
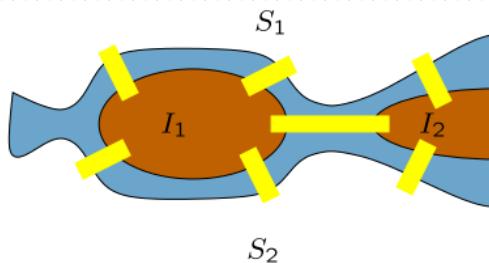
# LÝ THUYẾT ĐỒ THỊ

## Tìm đường đi ngắn nhất

Phạm Nguyên Khang -Võ Trí Thức  
BM. Khoa học máy tính, Khoa CNTT-TT

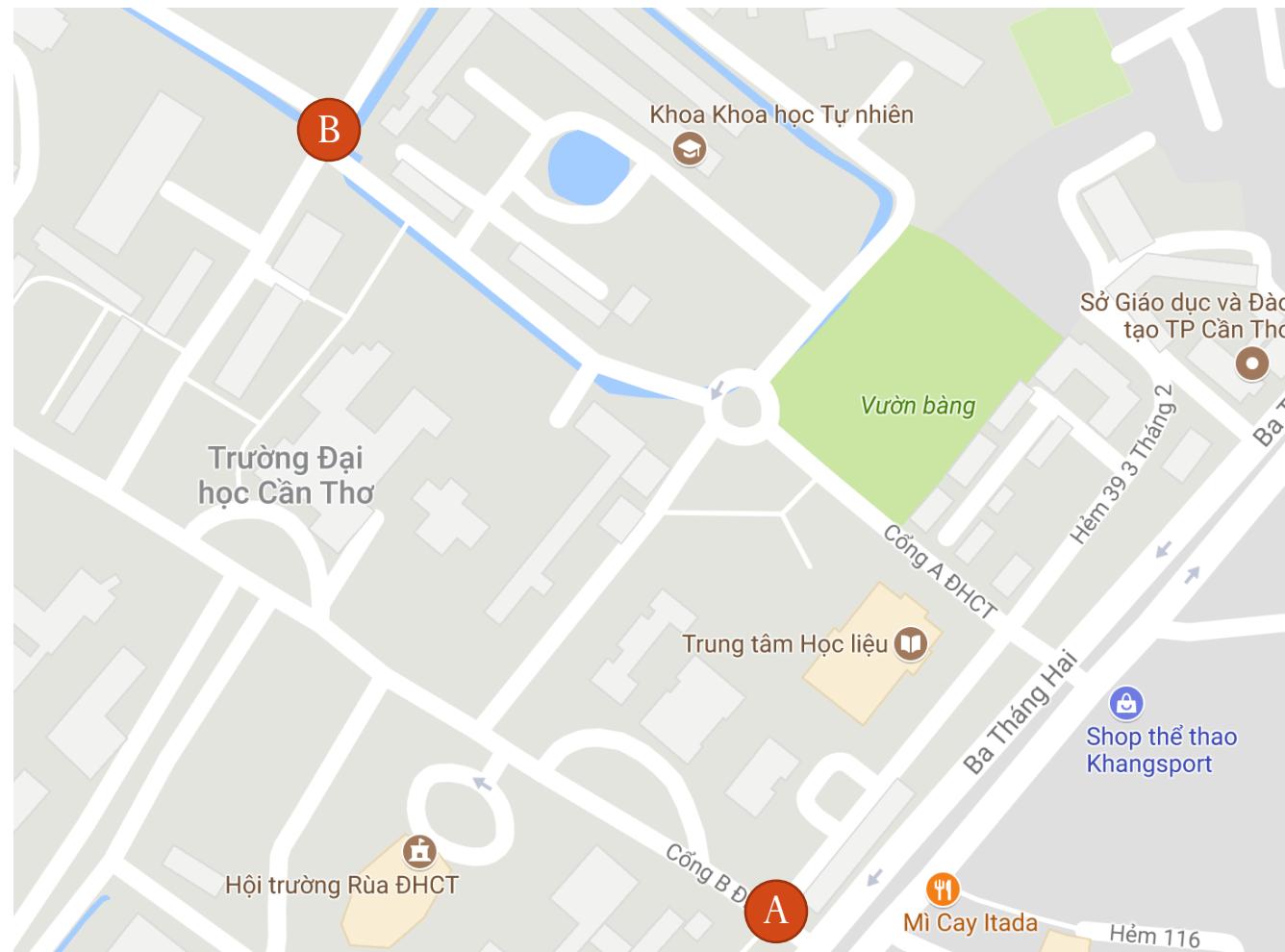
[pnkhang@cit.ctu.edu.vn](mailto:pnkhang@cit.ctu.edu.vn)

[vthuc@cit.ctu.edu.vn](mailto:vthuc@cit.ctu.edu.vn)



Cần Thơ, 2018

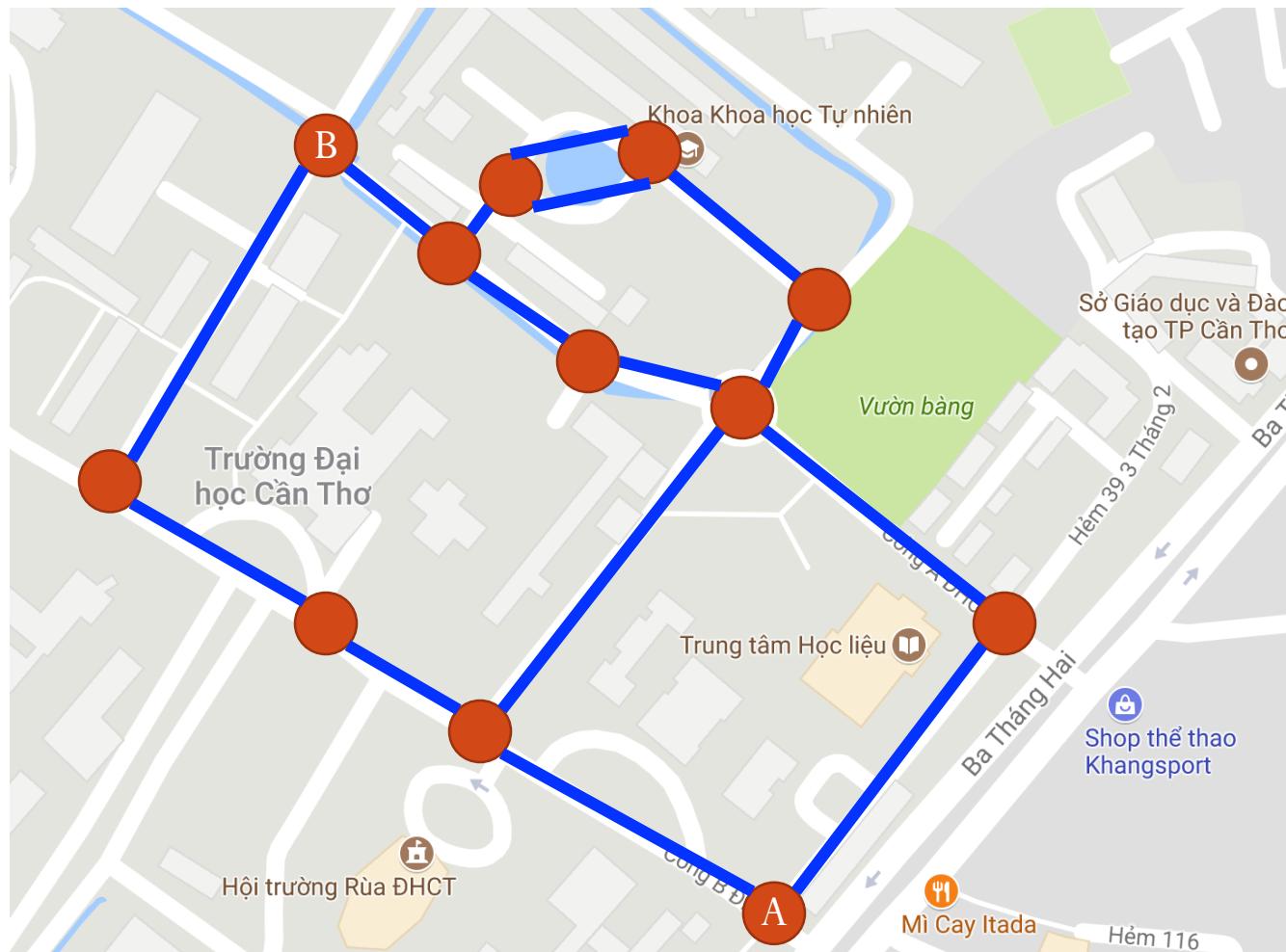
# Từ bản đồ đến đô thị



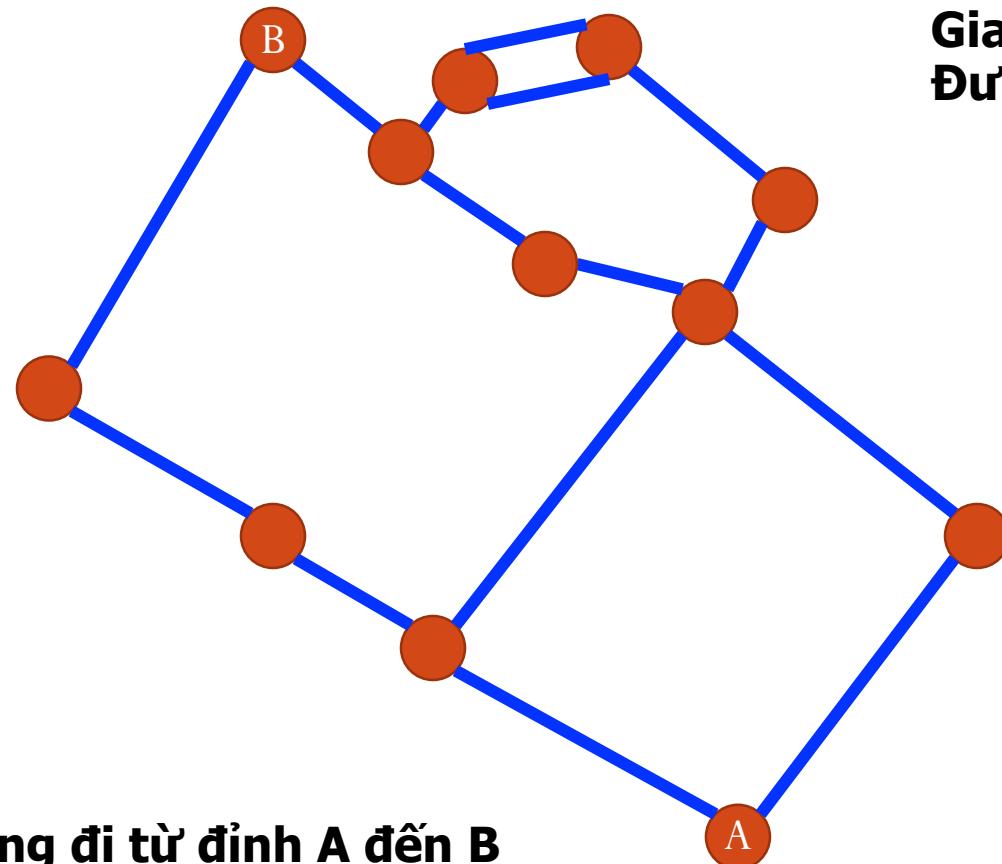
# Từ bản đồ đến đô thị



# Từ bản đồ đến đồ thị



# Từ bản đồ đến đồ thị



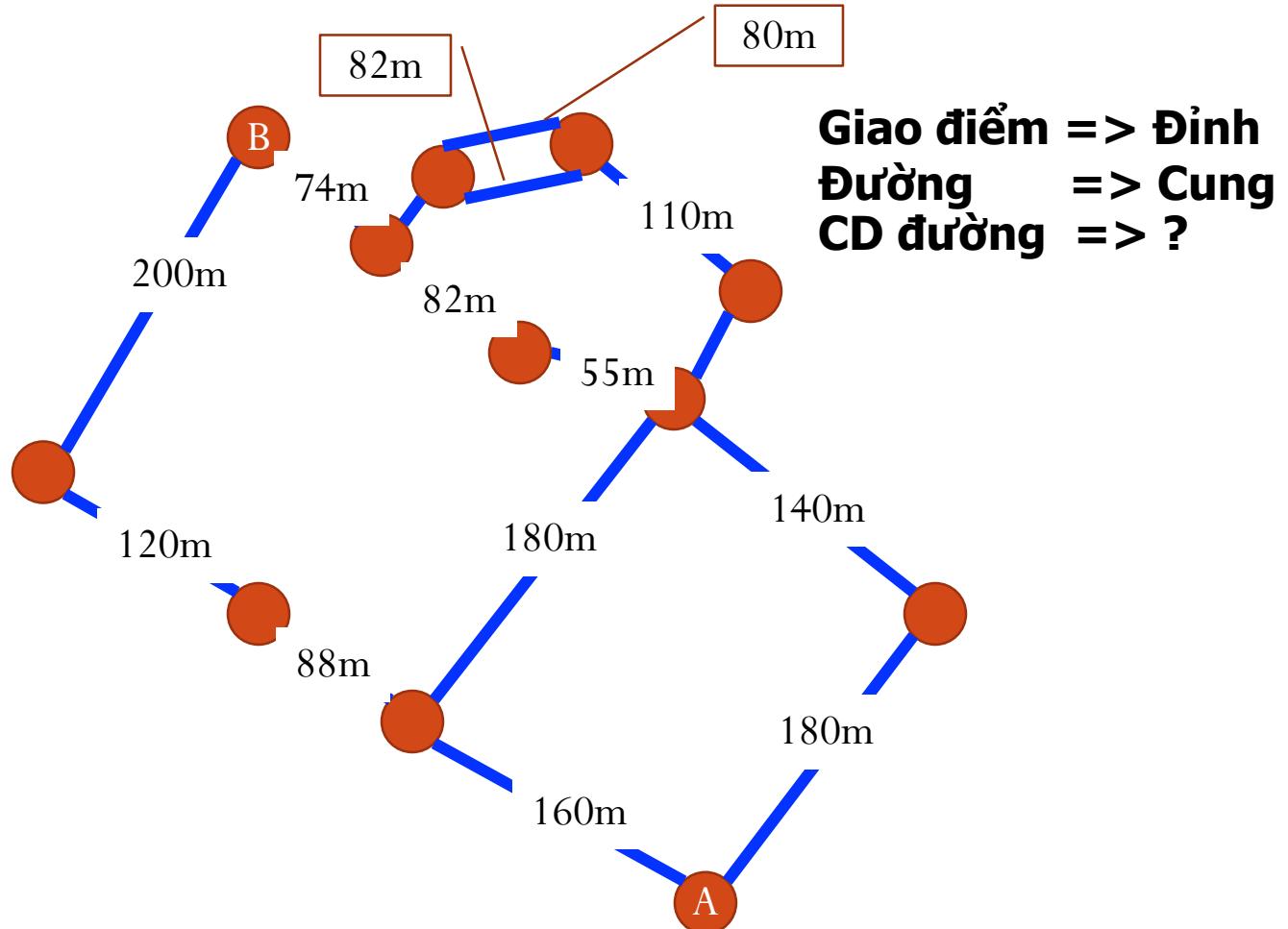
Giao điểm => Đỉnh  
Đường => Cung

Tìm đường đi từ đỉnh A đến B

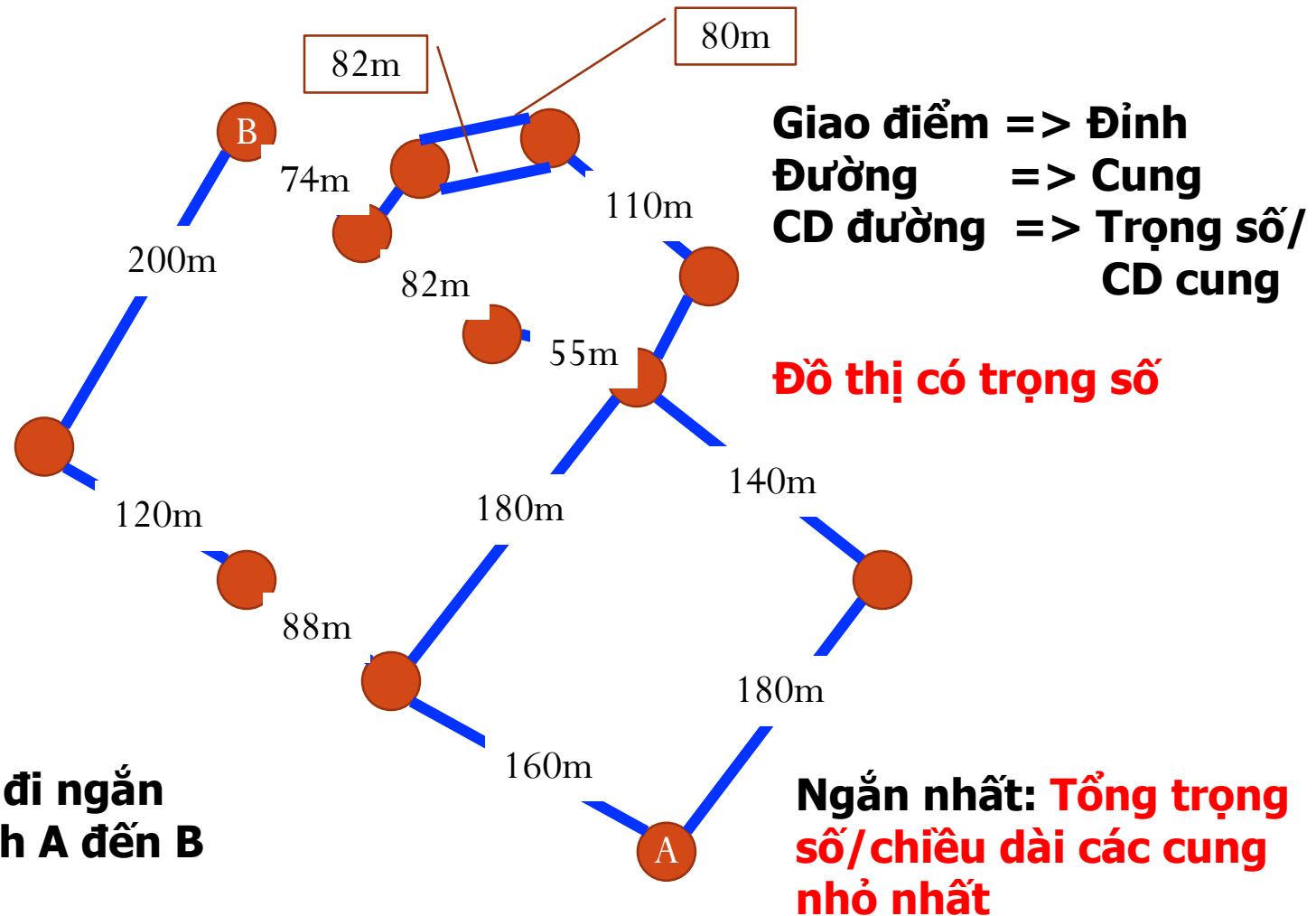
# Từ bản đồ đến đồ thị



# Từ bản đồ đến đồ thị

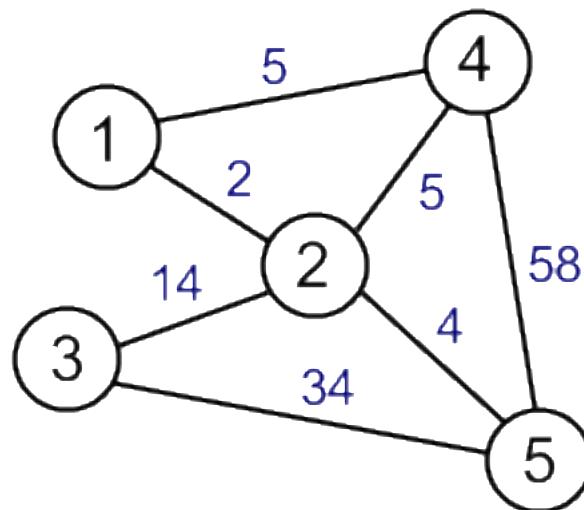


# Từ bản đồ đến đồ thị



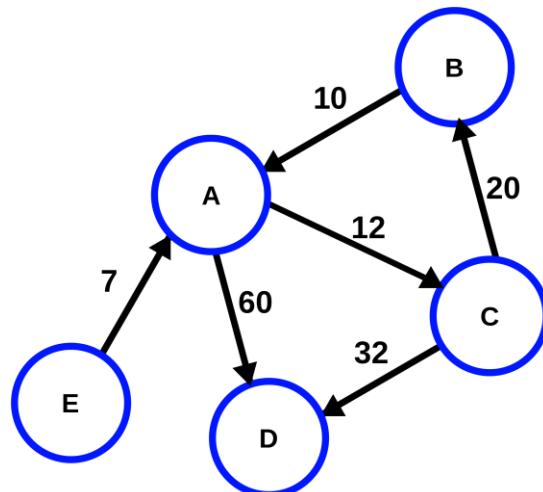
# Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
  - Ma trận kề (đỉnh - đỉnh) => **ma trận trọng số**
    - Nếu có  $(u, v)$  thì  $L[u, v] = \text{trọng số của cung } (u, v)$
    - Nếu không có  $(u, v)$  thì  $L[u, v] = \text{NO_EDGE}$  (vd: -1)



# Biểu diễn đồ thị có trọng số

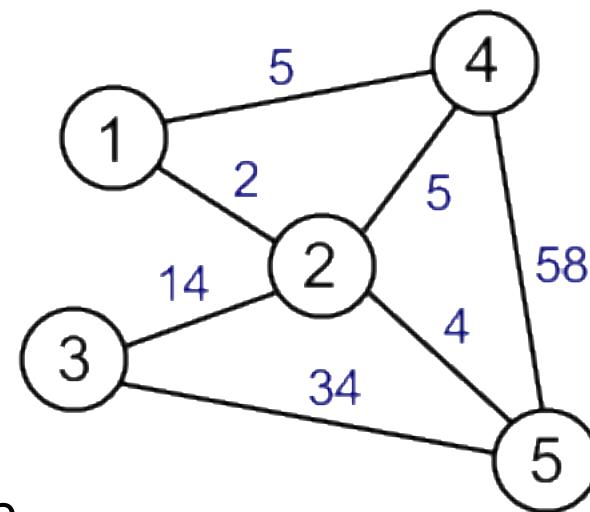
- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
  - Ma trận kề (đỉnh - đỉnh) => **ma trận trọng số**
    - Nếu có  $(u, v)$  thì  $L[u, v] = \text{trọng số của cung } (u, v)$
    - Nếu không có  $(u, v)$  thì  $L[u, v] = \text{NO_EDGE}$  (vd: -1)



# Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
  - Danh sách cung
    - Mỗi cung lưu trữ: đỉnh đầu, đỉnh cuối và trọng số

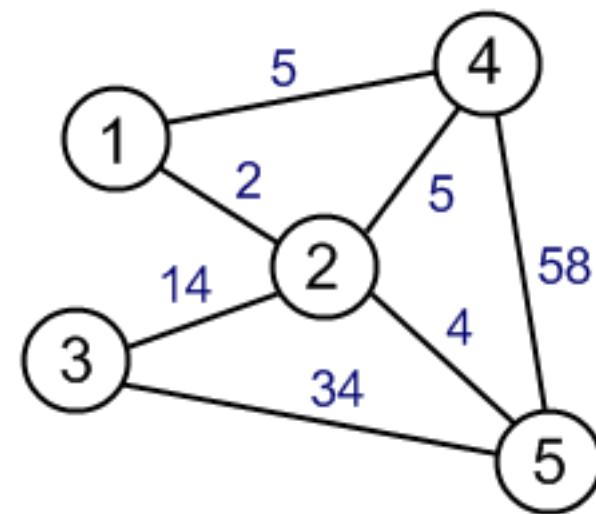
```
typedef struct {  
    int u, v;  
    int/float/double w;  
} Edge;
```



Graph = Danh sách các Edge

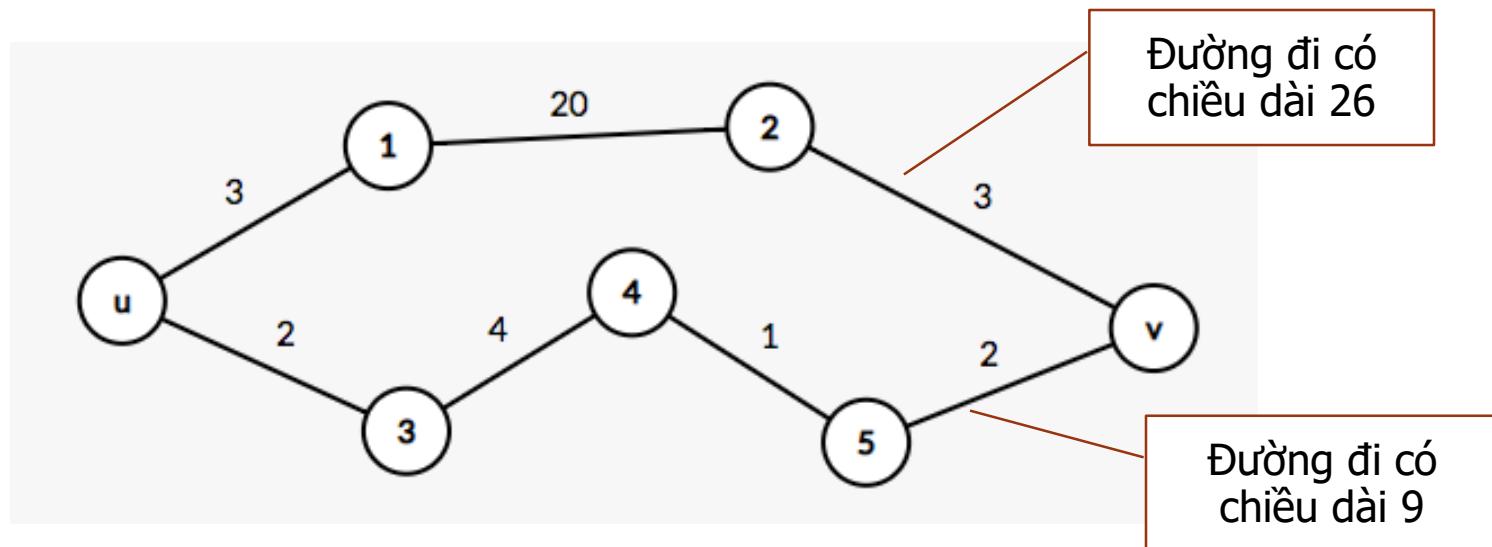
# Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
  - Danh sách kề
  - Bài tập:
    - **Tìm cách mở rộng phương pháp danh sách kề để biểu diễn đồ thị có trọng số**



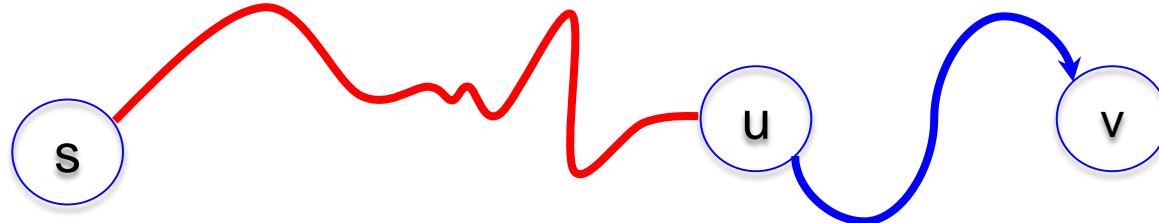
# Bài toán đường đi ngắn nhất

- *Đường đi ngắn nhất* (shortest path) từ đỉnh u đến đỉnh v trong đồ thị có trọng số là gì ?
  - *Chiều dài/Chi phí* (cost) của đường đi (path) là tổng các trọng số của các cung trên đường đi.
  - *Đường đi ngắn nhất* (shortest path) là đường đi có chiều dài nhỏ nhất.



# Đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
  - Tính chất *cấu trúc con tối ưu* (optimal substructure)  
=> có thể áp dụng kỹ thuật quy hoạch động để thiết kế các giải thuật tìm đường đi ngắn nhất.

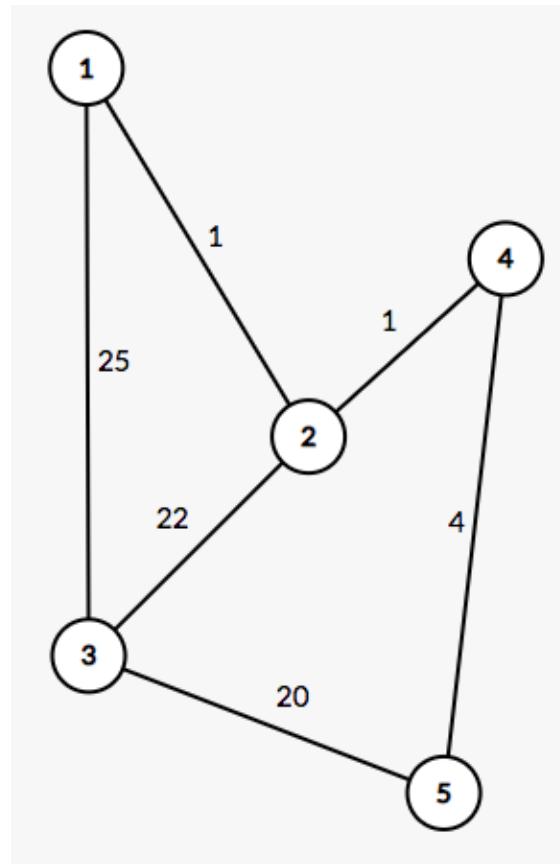


# Giải thuật Moore-Dijkstra

- Tìm đường đi ngắn nhất từ *1 đỉnh đến các đỉnh khác* trên đồ thị có trọng số (single source shortest path problem – SSSP)
- Điều kiện áp dụng:
  - Đồ thị có trọng số không âm
  - Có hướng hoặc vô hướng đều được
- Ý tưởng chính:
  - Khởi tạo đường đi trực tiếp.
  - Lần lượt cập nhật lại đường đi nếu **tìm được đường đi mới tốt hơn đường đi cũ**.

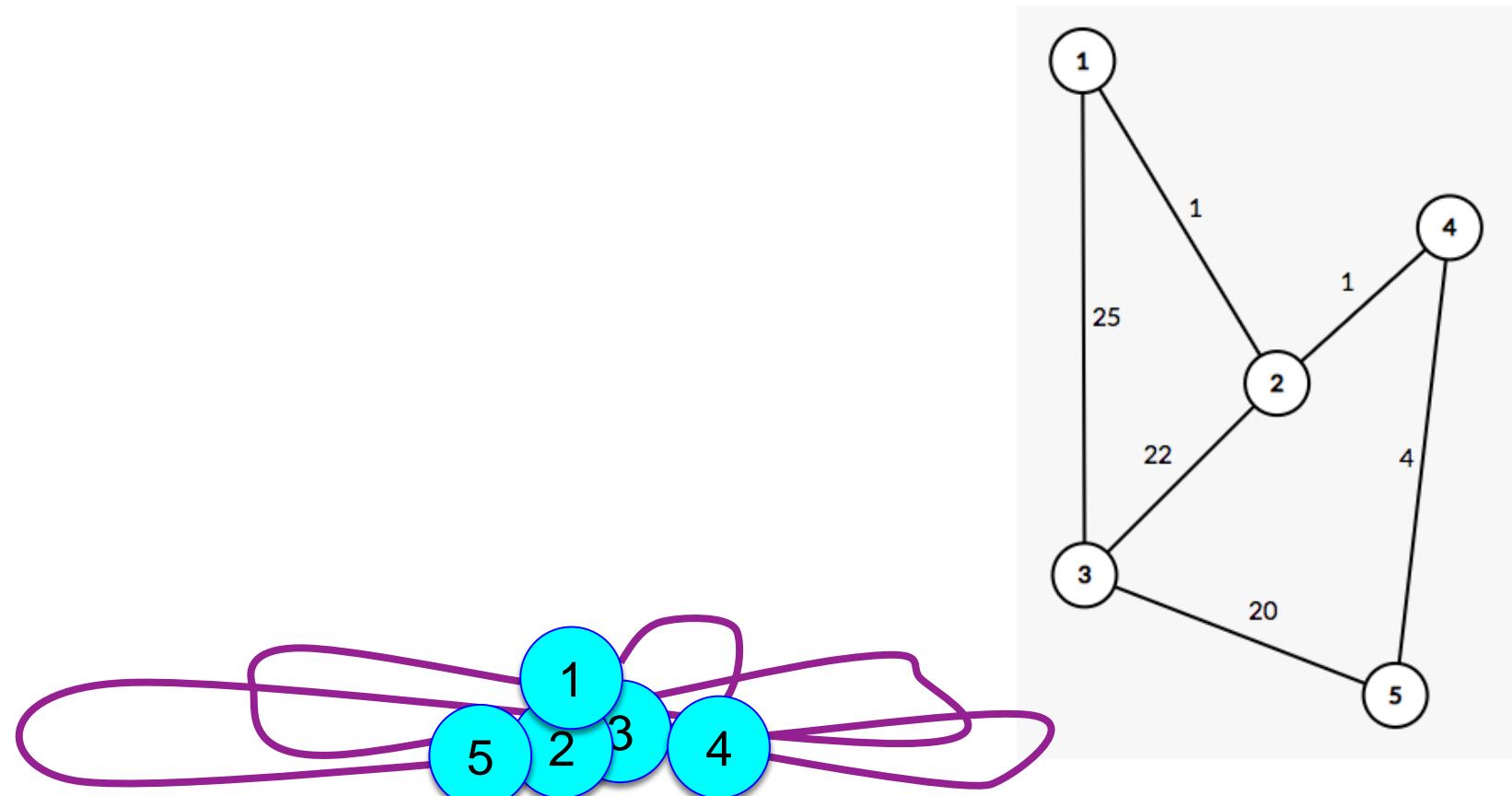
# Giải thuật Moore-Dijkstra

- Quan sát:
  - Tìm đường đi ngắn nhất từ 1



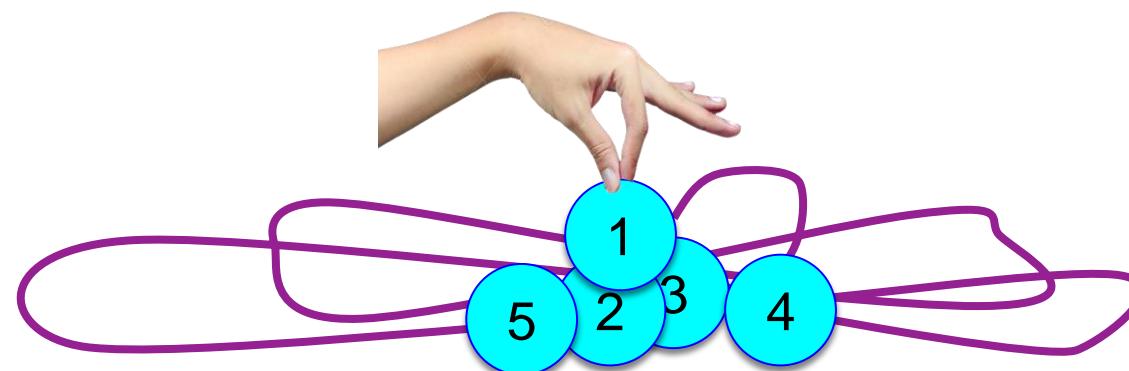
# Giải thuật Moore-Dijkstra

- Quan sát:
  - Tìm đường đi ngắn nhất từ 1



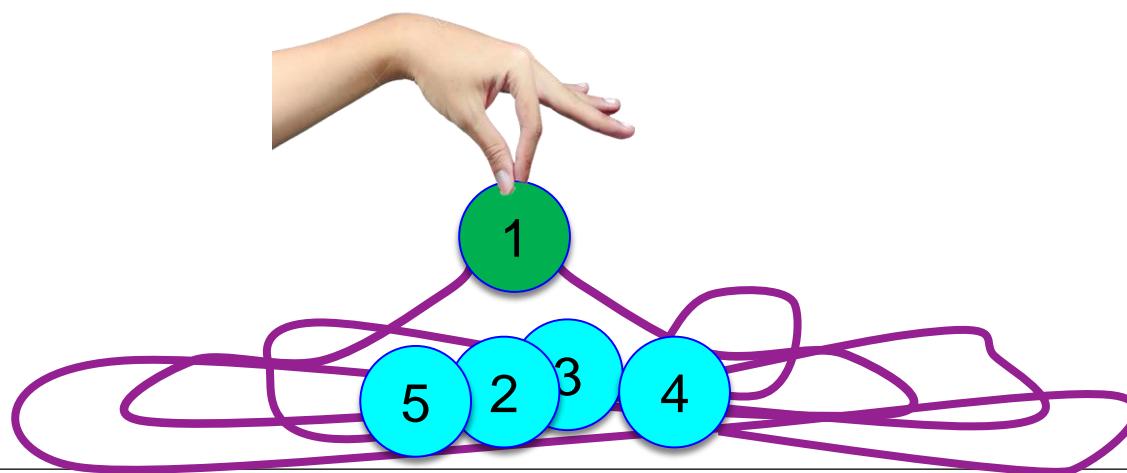
# Giải thuật Moore-Dijkstra

- Quan sát:



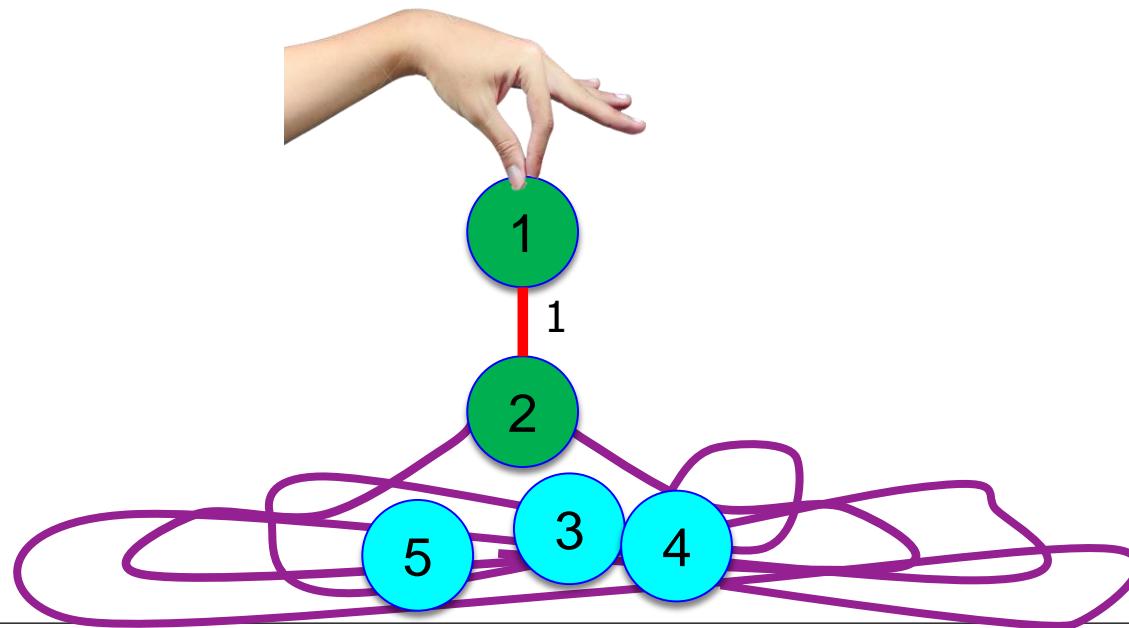
# Giải thuật Moore-Dijkstra

- Quan sát:



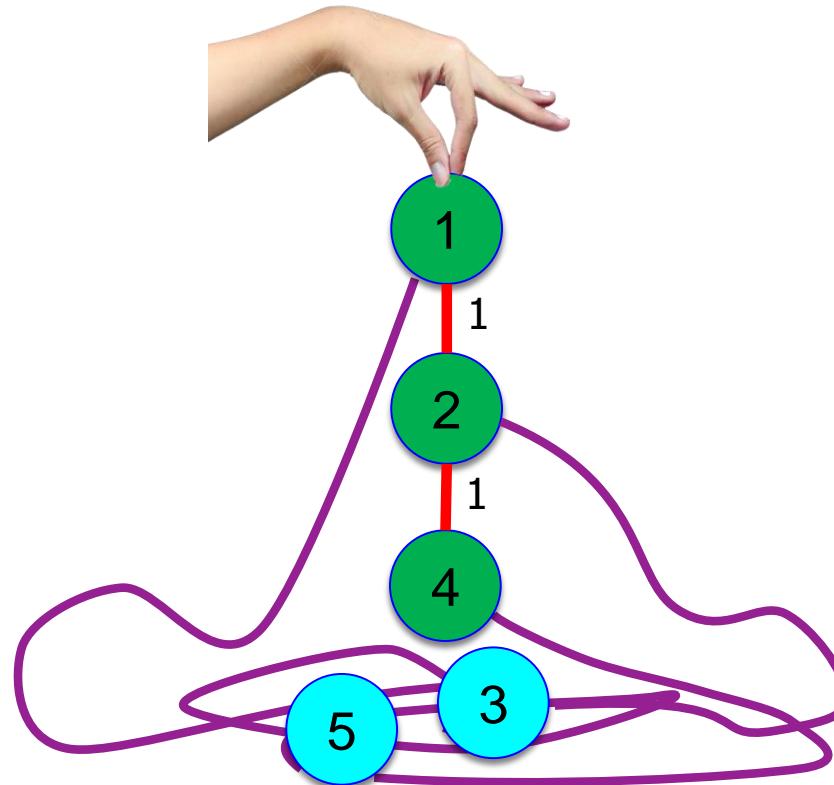
# Giải thuật Moore-Dijkstra

- Quan sát:



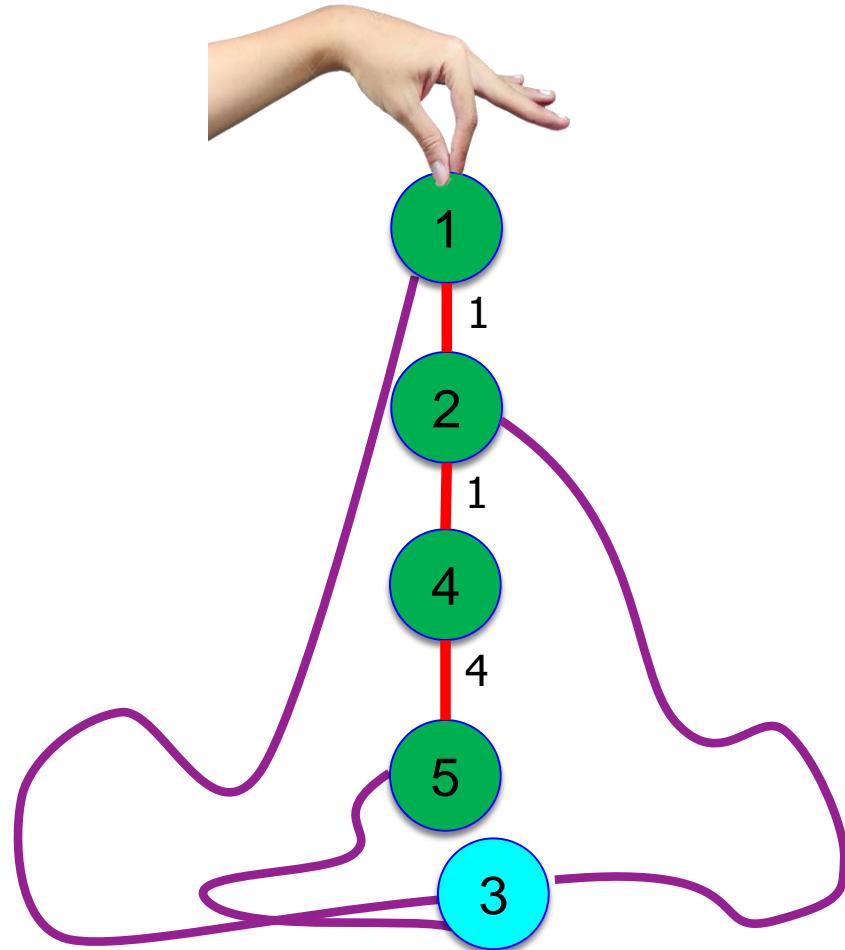
# Giải thuật Moore-Dijkstra

- Quan sát:



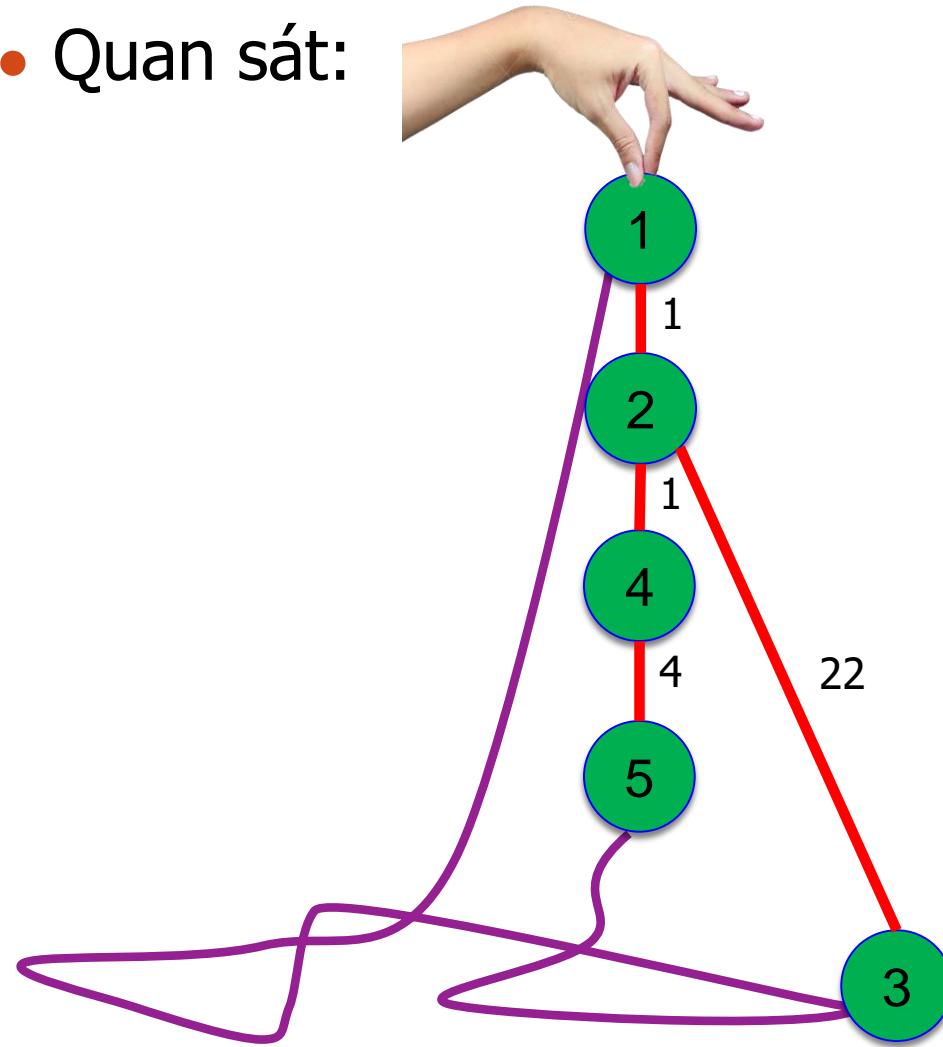
# Giải thuật Moore-Dijkstra

- Quan sát:



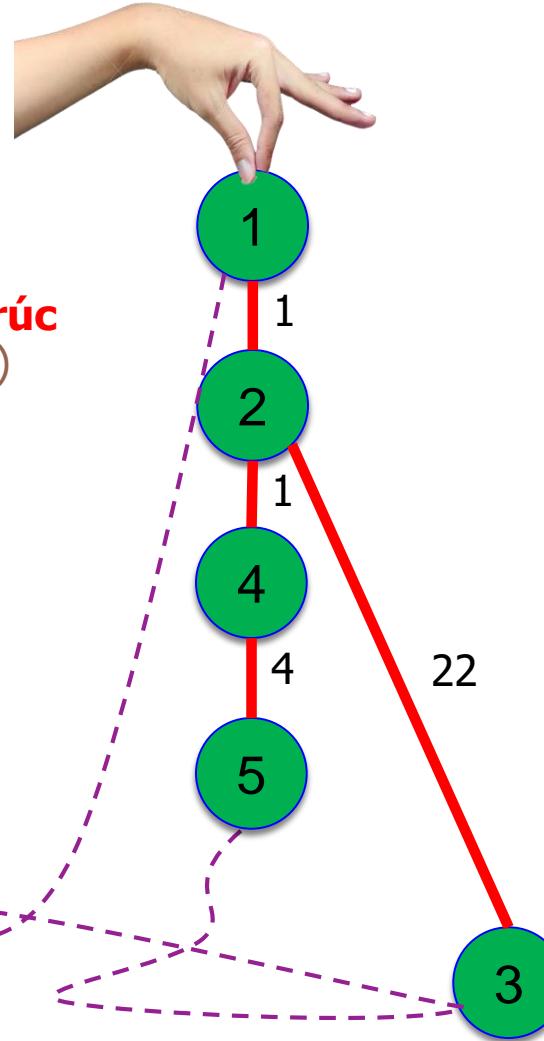
# Giải thuật Moore-Dijkstra

- Quan sát:



# Giải thuật Moore-Dijkstra

- Quan sát:

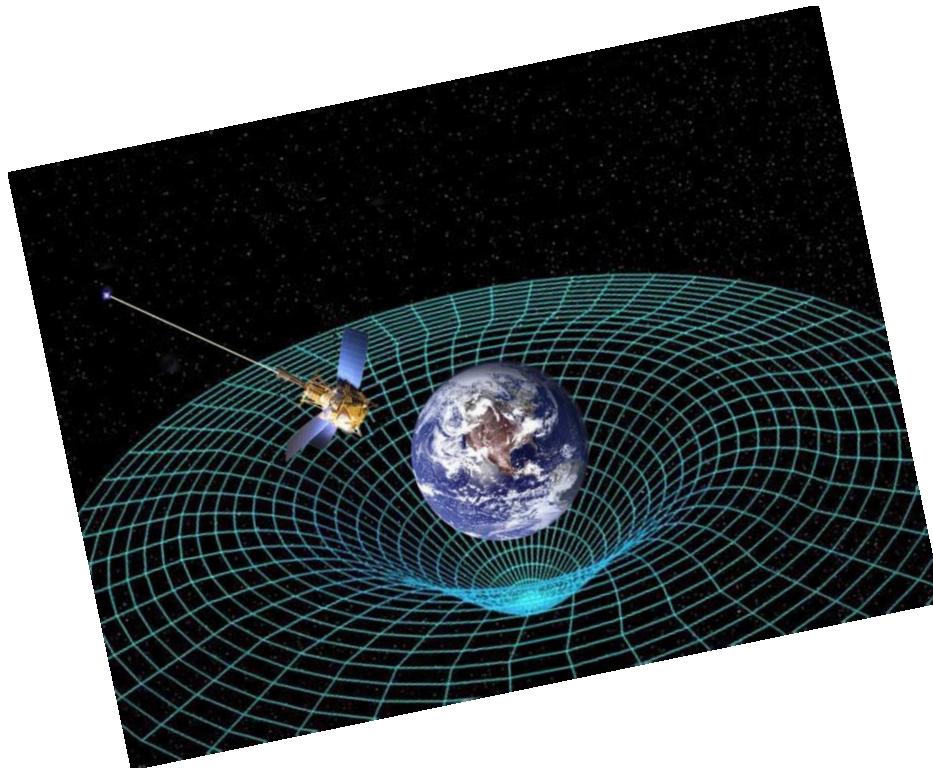


Quá trình này sinh ra **cấu trúc cây** (đường đi ngắn nhất)

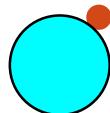
Các đường đi ngắn nhất là đường đi từ gốc đến các đỉnh của cây này.

# Cài đặt giải thuật như thế nào ?

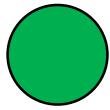
- **Không có** dây và trọng lực



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn

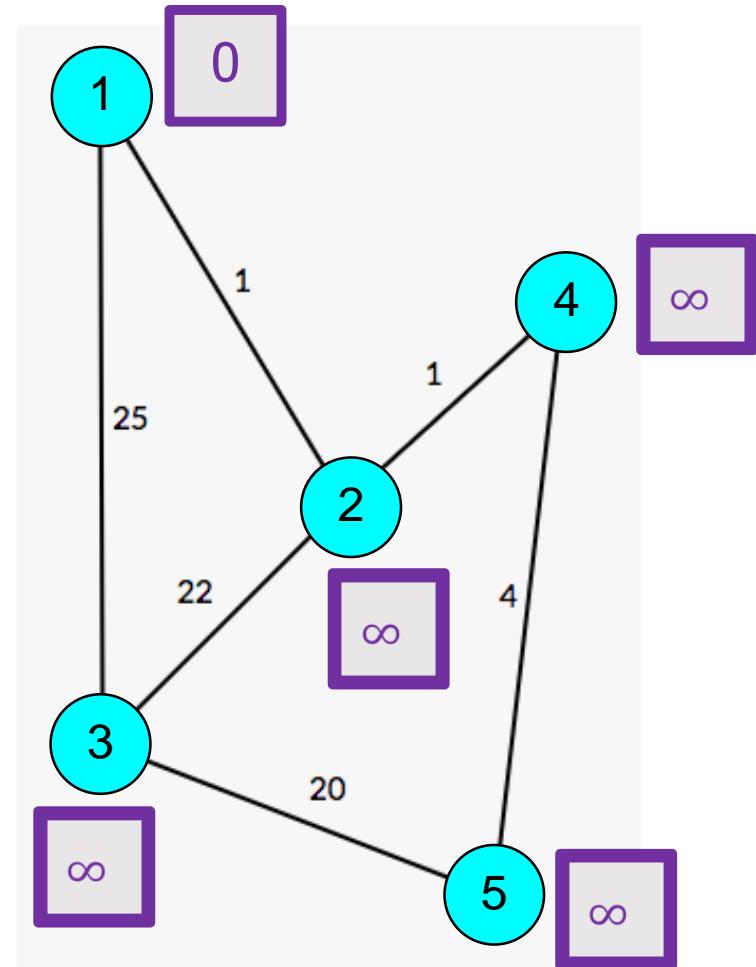


Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.

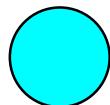


$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

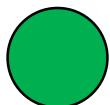
- Đỉnh bắt đầu: s (vd: 1)
- Khởi tạo:
  - các đỉnh đều chưa chắc chắn
  - $\pi[s] = 0$ ,
  - Các đỉnh khác,  $\pi[v] = \infty$ .



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn

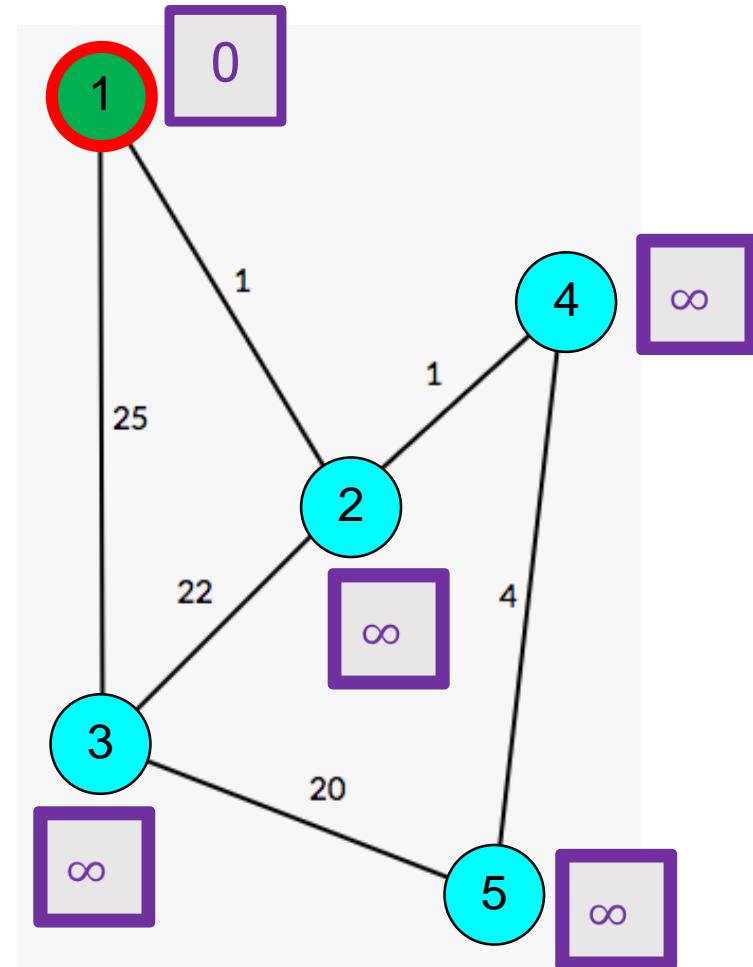


Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.

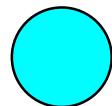


$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

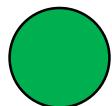
- Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
- Đánh dấu  $u$  là đỉnh **chắc chắn**
- Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn

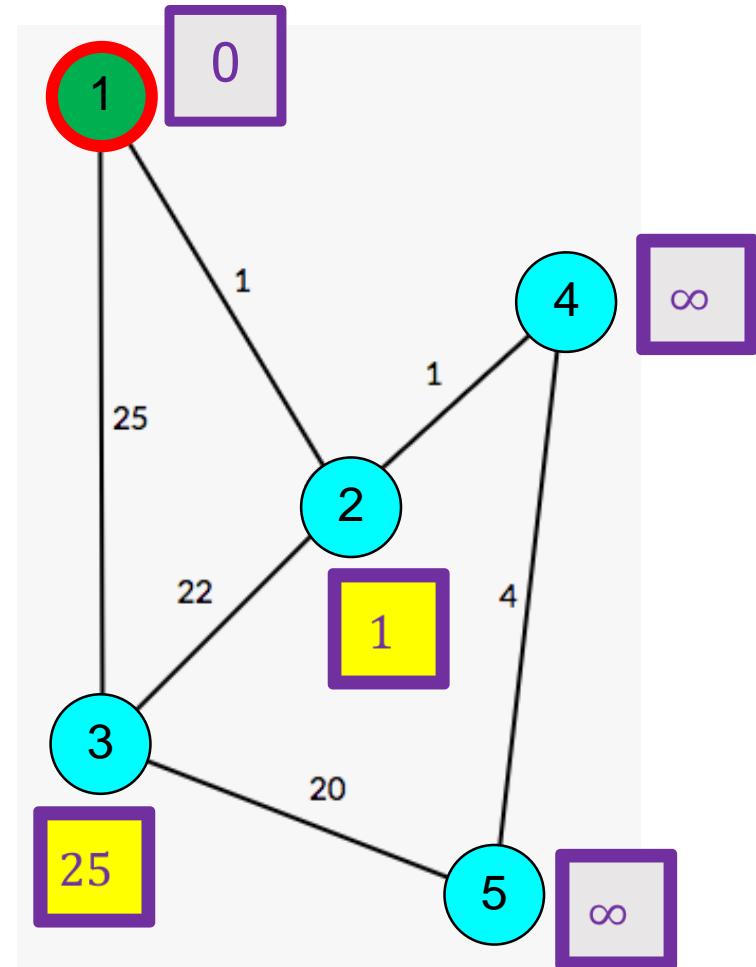


Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.

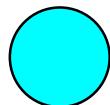


$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

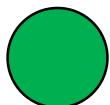
- Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
- Đánh dấu  $u$  là đỉnh **chắc chắn**
- Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn

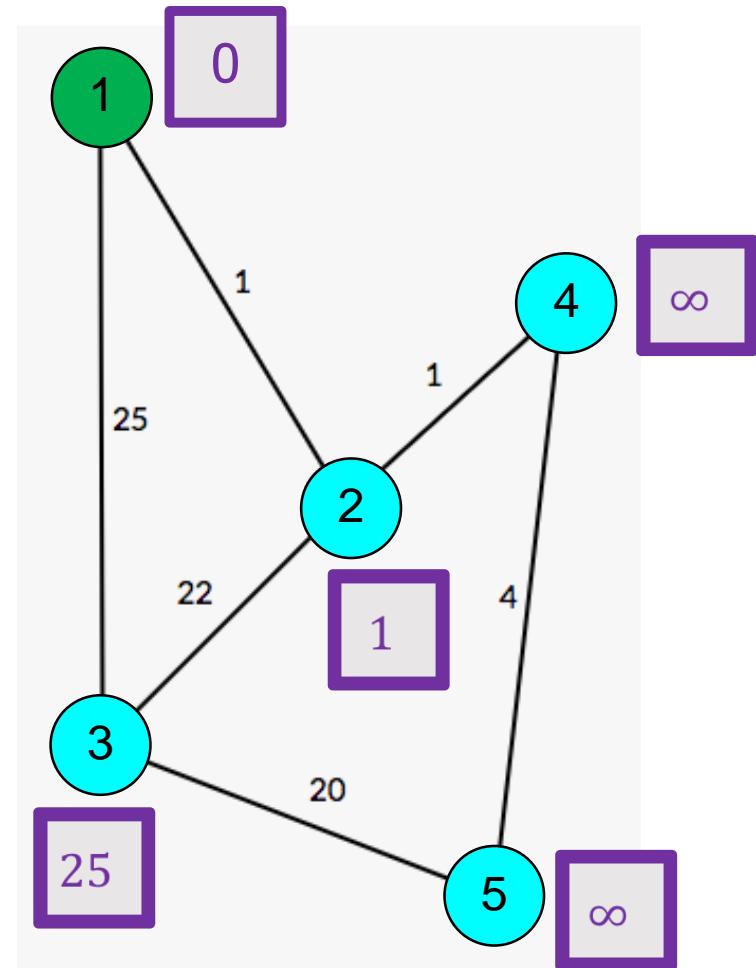


Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.

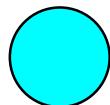


$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

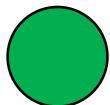
1. Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
  2. Đánh dấu  $u$  là đỉnh **chắc chắn**
  3. Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$
- Lặp lại 1, 2, 3**



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn

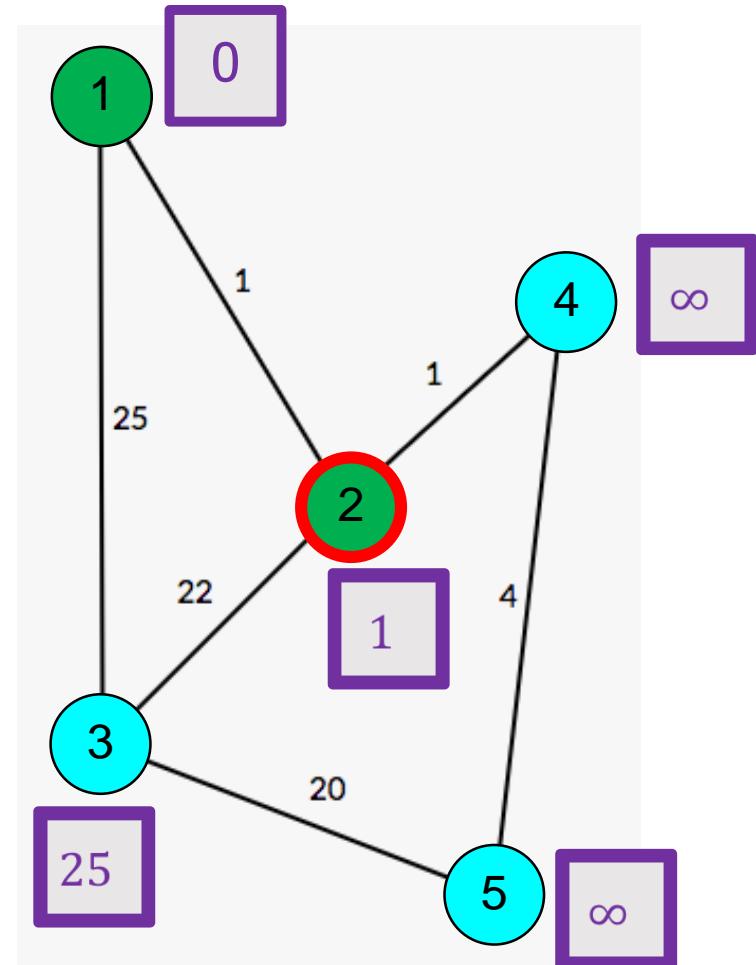


Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.

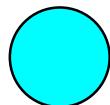


$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

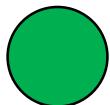
1. Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
  2. Đánh dấu  $u$  là đỉnh **chắc chắn**
  3. Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$
- Lặp lại 1, 2, 3**



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn

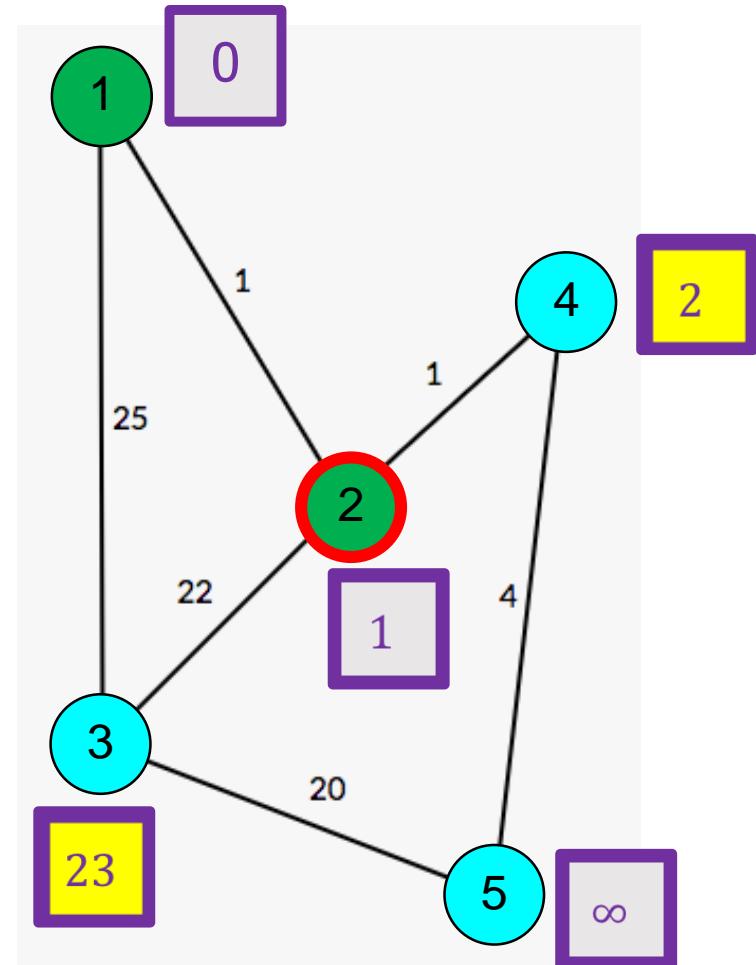


Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.

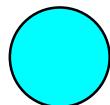


$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

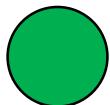
1. Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
  2. Đánh dấu  $u$  là đỉnh **chắc chắn**
  3. Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$
- Lặp lại 1, 2, 3**



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn

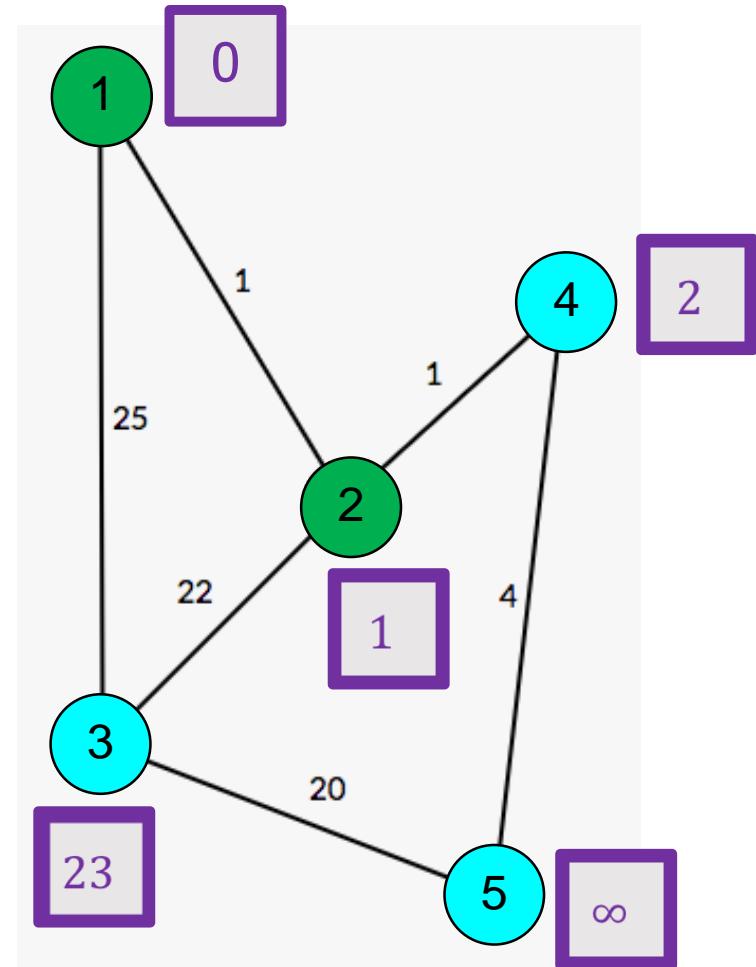


Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.

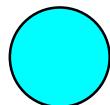


$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

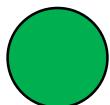
1. Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
  2. Đánh dấu  $u$  là đỉnh **chắc chắn**
  3. Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$
- Lặp lại 1, 2, 3**



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



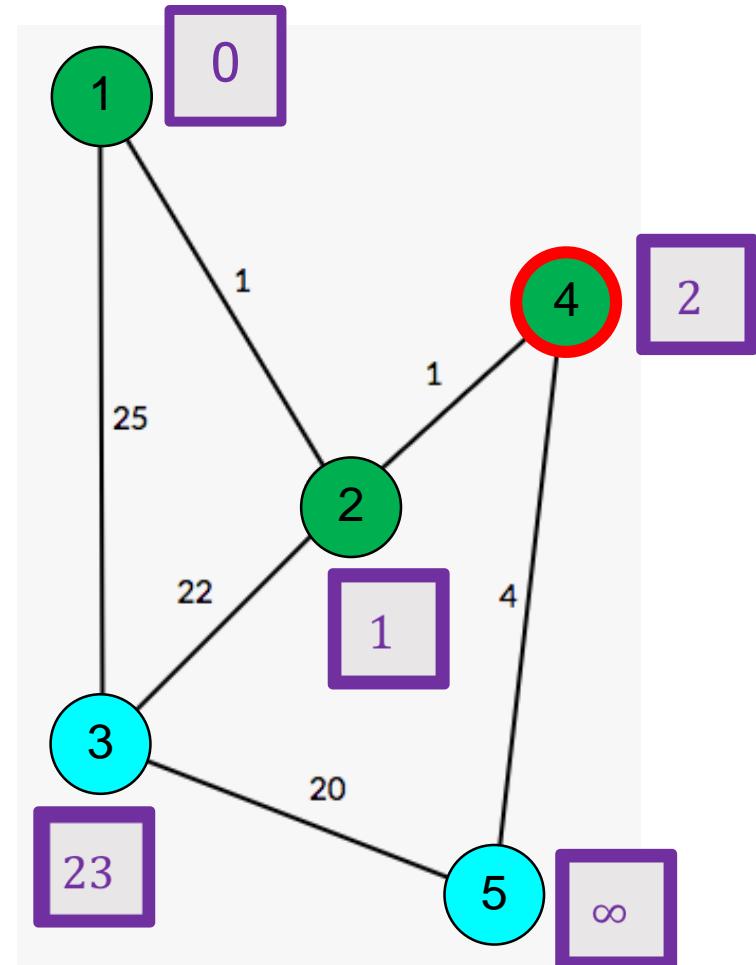
Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.



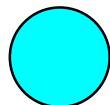
$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

1. Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
2. Đánh dấu  $u$  là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$

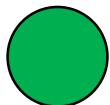
**Lặp lại 1, 2, 3**



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn

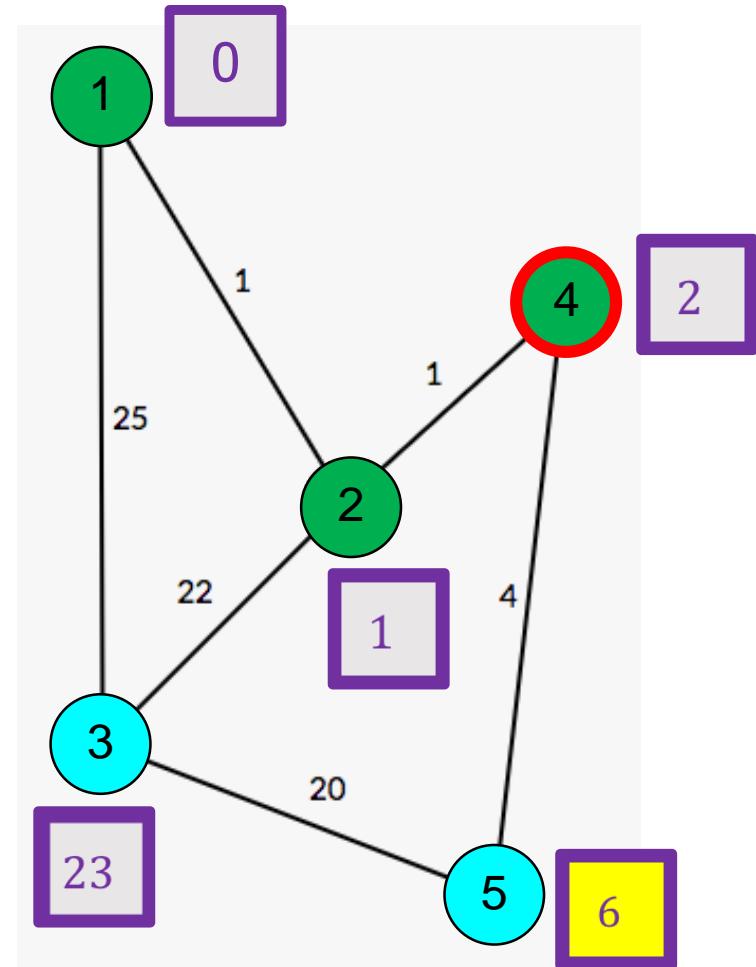


Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.

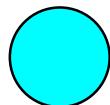


$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

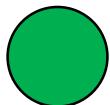
1. Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
  2. Đánh dấu  $u$  là đỉnh **chắc chắn**
  3. Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$
- Lặp lại 1, 2, 3**



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn

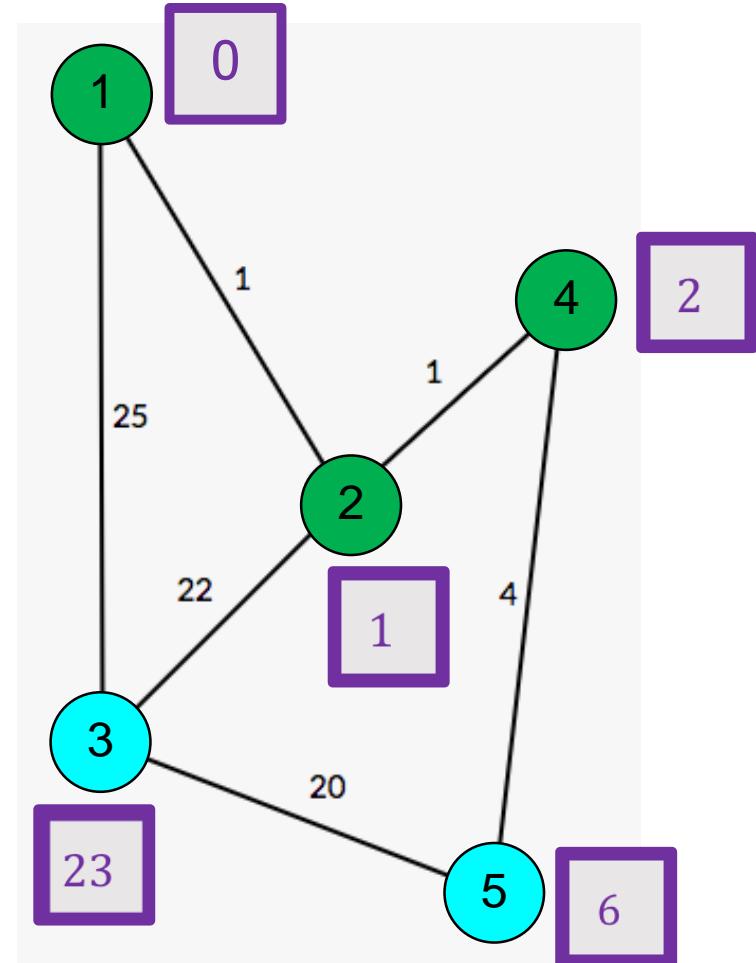


Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.

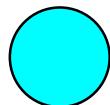


$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

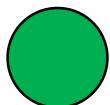
1. Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
  2. Đánh dấu  $u$  là đỉnh **chắc chắn**
  3. Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$
- Lặp lại 1, 2, 3**



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn

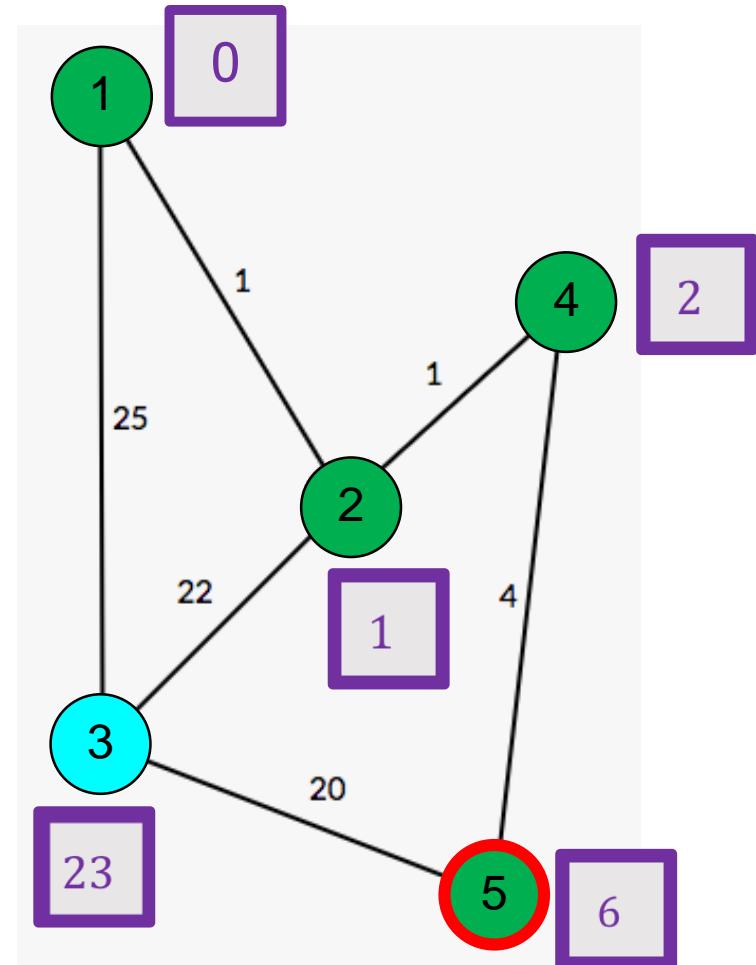


Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.

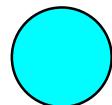


$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

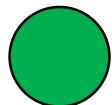
1. Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
  2. Đánh dấu  $u$  là đỉnh **chắc chắn**
  3. Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$
- Lặp lại 1, 2, 3**



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.

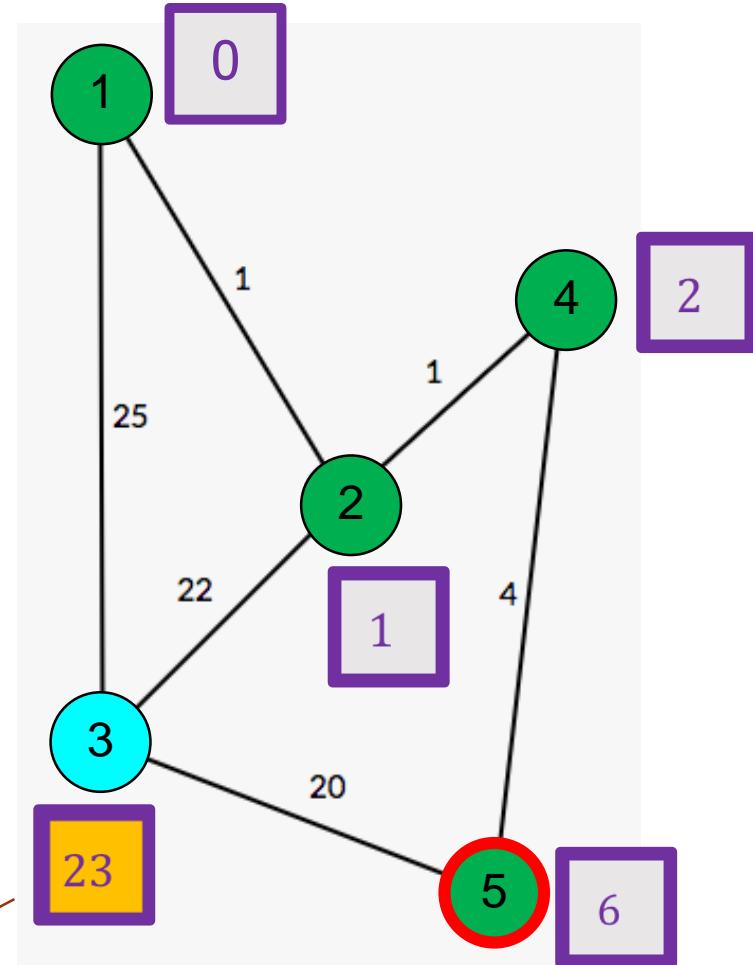


$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

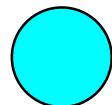
1. Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
2. Đánh dấu  $u$  là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$

**Lặp lại 1, 2, 3**

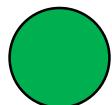
Không cập nhật



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.

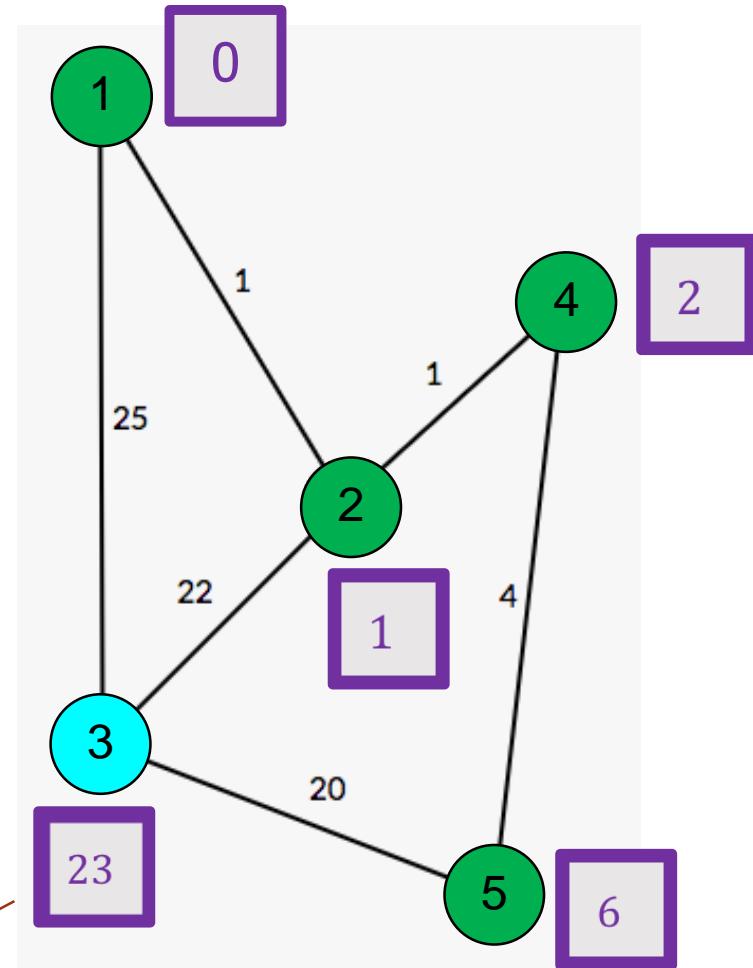


$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

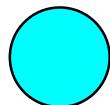
1. Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
2. Đánh dấu  $u$  là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$

**Lặp lại 1, 2, 3**

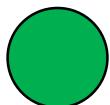
Không cập nhật



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.

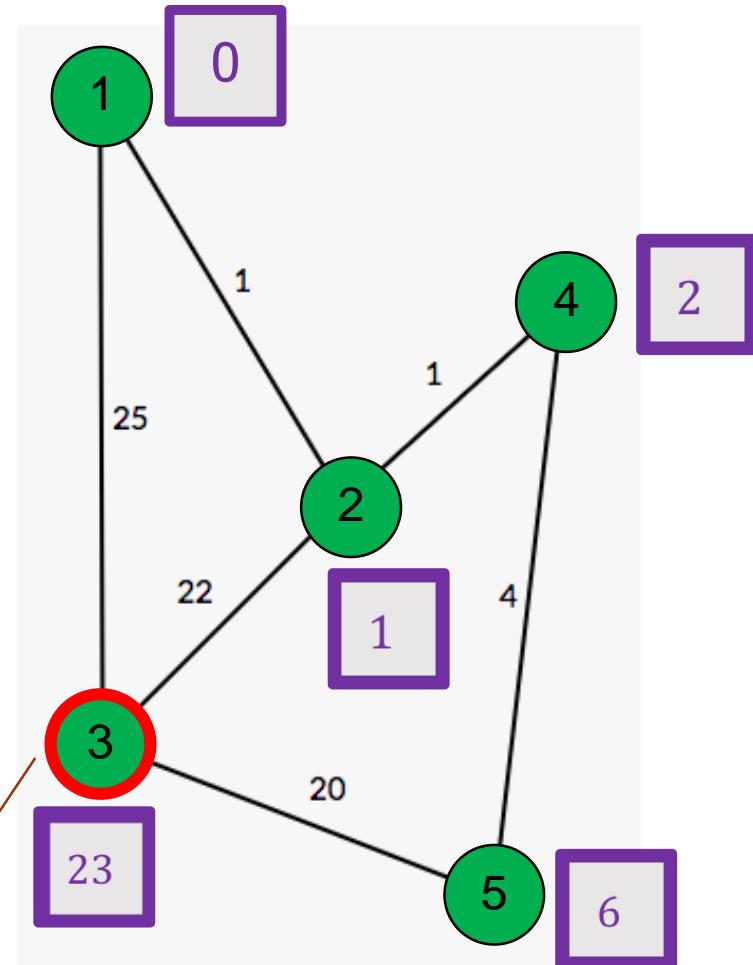


$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

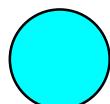
1. Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
2. Đánh dấu  $u$  là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$

**Lặp lại 1, 2, 3**

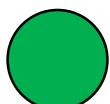
Không còn đỉnh kề



# GT Dijkstra qua ví dụ



Đỉnh chưa chắc chắn

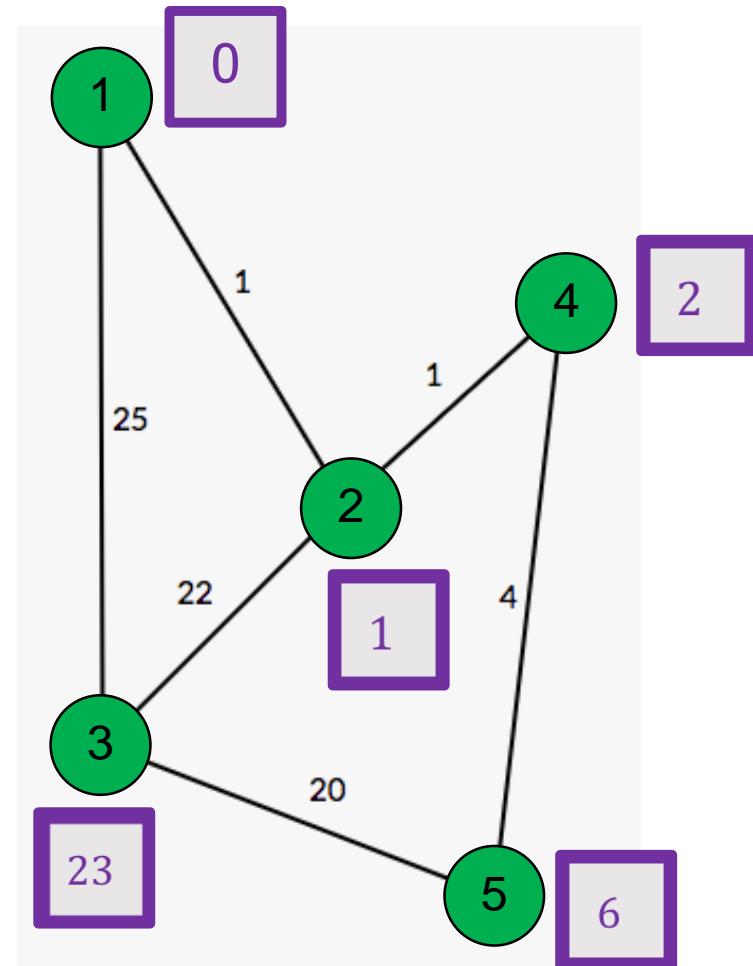


Đỉnh chắc chắn, đã tìm được  
đường đi ngắn nhất từ s đến nó.



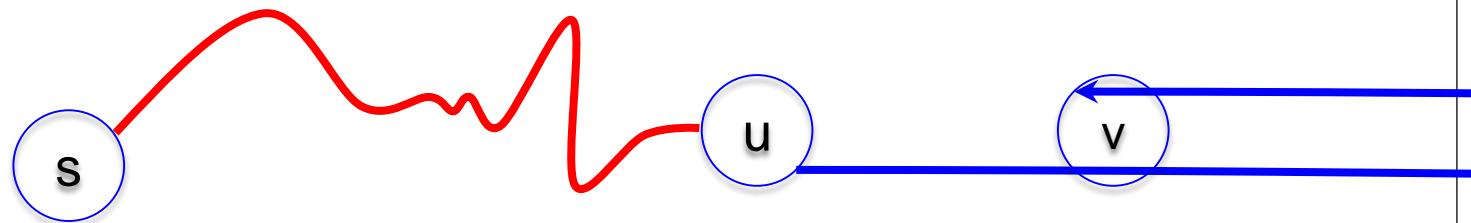
$x = \pi[u]$ : chiều dài đường đi  
ngắn nhất tính đến thời điểm này.

1. Chọn 1 đỉnh **chưa chắc chắn**  $u$  có  $\pi[u]$  nhỏ nhất
  2. Đánh dấu  $u$  là đỉnh **chắc chắn**
  3. Cập nhật các đỉnh kề  $v$  của  $u$   
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$
- Lặp lại 1, 2, 3**



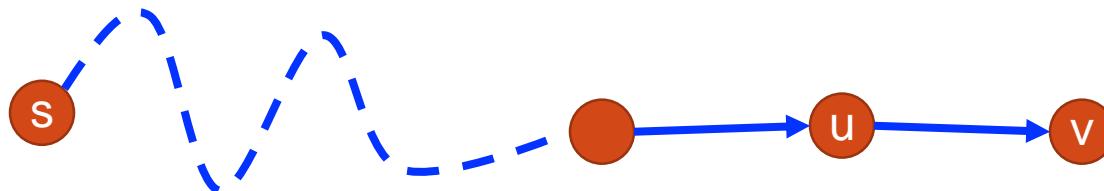
# Biểu diễn đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
  - Tính chất *cấu trúc con tối ưu* (optimal substructure)



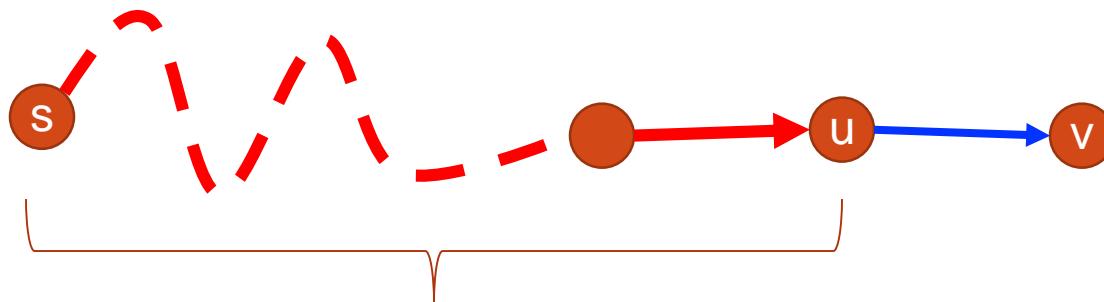
# Biểu diễn đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
  - Tính chất *cấu trúc con tối ưu* (optimal substructure)



# Biểu diễn đường đi ngắn nhất

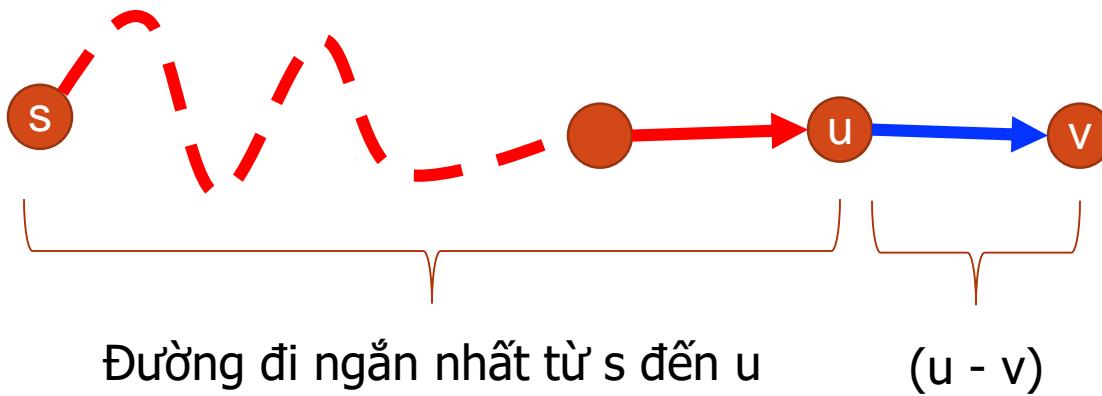
- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
  - Tính chất *cấu trúc con tối ưu* (optimal substructure)



Đường đi ngắn nhất từ s đến u

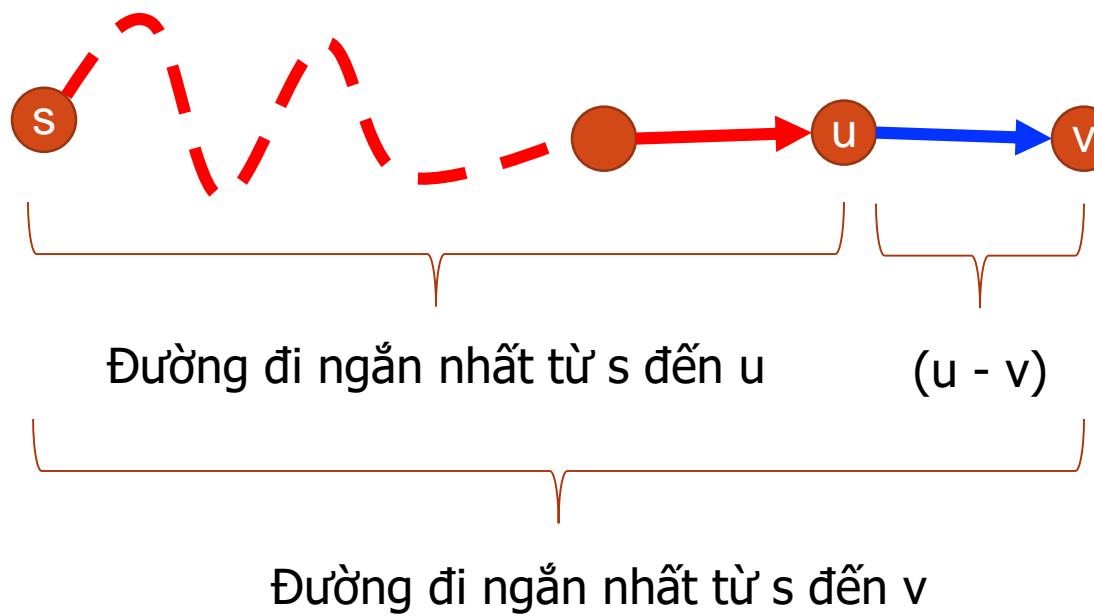
# Biểu diễn đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
    - Tính chất *cấu trúc con tối ưu* (optimal substructure)



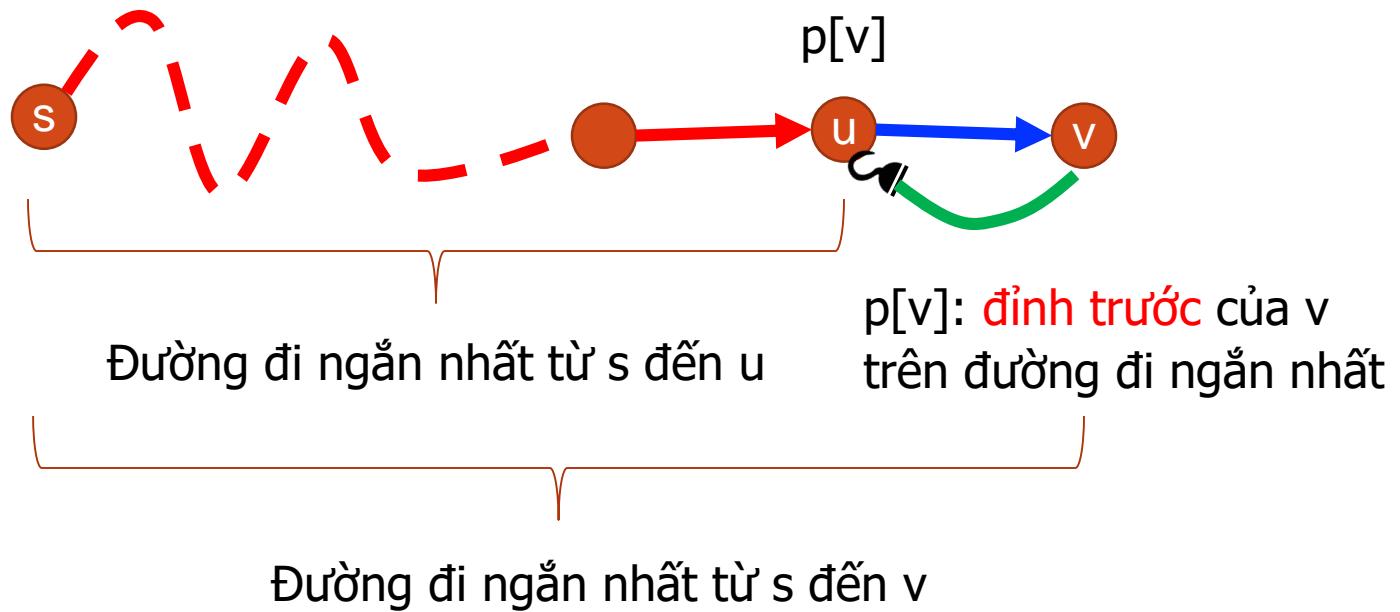
# Biểu diễn đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
  - Tính chất *cấu trúc con tối ưu* (optimal substructure)



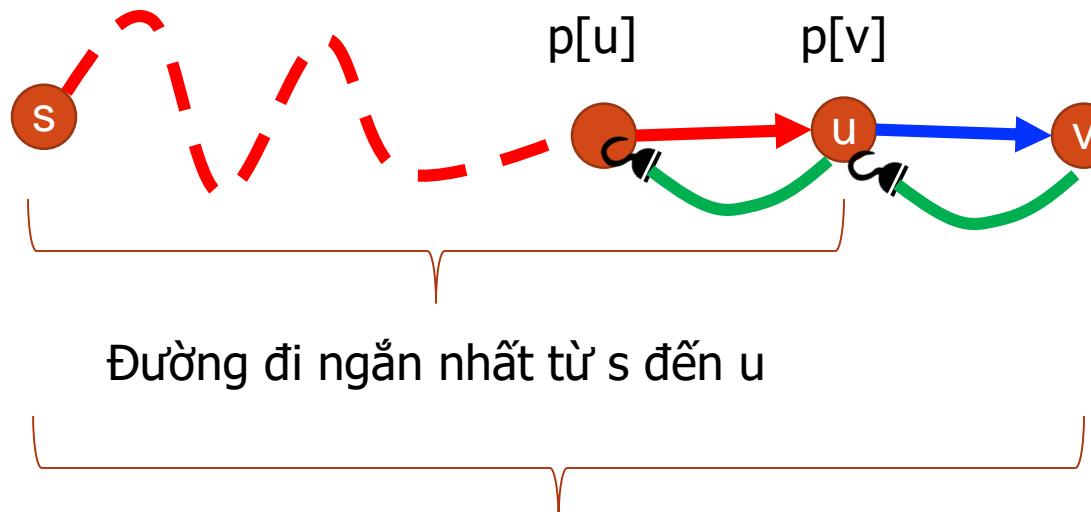
# Biểu diễn đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
  - Tính chất *cấu trúc con tối ưu* (optimal substructure)



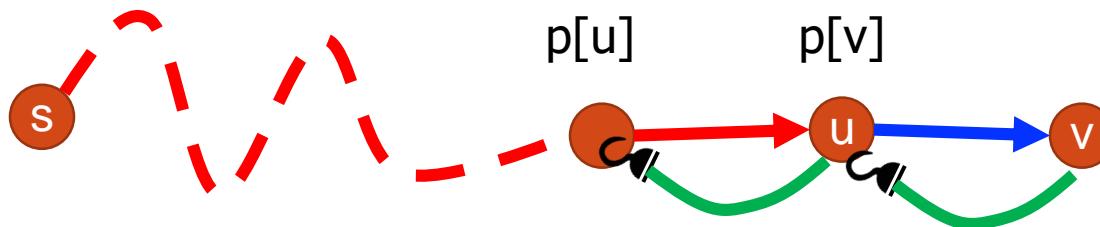
# Biểu diễn đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
  - Tính chất *cấu trúc con tối ưu* (optimal substructure)



# Biểu diễn đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
  - Tính chất *cấu trúc con tối ưu* (optimal substructure)



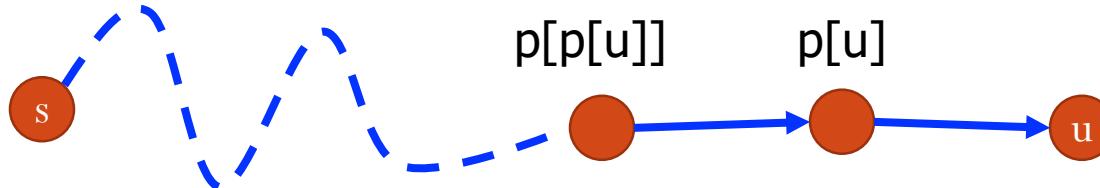
- Biểu diễn đường đi:
  - Lưu các  $p[1], p[2], \dots, p[n]$

# Giải thuật Moore-Dijkstra

- Tìm đường đi ngắn nhất từ 1 đỉnh đến các đỉnh khác trên đồ thị có trọng số
- Điều kiện áp dụng:
  - Đồ thị có trọng số không âm
  - Có hướng hoặc vô hướng đều được
- Ý tưởng chính:
  - Khởi tạo đường đi trực tiếp.
  - Lần lượt cập nhật lại đường đi nếu **tìm được đường đi mới tốt hơn đường đi cũ**.

# Giải thuật Moore-Dijkstra

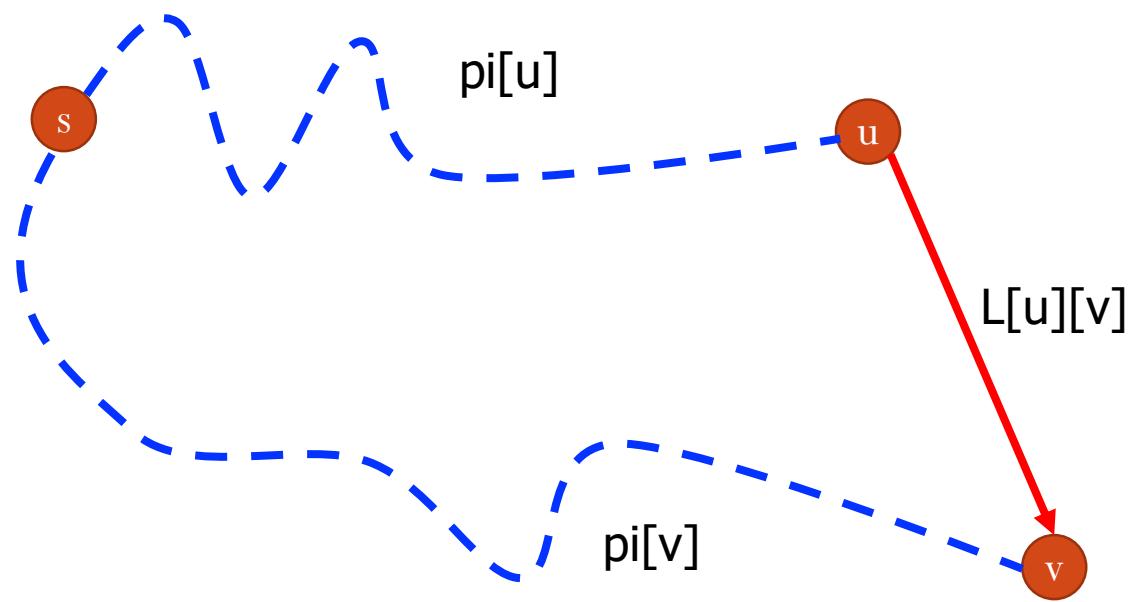
- Gọi đỉnh bắt đầu là s
- Các biến hỗ trợ
  - $\pi[u]$ : **chiều dài đường đi ngắn nhất** (tạm thời) từ s đến u.
  - $p[u]$ : **đỉnh trước** đỉnh u trên đường đi ngắn nhất (tạm thời) từ s đến u.
  - $mark[u]$ : đánh dấu đỉnh u (đã tìm được đường đi ngắn nhất đến u)
- Nếu  $mark[u] == 1$  thì  $\pi[u]$  chứa chiều dài đường đi ngắn nhất từ s đến u và  $p[u]$  là đỉnh trước của đỉnh.



# Giải thuật Moore-Dijkstra

- Giải thuật
  - Khởi tạo:
    - $\pi[u] = \infty$  với mọi  $u$
    - $p[u] = -1$  với mọi  $u$
    - $\text{mark}[u] = 0$  với mọi  $u$
    - $\pi[s] = 0$ ; đường đi ngắn nhất từ  $s$  đến  $s$  bằng 0.
  - Lặp  $n-1$  lần
    - Chọn đỉnh **chưa đánh dấu** và có  **$\pi[u]$  nhỏ nhất**
    - Đánh dấu  $u$
    - Xét các đỉnh kề  $v$  (chưa đánh dấu) của  $u$  để cập nhật đường đi nếu đường đi qua  $u$  rồi đến  $v$  tốt hơn đường đi cũ

# Giải thuật Moore-Dijkstra



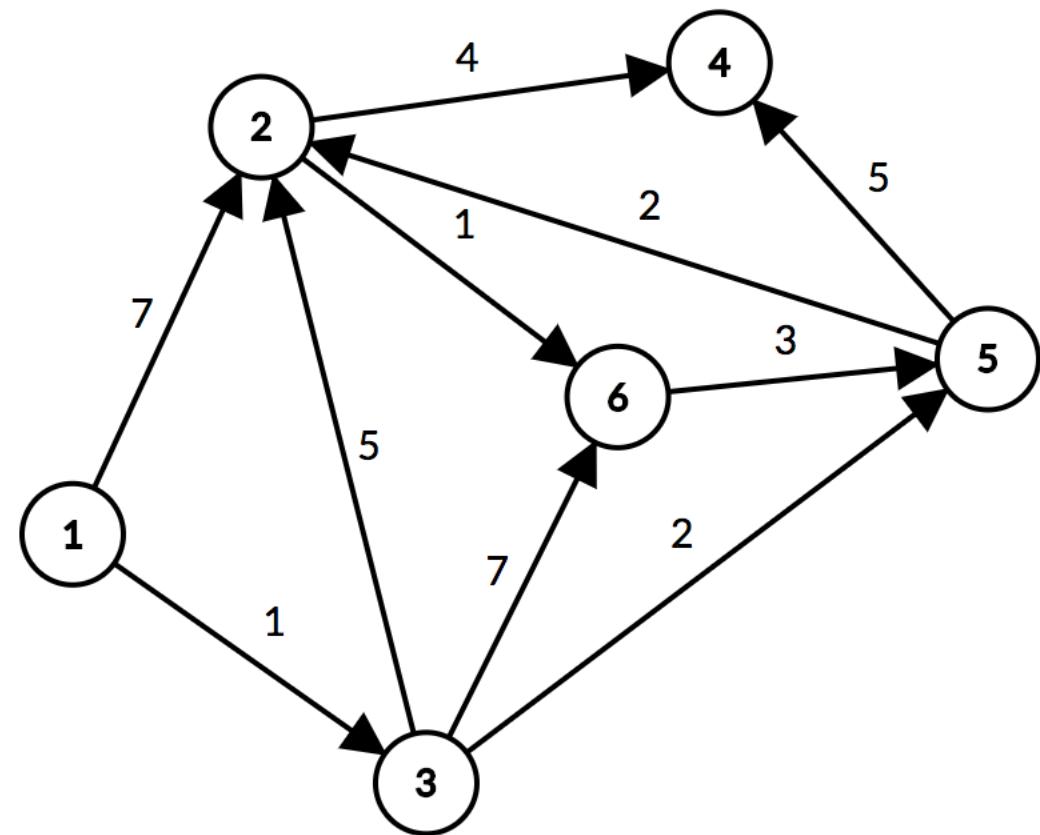
$$\pi[s][u] + L[u][v] < \pi[s][v]$$

# Giải thuật Moore-Dijkstra

```
void Dijkstra(Graph* G, int s) {
    int u, v, it;
    for (u = 1; u <= G->n; u++) {
        pi[u] = INFINITY;
        mark[u] = 0;
    }
    pi[s] = 0;
    p[s]=-1;
    for (it = 1; it < G->n; it++) {
        //1. Tìm u có mark[u] == 0 và có pi[u] nhỏ nhất
        //2. Đánh dấu u đã xét mark[u] = 1;
        //3. Cập nhật pi và p của các đỉnh kề của u (nếu thoả)
        for (v = 1; v <= G->n; v++)
            if (G->L[u][v] != NO_EDGE && mark[v] == 0) {
                if (pi[u] + G->L[u][v] < pi[v]) {
                    pi[v] = pi[u] + G->L[u][v];
                    p[v] = u;
                }
            }
    }
}
```

# Giải thuật Moore-Dijkstra

Tìm đường đi ngắn nhất từ 1 đến các đỉnh khác trên đồ thị có hướng

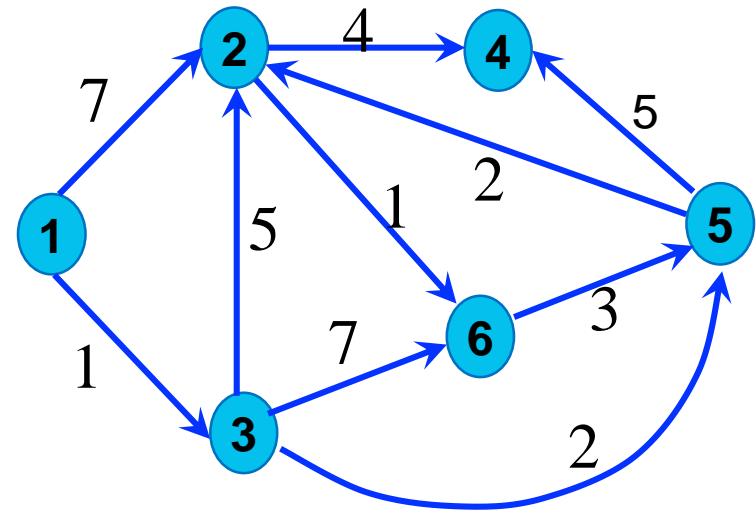


# Giải thuật Moore-Dijkstra

Tìm đường đi ngắn nhất từ A đến các đỉnh khác trên đồ thị vô hướng có ma trận trọng số.

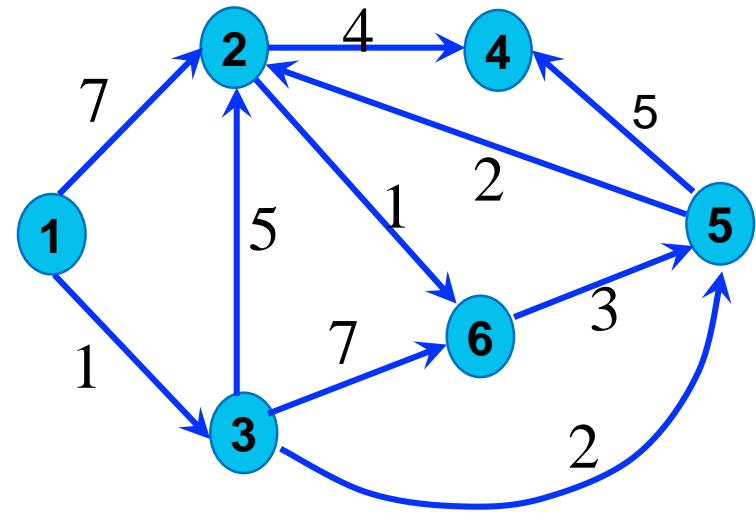
	A	B	C	D	E	F	G	H	K
A		1	1						4
B							9	2	3
C				7					2
D					1	5		4	3
E						2			
F							9	3	
G								5	
H									7
K									

# Moore-Dijkstra



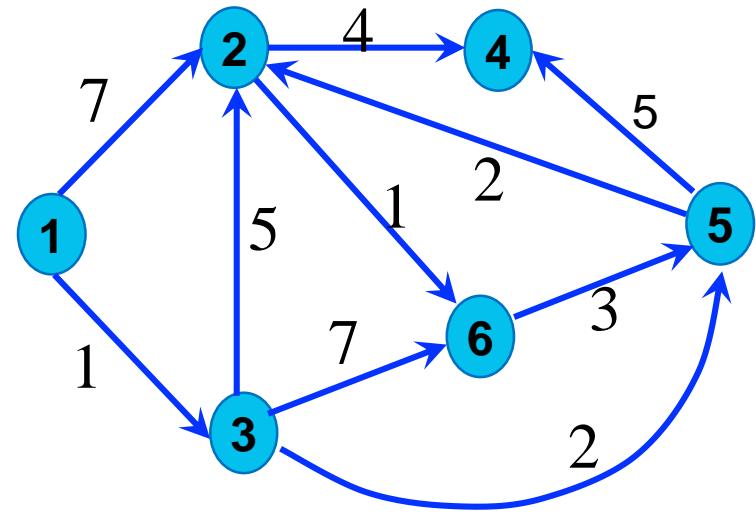
<b>G</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
<b>Mark[u]</b>	0	0	0	0	0	0
<b>Neighbors(u)</b>	2,3	4,6	2,5,6	∅	2,4	5
<b>pi(u)</b>	∞	∞	∞	∞	∞	∞
<b>p(i)</b>						

# Moore-Dijkstra



<b>G</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
<b>Mark[u]</b>	0	0	0	0	0	0
<b>Neigbors(u)</b>	2,3	4,6	2,5,6	∅	2,4	5
<b>pi(u)</b>	0	∞	∞	∞	∞	∞
<b>p(i)</b>						

# Moore-Dijkstra



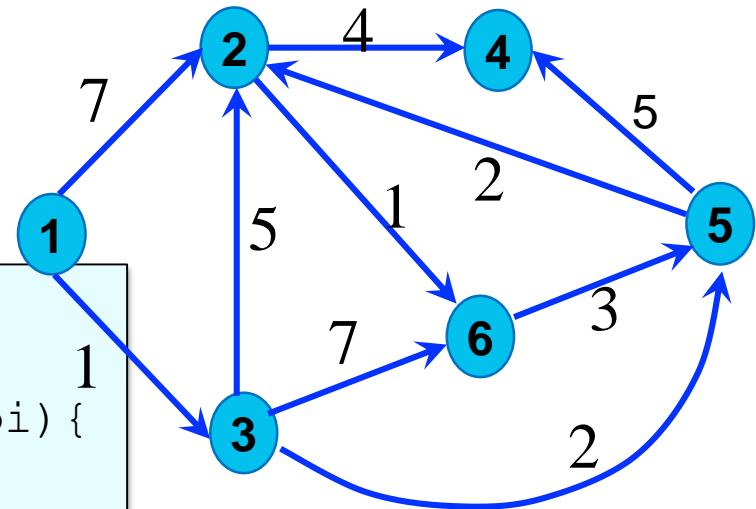
<b>G</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
<b>Mark[u]</b>	0	0	0	0	0	0
<b>Neigbors(u)</b>	2,3	4,6	2,5,6	∅	2,4	5
<b>pi(u)</b>	0	∞	∞	∞	∞	∞
<b>p(i)</b>	-1					

# Moore-Dijkstra

```

int min_pi = INFINITY;
for (j = 1; j <= pG->n; j++)
    if (mark[j] == 0 && pi[j] < min_pi) {
        min_pi = pi[j];
        u = j;
    }
}

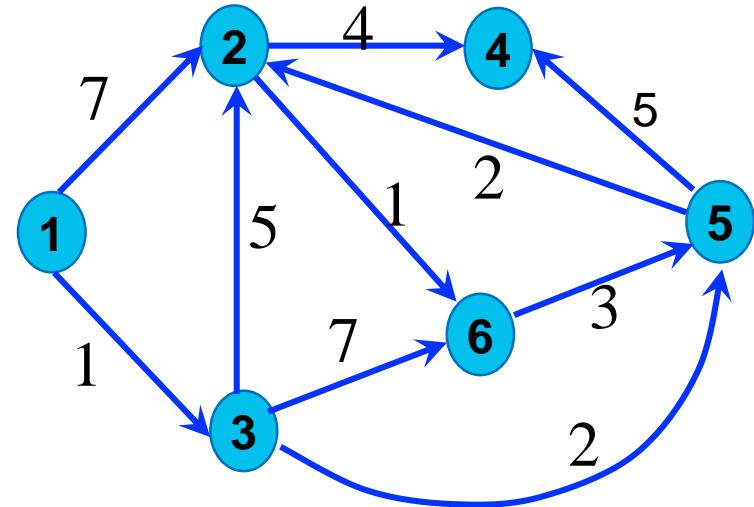
```



G	1	2	3	4	5	6
Mark[u]	1	0	0	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
p(i)	-1					

# Moore-Dijkstra

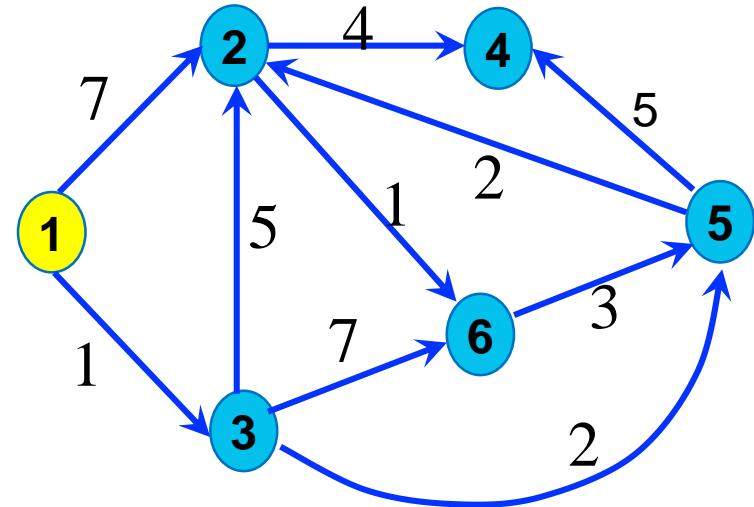
min\_pi =  
pi[1]



G	1	2	3	4	5	6
Mark[u]	0	0	0	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
p(i)	-1					

# Moore-Dijkstra

min\_pi =  
pi[1]



G	1	2	3	4	5	6
Mark[u]	1	0	0	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
p(i)	-1					

# Moore-Dijkstra

```

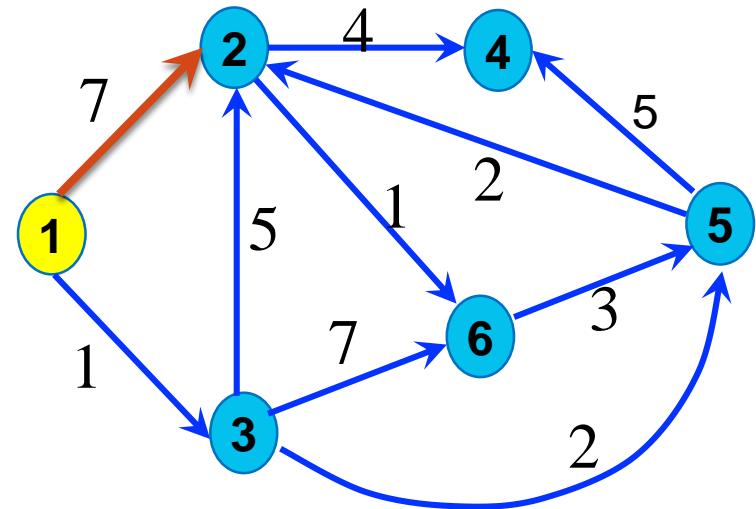
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[1] + G->L[1][2] < pi[2]) { //true
    pi[2] = pi[1] + G->L[1][2] = 0+7 = 7;
    p[2] = 1;
}

```



G	1	2	3	4	5	6
Mark[u]	1	0	0	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
p(i)	-1					

# Moore-Dijkstra

```

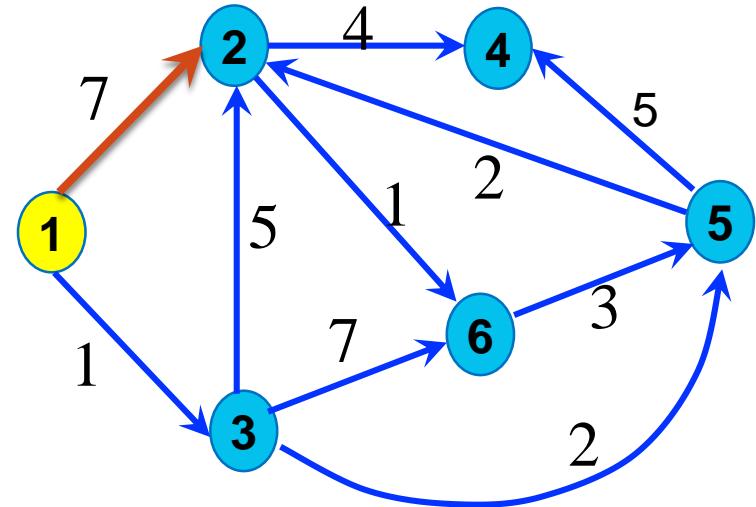
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[1] + G->L[1][2] < pi[2]) { //true
    pi[2] = pi[1] + G->L[1][2] = 0+7 = 7;
    p[2] = 1;
}

```



G	1	2	3	4	5	6
Mark[u]	1	0	0	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	7	$\infty$	$\infty$	$\infty$	$\infty$
p(i)	-1	1				

# Moore-Dijkstra

```

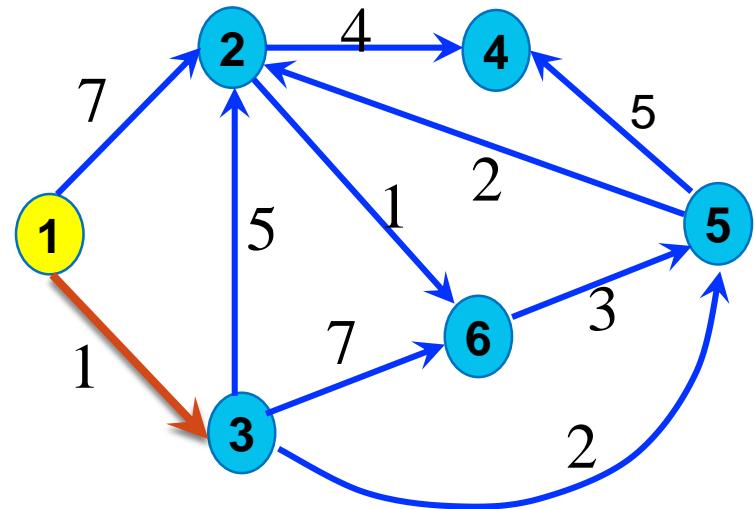
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[1] + G->L[1][3] < pi[3]) { //true
    pi[3] = pi[1] + G->L[1][3] = 0+1 = 1;
    p[3] = 1;
}

```



G	1	2	3	4	5	6
Mark[u]	1	0	0	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	7	$\infty$	$\infty$	$\infty$	$\infty$
p(i)	-1	1				

# Moore-Dijkstra

```

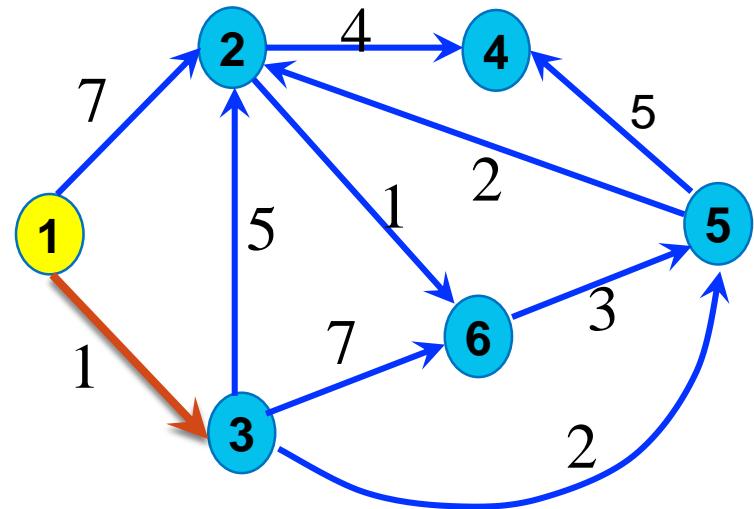
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[1] + G->L[1][3] < pi[3]) { //true
    pi[3] = pi[1] + G->L[1][3] = 0+1 = 1;
    p[3] = 1;
}

```



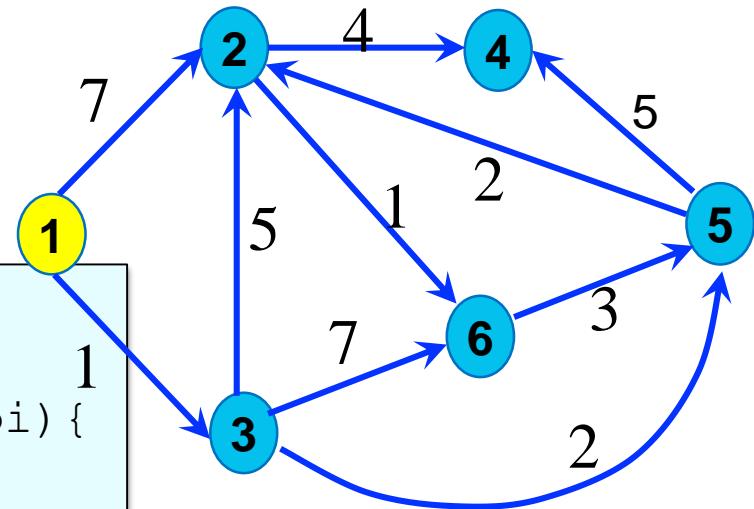
G	1	2	3	4	5	6
Mark[u]	1	0	0	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	7	1	$\infty$	$\infty$	$\infty$
p(i)	-1	1	1			

# Moore-Dijkstra

```

int min_pi = INFINITY;
for (j = 1; j <= pG->n; j++)
    if (mark[j] == 0 && pi[j] < min_pi) {
        min_pi = pi[j];
        u = j;
    }
}

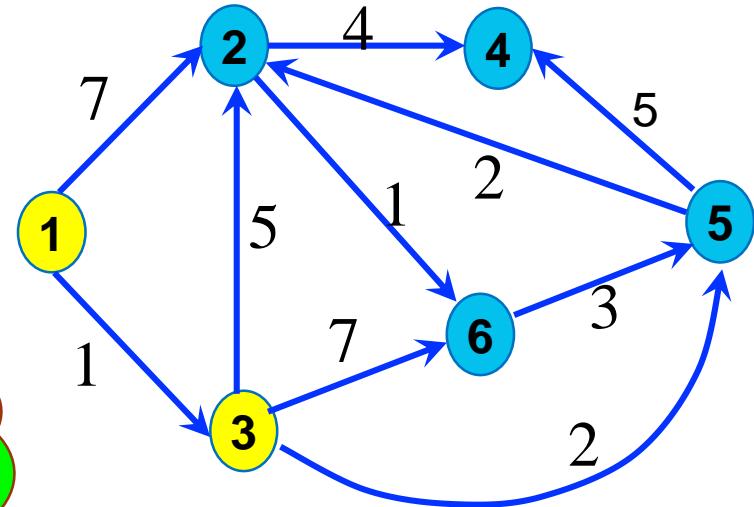
```



G	1	2	3	4	5	6
Mark[u]	1	0	0	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	7	1	$\infty$	$\infty$	$\infty$
p(i)	-1	1	1			

# Moore-Dijkstra

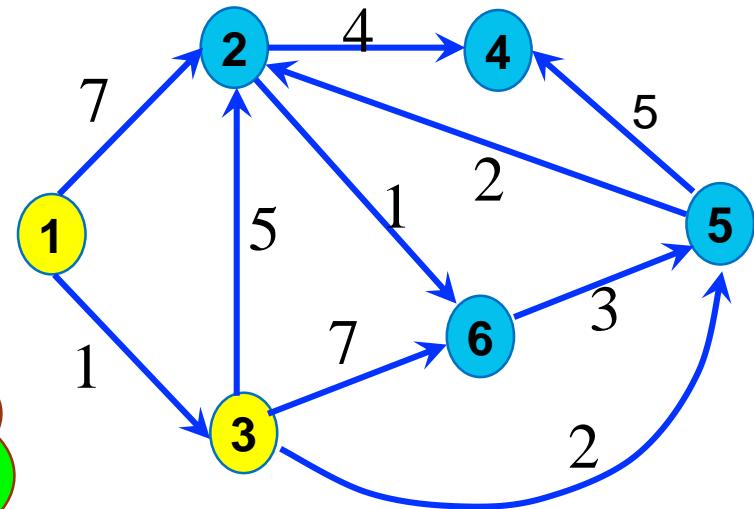
min\_pi =  
pi[3]



G	1	2	3	4	5	6
Mark[u]	1	0	0	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	7	1	$\infty$	$\infty$	$\infty$
p(i)	-1	1	1			

# Moore-Dijkstra

min\_pi =  
pi[3]



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	7	1	$\infty$	$\infty$	$\infty$
p(i)	-1	1	1			

# Moore-Dijkstra

```

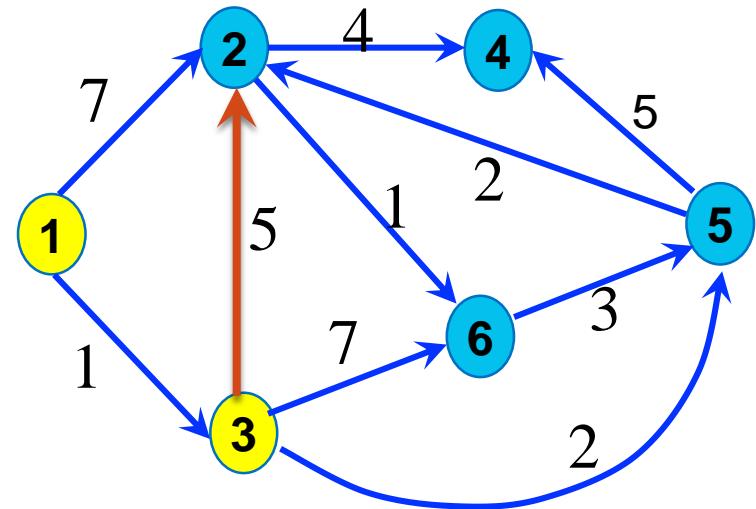
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[3] + G->L[3][2] < pi[2]) { //true
    pi[2] = pi[3] + G->L[3][2] = 1+5 = 6;
    p[2] = 3;
}

```



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	7	1	$\infty$	$\infty$	$\infty$
p(i)	-1	1	1			

# Moore-Dijkstra

```

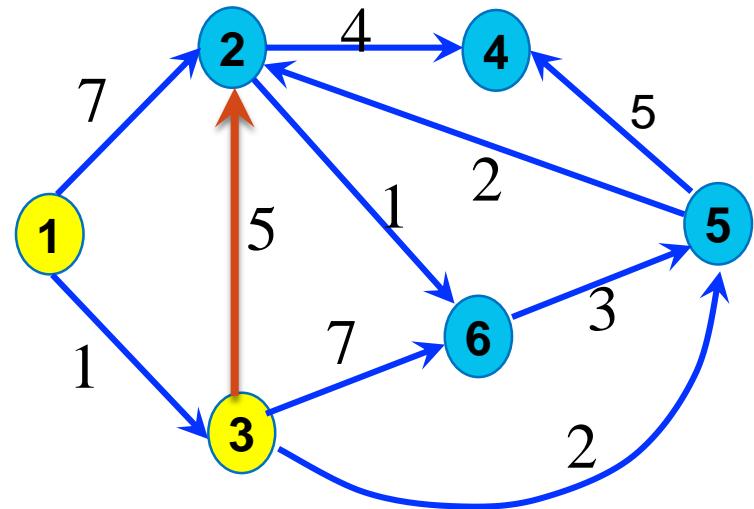
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[3] + G->L[3][2] < pi[2]) { //true
    pi[2] = pi[3] + G->L[3][2] = 1+5 = 6;
    p[2] = 3;
}

```



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	6	1	$\infty$	$\infty$	$\infty$
p(i)	-1	3	1			

# Moore-Dijkstra

```

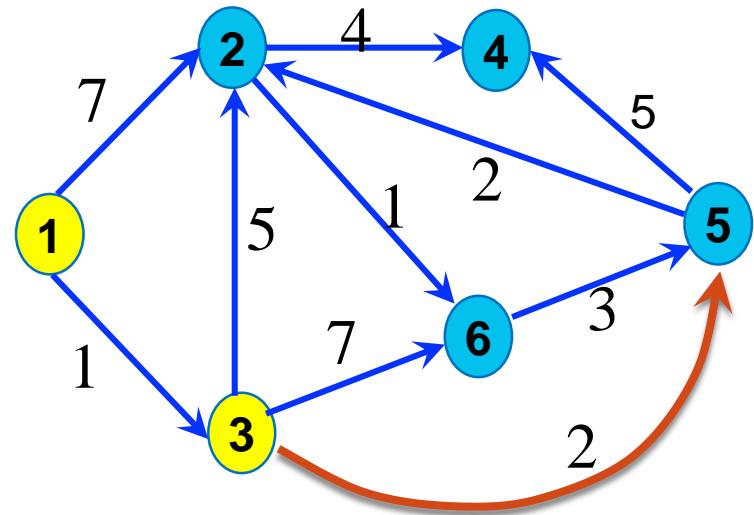
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[3] + G->L[3][5] < pi[5]) { //true
    pi[5] = pi[3] + G->L[3][5] = 1+2 = 3;
    p[5] = 3;
}

```



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	6	1	$\infty$	$\infty$	$\infty$
p(i)	-1	3	1			

# Moore-Dijkstra

```

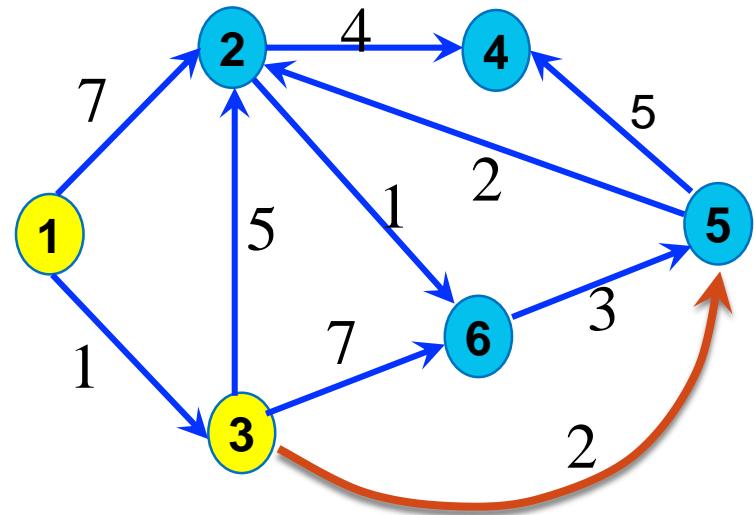
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[3] + G->L[3][5] < pi[5]) { //true
    pi[5] = pi[3] + G->L[3][5] = 1+2 = 3;
    p[5] = 3;
}

```



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	6	1	$\infty$	3	$\infty$
p(i)	-1	3	1		3	

# Moore-Dijkstra

```

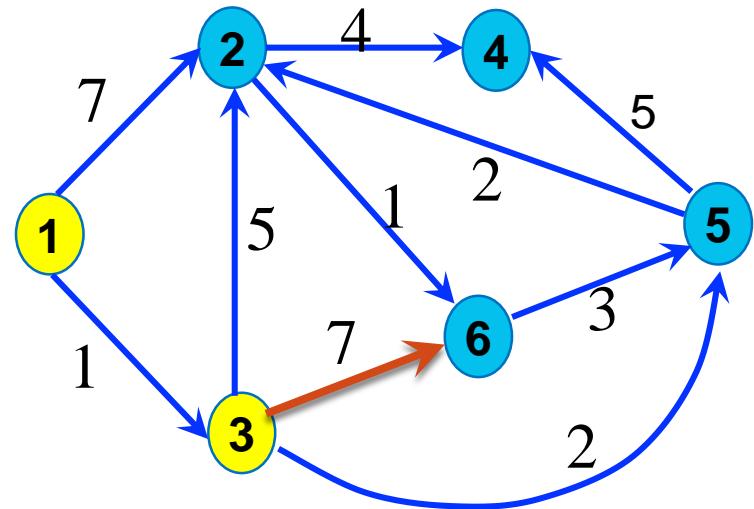
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[3] + G->L[3][6] < pi[6]) { //true
    pi[6] = pi[3] + G->L[3][6] = 1+7 = 8;
    p[6] = 3;
}

```



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	6	1	$\infty$	3	$\infty$
p(i)	-1	3	1		3	

# Moore-Dijkstra

```

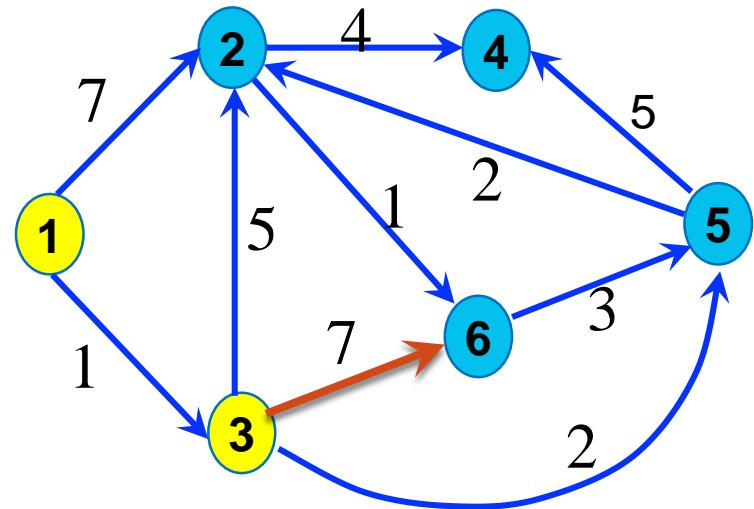
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[3] + G->L[3][6] < pi[6]) { //true
    pi[6] = pi[3] + G->L[3][6] = 1+7 = 8;
    p[6] = 3;
}

```



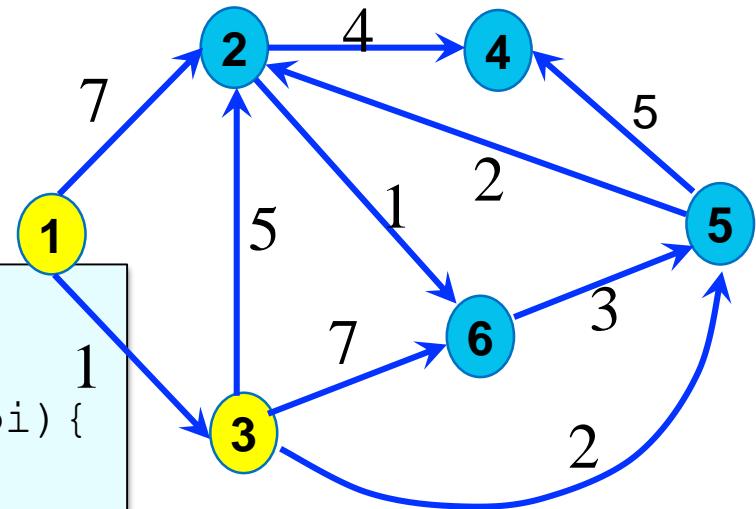
G	1	2	3	4	5	6
Mark[u]	1	0	1	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	6	1	$\infty$	3	8
p(i)	-1	3	1		3	3

# Moore-Dijkstra

```

int min_pi = INFINITY;
for (j = 1; j <= pG->n; j++)
    if (mark[j] == 0 && pi[j] < min_pi) {
        min_pi = pi[j];
        u = j;
    }
}

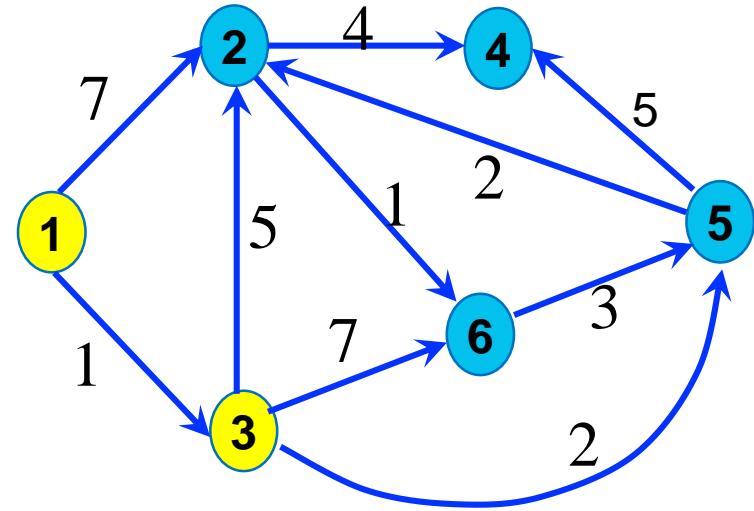
```



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	6	1	$\infty$	3	8
p(i)	-1	3	1		3	3

# Moore-Dijkstra

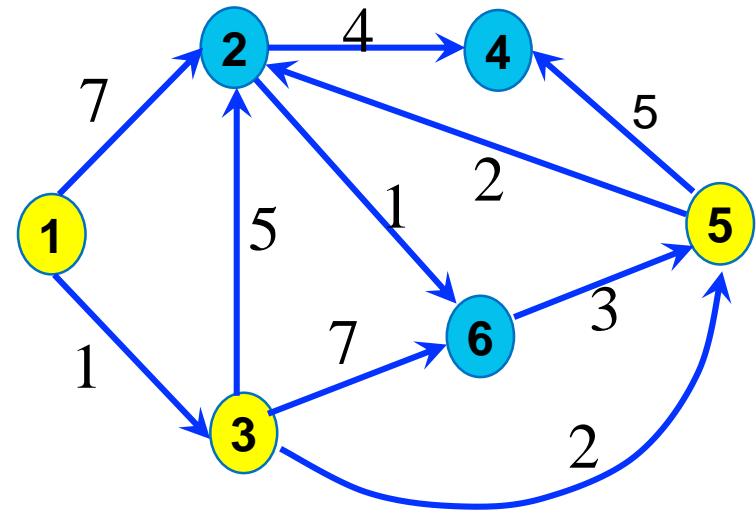
min\_pi =  
pi[5]



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	0	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	6	1	$\infty$	3	8
p(i)	-1	3	1		3	3

# Moore-Dijkstra

min\_pi =  
pi[5]



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	1	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	6	1	$\infty$	3	8
p(i)	-1	3	1		3	3

# Moore-Dijkstra

```

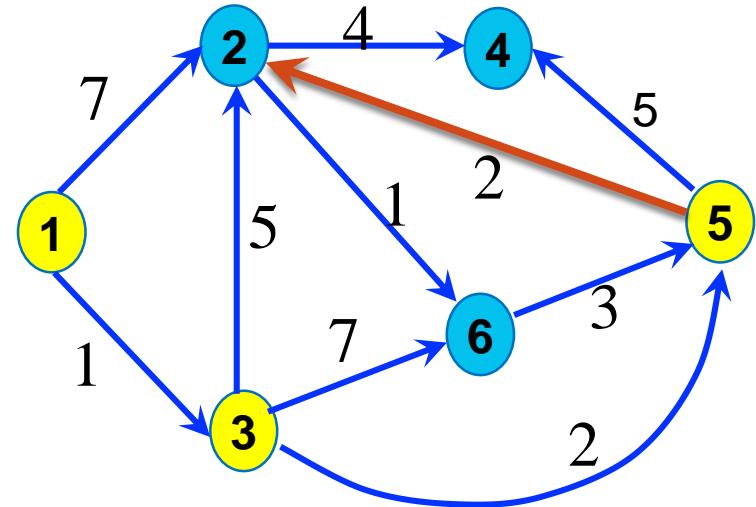
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[5] + G->L[5][2] < pi[2]) { //true
    pi[2] = pi[5] + G->L[5][2] = 3+2 = 5;
    p[2] = 5;
}

```



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	1	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	6	1	$\infty$	3	8
p(i)	-1	3	1		3	3

# Moore-Dijkstra

```

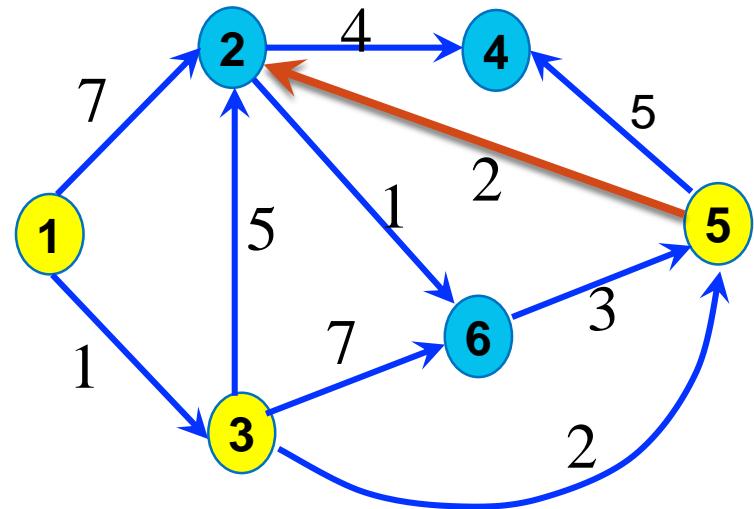
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[5] + G->L[5][2] < pi[2]) { //true
    pi[2] = pi[5] + G->L[5][2] = 3+2 = 5;
    p[2] = 5;
}

```



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	1	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	$\infty$	3	8
p(i)	-1	5	1		3	3

# Moore-Dijkstra

```

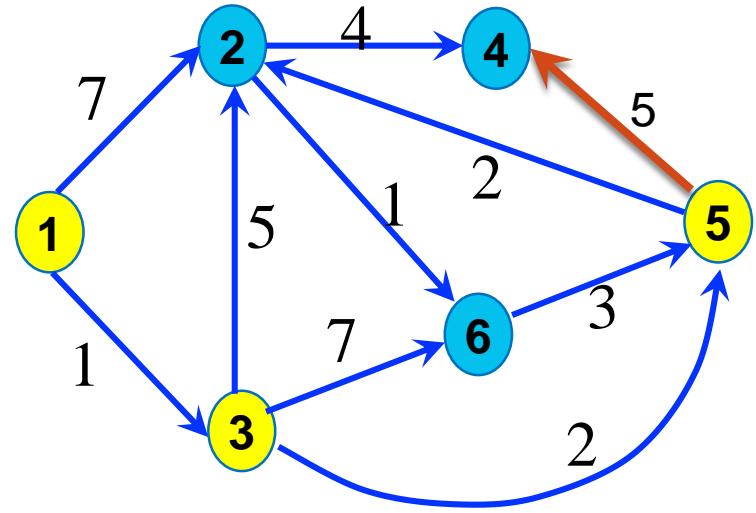
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[5] + G->L[5][4] < pi[4]) { //true
    pi[4] = pi[5] + G->L[5][4] = 3+5 = 8;
    p[4] = 5;
}

```



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	1	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	$\infty$	3	8
p(i)	-1	5	1		3	3

# Moore-Dijkstra

```

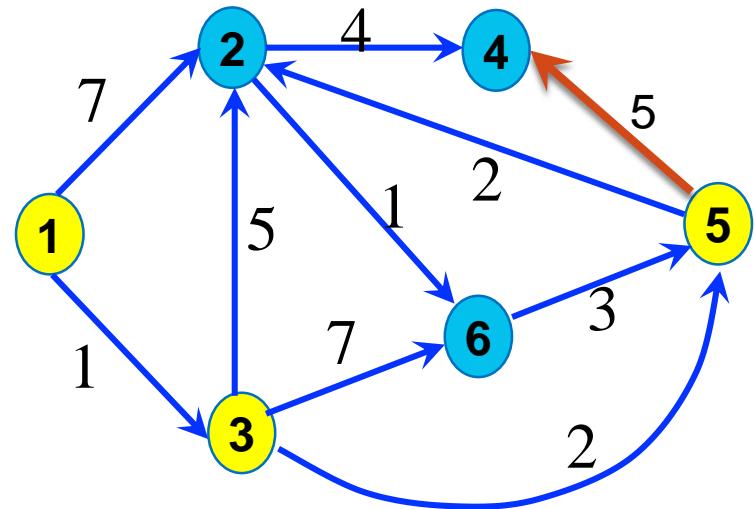
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[5] + G->L[5][4] < pi[4]) { //true
    pi[4] = pi[5] + G->L[5][4] = 3+5 = 8;
    p[4] = 5;
}

```



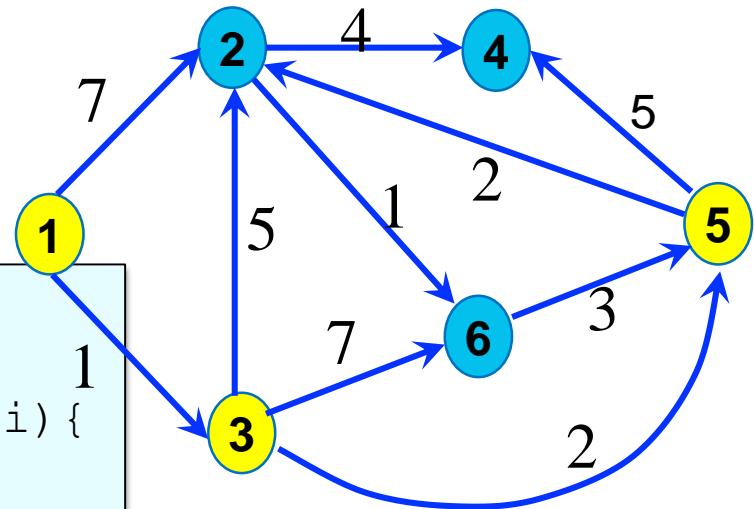
G	1	2	3	4	5	6
Mark[u]	1	0	1	0	1	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	8	3	8
p(i)	-1	5	1	5	3	3

# Moore-Dijkstra

```

int min_pi = INFINITY;
for (j = 1; j <= pG->n; j++)
    if (mark[j] == 0 && pi[j] < min_pi) {
        min_pi = pi[j];
        u = j;
    }
}

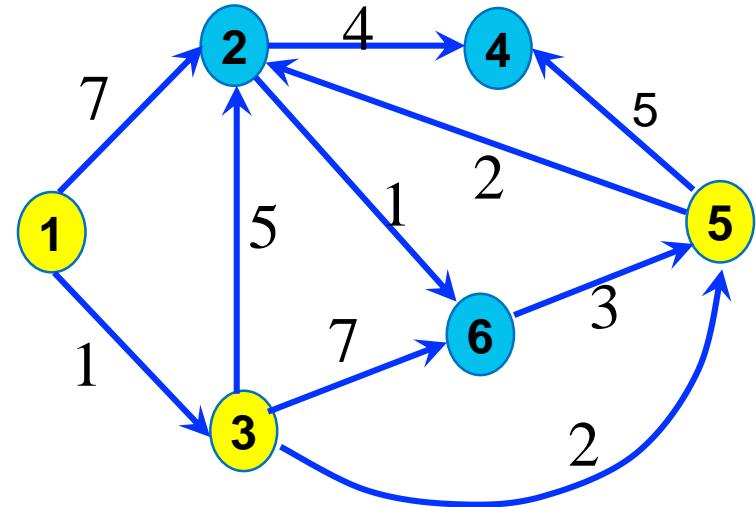
```



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	1	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	8	3	8
p(i)	-1	5	1	5	3	3

# Moore-Dijkstra

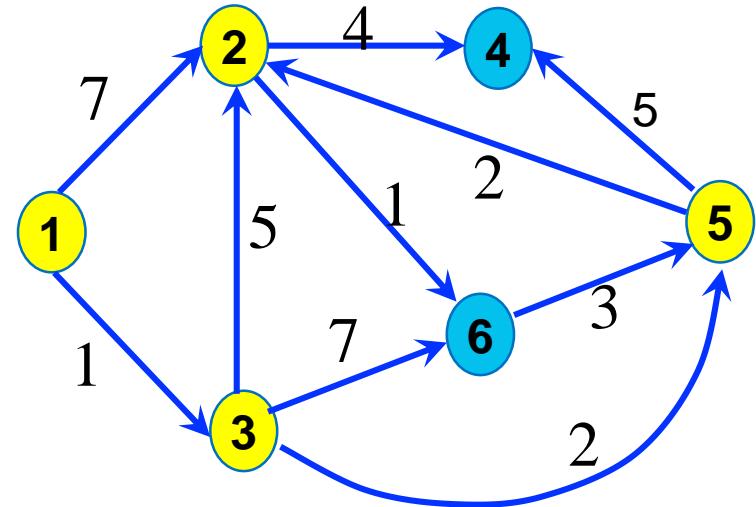
min\_pi =  
pi[2]



G	1	2	3	4	5	6
Mark[u]	1	0	1	0	1	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	8	3	8
p(i)	-1	5	1	5	3	3

# Moore-Dijkstra

min\_pi =  
pi[2]



G	1	2	3	4	5	6
Mark[u]	1	1	1	0	1	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	8	3	8
p(i)	-1	5	1	5	3	3

# Moore-Dijkstra

```

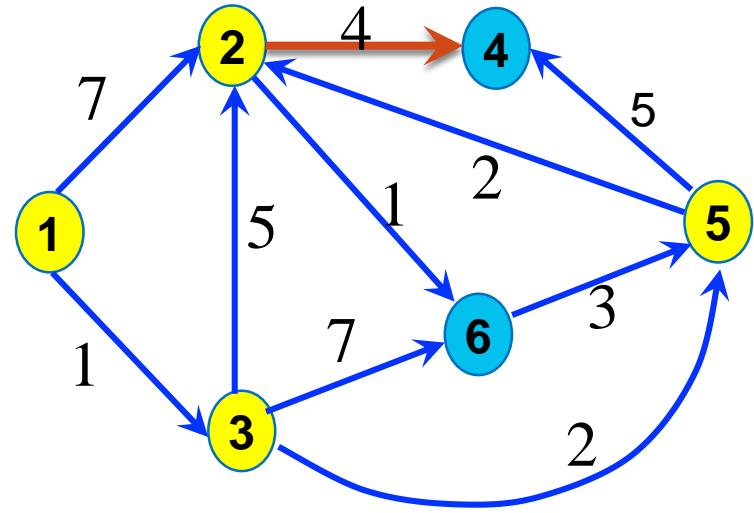
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[2] + G->L[2][4] < pi[4]) { //false
    //vì 5 + 4 < 8;
}

```



G	1	2	3	4	5	6
Mark[u]	1	1	1	0	1	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	8	3	8
p(i)	-1	5	1	5	3	3

# Moore-Dijkstra

```

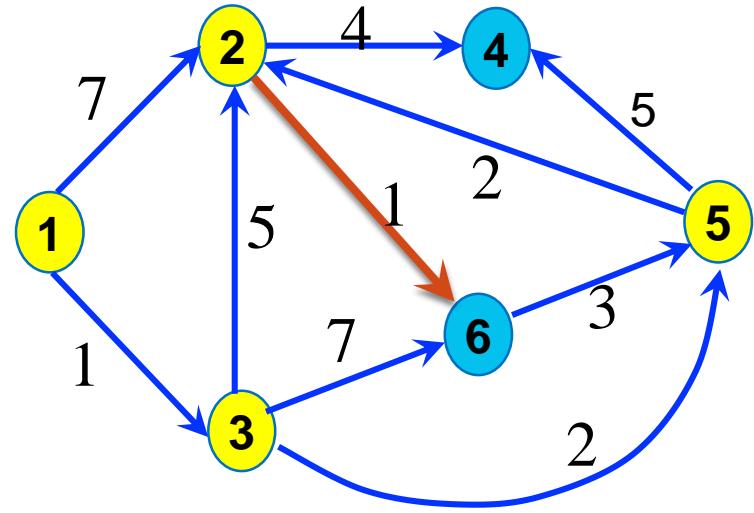
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[2] + G->L[2][6] < pi[6]) { //true
    pi[6] = pi[2] + G->L[2][6] = 5+1 = 6;
    p[6] = 2;
}

```



G	1	2	3	4	5	6
Mark[u]	1	1	1	0	1	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	8	3	8
p(i)	-1	5	1	5	3	3

# Moore-Dijkstra

```

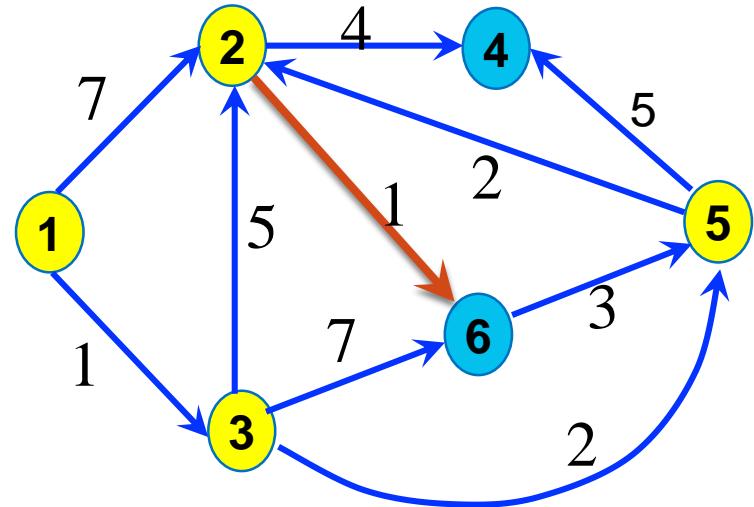
if (pi[u] + G->L[u][v] < pi[v]) {
    pi[v] = pi[u] + G->L[u][v];
    p[v] = u;
}

```

```

if (pi[2] + G->L[2][6] < pi[6]) { //true
    pi[6] = pi[2] + G->L[2][6] = 5+1 = 6;
    p[6] = 2;
}

```



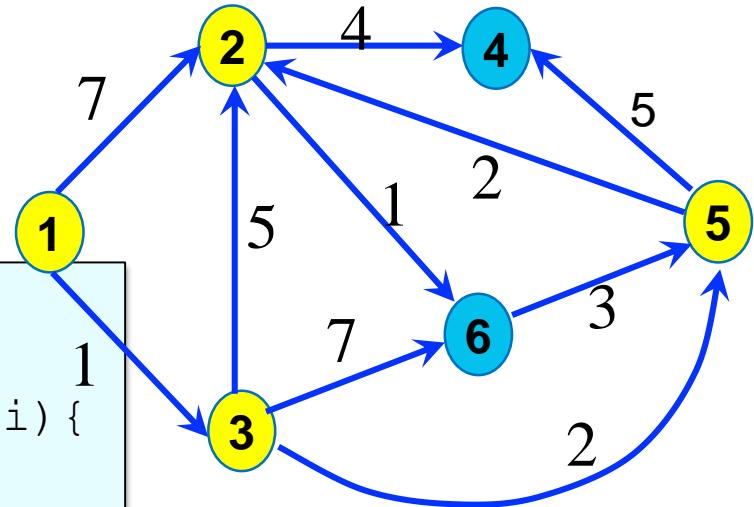
G	1	2	3	4	5	6
Mark[u]	1	1	1	0	1	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	8	3	6
p(i)	-1	5	1	5	3	2

# Moore-Dijkstra

```

int min_pi = INFINITY;
for (j = 1; j <= pG->n; j++)
    if (mark[j] == 0 && pi[j] < min_pi) {
        min_pi = pi[j];
        u = j;
    }
}

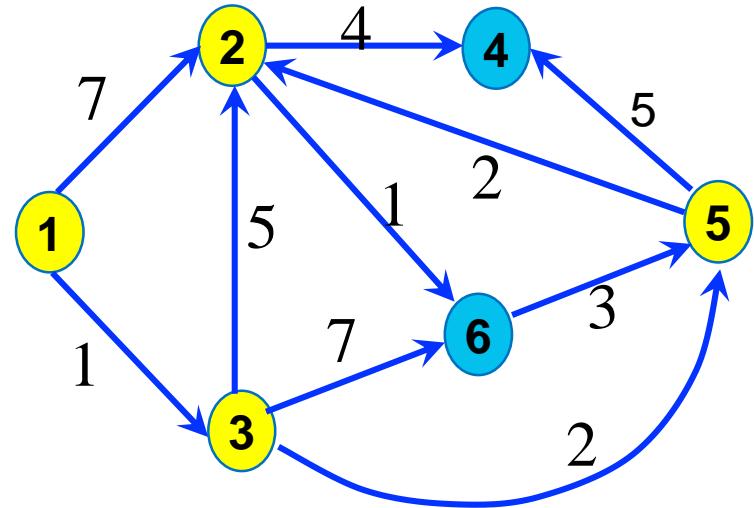
```



G	1	2	3	4	5	6
Mark[u]	1	1	1	0	1	0
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	8	3	6
p(i)	-1	5	1	5	3	2

# Moore-Dijkstra

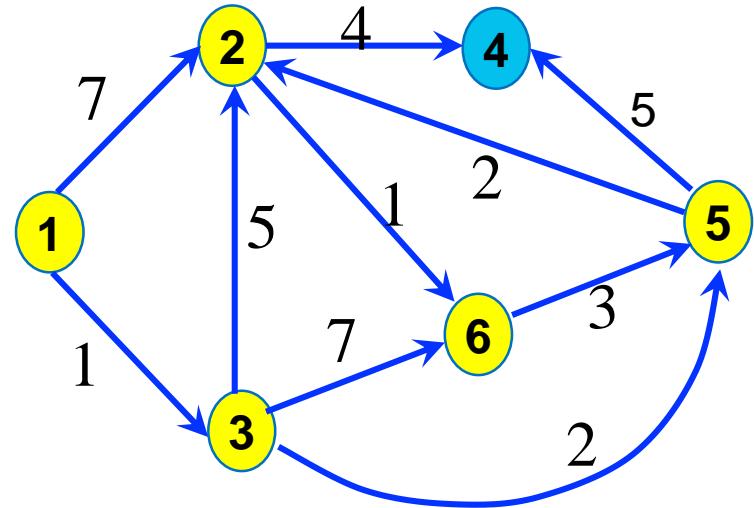
min\_pi =  
pi[6]



G	1	2	3	4	5	6
Mark[u]	1	1	1	0	1	0
Neighbors(u)	2,3	4,6	2,5,6	∅	2,4	5
pi(u)	0	5	1	8	3	6
p(i)	-1	5	1	5	3	2

# Moore-Dijkstra

min\_pi =  
pi[6]

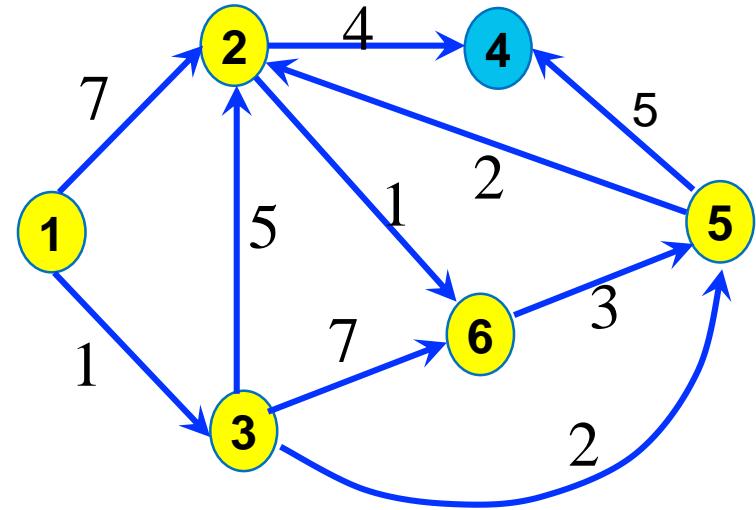


G	1	2	3	4	5	6
Mark[u]	1	1	1	0	1	1
Neighbors(u)	2,3	4,6	2,5,6	∅	2,4	5
pi(u)	0	5	1	8	3	6
p(i)	-1	5	1	5	3	2

# Moore-Dijkstra

`mark[5] ==1`

//Nên không làm gì cả



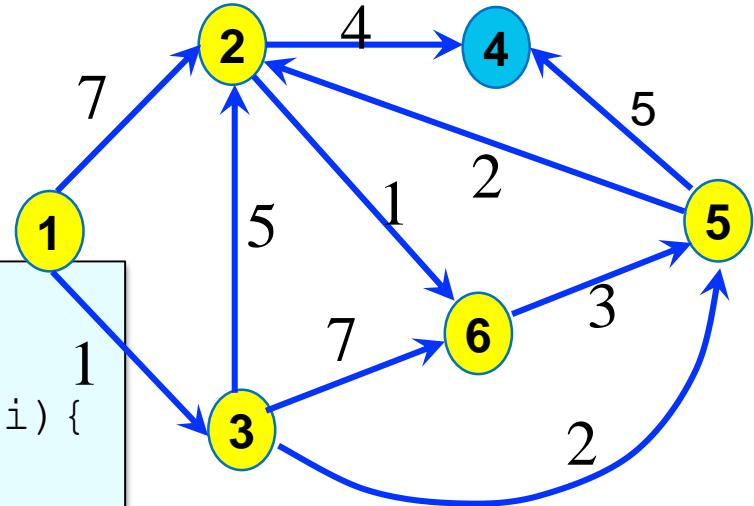
G	1	2	3	4	5	6
Mark[u]	1	1	1	0	1	1
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
$\pi(u)$	0	5	1	8	3	6
p(i)	-1	5	1	5	3	2

# Moore-Dijkstra

```

int min_pi = INFINITY;
for (j = 1; j <= pG->n; j++)
    if (mark[j] == 0 && pi[j] < min_pi) {
        min_pi = pi[j];
        u = j;
    }
}

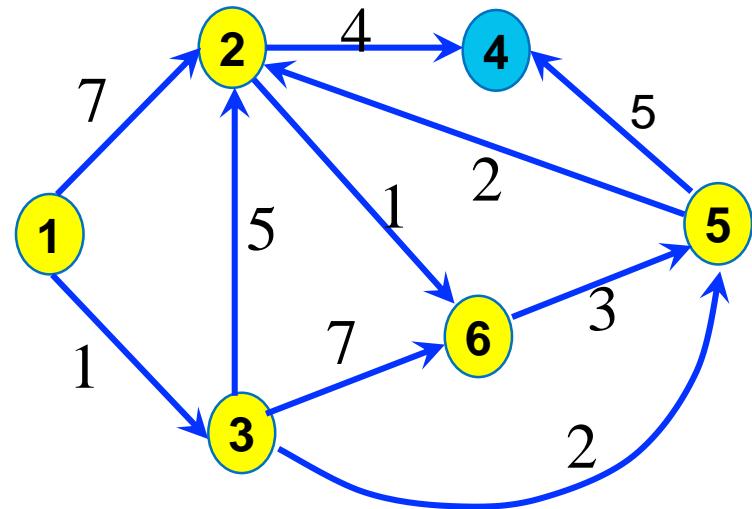
```



G	1	2	3	4	5	6
Mark[u]	1	1	1	0	1	1
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	8	3	6
p(i)	-1	5	1	5	3	2

# Moore-Dijkstra

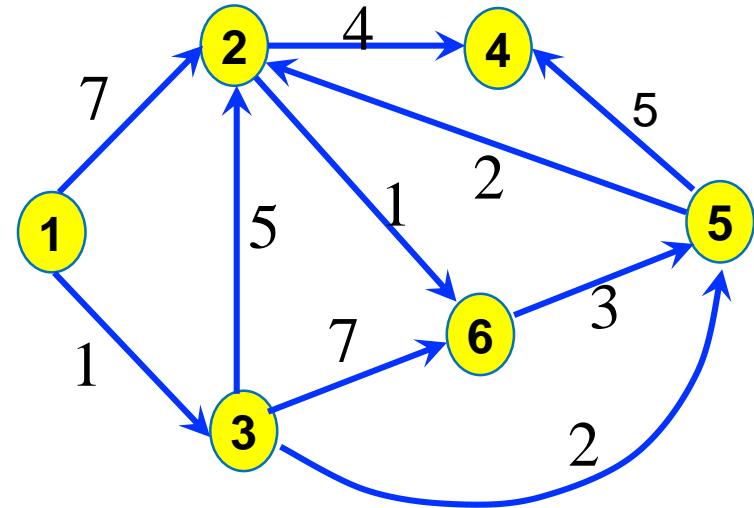
min\_pi =  
pi[4]



G	1	2	3	4	5	6
Mark[u]	1	1	1	0	1	1
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	8	3	6
p(i)	-1	5	1	5	3	2

# Moore-Dijkstra

min\_pi =  
pi[4]



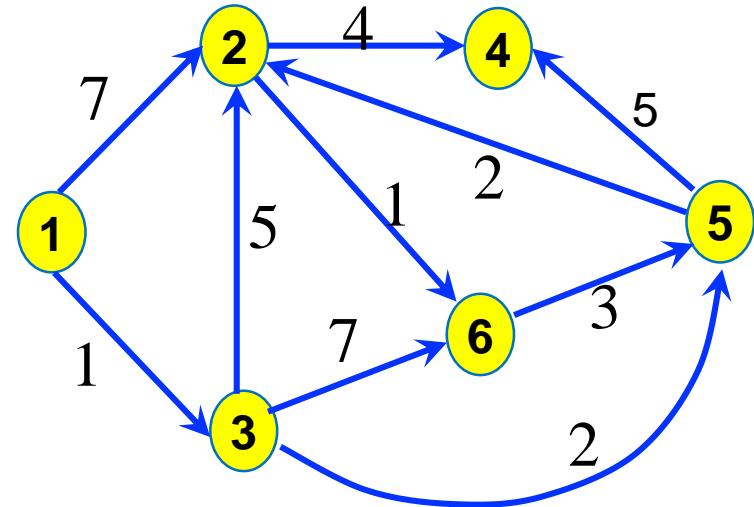
G	1	2	3	4	5	6
Mark[u]	1	1	1	1	1	1
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	8	3	6
p(i)	-1	5	1	5	3	2

# Moore-Dijkstra

$\text{Neighbors}(4) = \emptyset$

$\text{Mark}[u]=1$  mọi đỉnh u

Giải thuật dừng



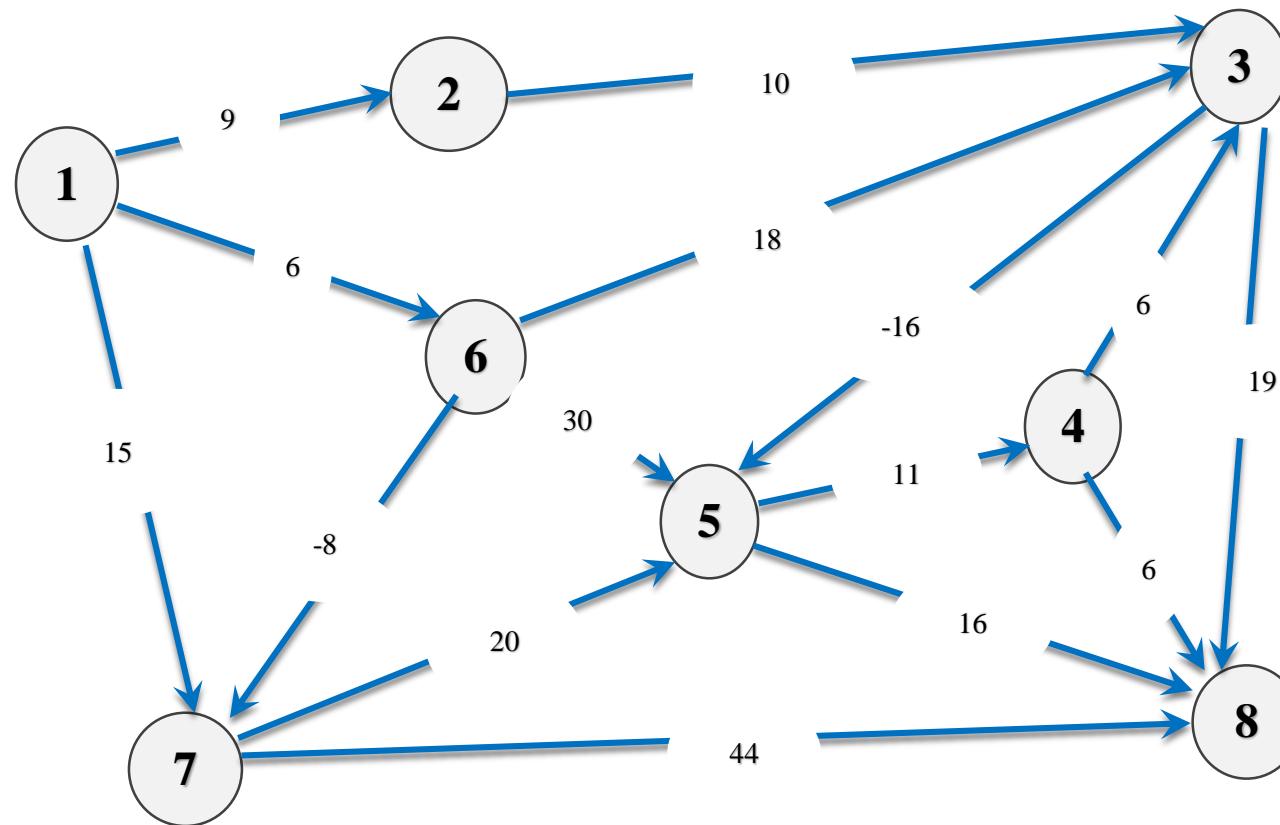
G	1	2	3	4	5	6
Mark[u]	1	1	1	1	1	1
Neighbors(u)	2,3	4,6	2,5,6	$\emptyset$	2,4	5
pi(u)	0	5	1	8	3	6
p(i)	-1	5	1	5	3	2

# Giải thuật Bellman – Ford

```
void BellmanFord(Graph* pG, int s) {  
    int u, v, w, it, k;  
    for (u = 1; u <= pG->n; u++) {  
        pi[u] = INFINITY;  
    }  
    pi[s] = 0;  
    p[s] = -1; //trước đỉnh s không có đỉnh nào cả  
  
    // lặp n-1 lần  
    for (it = 1; it < G->n; it++) {  
        // Duyệt qua các cung và cập nhật (nếu thoả)  
        for (k = 0; k < G->m; k++) {  
            u = G->edges[k].u;  
            v = G->edges[k].v;  
            w = G->edges[k].w;  
            if (pi[u] + w < pi[v]) {  
                pi[v] = pi[u] + w;  
                p[v] = u;  
            }  
        }  
    }  
    //Kiểm tra chu trình âm (nếu cần thiết)  
}
```

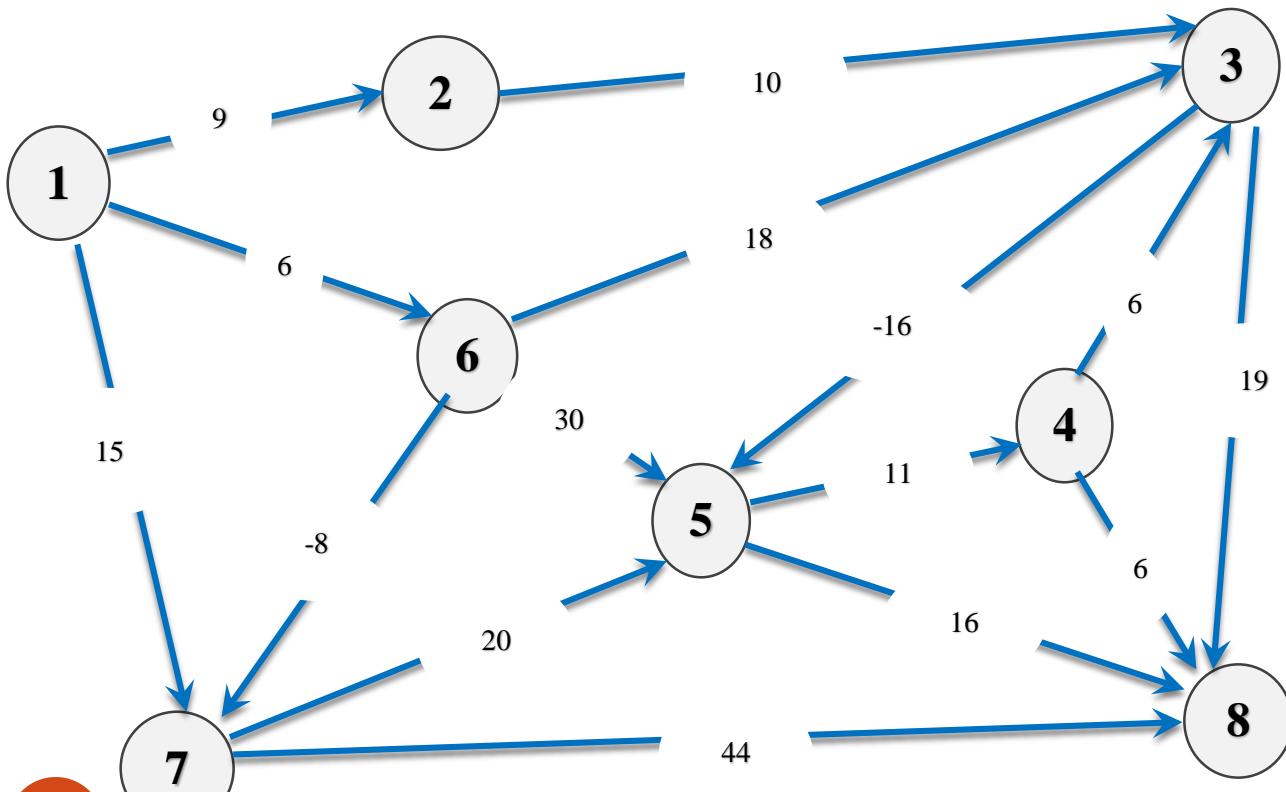
# Bellman – Ford

Tìm đường đi ngắn nhất từ đỉnh 1 đến đỉnh 8 bằng thuật toán Bellman-Ford

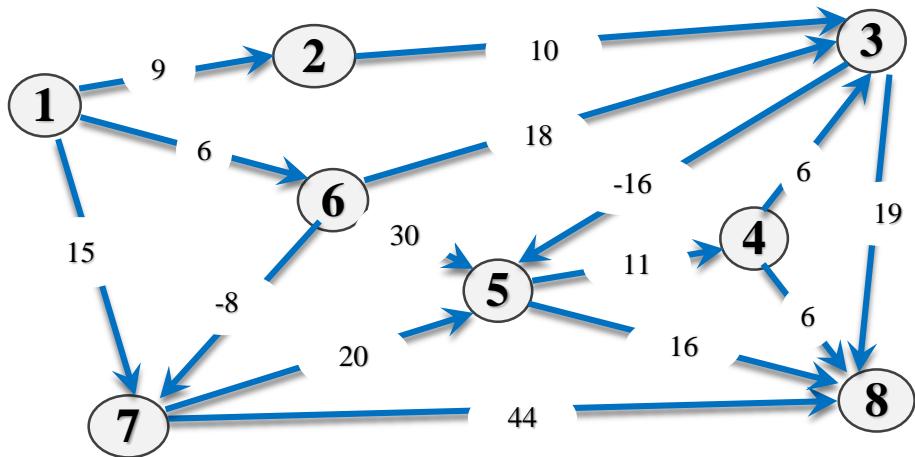


# Bellman – Ford

Tìm đường đi ngắn nhất từ đỉnh 1 đến đỉnh 8 bằng thuật toán Bellman-Ford



u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



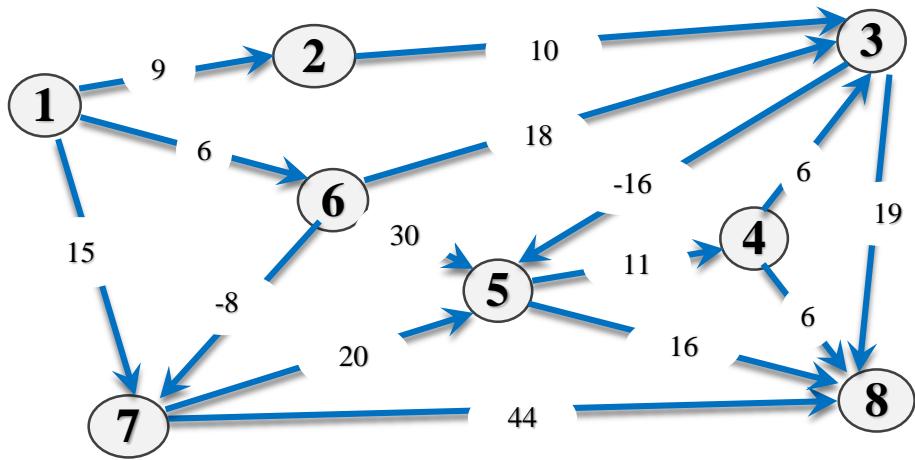
```

void BellmanFord(Graph* pG, int s) {
    int u, v, w, it, k;
    for (u = 1; u <= pG->n; u++) {
        pi[u] = INFINITY;
    }
    pi[s] = 0;
    p[s] = -1;
    .....
}

```

G	1	2	3	4	5	6	7	8
pi(u)	$\infty$							
p(i)								

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



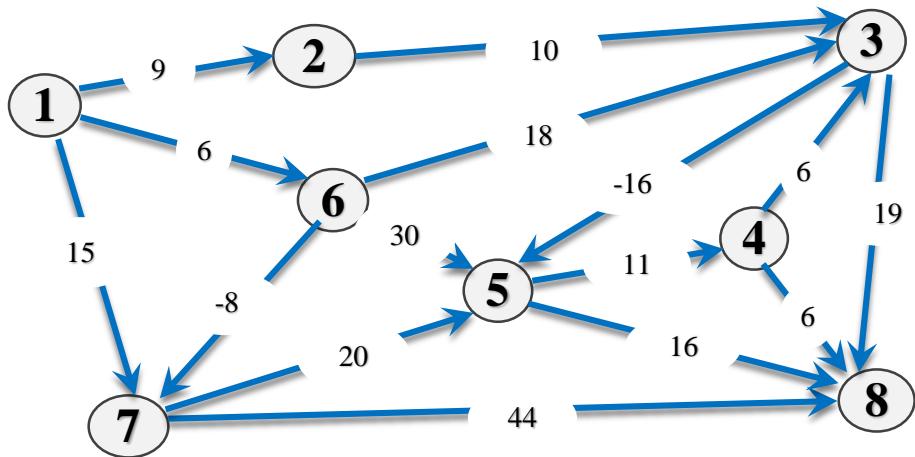
```

void BellmanFord(Graph* pG, int s) {
    int u, v, w, it, k;
    for (u = 1; u <= pG->n; u++) {
        pi[u] = INFINITY;
    }
    pi[s] = 0;
    p[s] = -1;
    .....
}

```

G	1	2	3	4	5	6	7	8
pi(u)	$\infty$							
p(i)								

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



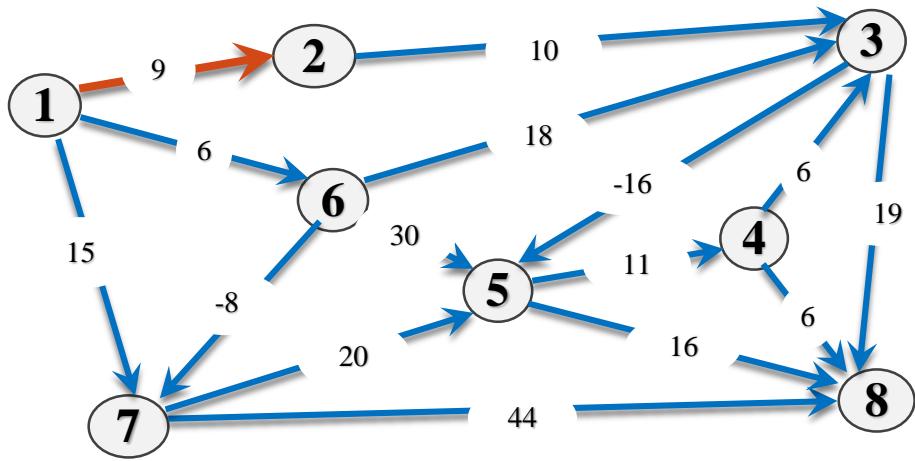
```

void BellmanFord(Graph* pG, int s) {
    int u, v, w, it, k;
    for (u = 1; u <= pG->n; u++) {
        pi[u] = INFINITY;
    }
    pi[s] = 0;
    p[s] = -1;
    .....
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	$\infty$						
p(i)	-1							

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19

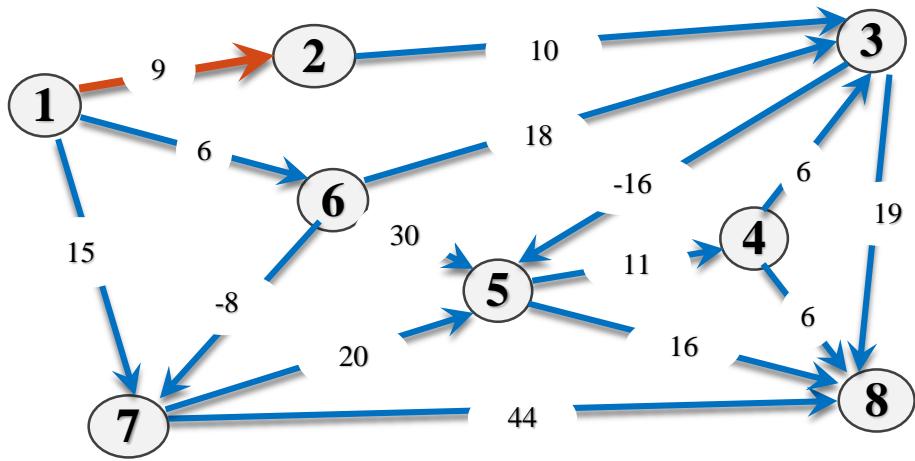


```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=0
        u = G->edges[0].u = 1;
        v = G->edges[0].v = 2;
        w = G->edges[0].w = 9;
        if (pi[u] + w < pi[v]) {
            pi[v] = pi[u] + w;
            p[v] = u;
        }
    }
}
    
```

G	1	2	3	4	5	6	7	8
pi(u)	0	$\infty$						
p(i)	-1							

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



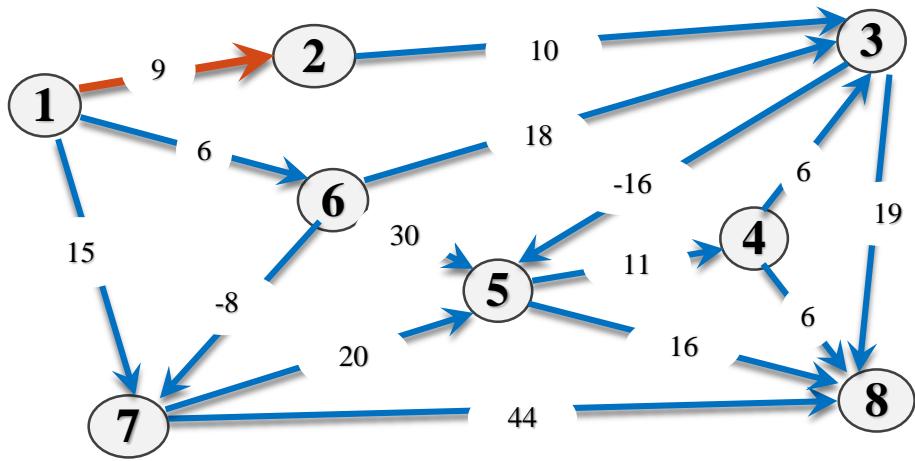
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=0
        u = G->edges[0].u = 1;
        v = G->edges[0].v = 2;
        w = G->edges[0].w = 9;
        if (pi[1] + 9 < pi[2]) { //true
            pi[2] = pi[1] + 9 = 0+9=9;
            p[2] = 1;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	$\infty$						
p(i)	-1							

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



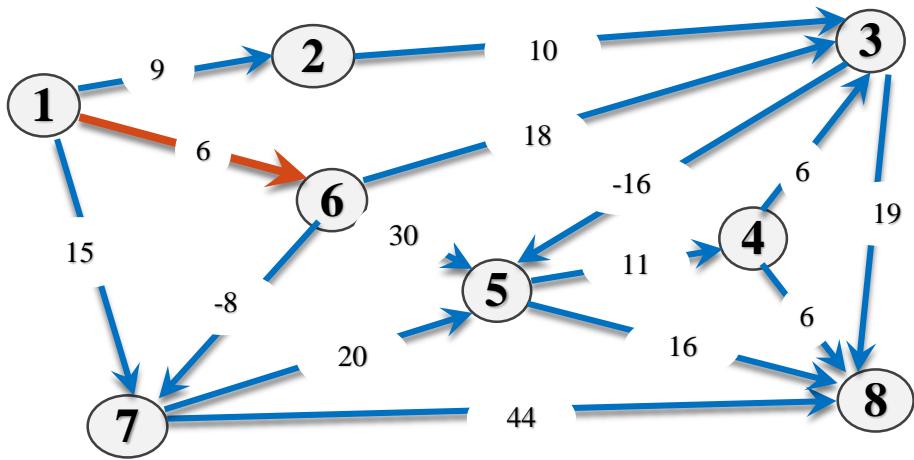
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=0
        u = G->edges[0].u = 1;
        v = G->edges[0].v = 2;
        w = G->edges[0].w = 9;
        if (pi[1] + 9 < pi[2]) { //true
            pi[2] = pi[1] + 9 = 0+9=9;
            p[2] = 1;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
p(i)	-1	1						

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19

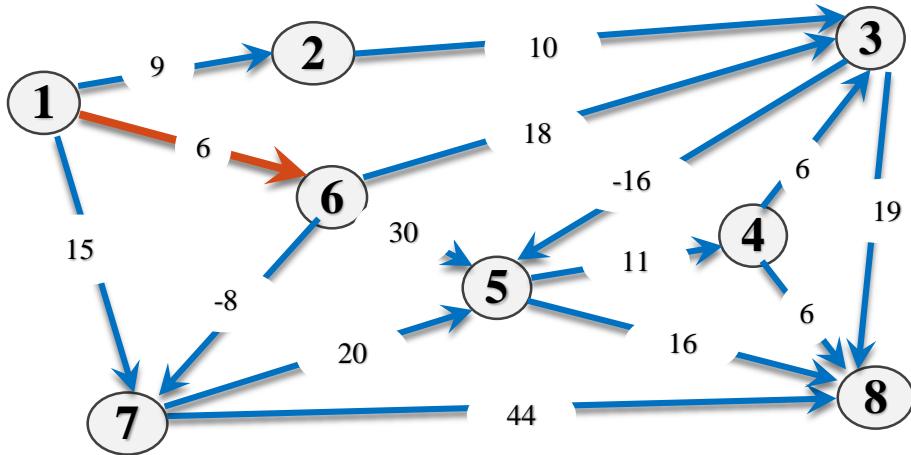


```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=1
        u = G->edges[1].u = 1;
        v = G->edges[1].v = 6;
        w = G->edges[1].w = 6;
        if (pi[u] + w < pi[v]) {
            pi[v] = pi[u] + w;
            p[v] = u;
        }
    }
}
    
```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
p(i)	-1	1						

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



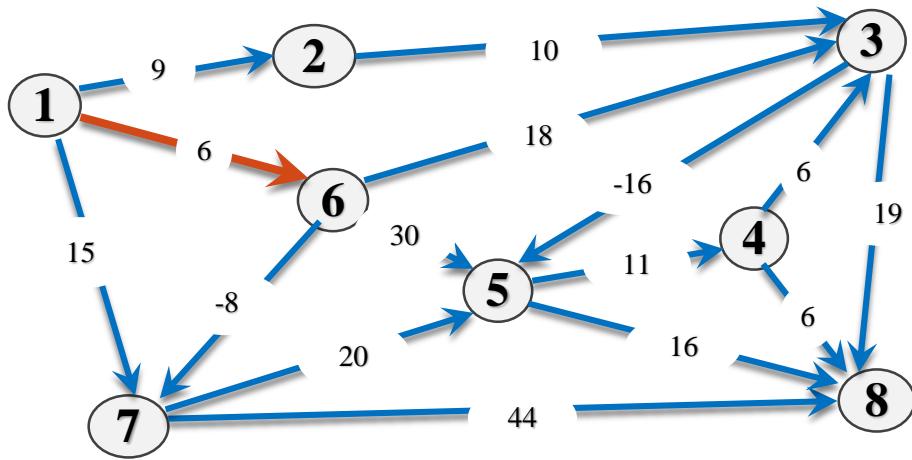
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=1
        u = G->edges[1].u = 1;
        v = G->edges[1].v = 6;
        w = G->edges[1].w = 6;
        if (pi[1] + w < pi[6]) { //true
            pi[6] = pi[1] + 6 = 0+6=6;
            p[6] = 1;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
p(i)	-1	1						

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



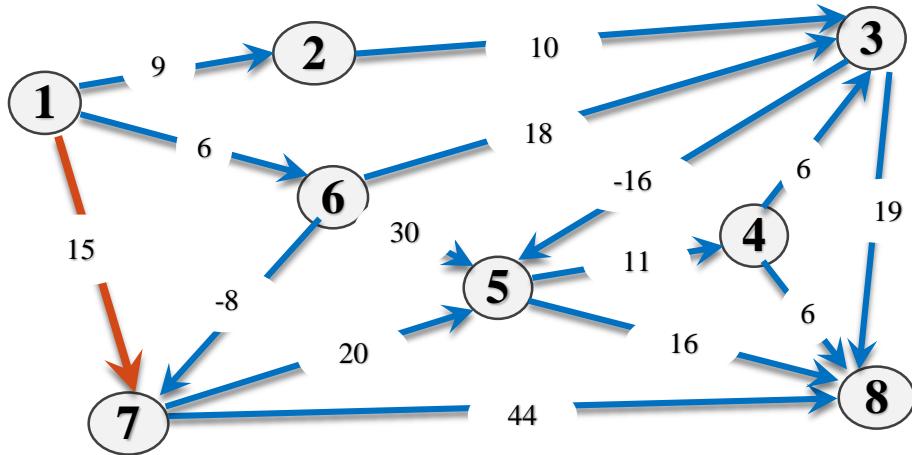
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=1
        u = G->edges[1].u = 1;
        v = G->edges[1].v = 6;
        w = G->edges[1].w = 6;
        if (pi[1] + w < pi[6]) { //true
            pi[6] = pi[1] + 6 = 0+6=6;
            p[6] = 1;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	$\infty$	$\infty$	$\infty$	6	$\infty$	$\infty$
p(i)	-1	1				1		

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



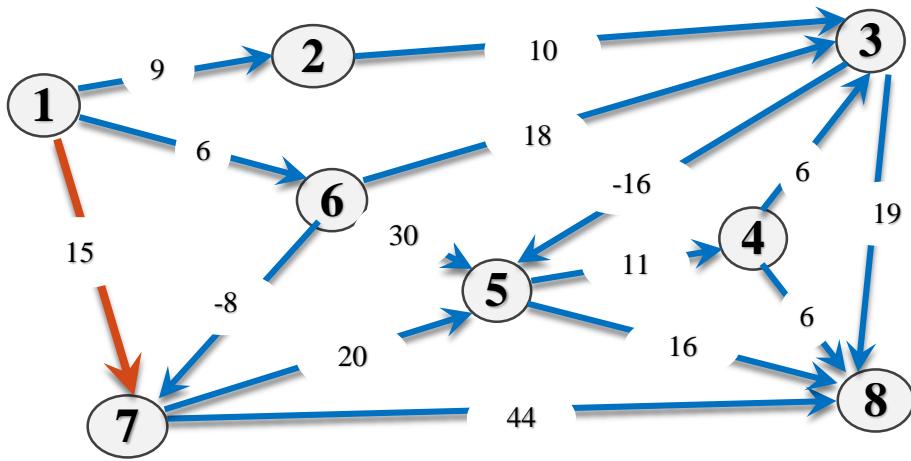
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=2
        u = G->edges[2].u = 1;
        v = G->edges[2].v = 7;
        w = G->edges[2].w = 17;
        if (pi[1] + w < pi[7]) { //true
            pi[7] = pi[1] + 17 = 0+17=17;
            p[7] = 1;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	$\infty$	$\infty$	$\infty$	6	$\infty$	$\infty$
p(i)	-1	1				1		

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



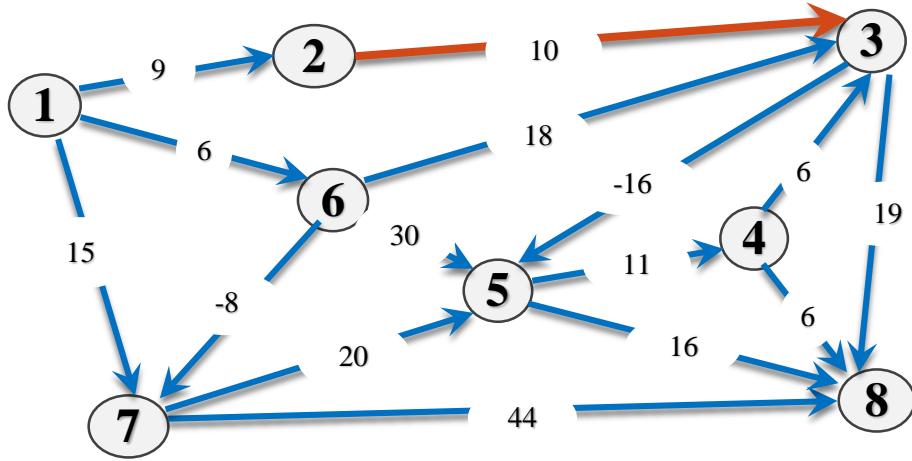
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=2
        u = G->edges[2].u = 1;
        v = G->edges[2].v = 7;
        w = G->edges[2].w = 17;
        if (pi[1] + w < pi[7]) { //true
            pi[7] = pi[1] + 17 = 0+17=17;
            p[7] = 1;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	$\infty$	$\infty$	$\infty$	6	17	$\infty$
p(i)	-1	1				1	1	

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



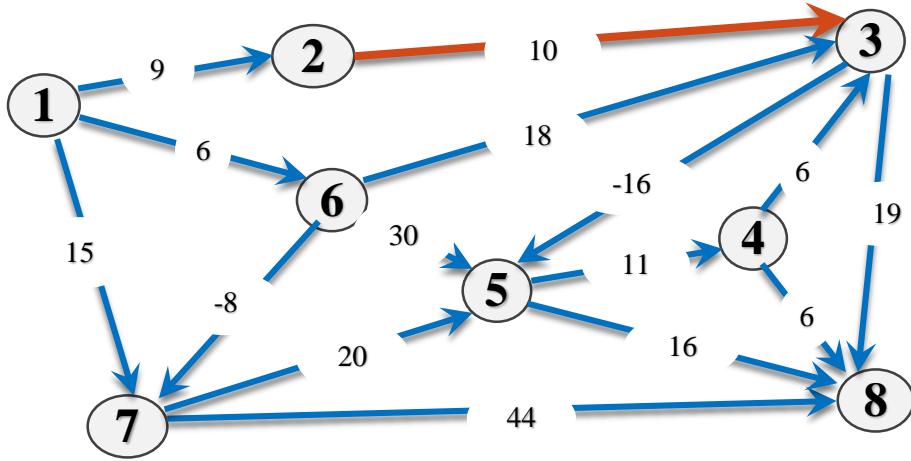
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=3
        u = G->edges[3].u = 2;
        v = G->edges[3].v = 3;
        w = G->edges[3].w = 10;
        if (pi[2] + w < pi[3]) { //true
            pi[3] = pi[2] + 10 = 9+10=19;
            p[3] = 2;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	$\infty$	$\infty$	$\infty$	6	$\infty$	$\infty$
p(i)	-1	1				1		

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



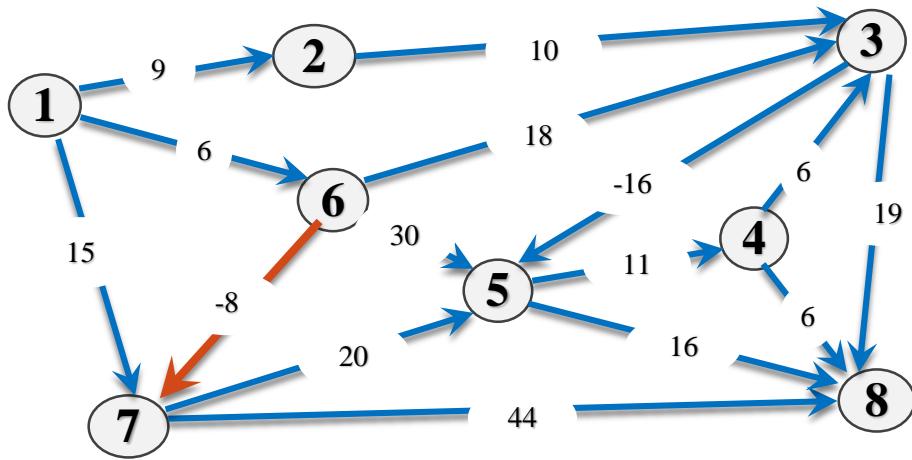
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=3
        u = G->edges[3].u = 2;
        v = G->edges[3].v = 3;
        w = G->edges[3].w = 10;
        if (pi[2] + w < pi[3]) { //true
            pi[3] = pi[2] + 10 = 9+10=19;
            p[3] = 2;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	$\infty$	$\infty$	6	$\infty$	$\infty$
p(i)	-1	1	2			1		

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



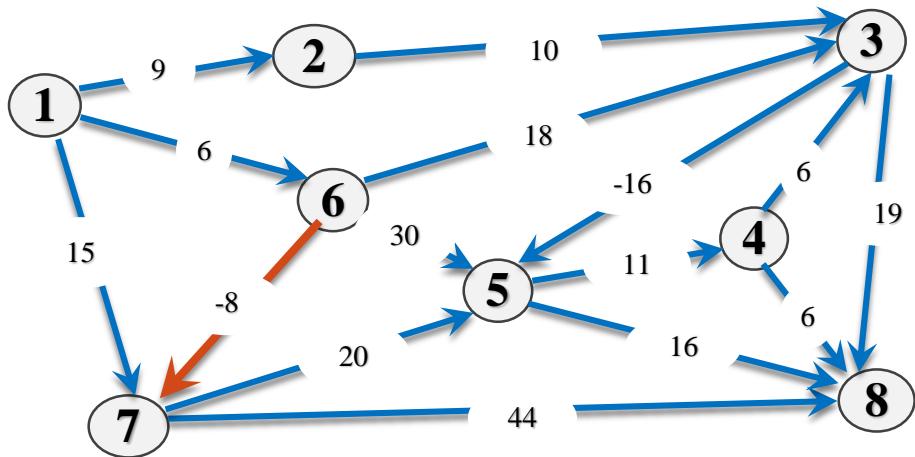
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=4
        u = G->edges[4].u = 6;
        v = G->edges[4].v = 7;
        w = G->edges[4].w = -8;
        if (pi[6] + w < pi[7]) { //true
            pi[7] = pi[6] + -8 = 6+-8=-2;
            p[7] = 6;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	$\infty$	$\infty$	6	$\infty$	$\infty$
p(i)	-1	1	2			1		

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



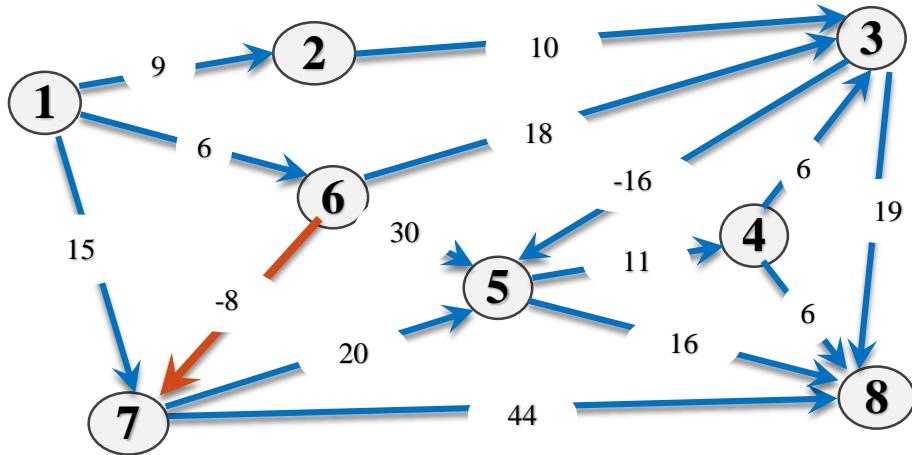
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=4
        u = G->edges[4].u = 6;
        v = G->edges[4].v = 7;
        w = G->edges[4].w = -8;
        if (pi[6] + w < pi[7]) { //true
            pi[7] = pi[6] + -8 = 6+(-8)=-2;
            p[7] = 6;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	$\infty$	$\infty$	6	$\infty$	$\infty$
p(i)	-1	1	2			1		

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



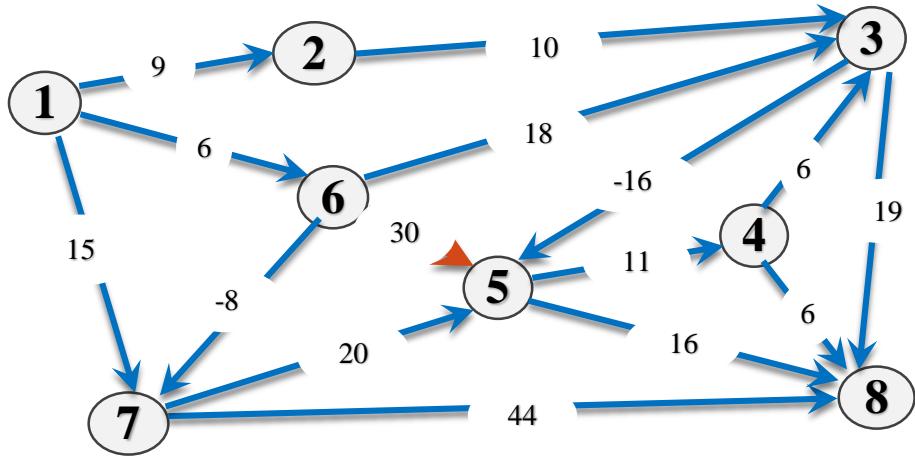
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=4
        u = G->edges[4].u = 6;
        v = G->edges[4].v = 7;
        w = G->edges[4].w = -8;
        if (pi[6] + w < pi[7]) { //true
            pi[7] = pi[6] + -8 = 6+(-8)=-2;
            p[7] = 6;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	$\infty$	$\infty$	6	-2	$\infty$
p(i)	-1	1	2			1	6	

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



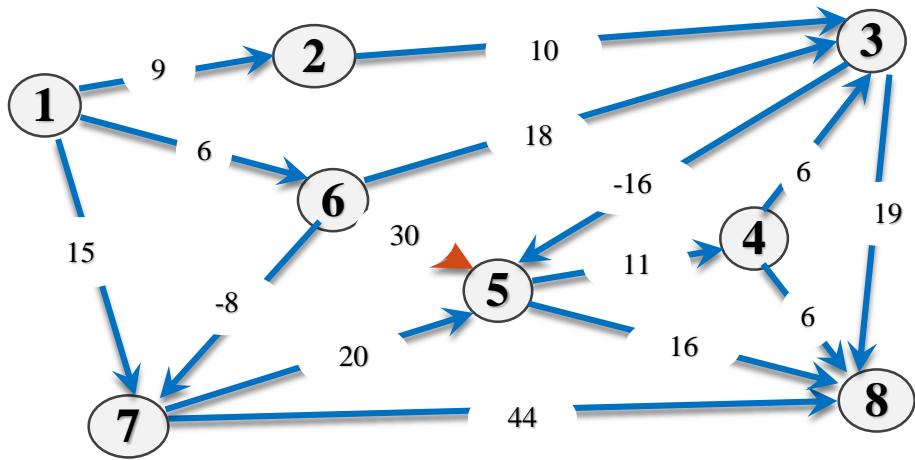
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=5
        u = G->edges[5].u = 6;
        v = G->edges[5].v = 5;
        w = G->edges[5].w = 30;
        if (pi[6] + w < pi[5]) { //true
            pi[5] = pi[6] + 30 = 6+30=36;
            p[5] = 6;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	$\infty$	$\infty$	6	-2	$\infty$
p(i)	-1	1	2			1	6	

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



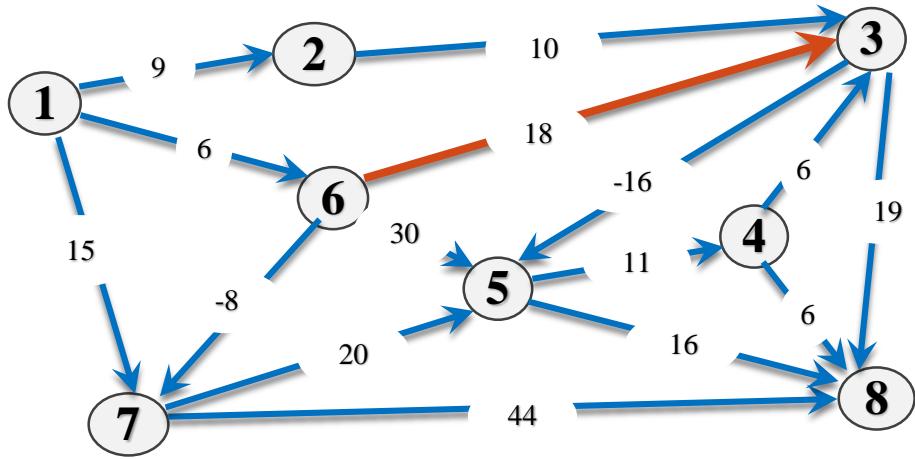
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=5
        u = G->edges[5].u = 6;
        v = G->edges[5].v = 5;
        w = G->edges[5].w = 30;
        if (pi[6] + w < pi[5]) { //true
            pi[5] = pi[6] + 30 = 6+30=36;
            p[5] = 6;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	$\infty$	36	6	-2	$\infty$
p(i)	-1	1	2		5	1	6	

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



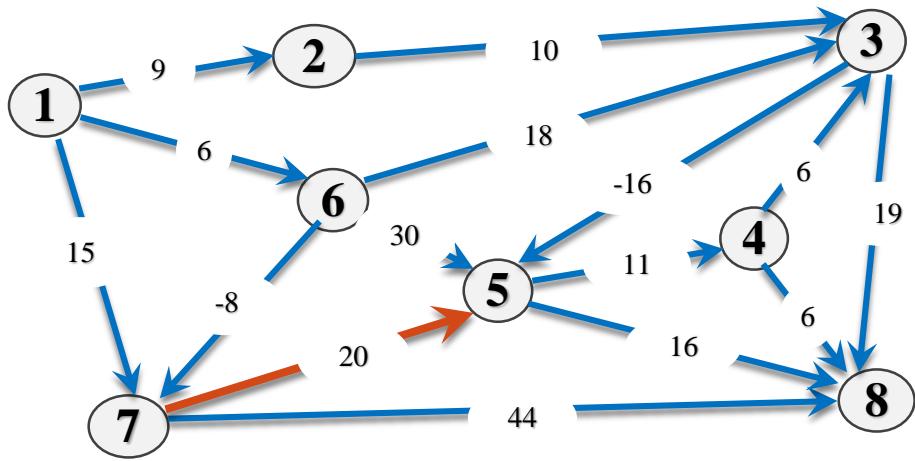
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=6
        u = G->edges[6].u = 6;
        v = G->edges[6].v = 3;
        w = G->edges[6].w = 18;
        if (pi[6] + w < pi[3]) { //false
            //Không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	$\infty$	36	6	-2	$\infty$
p(i)	-1	1	2		5	1	6	

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



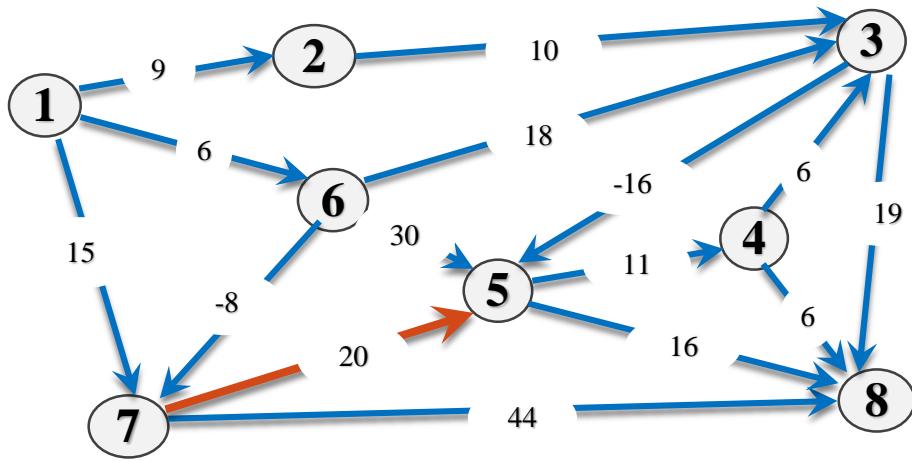
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=7
        u = G->edges[7].u = 7;
        v = G->edges[7].v = 5;
        w = G->edges[7].w = 20;
        if (pi[7] + w < pi[5]) { //true
            pi[5] = pi[7] + w = -2+20=18;
            p[5] = 7;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	$\infty$	36	6	-2	$\infty$
p(i)	-1	1	2		5	1	6	

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



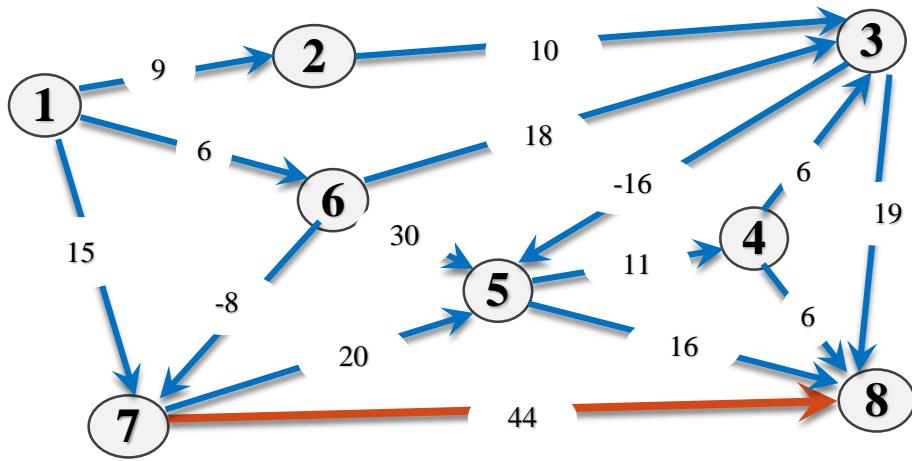
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=7
        u = G->edges[7].u = 7;
        v = G->edges[7].v = 5;
        w = G->edges[7].w = 20;
        if (pi[7] + w < pi[5]) { //true
            pi[5] = pi[7] + w = -2+20=18;
            p[5] = 7;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	$\infty$	18	6	-2	$\infty$
p(i)	-1	1	2		7	1	6	

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



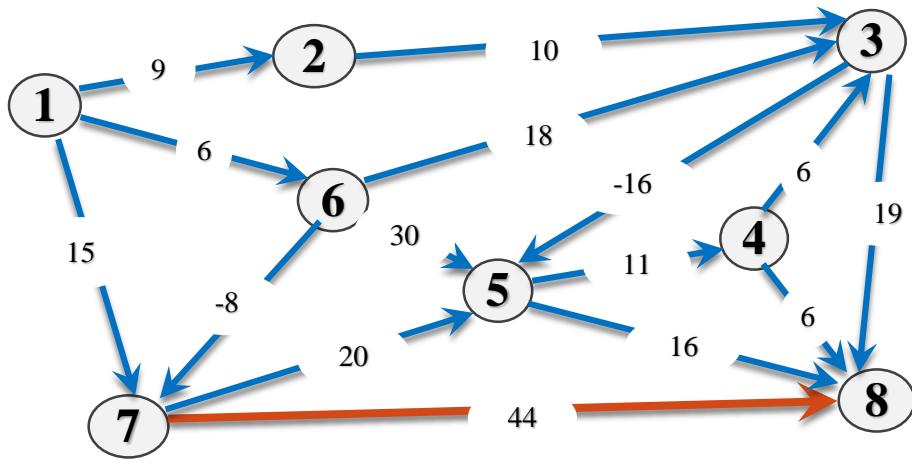
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=8
        u = G->edges[8].u = 7;
        v = G->edges[8].v = 8;
        w = G->edges[8].w = 44;
        if (pi[7] + w < pi[8]) { //true
            pi[8] = pi[7] + w = -2+44=42;
            p[8] = 7;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	$\infty$	18	6	-2	$\infty$
p(i)	-1	1	2		7	1	6	

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



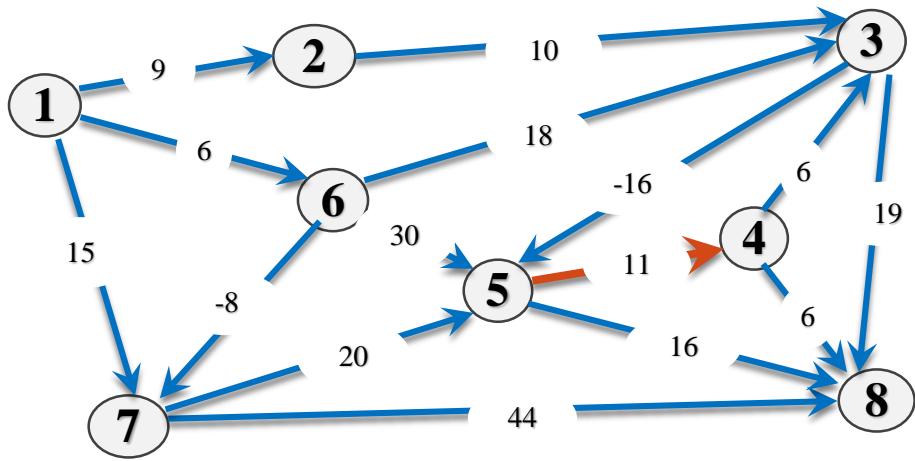
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=8
        u = G->edges[8].u = 7;
        v = G->edges[8].v = 8;
        w = G->edges[8].w = 44;
        if (pi[7] + w < pi[8]) { //true
            pi[8] = pi[7] + w = -2+44=42;
            p[8] = 7;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	$\infty$	18	6	-2	42
p(i)	-1	1	2		7	1	6	7

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



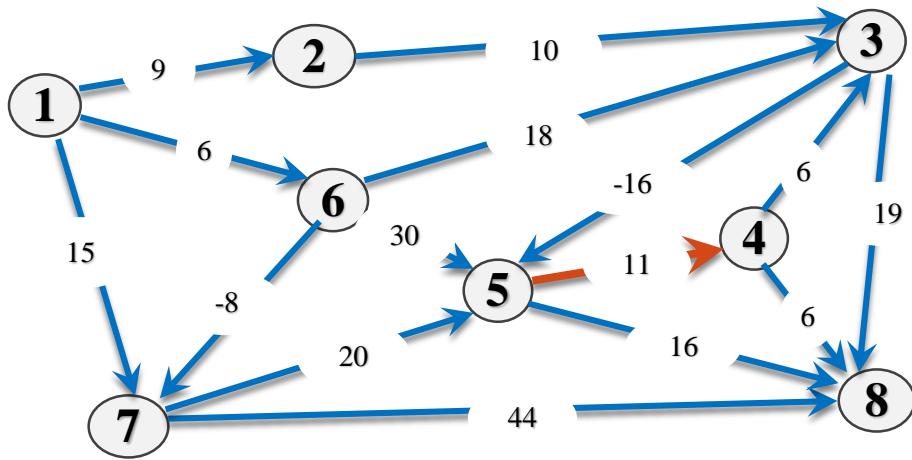
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=9
        u = G->edges[9].u = 5;
        v = G->edges[9].v = 4;
        w = G->edges[9].w = 11;
        if (pi[5] + w < pi[4]) { //true
            pi[4] = pi[5] + w = 18+11=29;
            p[4] = 5;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	$\infty$	18	6	-2	42
p(i)	-1	1	2		7	1	6	7

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



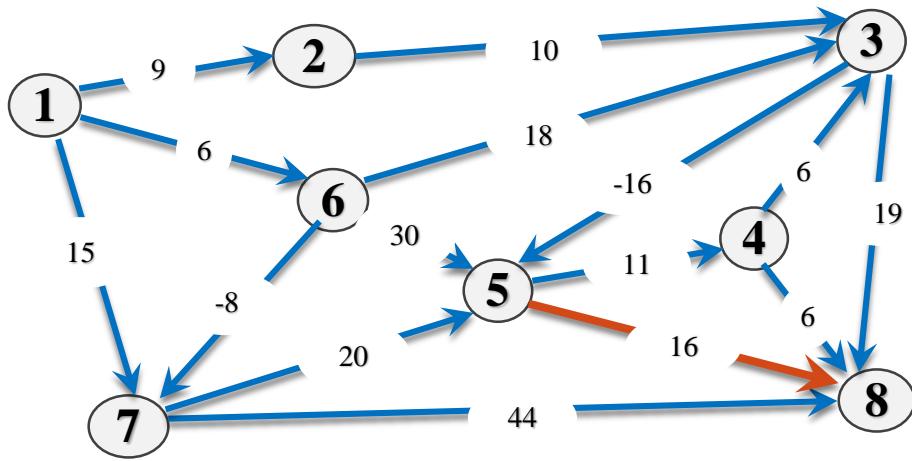
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=9
        u = G->edges[9].u = 5;
        v = G->edges[9].v = 4;
        w = G->edges[9].w = 11;
        if (pi[5] + w < pi[4]) { //true
            pi[4] = pi[5] + w = 18+11=29;
            p[4] = 5;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	18	6	-2	42
p(i)	-1	1	2	5	7	1	6	7

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



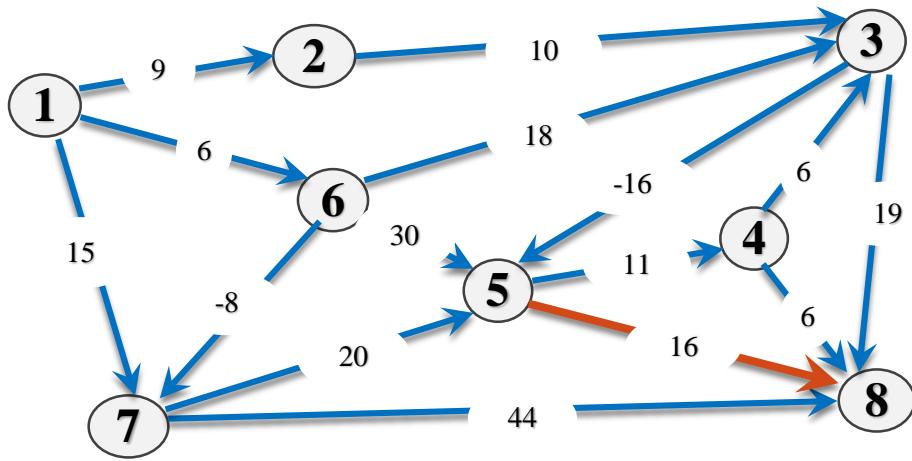
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=10
        u = G->edges[10].u = 5;
        v = G->edges[10].v = 8;
        w = G->edges[10].w = 16;
        if (pi[5] + w < pi[8]) { //true
            pi[8] = pi[5] + w = 18+16=34;
            p[8] = 5;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	18	6	-2	42
p(i)	-1	1	2	5	7	1	6	7

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



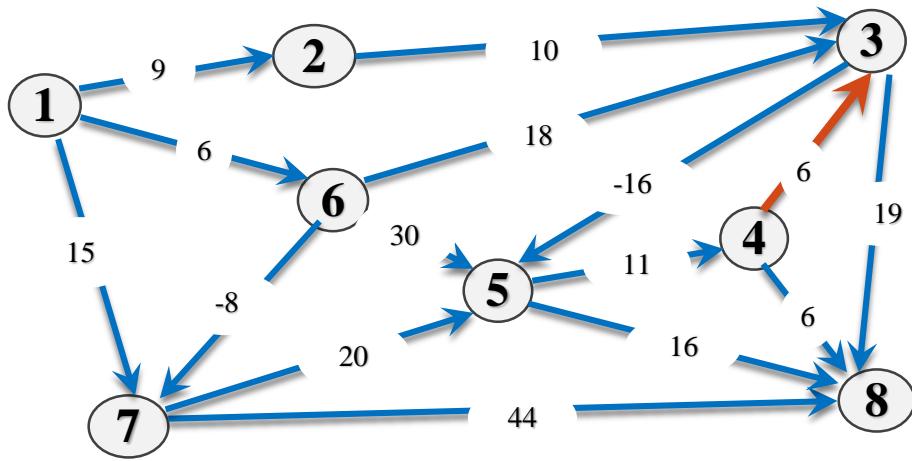
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=10
        u = G->edges[10].u = 5;
        v = G->edges[10].v = 8;
        w = G->edges[10].w = 16;
        if (pi[5] + w < pi[8]) { //true
            pi[8] = pi[5] + w = 18+16=34;
            p[8] = 5;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	18	6	-2	34
p(i)	-1	1	2	5	7	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



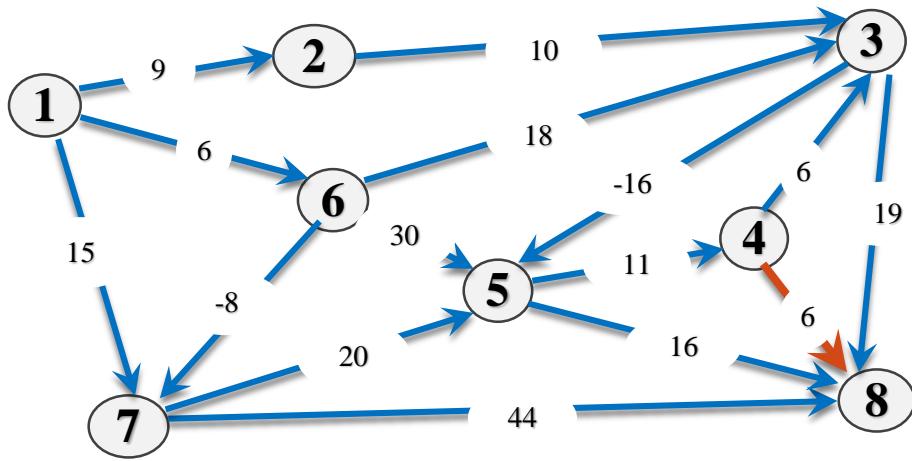
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=11
        u = G->edges[11].u = 4;
        v = G->edges[11].v = 3;
        w = G->edges[11].w = 6;
        if (pi[4] + w < pi[3]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	18	6	-2	34
p(i)	-1	1	2	5	7	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



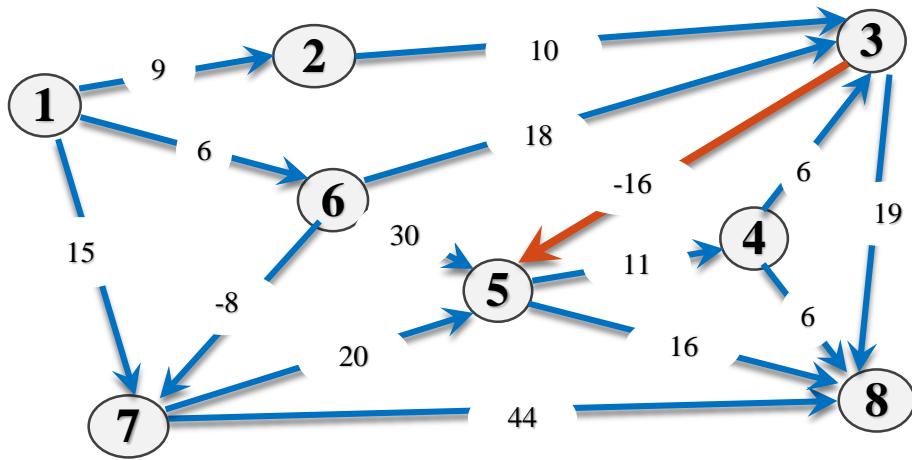
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=12
        u = G->edges[12].u = 4;
        v = G->edges[12].v = 8;
        w = G->edges[12].w = 6;
        if (pi[4] + w < pi[8]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	18	6	-2	34
p(i)	-1	1	2	5	7	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



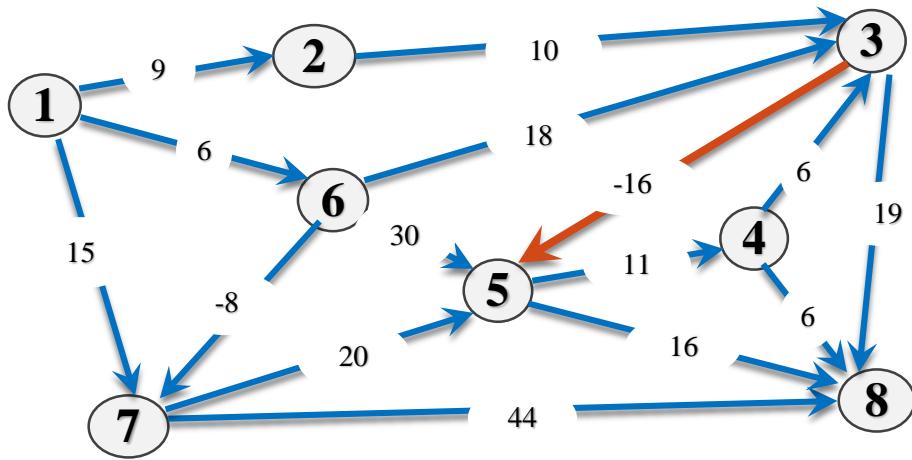
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=13
        u = G->edges[13].u = 3;
        v = G->edges[13].v = 5;
        w = G->edges[13].w = -16;
        if (pi[3] + w < pi[5]) { //true
            pi[5] = pi[3] + w = 19+(-16)=3;
            p[5] = 3;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	18	6	-2	34
p(i)	-1	1	2	5	7	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



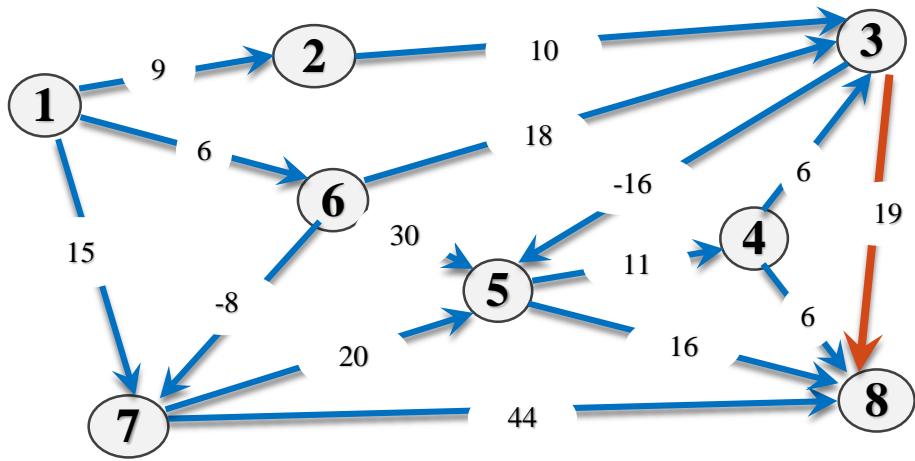
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=13
        u = G->edges[13].u = 3;
        v = G->edges[13].v = 5;
        w = G->edges[13].w = -16;
        if (pi[3] + w < pi[5]) { //true
            pi[5] = pi[3] + w = 19+(-16)=3;
            p[5] = 3;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



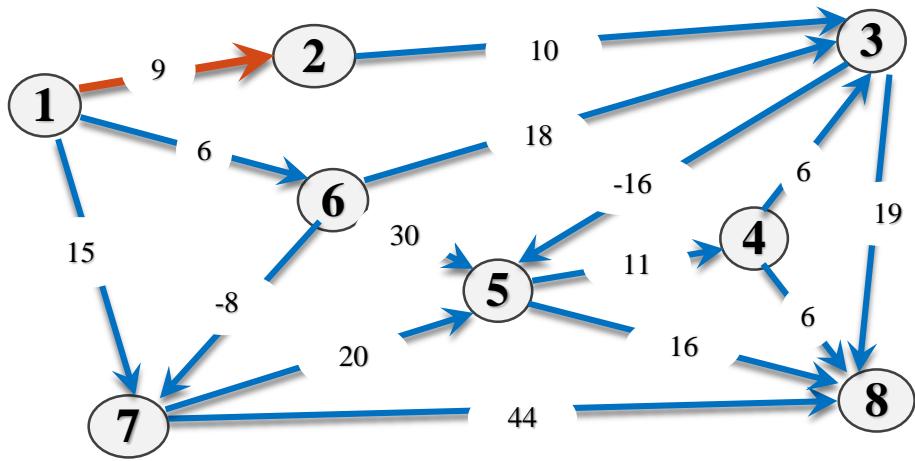
```

for (it = 1; it < G->n; it++) { //it=1
    for (k = 0; k < G->m; k++) { //k=14
        u = G->edges[14].u = 3;
        v = G->edges[14].v = 8;
        w = G->edges[14].w = 19;
        if (pi[3] + w < pi[8]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



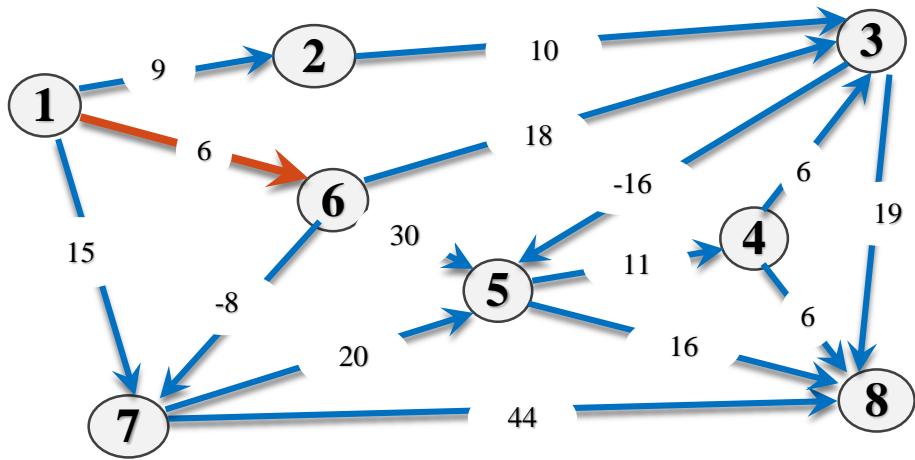
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=0
        u = G->edges[0].u = 1;
        v = G->edges[0].v = 2;
        w = G->edges[0].w = 9;
        if (pi[u] + w < pi[v]) { //False
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



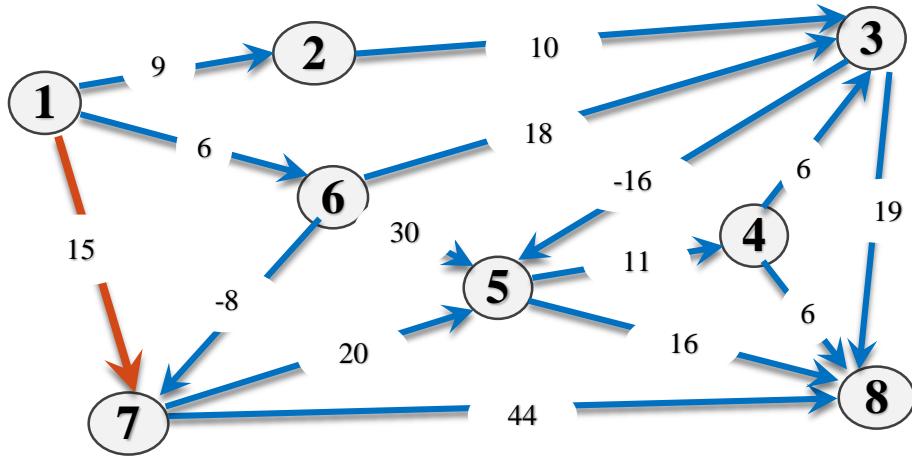
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=1
        u = G->edges[1].u = 1;
        v = G->edges[1].v = 6;
        w = G->edges[1].w = 6;
        if (pi[u] + w < pi[v]) { //False
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



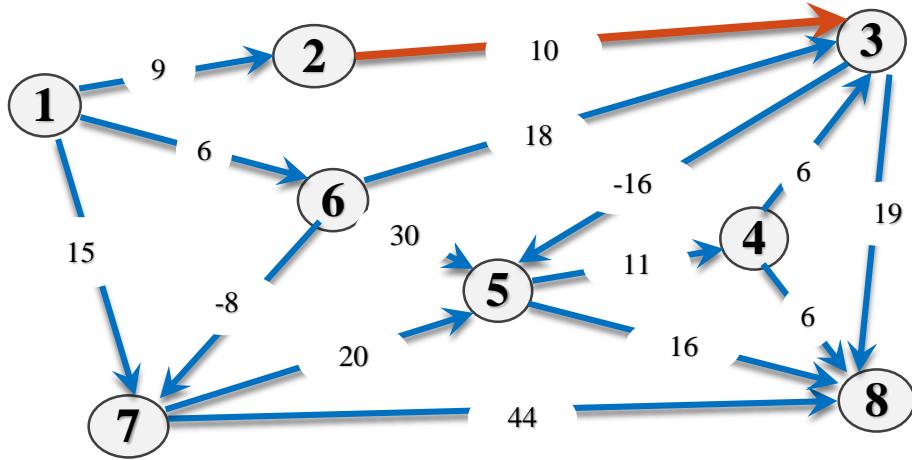
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=2
        u = G->edges[2].u = 1;
        v = G->edges[2].v = 7;
        w = G->edges[2].w = 17;
        if (pi[1] + w < pi[7]) { //False
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



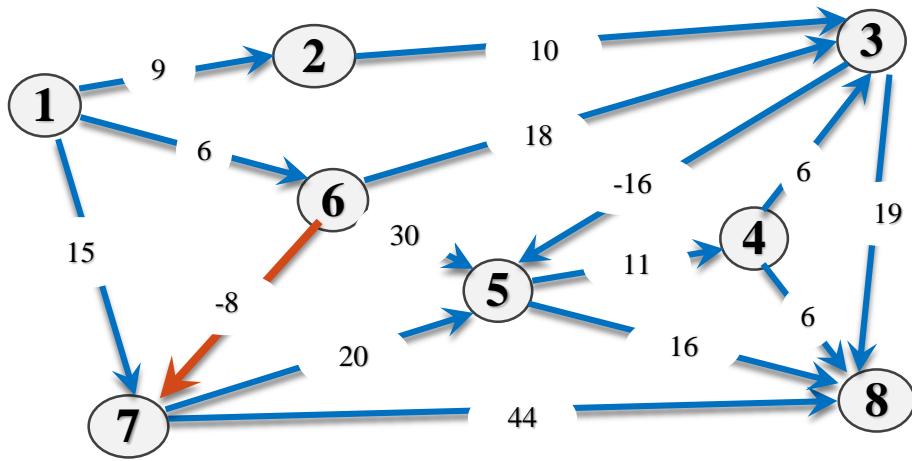
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=3
        u = G->edges[3].u = 2;
        v = G->edges[3].v = 3;
        w = G->edges[3].w = 10;
        if (pi[2] + w < pi[3]) { //False
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



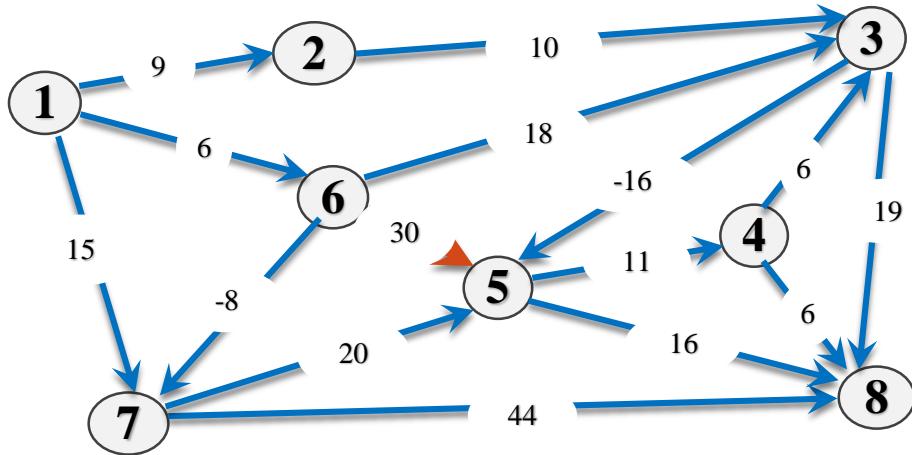
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=4
        u = G->edges[4].u = 6;
        v = G->edges[4].v = 7;
        w = G->edges[4].w = -8;
        if (pi[6] + w < pi[7]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



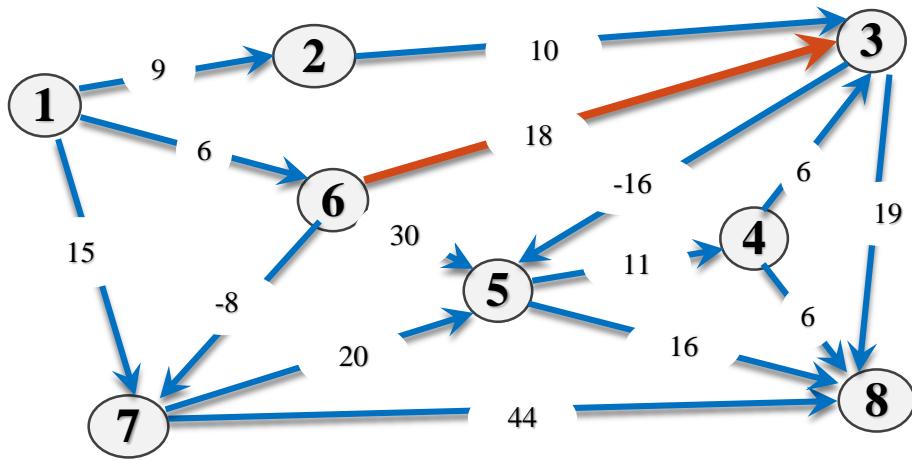
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=5
        u = G->edges[5].u = 6;
        v = G->edges[5].v = 5;
        w = G->edges[5].w = 30;
        if (pi[6] + w < pi[5]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



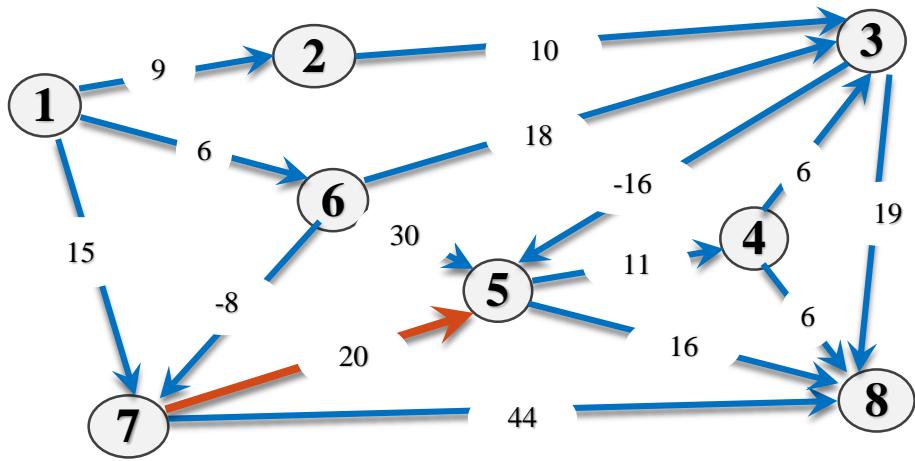
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=6
        u = G->edges[6].u = 6;
        v = G->edges[6].v = 3;
        w = G->edges[6].w = 18;
        if (pi[6] + w < pi[3]) { //false
            //Không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



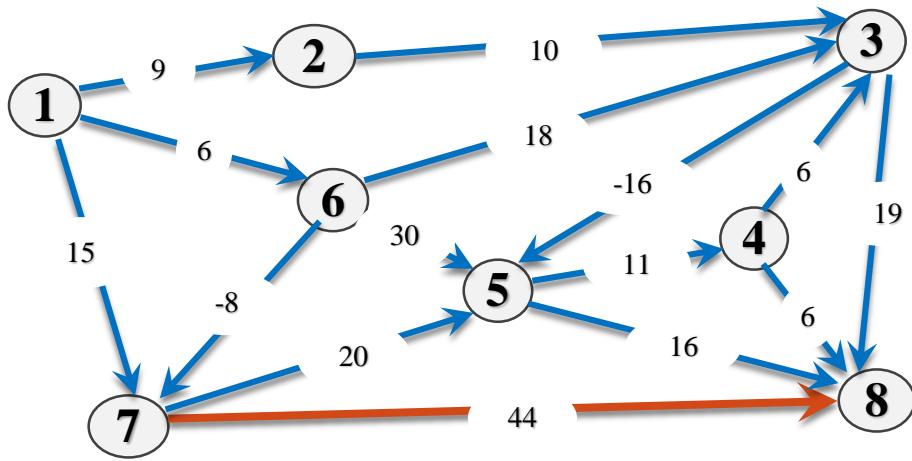
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=7
        u = G->edges[7].u = 7;
        v = G->edges[7].v = 5;
        w = G->edges[7].w = 20;
        if (pi[7] + w < pi[5]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



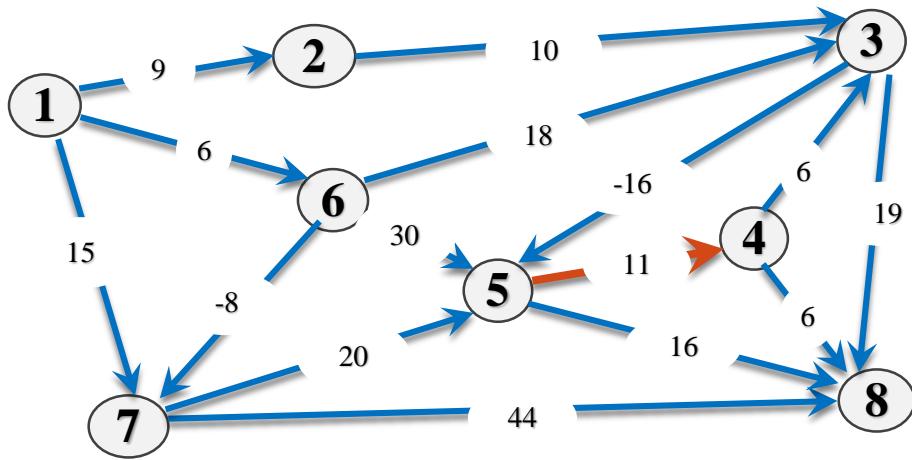
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=8
        u = G->edges[8].u = 7;
        v = G->edges[8].v = 8;
        w = G->edges[8].w = 44;
        if (pi[7] + w < pi[8]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



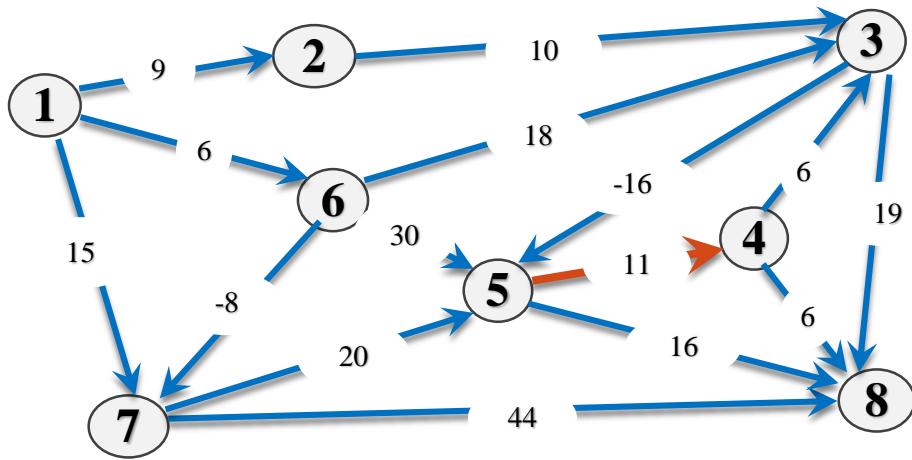
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=9
        u = G->edges[9].u = 5;
        v = G->edges[9].v = 4;
        w = G->edges[9].w = 11;
        if (pi[5] + w < pi[4]) { //true
            pi[4] = pi[5] + w = 3+11=14;
            p[4] = 5;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



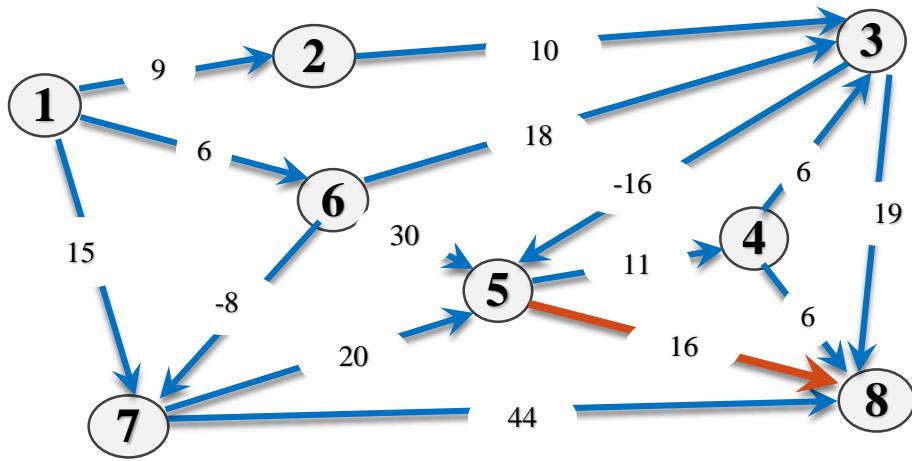
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=9
        u = G->edges[9].u = 5;
        v = G->edges[9].v = 4;
        w = G->edges[9].w = 11;
        if (pi[5] + w < pi[4]) { //true
            pi[4] = pi[5] + w = 3+11=14;
            p[4] = 5;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



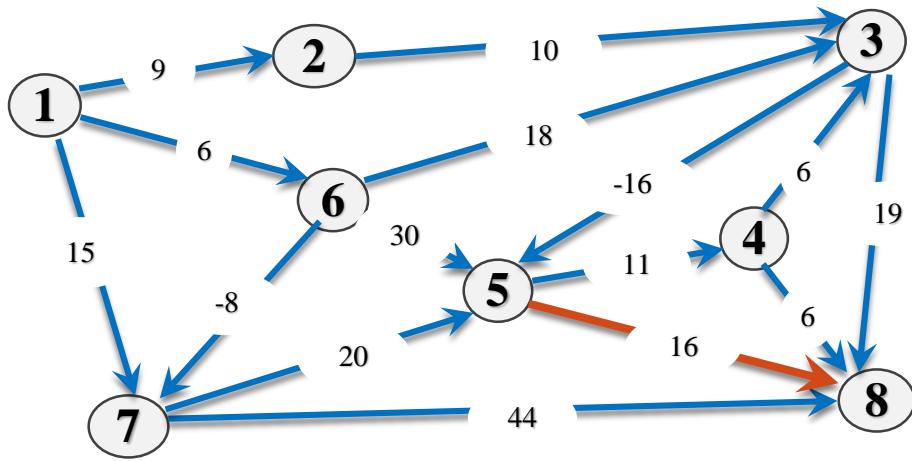
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=10
        u = G->edges[10].u = 5;
        v = G->edges[10].v = 8;
        w = G->edges[10].w = 16;
        if (pi[5] + w < pi[8]) { //true
            pi[8] = pi[5] + w = 3+16=19;
            p[8] = 5;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



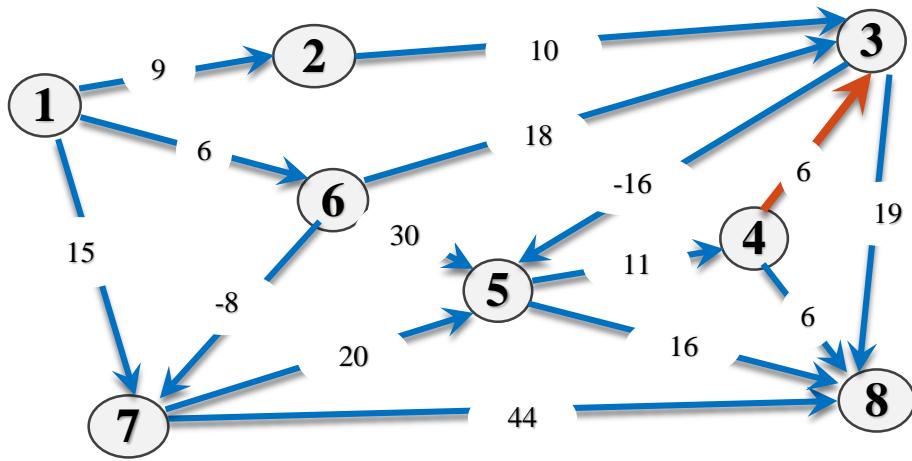
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=10
        u = G->edges[10].u = 5;
        v = G->edges[10].v = 8;
        w = G->edges[10].w = 16;
        if (pi[5] + w < pi[8]) { //true
            pi[8] = pi[5] + w = 3+16=19;
            p[8] = 5;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



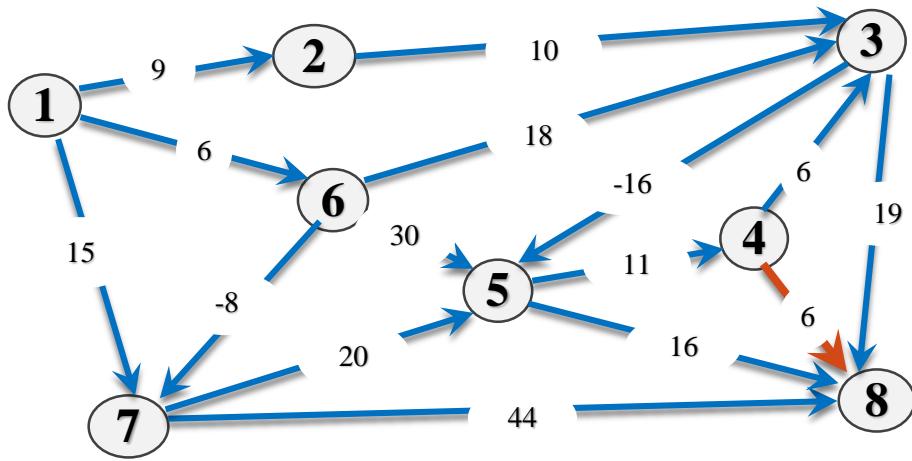
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=11
        u = G->edges[11].u = 4;
        v = G->edges[11].v = 3;
        w = G->edges[11].w = 6;
        if (pi[4] + w < pi[3]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



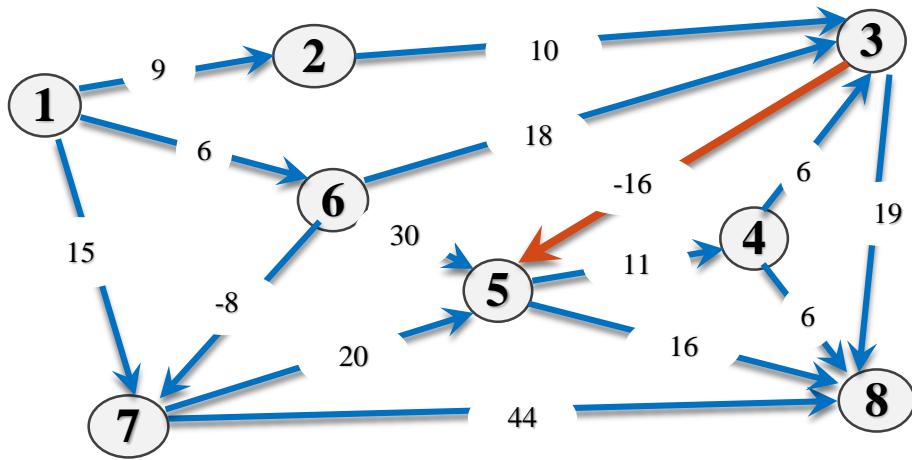
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=12
        u = G->edges[12].u = 4;
        v = G->edges[12].v = 8;
        w = G->edges[12].w = 6;
        if (pi[4] + w < pi[8]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



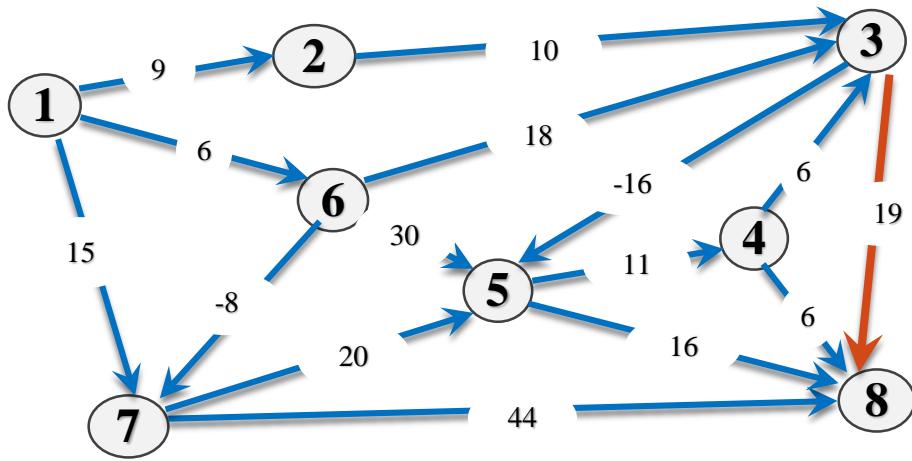
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=13
        u = G->edges[13].u = 3;
        v = G->edges[13].v = 5;
        w = G->edges[13].w = -16;
        if (pi[3] + w < pi[5]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



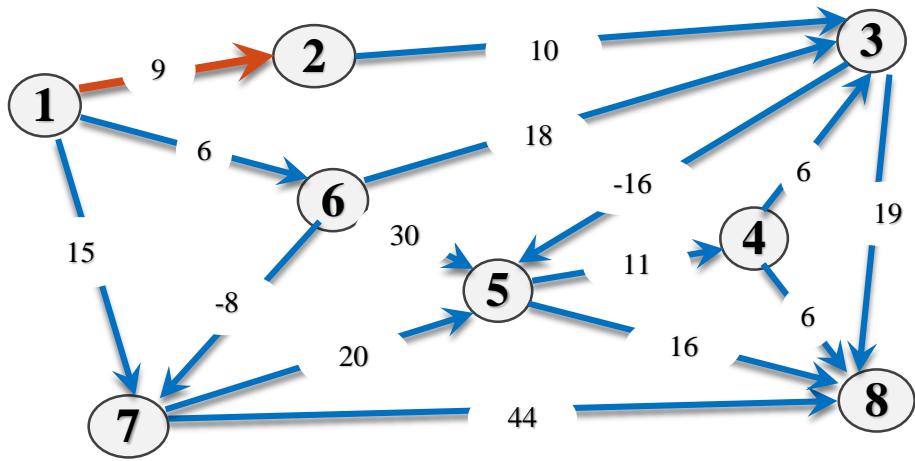
```

for (it = 1; it < G->n; it++) { //it=2
    for (k = 0; k < G->m; k++) { //k=14
        u = G->edges[14].u = 3;
        v = G->edges[14].v = 8;
        w = G->edges[14].w = 19;
        if (pi[3] + w < pi[8]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



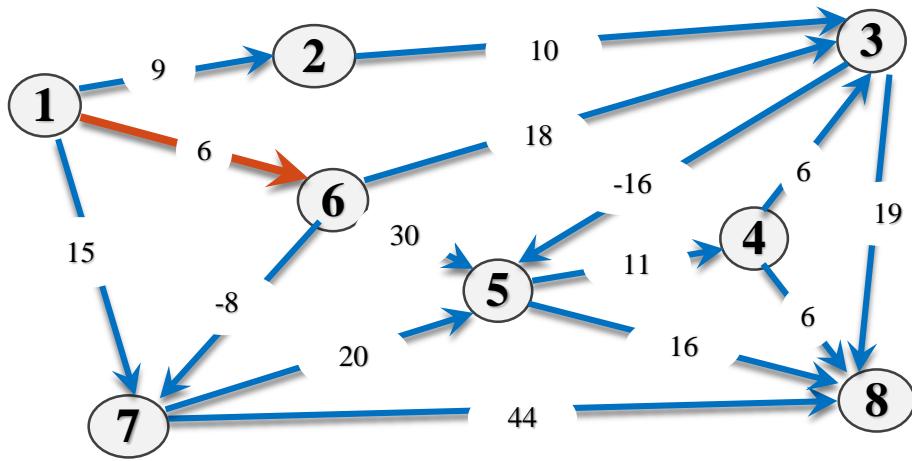
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=0
        u = G->edges[0].u = 1;
        v = G->edges[0].v = 2;
        w = G->edges[0].w = 9;
        if (pi[u] + w < pi[v]) { //False
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



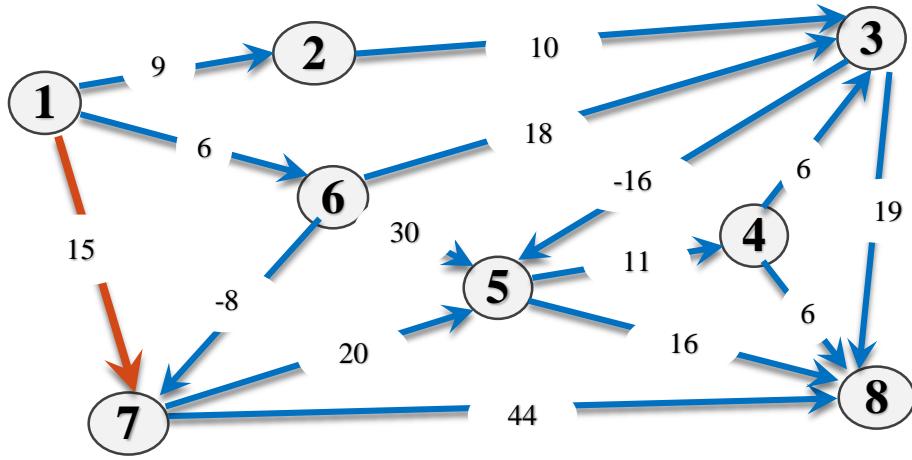
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=1
        u = G->edges[1].u = 1;
        v = G->edges[1].v = 6;
        w = G->edges[1].w = 6;
        if (pi[u] + w < pi[v]) { //False
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19

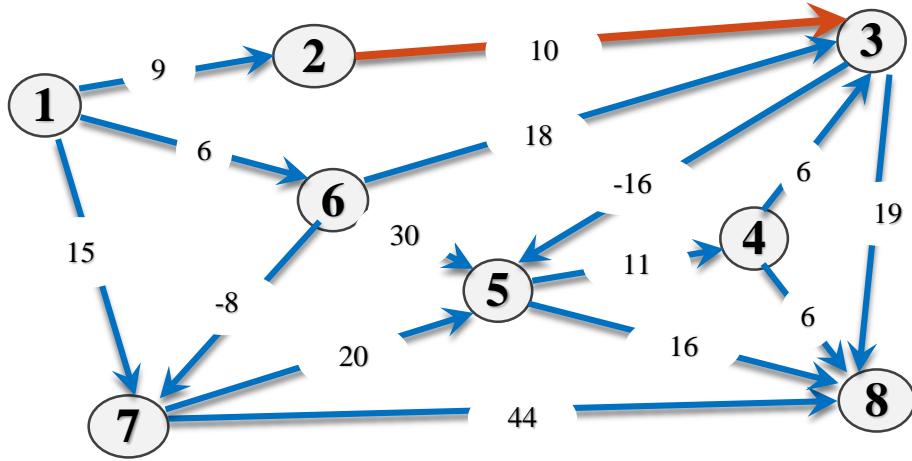


```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=2
        u = G->edges[2].u = 1;
        v = G->edges[2].v = 7;
        w = G->edges[2].w = 17;
        if (pi[1] + w < pi[7]) { //False
            //không làm gì cả
        }
    }
}
    
```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



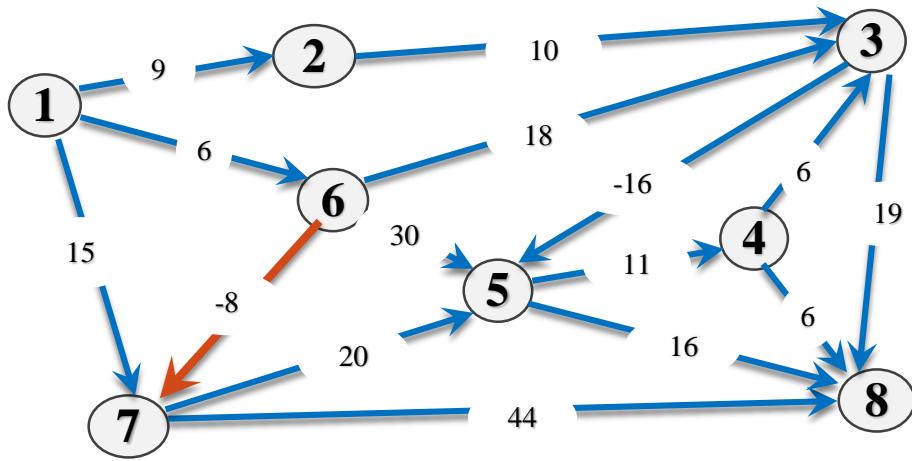
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=3
        u = G->edges[3].u = 2;
        v = G->edges[3].v = 3;
        w = G->edges[3].w = 10;
        if (pi[2] + w < pi[3]) { //False
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



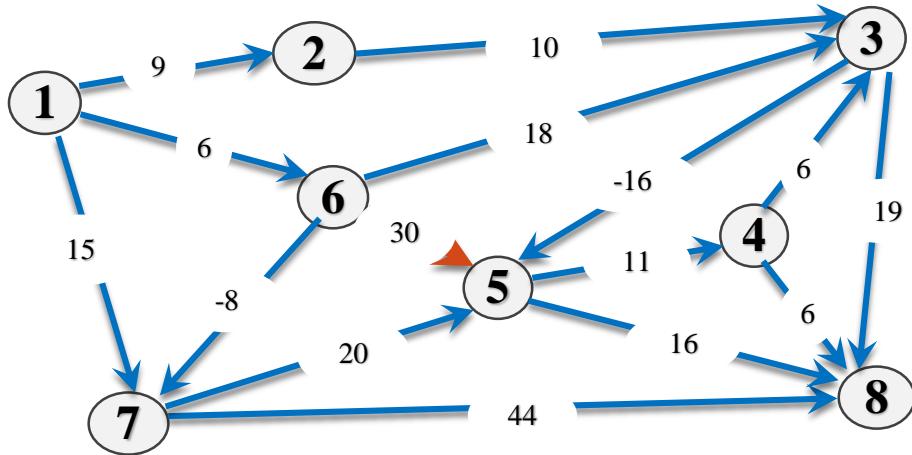
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=4
        u = G->edges[4].u = 6;
        v = G->edges[4].v = 7;
        w = G->edges[4].w = -8;
        if (pi[6] + w < pi[7]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



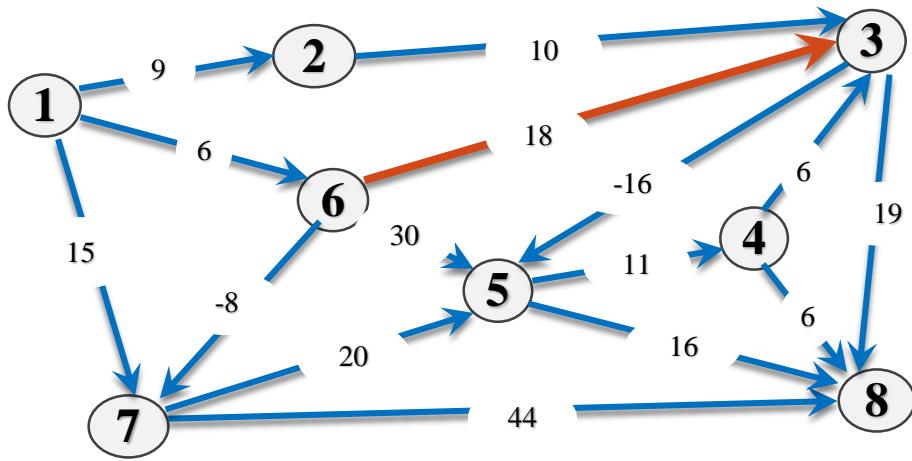
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=5
        u = G->edges[5].u = 6;
        v = G->edges[5].v = 5;
        w = G->edges[5].w = 30;
        if (pi[6] + w < pi[5]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



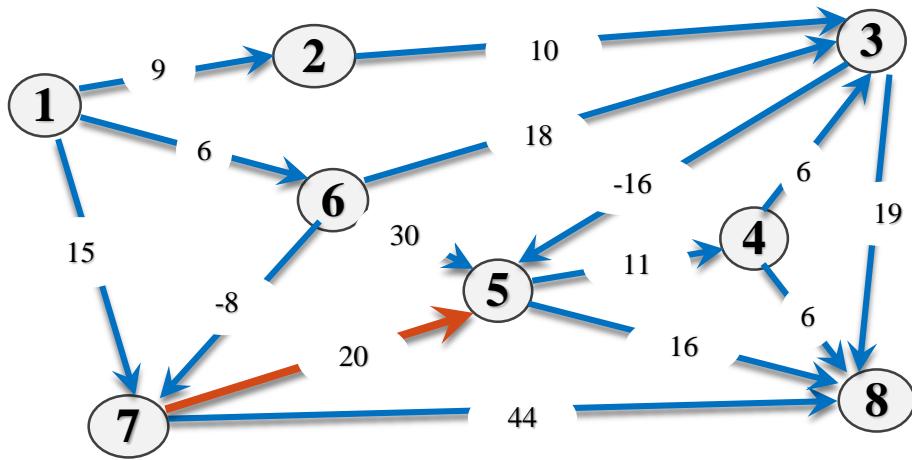
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=6
        u = G->edges[6].u = 6;
        v = G->edges[6].v = 3;
        w = G->edges[6].w = 18;
        if (pi[6] + w < pi[3]) { //false
            //Không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



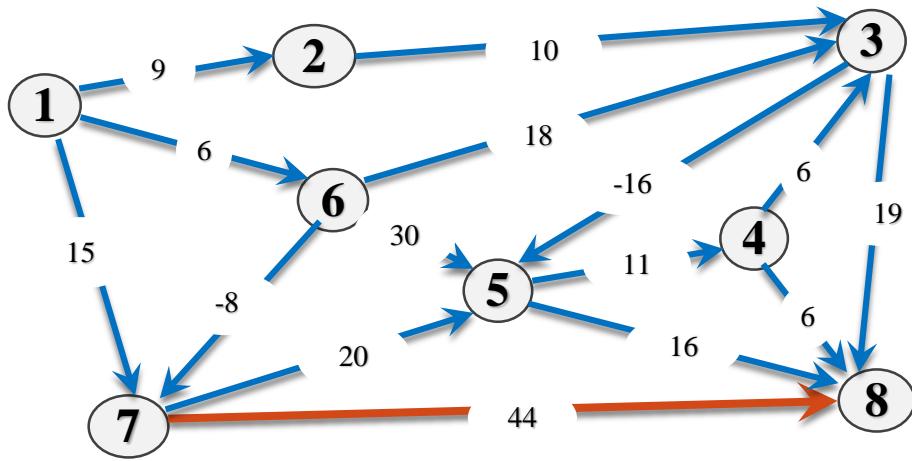
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=7
        u = G->edges[7].u = 7;
        v = G->edges[7].v = 5;
        w = G->edges[7].w = 20;
        if (pi[7] + w < pi[5]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



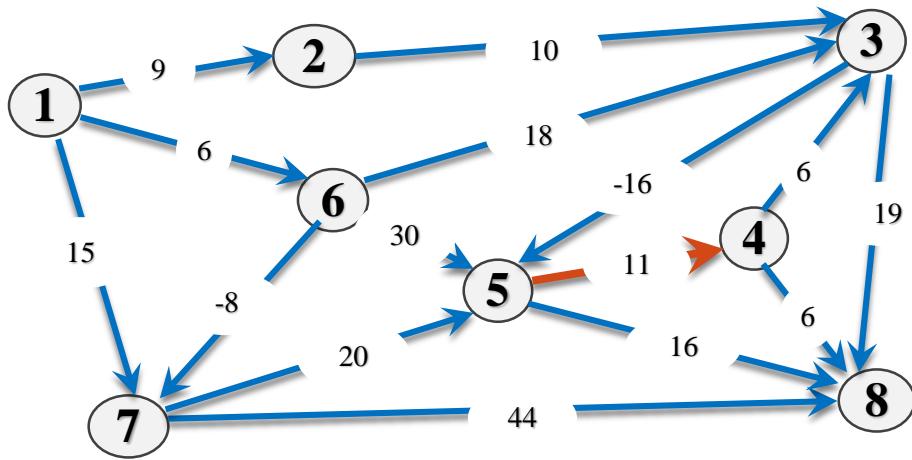
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=8
        u = G->edges[8].u = 7;
        v = G->edges[8].v = 8;
        w = G->edges[8].w = 44;
        if (pi[7] + w < pi[8]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



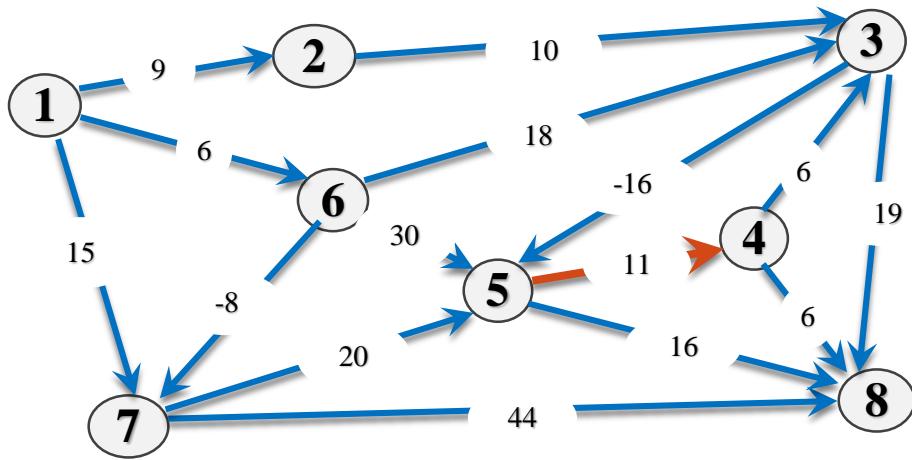
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=9
        u = G->edges[9].u = 5;
        v = G->edges[9].v = 4;
        w = G->edges[9].w = 11;
        if (pi[5] + w < pi[4]) { //true
            pi[4] = pi[5] + w = 3+11=14;
            p[4] = 5;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	29	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



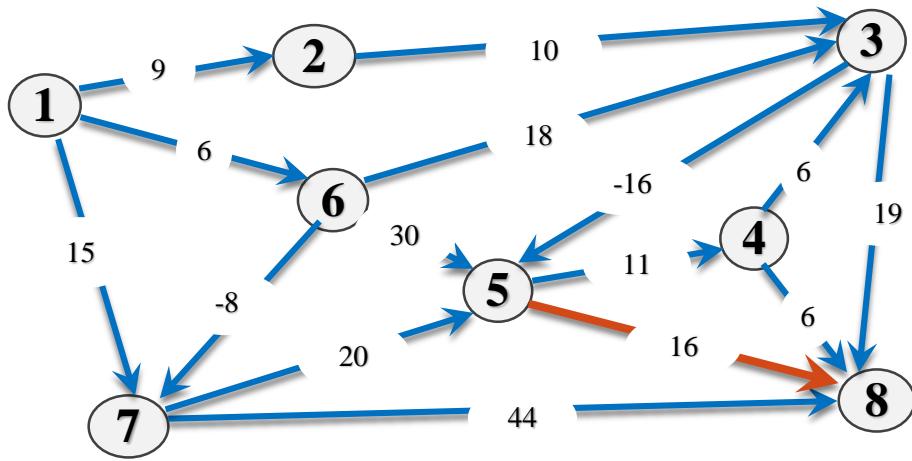
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=9
        u = G->edges[9].u = 5;
        v = G->edges[9].v = 4;
        w = G->edges[9].w = 11;
        if (pi[5] + w < pi[4]) { //true
            pi[4] = pi[5] + w = 3+11=14;
            p[4] = 5;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



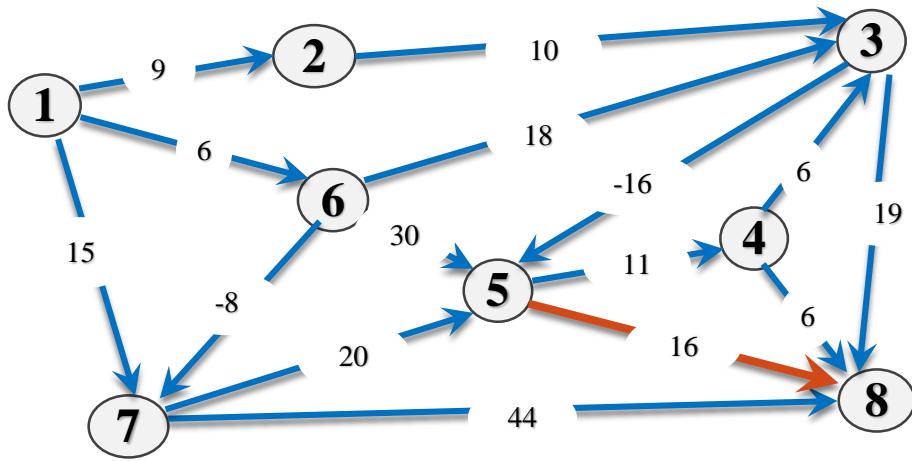
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=10
        u = G->edges[10].u = 5;
        v = G->edges[10].v = 8;
        w = G->edges[10].w = 16;
        if (pi[5] + w < pi[8]) { //true
            pi[8] = pi[5] + w = 3+16=19;
            p[8] = 5;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	34
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



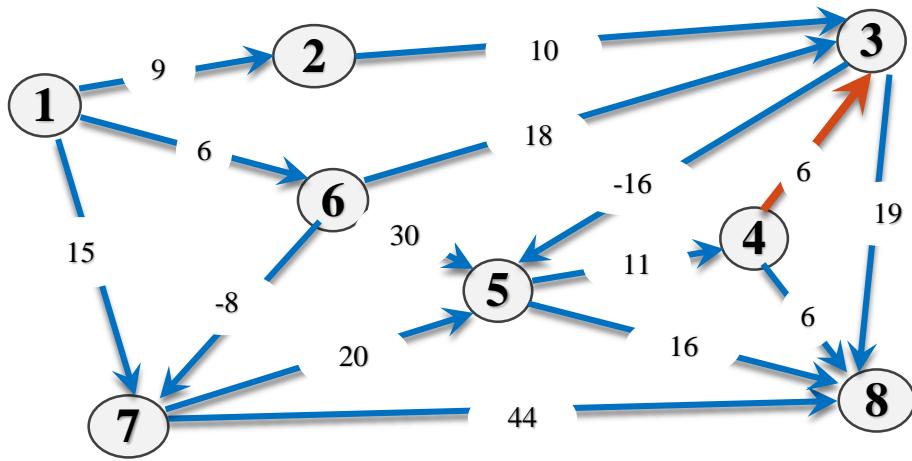
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=10
        u = G->edges[10].u = 5;
        v = G->edges[10].v = 8;
        w = G->edges[10].w = 16;
        if (pi[5] + w < pi[8]) { //true
            pi[8] = pi[5] + w = 3+16=19;
            p[8] = 5;
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



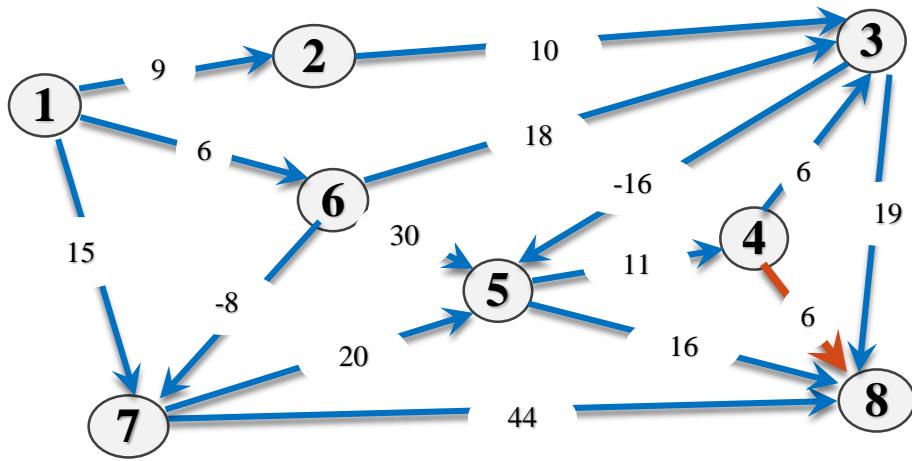
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=11
        u = G->edges[11].u = 4;
        v = G->edges[11].v = 3;
        w = G->edges[11].w = 6;
        if (pi[4] + w < pi[3]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



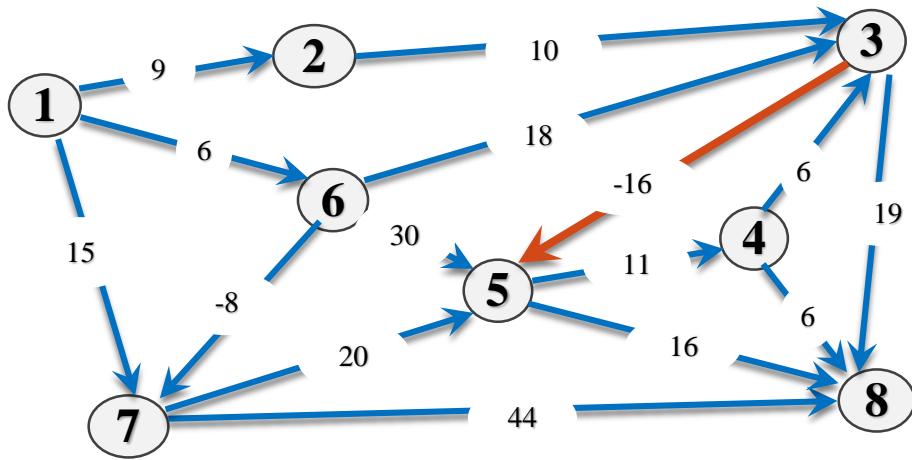
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=12
        u = G->edges[12].u = 4;
        v = G->edges[12].v = 8;
        w = G->edges[12].w = 6;
        if (pi[4] + w < pi[8]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



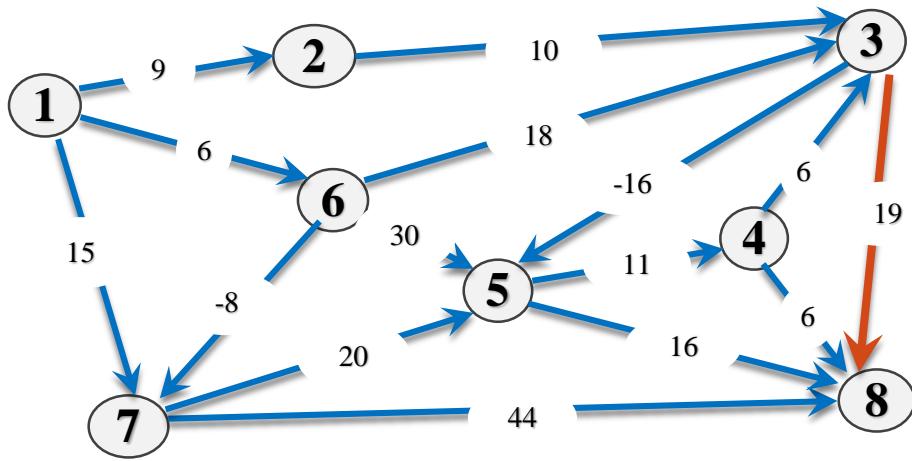
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=13
        u = G->edges[13].u = 3;
        v = G->edges[13].v = 5;
        w = G->edges[13].w = -16;
        if (pi[3] + w < pi[5]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



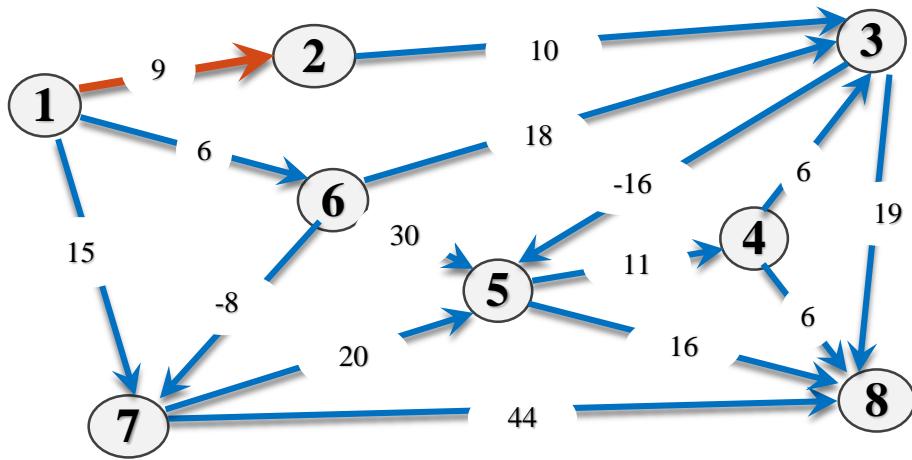
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=14
        u = G->edges[14].u = 3;
        v = G->edges[14].v = 8;
        w = G->edges[14].w = 19;
        if (pi[3] + w < pi[8]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



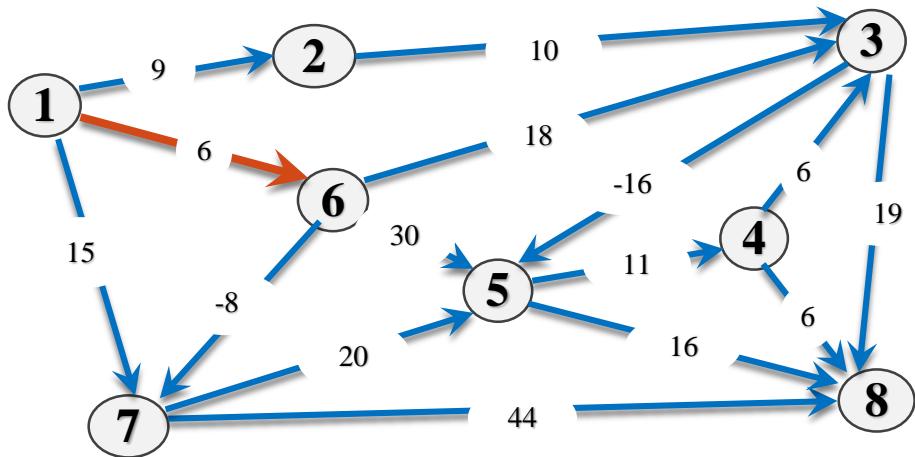
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=0
        u = G->edges[0].u = 1;
        v = G->edges[0].v = 2;
        w = G->edges[0].w = 9;
        if (pi[u] + w < pi[v]) { //False
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



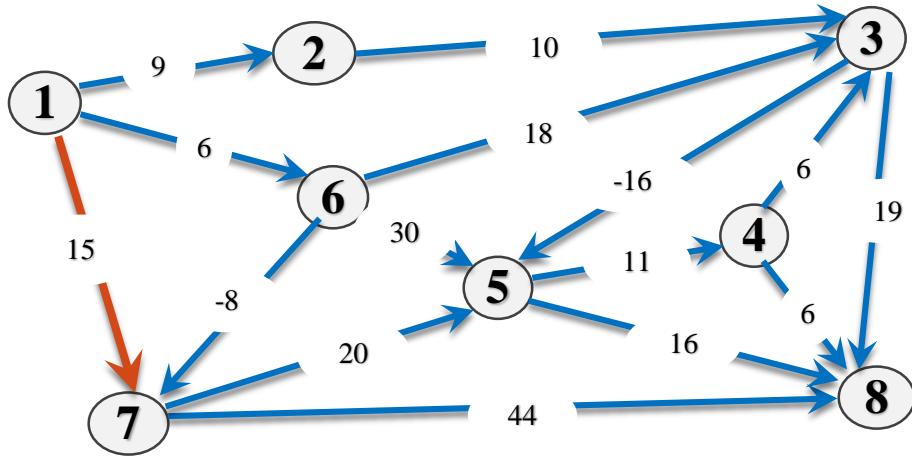
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=1
        u = G->edges[1].u = 1;
        v = G->edges[1].v = 6;
        w = G->edges[1].w = 6;
        if (pi[u] + w < pi[v]) { //False
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



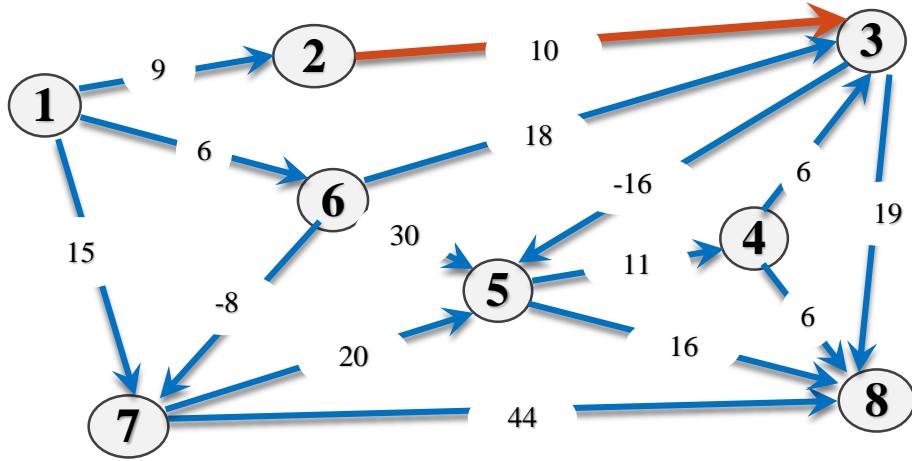
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=2
        u = G->edges[2].u = 1;
        v = G->edges[2].v = 7;
        w = G->edges[2].w = 17;
        if (pi[1] + w < pi[7]) { //False
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



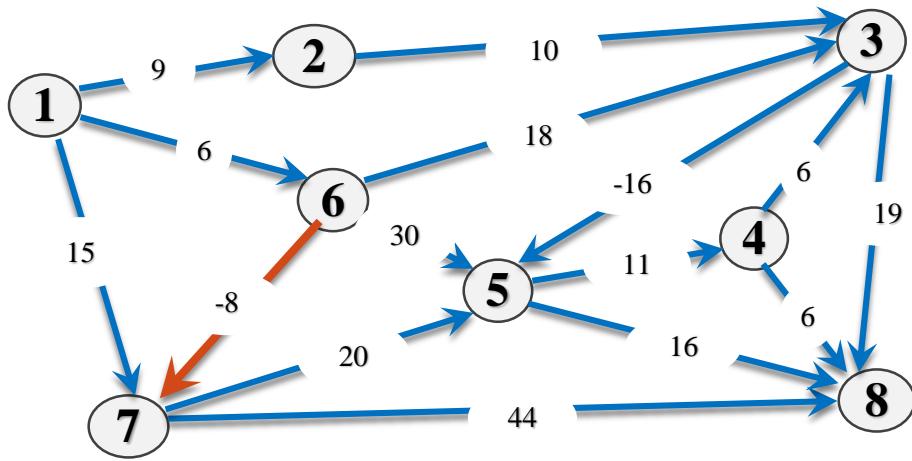
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=3
        u = G->edges[3].u = 2;
        v = G->edges[3].v = 3;
        w = G->edges[3].w = 10;
        if (pi[2] + w < pi[3]) { //False
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



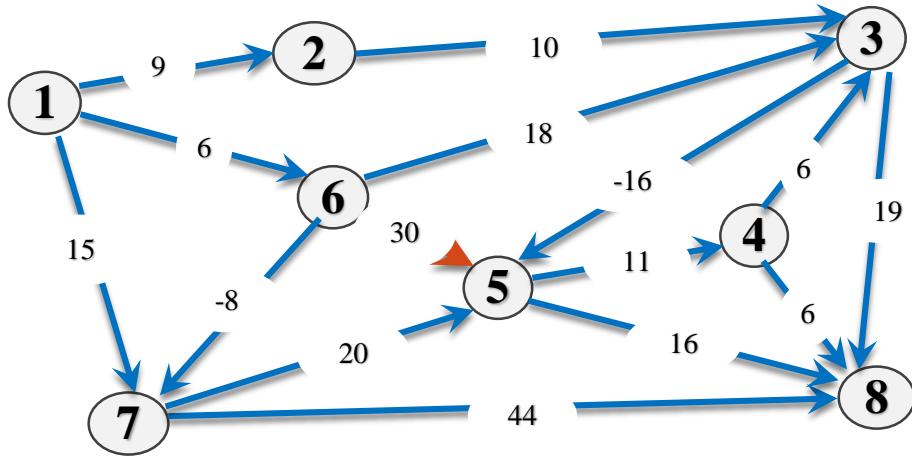
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=4
        u = G->edges[4].u = 6;
        v = G->edges[4].v = 7;
        w = G->edges[4].w = -8;
        if (pi[6] + w < pi[7]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



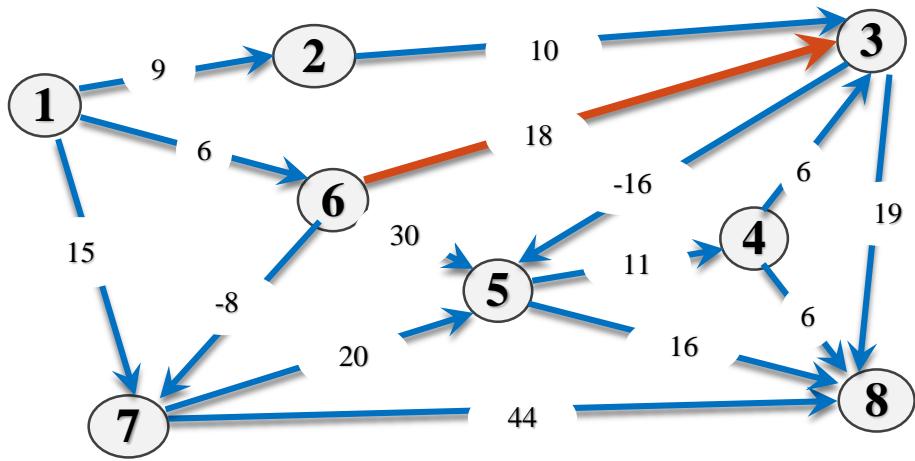
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=5
        u = G->edges[5].u = 6;
        v = G->edges[5].v = 5;
        w = G->edges[5].w = 30;
        if (pi[6] + w < pi[5]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



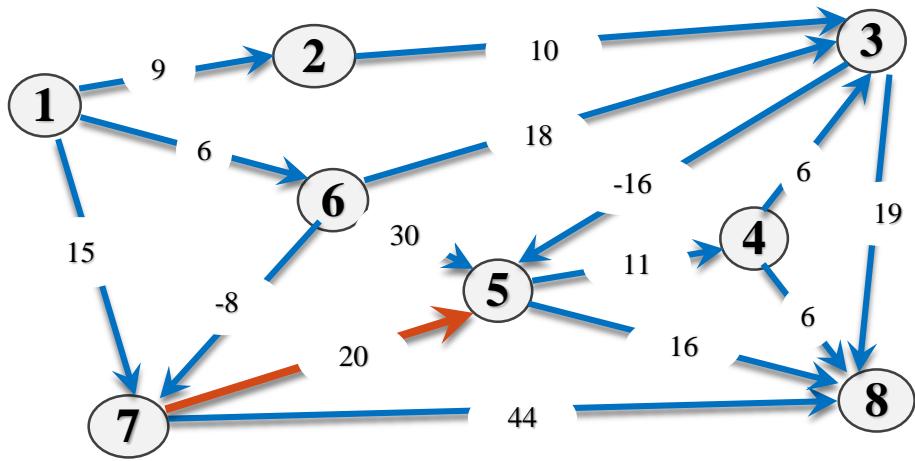
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=6
        u = G->edges[6].u = 6;
        v = G->edges[6].v = 3;
        w = G->edges[6].w = 18;
        if (pi[6] + w < pi[3]) { //false
            //Không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



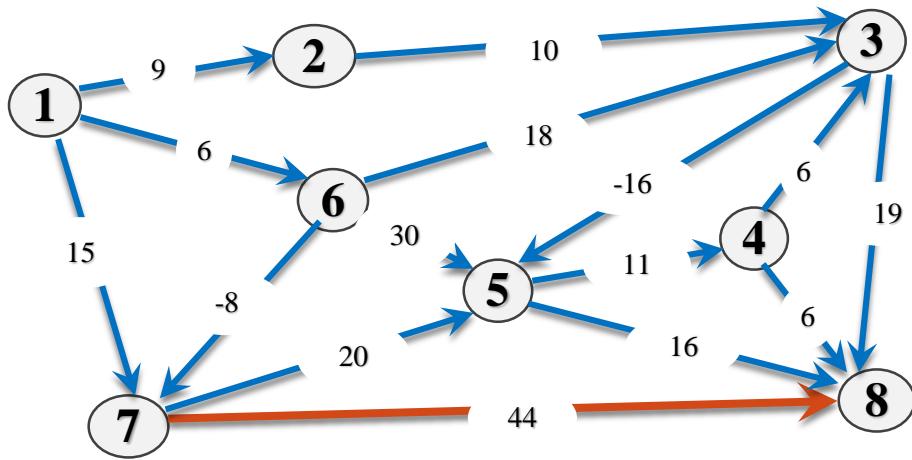
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=7
        u = G->edges[7].u = 7;
        v = G->edges[7].v = 5;
        w = G->edges[7].w = 20;
        if (pi[7] + w < pi[5]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



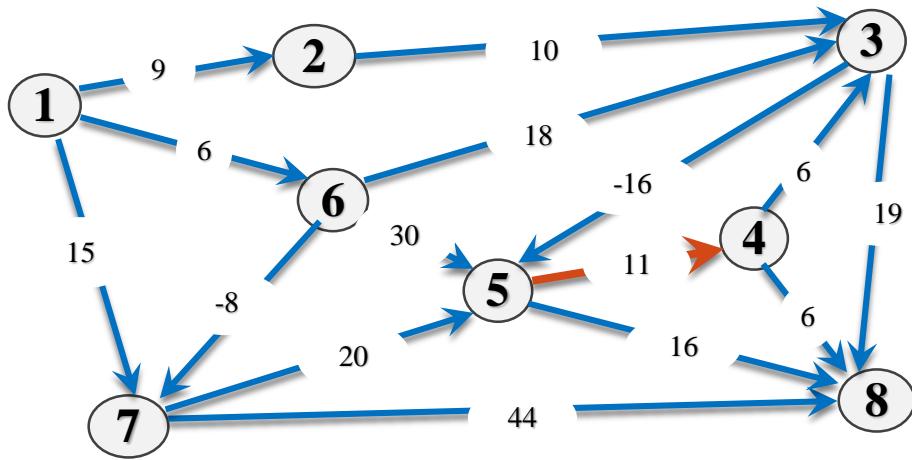
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=8
        u = G->edges[8].u = 7;
        v = G->edges[8].v = 8;
        w = G->edges[8].w = 44;
        if (pi[7] + w < pi[8]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



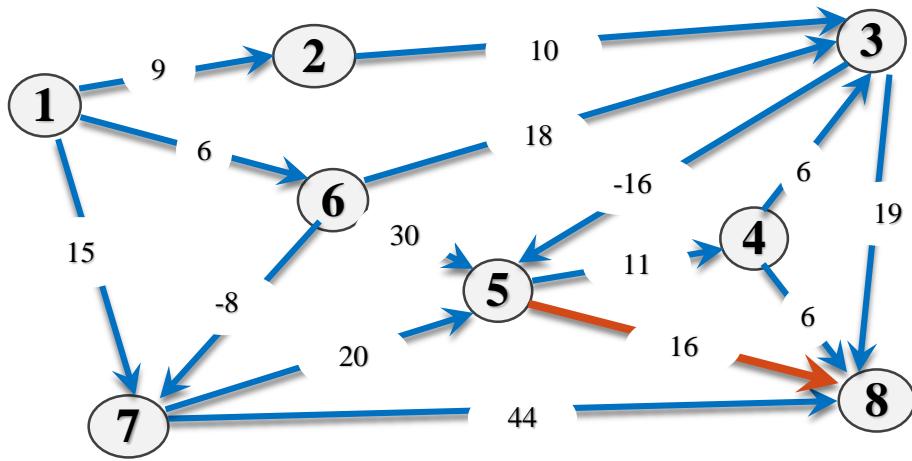
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=9
        u = G->edges[9].u = 5;
        v = G->edges[9].v = 4;
        w = G->edges[9].w = 11;
        if (pi[5] + w < pi[4]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



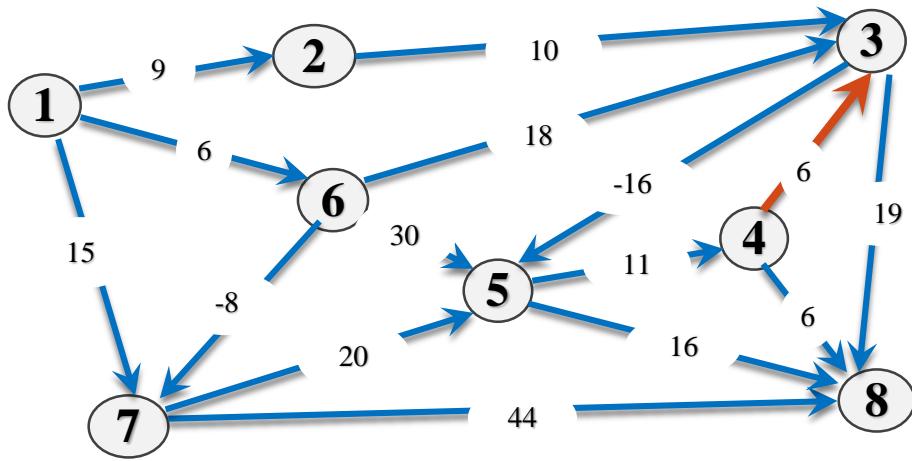
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=10
        u = G->edges[10].u = 5;
        v = G->edges[10].v = 8;
        w = G->edges[10].w = 16;
        if (pi[5] + w < pi[8]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



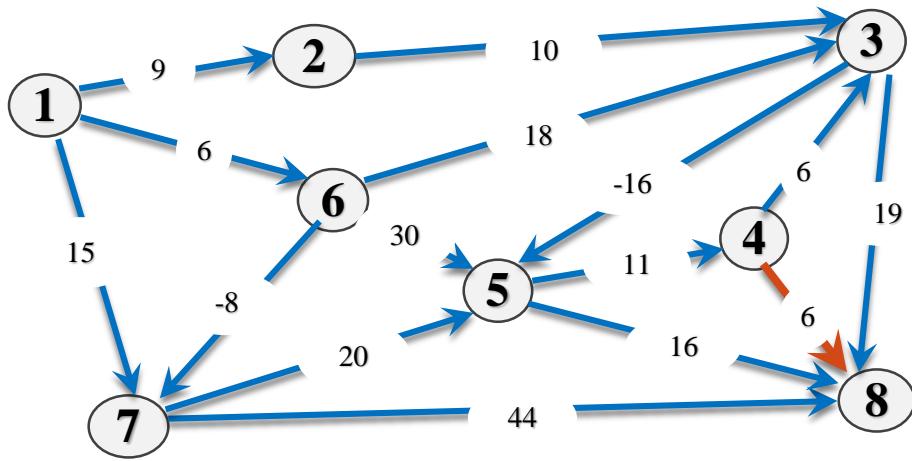
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=11
        u = G->edges[11].u = 4;
        v = G->edges[11].v = 3;
        w = G->edges[11].w = 6;
        if (pi[4] + w < pi[3]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



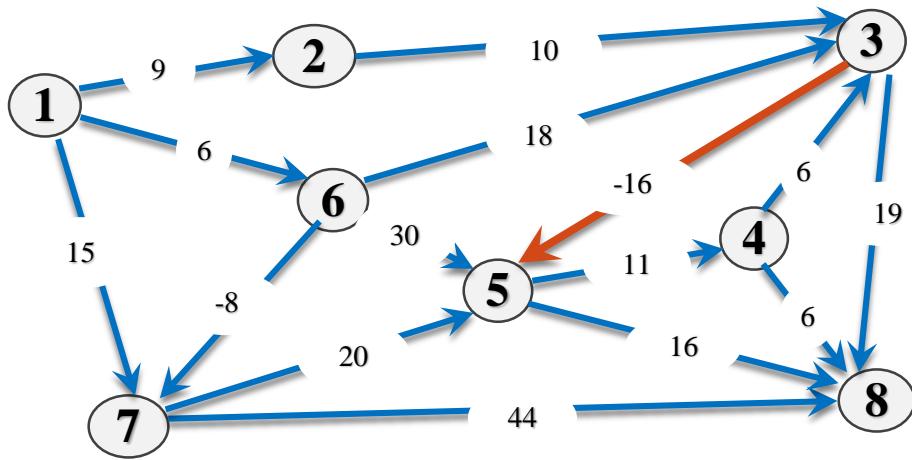
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=12
        u = G->edges[12].u = 4;
        v = G->edges[12].v = 8;
        w = G->edges[12].w = 6;
        if (pi[4] + w < pi[8]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



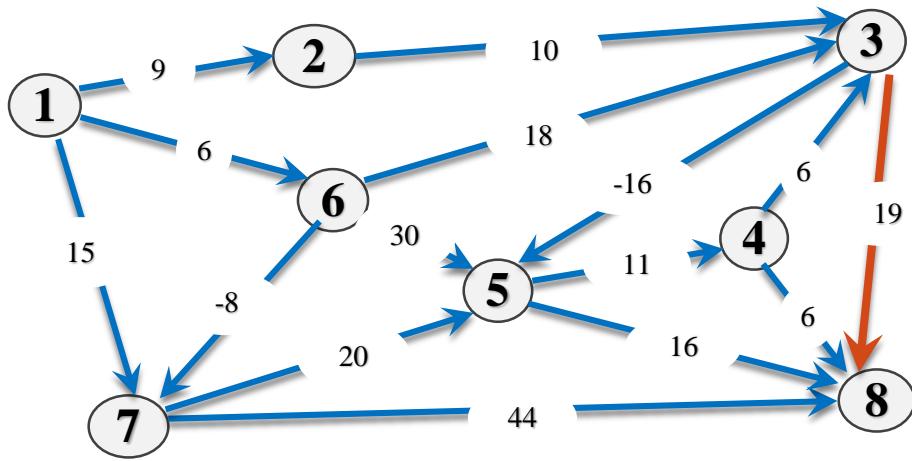
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=13
        u = G->edges[13].u = 3;
        v = G->edges[13].v = 5;
        w = G->edges[13].w = -16;
        if (pi[3] + w < pi[5]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



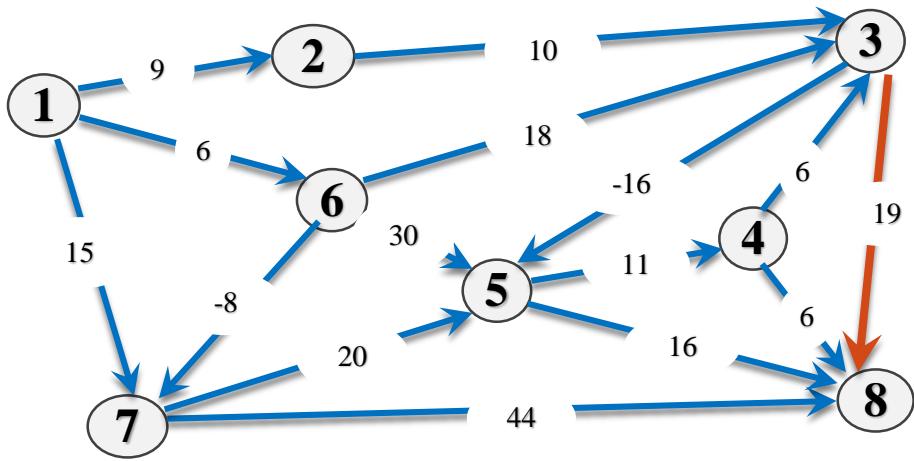
```

for (it = 1; it < G->n; it++) { //it=3
    for (k = 0; k < G->m; k++) { //k=14
        u = G->edges[14].u = 3;
        v = G->edges[14].v = 8;
        w = G->edges[14].w = 19;
        if (pi[3] + w < pi[8]) { //false
            //không làm gì cả
        }
    }
}

```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19



```

for (it = 1; it < G->n; it++) {
//it=4, 5, 6, 7
    for (k = 0; k < G->m; k++) {
        u = G->edges[k].u;
        v = G->edges[k].v;
        w = G->edges[k].w;
        if (pi[u] + w < pi[v]) { //false
            //không làm gì cả
        }
    }
}

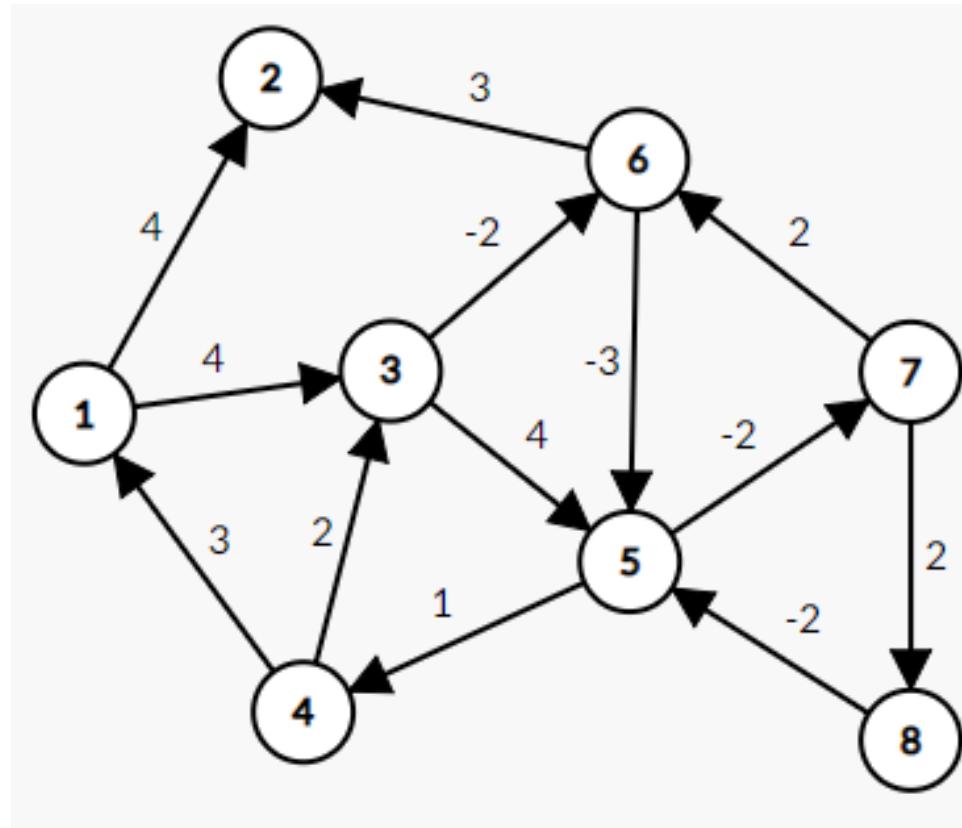
```

G	1	2	3	4	5	6	7	8
pi(u)	0	9	19	14	3	6	-2	19
p(i)	-1	1	2	5	3	1	6	5

u	v	w
1	2	9
1	6	6
1	7	17
2	3	10
6	7	-8
6	5	30
6	3	18
7	5	20
7	8	44
5	4	11
5	8	16
4	3	6
4	8	6
3	5	-16
3	8	19

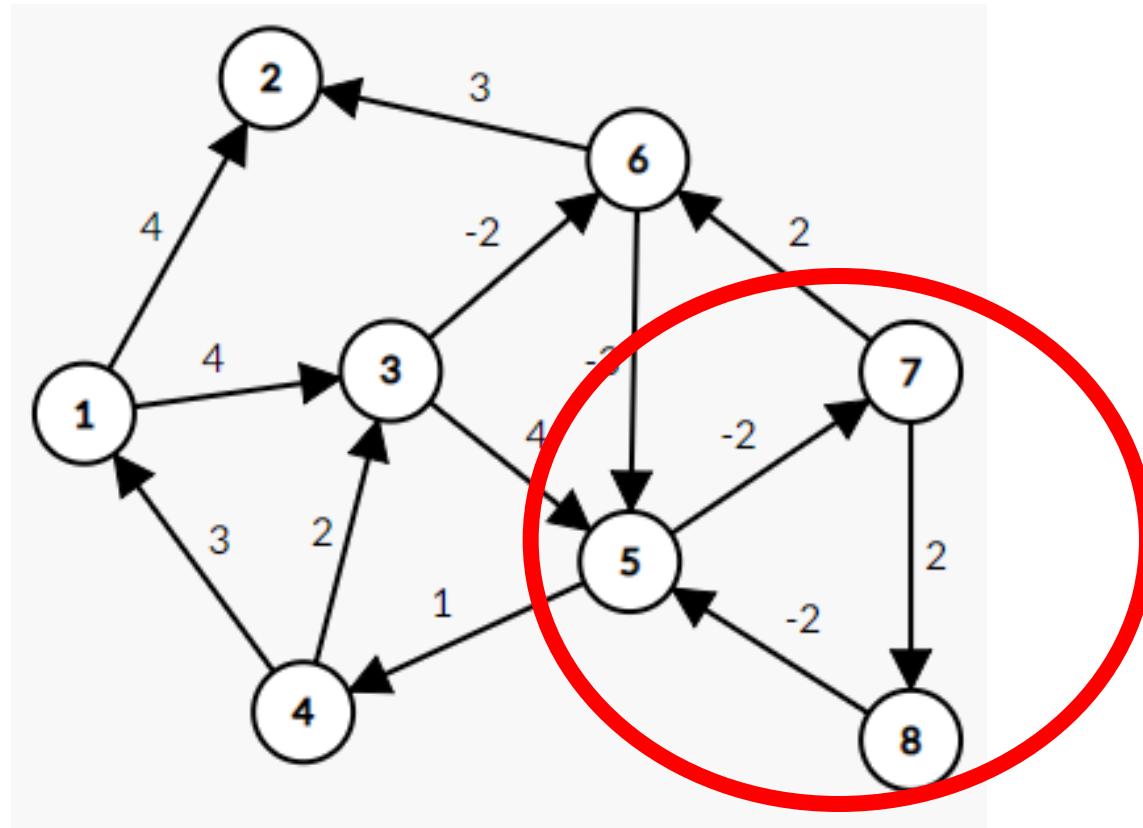
# Giải thuật Bellman – Ford

Đồ thị sau đây có chu trình âm hay không? Nếu có, hãy chỉ ra chu trình đó.



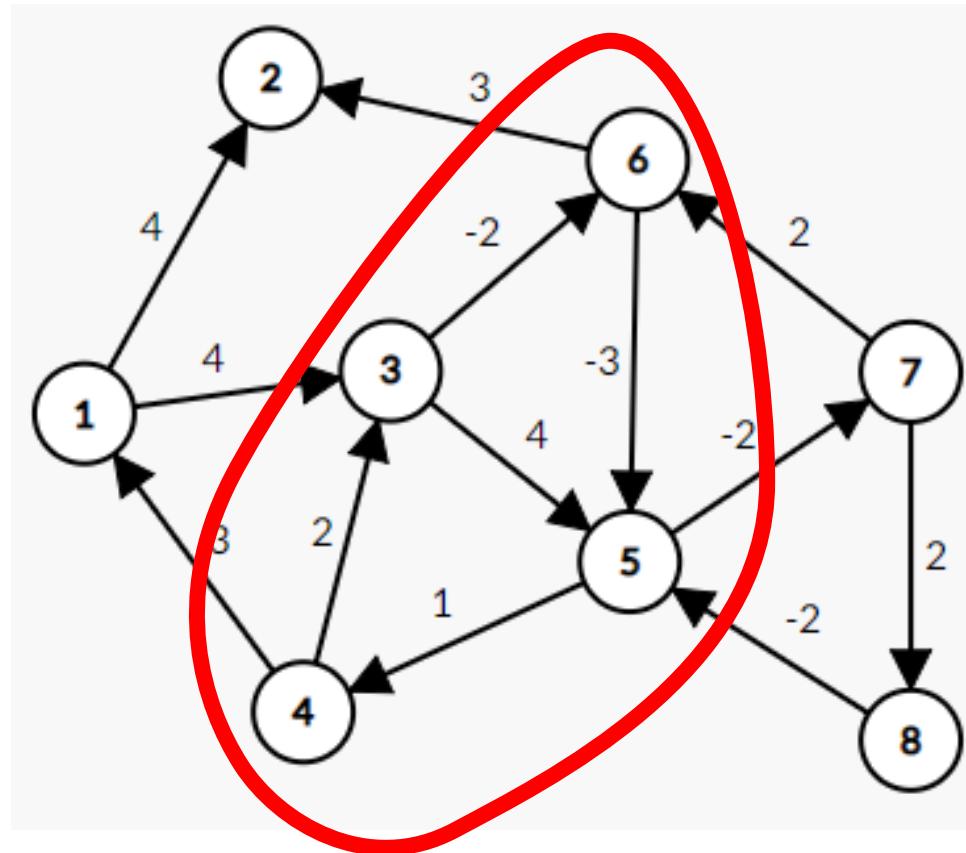
# Giải thuật Bellman – Ford

Đồ thị sau đây có chu trình âm hay không? Nếu có, hãy chỉ ra chu trình đó.



# Giải thuật Bellman – Ford

Đồ thị sau đây có chu trình âm hay không? Nếu có, hãy chỉ ra chu trình đó.



# Giải thuật Bellman – Ford

```
// Kiểm tra chu trình âm bằng cách
// Duyệt qua các cung một lần nữa
for (k = 0; k < G->m; k++) {
    int u = G->edges[k].u;
    int v = G->edges[k].v;
    int w = G->edges[k].w;
    if (pi[u] + w < pi[v]) {
        // Có chu trình âm
        ...
        break;
    }
}
```

# Thuật toán Floyd\_Warshall

```
void Floyd_Warshall(Graph* pG) {  
    int u, v, k;  
    for (u = 1; u <= pG->n; u++)  
        for (v = 1; v <= pG->n; v++) {  
            pi[u][v] = INFINITY;  
            next[u][v] = -1;  
        }  
    for (u = 1; u <= pG->n; u++)  
        p[u][u] = 0;  
  
    for (u = 1; u <= pG->n; u++)  
        for (v = 1; v <= pG->n; v++)  
            if (pG->L[u][v] != NO_EDGE) {  
                pi[u][v] = pG->L[u][v]; //đi trực tiếp từ u -> v  
                next[u][v] = v;  
            }  
  
    for (k = 1; k <= pG->n; k++)  
        for (u = 1; u <= pG->n; u++)  
            for (v = 1; v <= pG->n; v++)  
                if (pi[u][k] + p[k][v] < pi[u][v]) {  
                    pi[u][k] = pi[u][k] + pi[k][v];  
                    next[u][v] = next[u][k];  
                }  
    //Kiểm tra chu trình âm (nếu cần thiết)  
}
```