

LECTURE 11

FLOYD-WARSHALL ALGORITHM

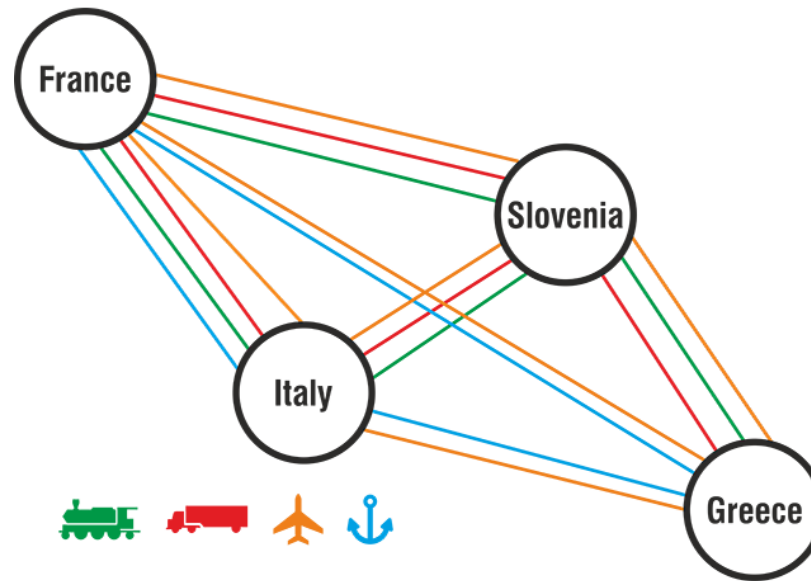


Big-O Coding

Website: www.bigocoding.com

Floyd-Warshall

Floyd Warshall là thuật toán tìm đường đi ngắn nhất giữa **tất cả các cặp đỉnh** trong đồ thị **có hướng**, có trọng số (trọng số có thể dương hoặc âm). Không chạy được với đồ thị có chu trình âm.

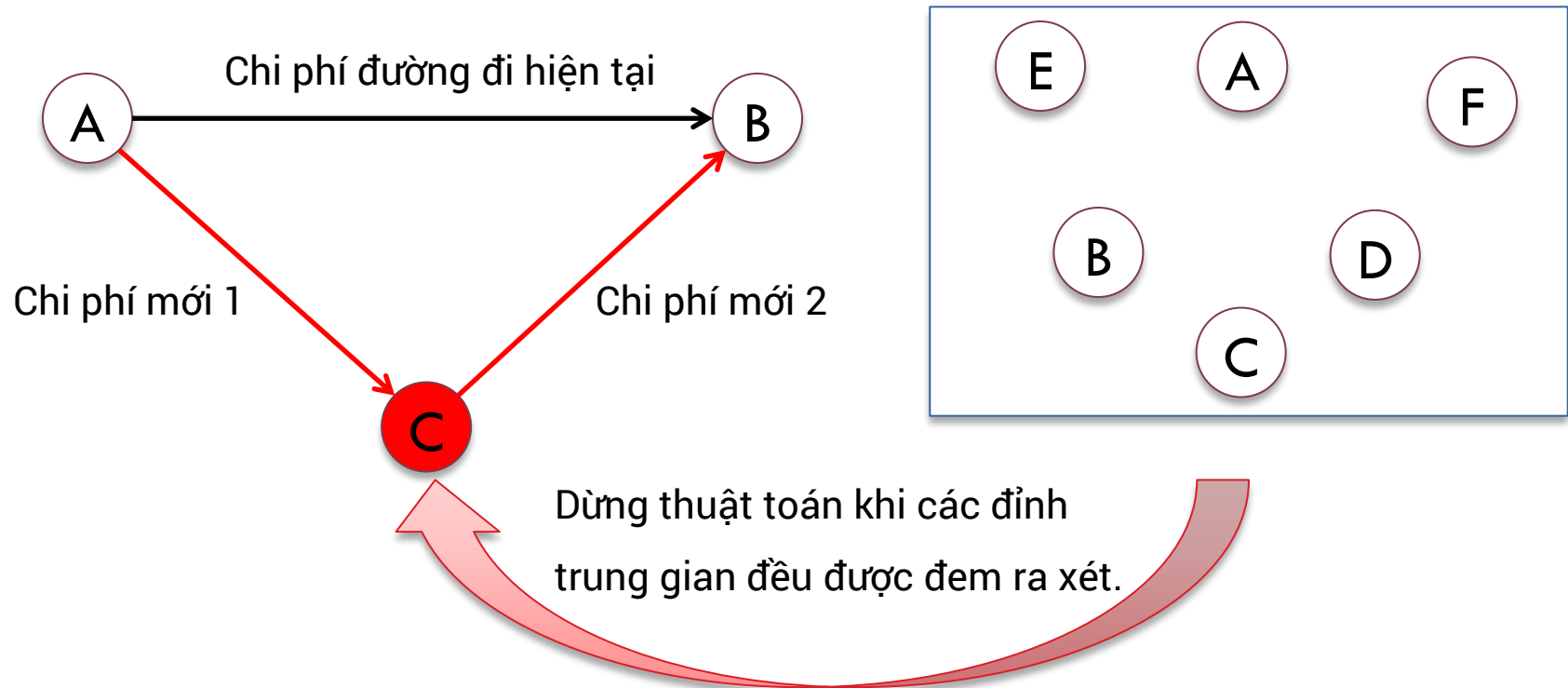


Độ phức tạp: $O(V^3)$

- Với **V (Vertex)** là số đỉnh.

Ý tưởng của thuật toán

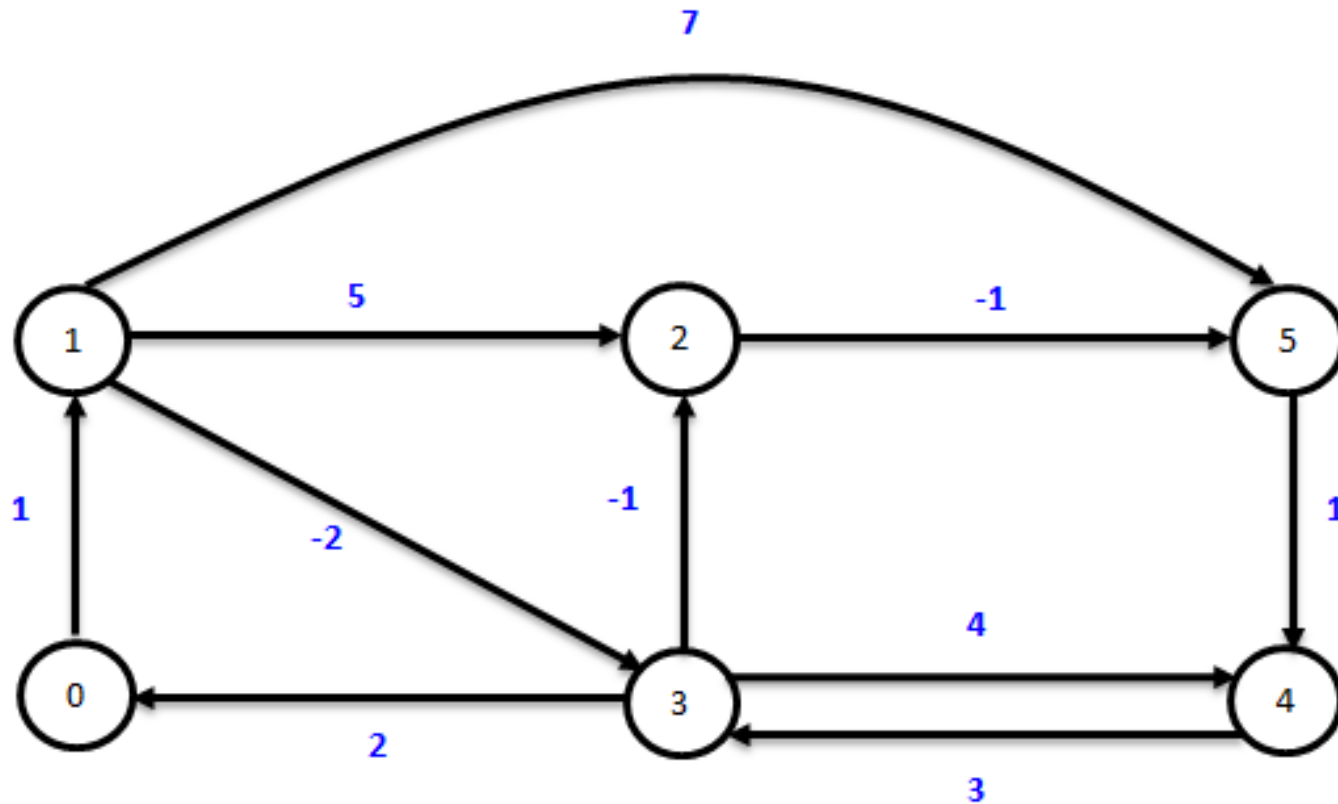
Thuật toán dùng phương pháp Quy Hoạch Động (Dynamic Programming) lưu tất cả các kết quả có được ban đầu vào ma trận.



Nếu: Chi phí đường đi hiện tại > chi phí mới 1 + chi phí mới 2
→ Cập nhật chi phí mới vào ma trận kết quả và lưu vết đỉnh cha.

Bài toán minh họa

Cho đồ thị **có hướng** như hình vẽ. Tìm đường đi **ngắn nhất** từ **đỉnh 0** đến **tất cả** các đỉnh khác.



BƯỚC 0: CHUẨN BỊ DỮ LIỆU

Bước 0: Chuẩn bị dữ liệu (1)

Từ **ma trận kề** cho trước, chuyển thông tin vào các cấu trúc dữ liệu cần thiết.

Adjacency Matrix

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

*Chuyển ma trận kề vào **graph**.*

	0	1	2	3	4	5
0	0	1	∞	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	∞	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

Bước 0: Chuẩn bị dữ liệu (2)

Ma trận chứa chi phí đường đi **dist** chuyển từ ma trận **graph**.
với $\text{dist}[i][j]$ là chi phí đường đi ngắn nhất hiện tại từ đỉnh i đến đỉnh j .

	0	1	2	3	4	5
0	0	1	∞	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	∞	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

Bước 0: Chuẩn bị dữ liệu (3)

Ma trận lưu vết đường đi **path**, cặp đỉnh nào có liên kết với nhau thì giá trị của đỉnh đến là đỉnh bắt đầu, ngược lại là -1.

	0	1	2	3	4	5
0	-1	0	-1	-1	-1	-1
1	-1	-1	1	1	-1	1
2	-1	-1	-1	-1	-1	2
3	3	-1	3	-1	3	-1
4	-1	-1	-1	4	-1	-1
5	-1	-1	-1	-1	5	-1

BƯỚC 1

CHẠY VÒNG LẶP DUYỆT QUA MA TRẬN KÈ LẦN 1

Bước 1: Chạy thuật toán ($k=0$)

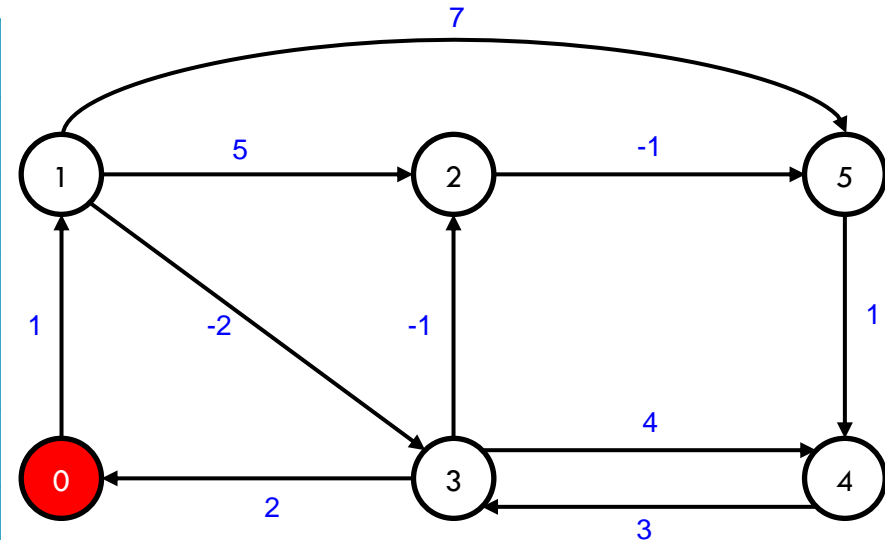
Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

$$\underset{0}{\text{dist}}[\underset{0}{0}][\underset{0}{0}] > \underset{0}{\text{dist}}[\underset{0}{0}][\underset{0}{0}] + \underset{0}{\text{dist}}[\underset{0}{0}][\underset{0}{0}] \quad \text{X}$$

$j=0$

	0	1	2	3	4	5
$i=0$ 0	0	1	∞	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	∞	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

dist



Bước 1: Chạy thuật toán ($k=0$)

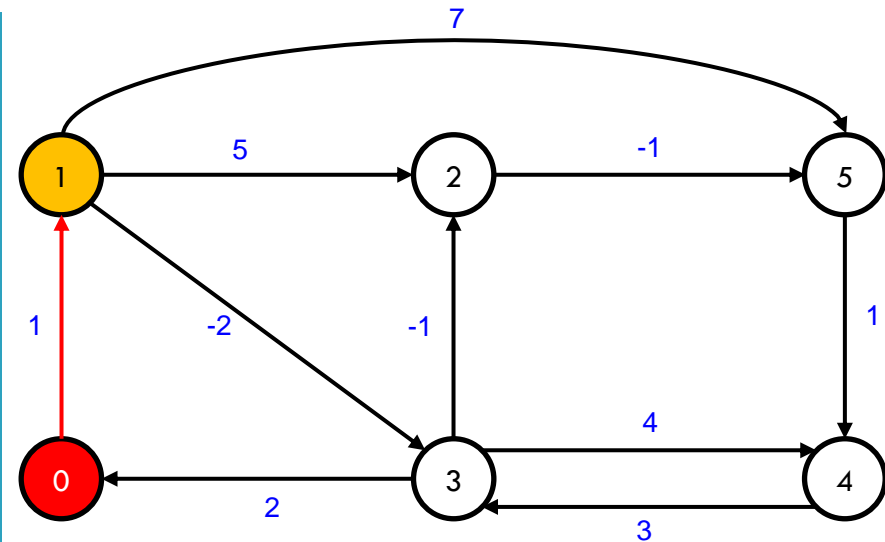
Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

$$\underset{1}{\text{dist}[0][1]} > \underset{0}{\text{dist}[0][0]} + \underset{1}{\text{dist}[0][1]} \quad \text{X}$$

$j=1$

	0	1	2	3	4	5
$i=0$ 0	0	1	∞	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	∞	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

dist



Bước 1: Chạy thuật toán (**k=0**)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

$$\text{dist}[0][2] > \text{dist}[0][0] + \text{dist}[0][2]$$

 ∞

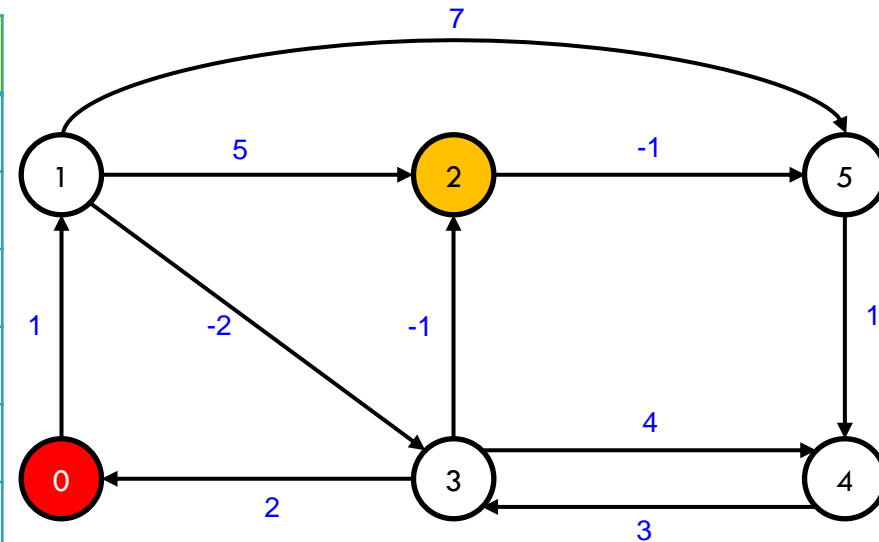
0

 ∞  $j=2$

$i=0$

	0	1	2	3	4	5
0	0	1	∞	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	∞	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

dist



Bước 1: Chạy thuật toán (**k=0**)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

$$\text{dist}[0][3] > \text{dist}[0][0] + \text{dist}[0][3] \quad \text{X}$$

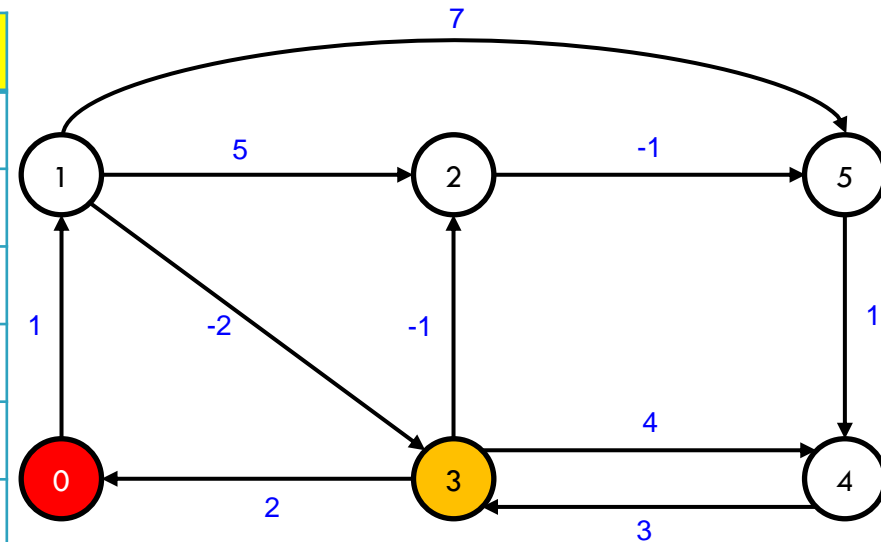
∞ 0 ∞

$j=3$

$i=0$

	0	1	2	3	4	5
0	0	1	∞	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	∞	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

dist



Bước 1: Chạy thuật toán (**k=0**)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

$$\text{dist}[0][4] > \text{dist}[0][0] + \text{dist}[0][4]$$

 ∞

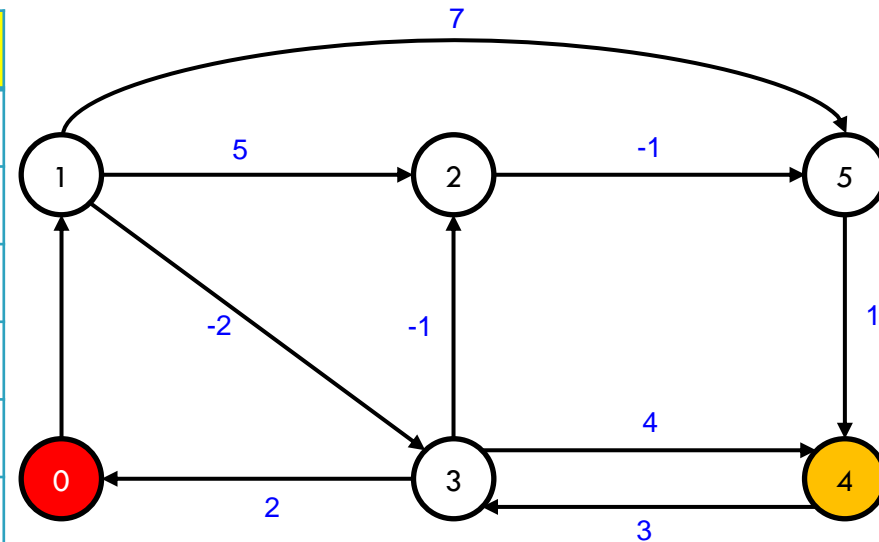
0

 ∞  $j=4$

$i=0$

	0	1	2	3	4	5
0	0	1	∞	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	∞	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

dist



Bước 1: Chạy thuật toán ($k=0$)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

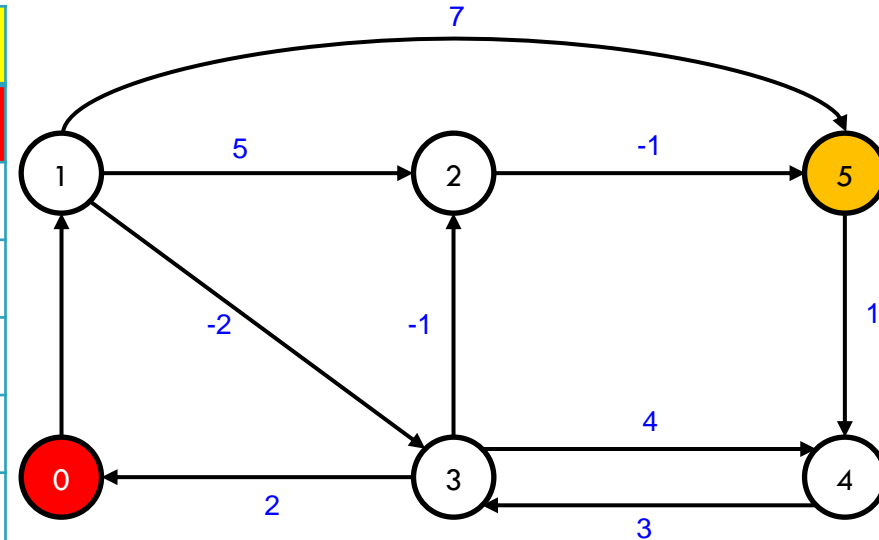
$$\underset{\infty}{\text{dist}[0][5]} > \underset{0}{\text{dist}[0][0]} + \underset{\infty}{\text{dist}[0][5]} \quad \text{X}$$

$i=0$

$j=5$

	0	1	2	3	4	5
0	0	1	∞	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	∞	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

dist



→ Với $i = 0$, không có bất kỳ sự thay đổi nào.

Bước 1: Chạy thuật toán ($k=0$)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

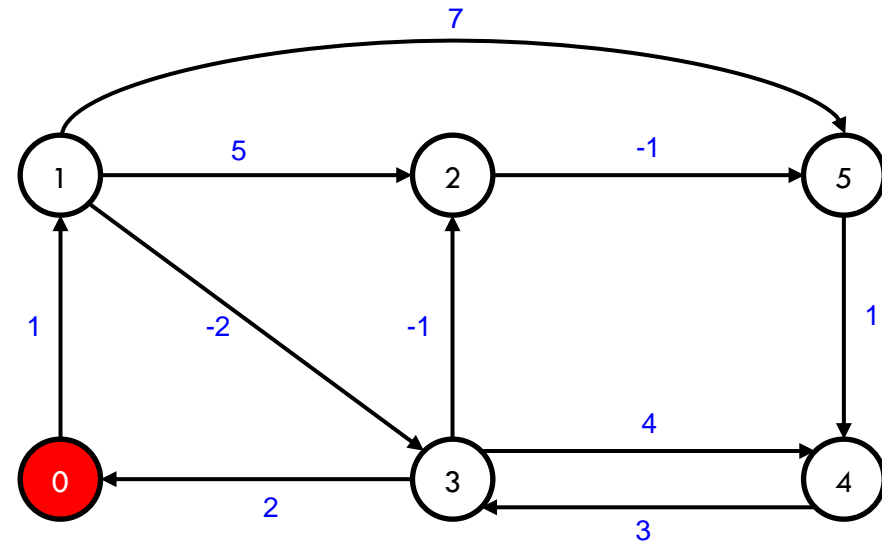
$j = 0, j = 1, j = 2, j = 3, j = 4, j = 5$ ❌

$j=0 \longrightarrow j=5$

$i=1$

	0	1	2	3	4	5
0	0	1	∞	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	∞	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

dist



→ Với $i = 1$, không có bất kỳ sự thay đổi nào.

Bước 1: Chạy thuật toán ($k=0$)

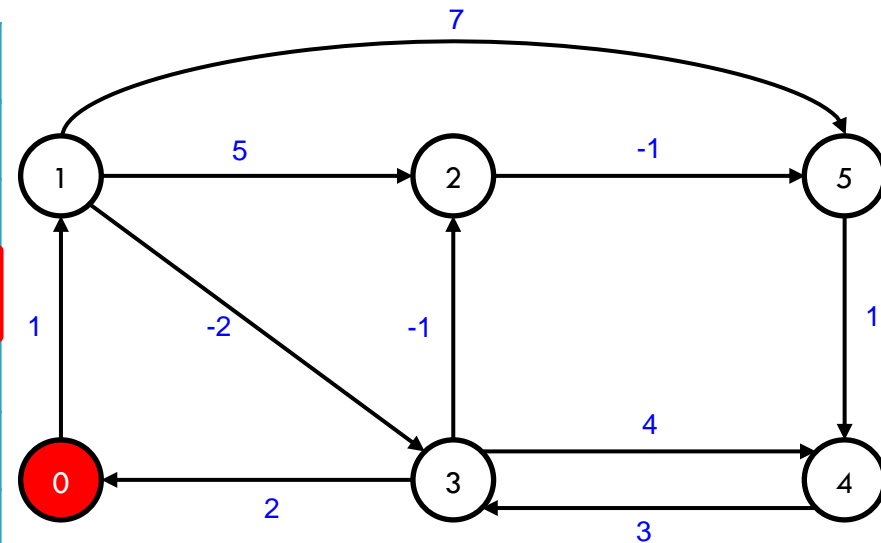
Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

$j = 0, j = 1, j = 2, j = 3, j = 4, j = 5$ ❌

$j=0 \longrightarrow j=5$

	0	1	2	3	4	5
0	0	1	∞	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	∞	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

dist



➔ Với $i = 2$, không có bất kỳ sự thay đổi nào.

Bước 1: Chạy thuật toán ($k=0$)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

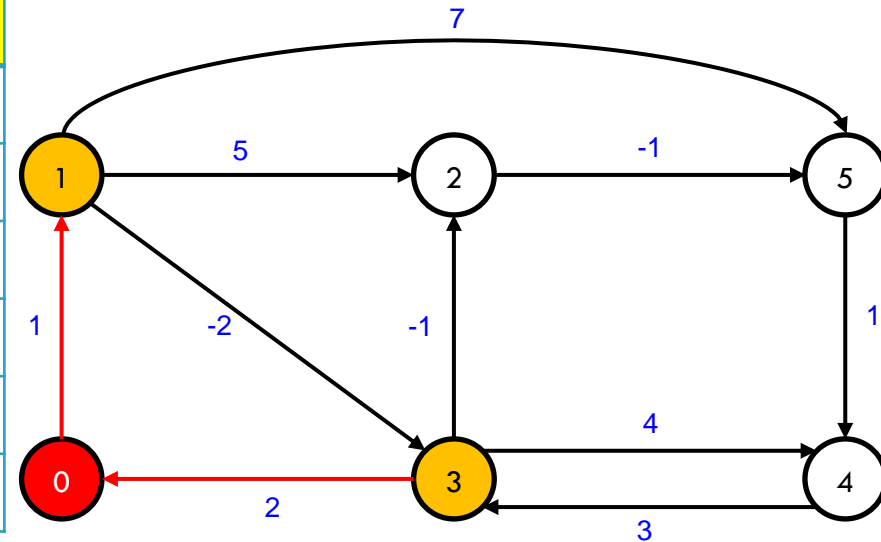
1 $\text{dist}[3][1] > \text{dist}[3][0] + \text{dist}[0][1]$ ✓

∞ 2 1

→ $\text{dist}[3][1] = 2 + 1 = 3$

$j=1$

	0	1	2	3	4	5
0	0	1	∞	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	(∞) 3	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0



dist

Bước 1: Chạy thuật toán ($k=0$)

1 Cập nhật $\text{path}[i][j] = \text{path}[k][j]$

$$\text{path}[3][1] = \text{path}[0][1] = 0$$

$j=1$

$i=3$

	0	1	2	3	4	5
0	-1	0	-1	-1	-1	-1
1	-1	-1	1	1	-1	1
2	-1	-1	-1	-1	-1	2
3	3	$(-1) 0$	3	-1	3	-1
4	-1	-1	-1	4	-1	-1
5	-1	-1	-1	-1	5	-1

path

Bước 1: Chạy thuật toán ($k=0$)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

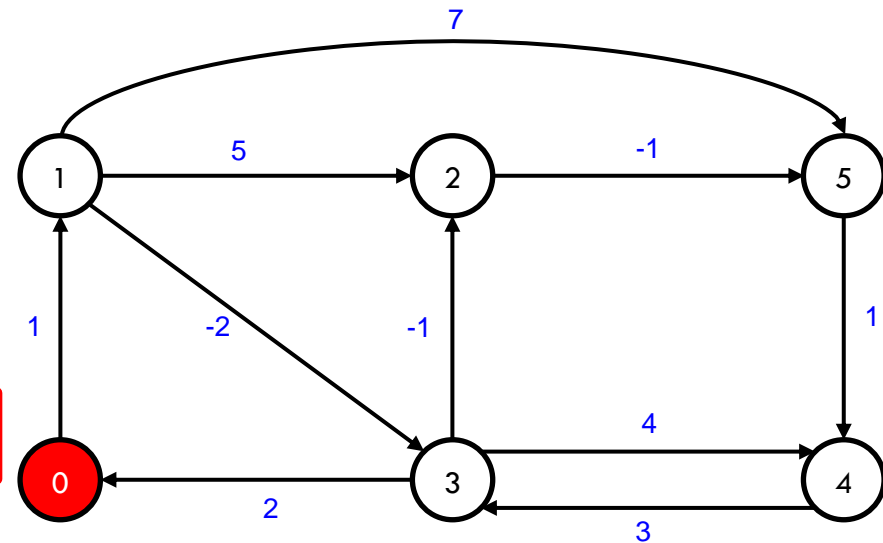
$j = 0, j = 1, j = 2, j = 3, j = 4, j = 5$ ❌

$j=0 \longrightarrow j=5$

	0	1	2	3	4	5
0	0	1	∞	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	3	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

$i=4$

dist



➔ Với $i = 4$, không có bất kỳ sự thay đổi nào.

Bước 1: Chạy thuật toán ($k=0$)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

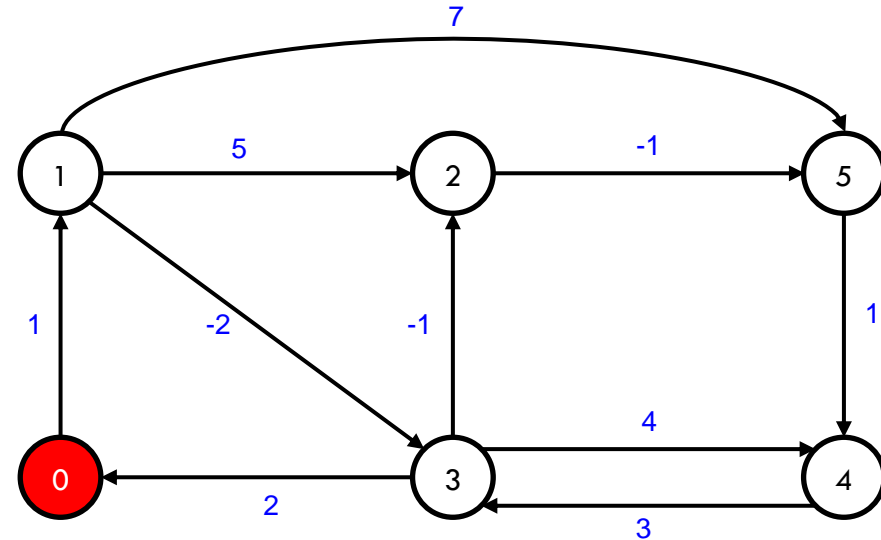
$j = 0, j = 1, j = 2, j = 3, j = 4, j = 5$ ❌

$j=0 \longrightarrow j=5$

	0	1	2	3	4	5
0	0	1	∞	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	3	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

$i=5$

dist



→ Với $i = 5$, không có bất kỳ sự thay đổi nào.

BƯỚC 2

CHẠY VÒNG LẶP DUYỆT QUA MA

TRẬN KÈ LẦN 2

Bước 2: Chạy thuật toán ($k=1$)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

1 $\text{dist}[0][2] > \text{dist}[0][1] + \text{dist}[1][2]$ ✓

∞ 1 5

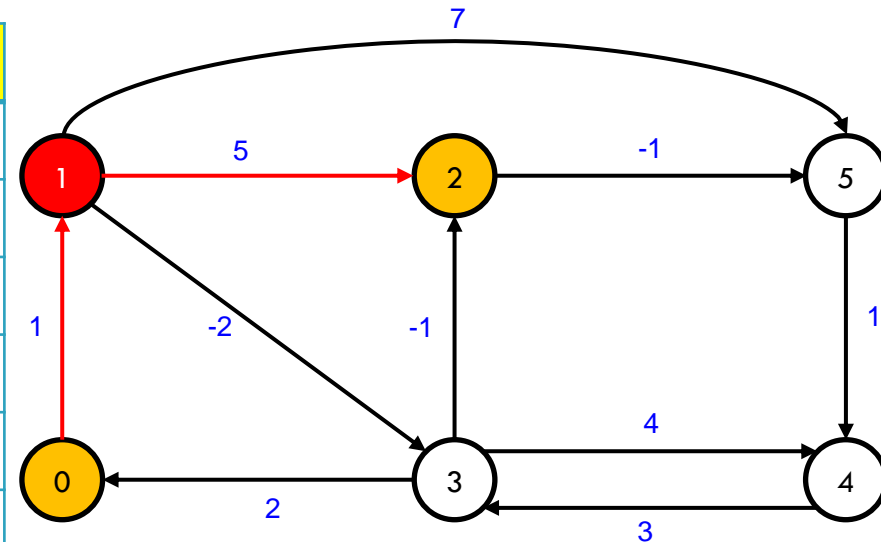
→ $\text{dist}[0][2] = 1 + 5 = 6$

$j=2$

$i=0$

	0	1	2	3	4	5
0	0	1	(∞) 6	∞	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	3	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

dist



Bước 2: Chạy thuật toán ($k=1$)

1 Cập nhật $\text{path}[i][j] = \text{path}[k][j]$

$$\text{path}[0][2] = \text{path}[1][2] = 1$$

$j=2$

$i=0$

	0	1	2	3	4	5
0	-1	0	(-1) 1	-1	-1	-1
1	-1	-1	1	1	-1	1
2	-1	-1	-1	-1	-1	2
3	3	0	3	-1	3	-1
4	-1	-1	-1	4	-1	-1
5	-1	-1	-1	-1	5	-1

path

Bước 2: Chạy thuật toán ($k=1$)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

2 $\text{dist}[0][3] > \text{dist}[0][1] + \text{dist}[1][3]$ ✓

∞ 1 -2

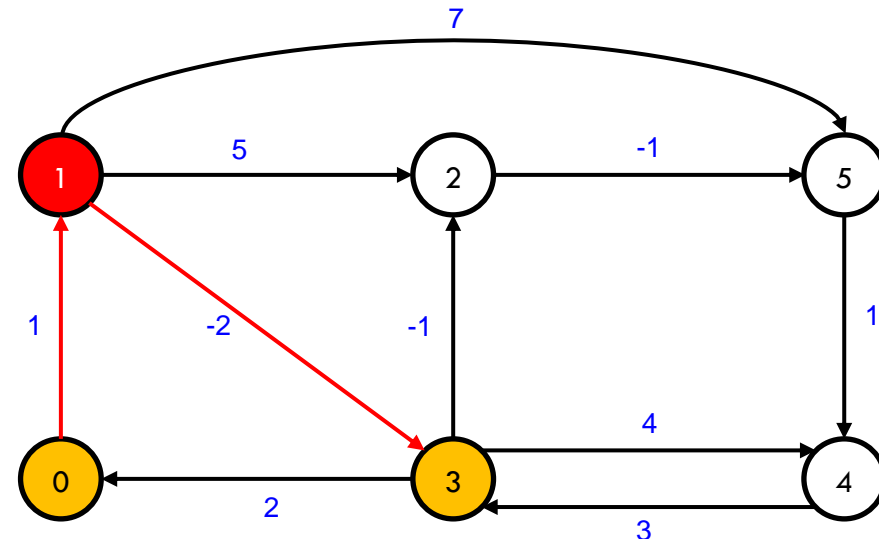
→ $\text{dist}[0][3] = 1 + (-2) = -1$

$j=3$

$i=0$

	0	1	2	3	4	5
0	0	1	6	$(\infty) -1$	∞	∞
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	3	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

dist



Bước 2: Chạy thuật toán ($k=1$)

2

Cập nhật $\text{path}[i][j] = \text{path}[k][j]$

$\text{path}[0][3] = \text{path}[1][3] = 1$

$j=3$

$i=0$

	0	1	2	3	4	5
0	-1	0	1	(-1) 1	-1	-1
1	-1	-1	1	1	-1	1
2	-1	-1	-1	-1	-1	2
3	3	0	3	-1	3	-1
4	-1	-1	-1	4	-1	-1
5	-1	-1	-1	-1	5	-1

path

Bước 2: Chạy thuật toán ($k=1$)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

3 $\text{dist}[0][5] > \text{dist}[0][1] + \text{dist}[1][5]$ ✓

∞

1

7

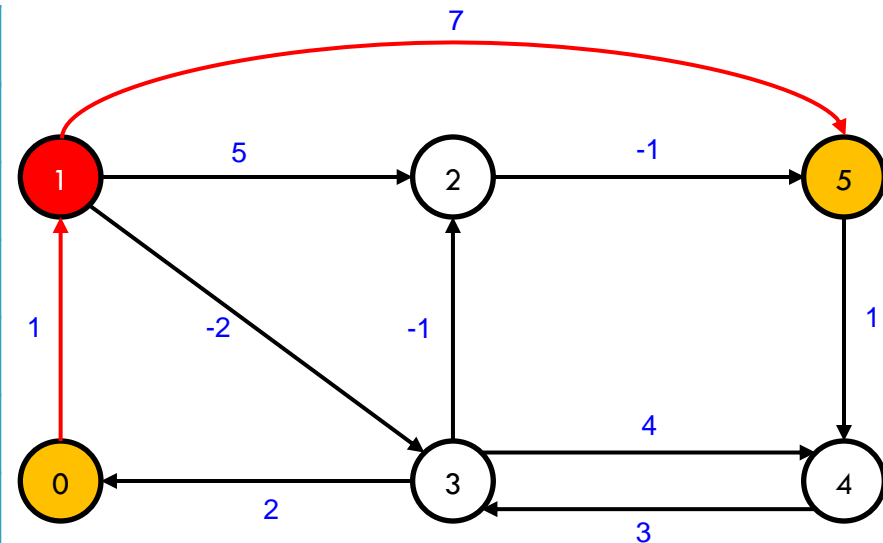
$\Rightarrow \text{dist}[0][5] = 1 + 7 = 8$

$j=5$

$i=0$

	0	1	2	3	4	5
0	0	1	6	-1	∞	(∞) 8
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	3	-1	0	4	∞
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

dist



Bước 2: Chạy thuật toán ($k=1$)

3

Cập nhật $\text{path}[i][j] = \text{path}[k][j]$

$\text{path}[0][5] = \text{path}[1][5] = 1$

$j=5$

$i=0$

	0	1	2	3	4	5
0	-1	0	1	1	-1	(-1) 1
1	-1	-1	1	1	-1	1
2	-1	-1	-1	-1	-1	2
3	3	0	3	-1	3	-1
4	-1	-1	-1	4	-1	-1
5	-1	-1	-1	-1	5	-1

path

Bước 2: Chạy thuật toán ($k=1$)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

4 $\text{dist}[3][5] > \text{dist}[3][1] + \text{dist}[1][5]$ ✓

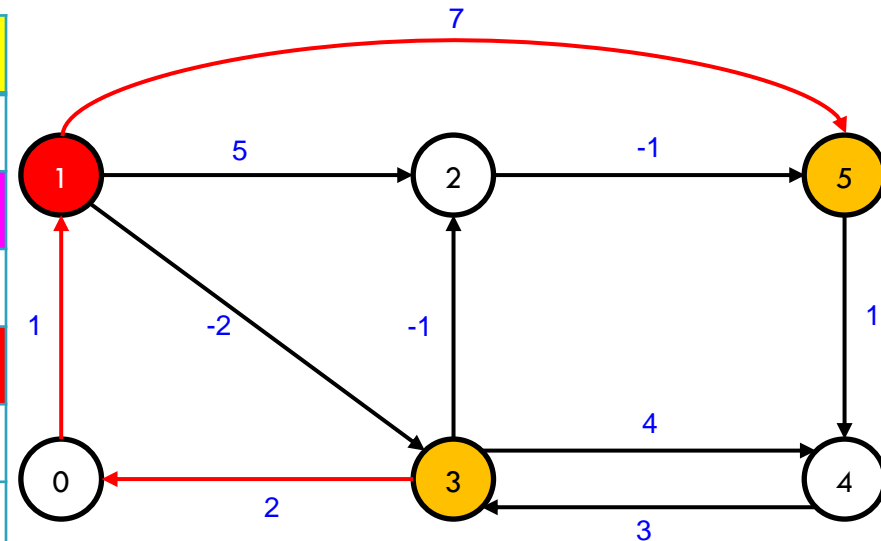
∞ 3 7

→ $\text{dist}[3][5] = 3 + 7 = 10$

$j=5$

	0	1	2	3	4	5
0	0	1	6	-1	∞	8
1	∞	0	5	-2	∞	7
2	∞	∞	0	∞	∞	-1
3	2	3	-1	0	4	(∞)10
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

dist



Bước 2: Chạy thuật toán ($k=1$)

4

Cập nhật: $\text{path}[i][j] = \text{path}[k][j]$

$\text{path}[3][5] = \text{path}[1][5] = 1$

$j=5$

$i=3$

	0	1	2	3	4	5
0	-1	0	1	1	-1	1
1	-1	-1	1	1	-1	1
2	-1	-1	-1	-1	-1	2
3	3	0	3	-1	3	$(-1) 1$
4	-1	-1	-1	4	-1	-1
5	-1	-1	-1	-1	5	-1

An arrow points from the cell at row 1, column 5 (value 1) to the cell at row 3, column 5 (value $(-1) 1$).

BƯỚC 3

CHẠY VÒNG LẶP DUYỆT QUA MA

TRẬN KÈ LẦN 3

Bước 3: Chạy thuật toán ($k=2$)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

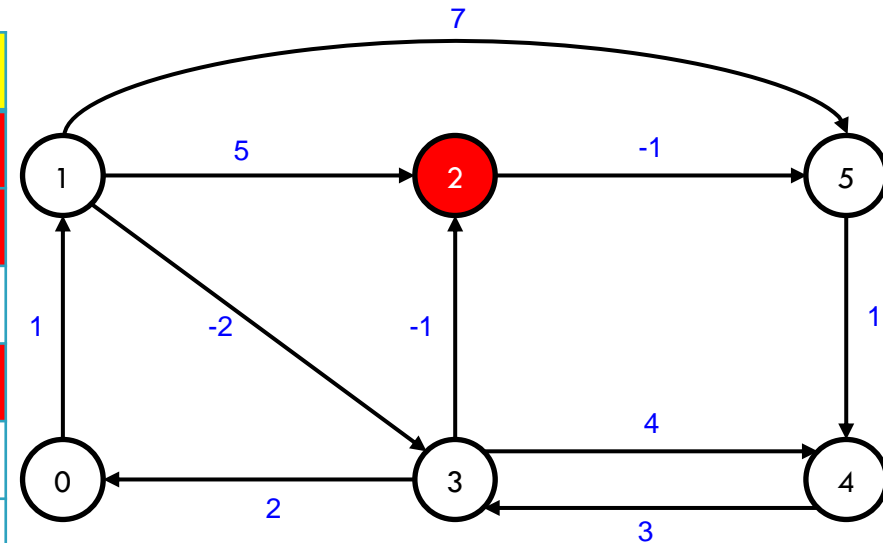
1. $\text{dist}[0][5] = \text{dist}[0][2] + \text{dist}[2][5] = 5$
2. $\text{dist}[1][5] = \text{dist}[1][2] + \text{dist}[2][5] = 4$
3. $\text{dist}[3][5] = \text{dist}[3][2] + \text{dist}[2][5] = -2$

$j=0 \longrightarrow j=5$

$i=0 \downarrow i=5$

	0	1	2	3	4	5
0	0	1	6	-1	∞	(8) 5
1	∞	0	5	-2	∞	(7) 4
2	∞	∞	0	∞	∞	-1
3	2	3	-1	0	4	(10) -2
4	∞	∞	∞	3	0	∞
5	∞	∞	∞	∞	1	0

dist



Bước 3: Chạy thuật toán (k=2)

Cập nhật $\text{path}[i][j] = \text{path}[k][j]$

$$1. \text{path}[0][5] = \text{path}[2][5] = 2$$

$$2. \text{path}[1][5] = \text{path}[2][5] = 2$$

$$3. \text{path}[3][5] = \text{path}[2][5] = 2$$

$j=0 \longrightarrow j=5$

$i=0 \downarrow i=5$

	0	1	2	3	4	5
0	-1	0	1	1	-1	(1) 2
1	-1	-1	1	1	-1	(1) 2
2	-1	-1	-1	-1	-1	2
3	3	0	3	-1	3	(1) 2
4	-1	-1	-1	4	-1	-1
5	-1	-1	-1	-1	5	-1

path

BƯỚC 4

CHẠY VÒNG LẶP DUYỆT QUA CÁC DANH SÁCH KÈ LẦN 4

Bước 4: Chạy thuật toán ($k=3$)

Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

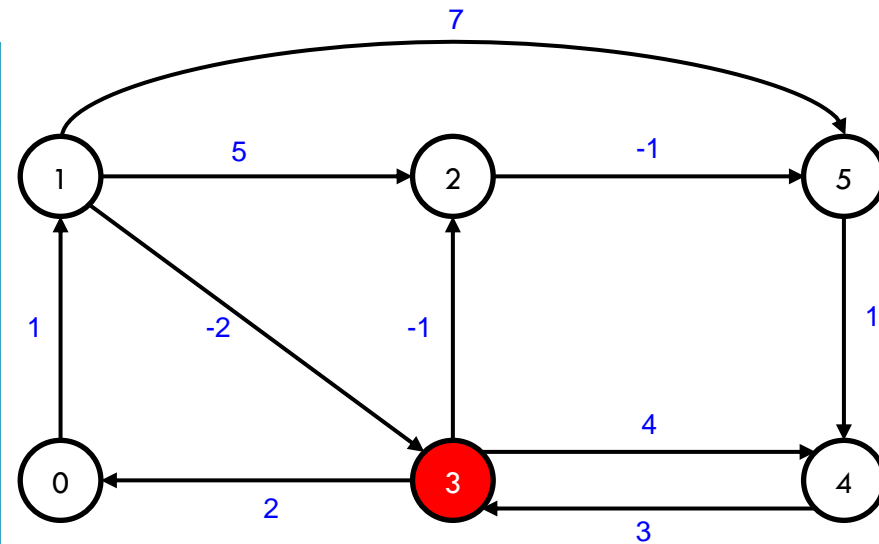
Có 11 giá trị trong ma trận **dist** được cập nhật khi $k = 3$.

$j=0 \longrightarrow j=5$

$i=0 \downarrow i=5$

	0	1	2	3	4	5
0	0	1	(6) -2	-1	(∞) 3	(5) -3
1	(∞) 0	0	(5) -3	-2	(∞) 2	4 (-4)
2	∞	∞	0	∞	∞	-1
3	2	3	-1	0	4	-2
4	(∞) 5	(∞) 6	(∞) 2	3	0	(∞) 1
5	∞	∞	∞	∞	1	0

dist



Bước 4: Chạy thuật toán ($k=3$)

Cập nhật: $\text{path}[i][j] = \text{path}[k][j]$

Tương tự cũng sẽ có 11 giá trị trong ma trận **path** được cập nhật khi $k = 3$.

$j=0 \longrightarrow j=5$

	0	1	2	3	4	5
$i=0$ 0	-1	0	(1) 3	1	(-1) 3	(2) 2
1	(-1) 3	-1	(1) 3	1	(-1) 3	(2) 2
2	-1	-1	-1	-1	-1	2
3	3	0	3	-1	3	2
4	(-1) 3	(-1) 0	(-1) 3	4	-1	(-1) 2
$i=5$ 5	-1	-1	-1	-1	5	-1

path

BƯỚC 5

CHẠY VÒNG LẶP DUYỆT QUA CÁC DANH SÁCH KÈ LẦN 5

Bước 5: Chạy thuật toán (**k=4**)

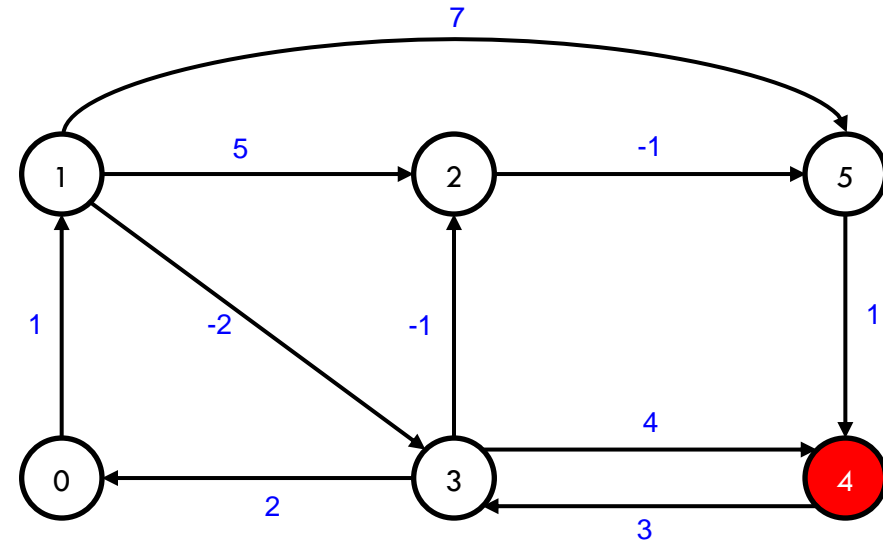
Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

Có 4 giá trị trong ma trận **dist** được cập nhật khi **k = 4**.

j=0 → j=5

	0	1	2	3	4	5
i=0	0	1	-2	-1	3	-3
1	0	0	-3	-2	2	-4
2	∞	∞	0	∞	∞	-1
3	2	3	-1	0	4	-2
4	5	6	2	3	0	1
i=5	5	(∞) 6	(∞) 7	(∞) 3	(∞) 4	1

dist



Bước 5: Chạy thuật toán (**k=4**)

Cập nhật $\text{path}[i][j] = \text{path}[k][j]$

Tương tự cũng sẽ có 4 giá trị trong ma trận **path** được cập nhật khi **k = 4**.

j=0

→

j=5

	0	1	2	3	4	5	
i=0	0	-1	0	3	1	3	2
	1	3	-1	3	1	3	2
	2	-1	-1	-1	-1	-1	2
	3	3	0	3	-1	3	2
	4	3	0	3	4	-1	2
i=5	5	(-1) 3	(-1) 0	(-1) 3	(-1) 4	5	-1

path

BƯỚC 6

CHẠY VÒNG LẶP DUYỆT QUA CÁC DANH SÁCH KÈ LẦN 6

Bước 6: Chạy thuật toán ($k=5$)

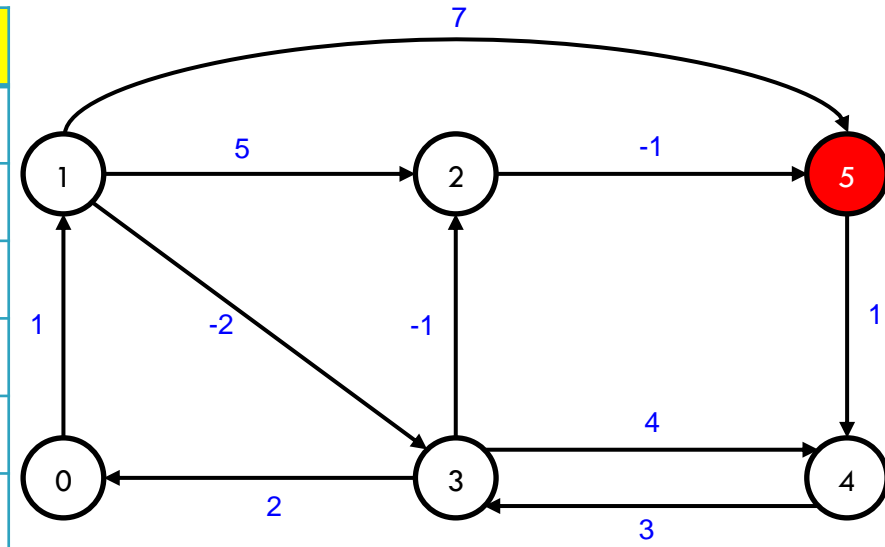
Xét điều kiện: $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$

Có 7 giá trị trong ma trận **dist** được cập nhật khi $k = 5$.

$j=0 \longrightarrow j=5$

	0	1	2	3	4	5
$i=0$	0	1	-2	-1	(3) -2	-3
1	0	0	-3	-2	(2) -3	-4
2	(∞) 5	(∞) 6	0	(∞) 3	(∞) 0	-1
3	2	3	-1	0	(4) -1	-2
4	5	6	2	3	0	1
$i=5$	5	6	7	3	1	0

dist



Bước 6: Chạy thuật toán ($k=5$)

Cập nhật $\text{path}[i][j] = \text{path}[k][j]$

Tương tự cũng sẽ có 7 giá trị trong ma trận **path** được cập nhật khi $k = 5$.

	0	1	2	3	4	5
0	-1	0	3	1	(3) 5	2
1	3	-1	3	1	(3) 5	2
2	(-1) 3	(-1) 0	-1	(-1) 4	(-1) 5	2
3	3	0	3	-1	(3) 5	2
4	3	0	3	4	-1	2
5	3	0	3	4	5	-1

path

Kết quả cuối cùng

Mảng chứa chi phí đường đi của tất cả các cặp đỉnh **dist**.

	0	1	2	3	4	5
0	0	1	-2	-1	-2	-3
1	0	0	-3	-2	-3	-4
2	5	6	0	3	0	-1
3	2	3	-1	0	-1	-2
4	5	6	2	3	0	1
5	6	7	3	4	1	0

Mảng lưu vết đỉnh cha của tất cả các cặp đỉnh **path**.

	0	1	2	3	4	5
0	-1	0	3	1	5	2
1	3	-1	3	1	5	2
2	3	0	-1	4	5	2
3	3	0	3	-1	5	2
4	3	0	3	4	-1	2
5	3	0	3	4	5	-1

Kết quả bài toán

Tìm đường đi ngắn nhất từ 0 đến 4.

dist

	0	1	2	3	4	5
0	0	1	-2	-1	-2	-3
1	0	0	-3	-2	-3	-4
2	5	6	0	3	0	-1
3	2	3	-1	0	-1	-2
4	5	6	2	3	0	1
5	6	7	3	4	1	0

path

	0	1	2	3	4	5
0	-1	0	3	1	5	2
1	3	-1	3	1	5	2
2	3	0	-1	4	5	2
3	3	0	3	-1	5	2
4	3	0	3	4	-1	2
5	3	0	3	4	5	-1

Kết quả bài toán & in đường đi

Tìm đường đi ngắn nhất từ 0 đến 4.



path

Đỉnh	0	1	2	3	4	5
Lưu vết	-1	0	3	1	5	2

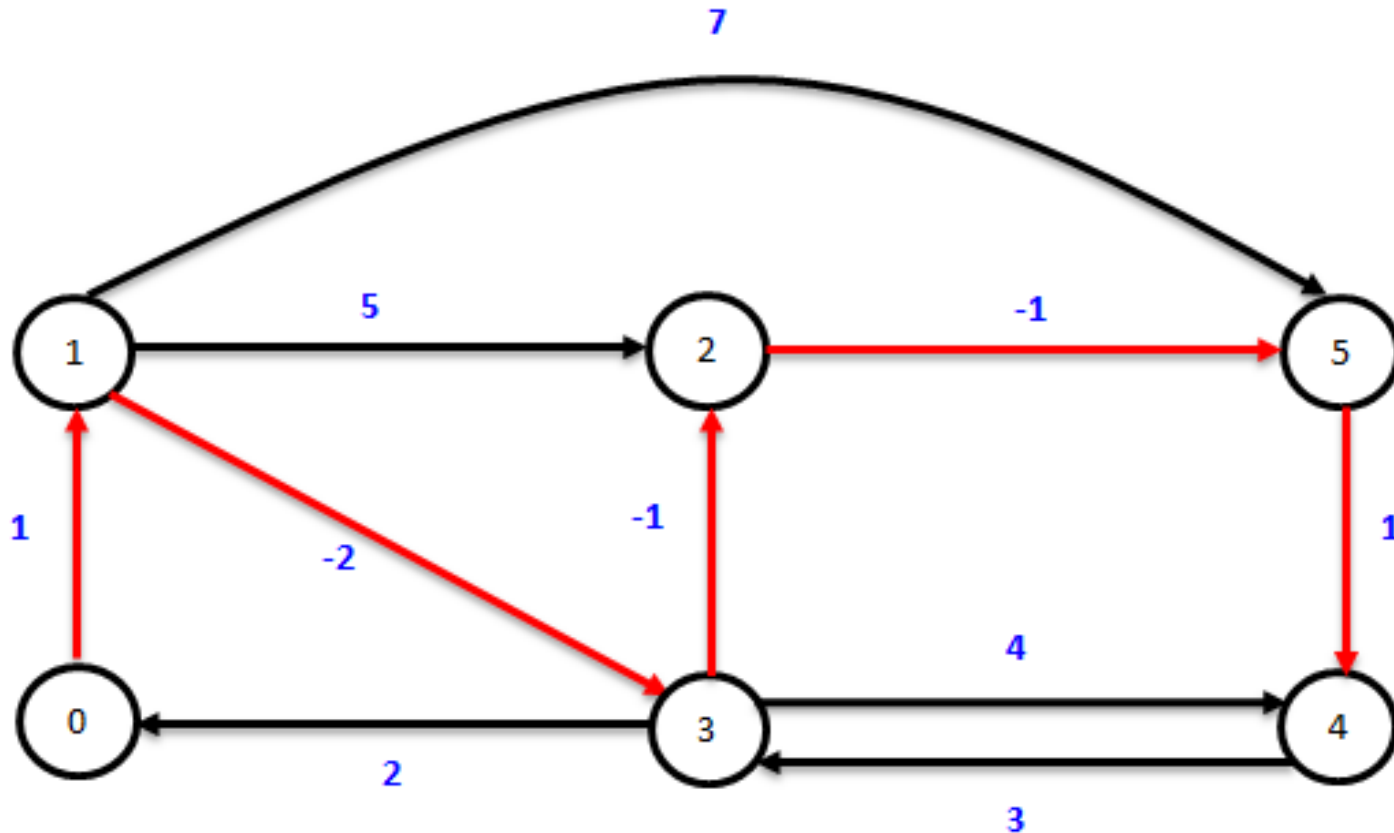
dist

Đỉnh	0	1	2	3	4	5
Chi phí	0	1	-2	-1	-2	-3

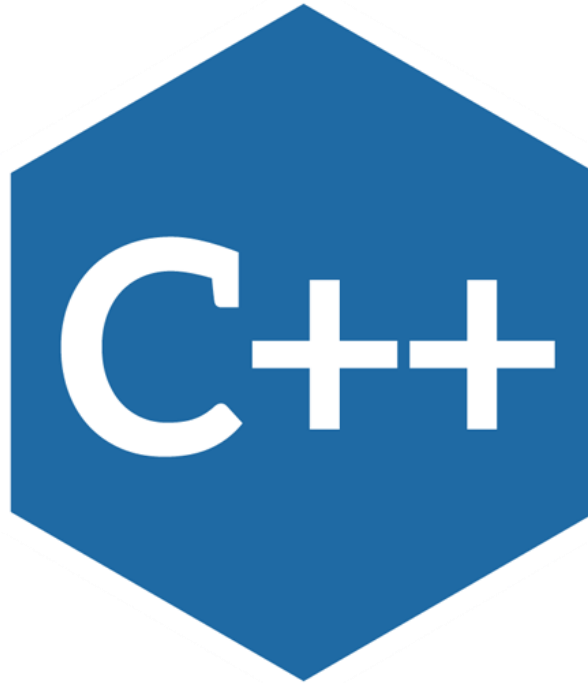
0 → 1 → 3 → 2 → 5 → 4
Chi phí: -2

Đường đi trên đồ thị

$0 \rightarrow 1 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 4$
Chi phí: -2



MÃ NGUỒN MINH HỌA BẰNG C++



Source Code Floyd-Warshall

Khai báo thư viện và các biến toàn cục:

```
1. #include <iostream>
2. #include <vector>
3. #include <algorithm>
4. using namespace std;
5. #pragma warning (disable : 4996)
6. #define MAX 105
7. #define INF 1e9
8. vector<vector<int> > graph;
9. vector<vector<int> > dist;
10. vector<vector<int> > path;
11. int V;
```



Source Code Floyd-Warshall

```
12. void PrintPath(vector<vector<int> >& path, int s, int f)
13. {
14.     if (path[s][f] == s)
15.         return;
16.     PrintPath(path, s, path[s][f]);
17.     cout << path[s][f] << " ";
18. }
```



Source Code Floyd-Warshall



```
19. void PrintSolution(vector<vector<int> >& path, vector<vector<int> >& dist)
20. {
21.     for (int i = 0; i < V; i++)
22.     {
23.         for (int j = 0; j < V; j++)
24.         {
25.             if (j != i && path[i][j] != -1)
26.             {
27.                 cout << i << " -> " << j << ": " << i << " ";
28.                 PrintPath(path, i, j);
29.                 cout << j << endl;
30.             }
31.             else if (j != i && path[i][j] == -1)
32.             {
33.                 cout << i << " -> " << j << ": No path" << endl;
34.             }
35.         }
36.     }
37. }
```

Source Code Floyd-Warshall

```
38. bool FloydWarshall(vector<vector<int> >& graph, vector<vector<int> >& dist)
39. {
40.     for (int i = 0; i < V; i++)
41.     {
42.         for (int j = 0; j < V; j++)
43.         {
44.             dist[i][j] = graph[i][j];
45.             if (i == j)
46.                 path[i][j] = 0;
47.             else if (dist[i][j] != INF)
48.                 path[i][j] = i;
49.             else
50.                 path[i][j] = -1;
51.         }
52.     }

//to be continued
```



Source Code Floyd-Warshall



```
53.     for (int k = 0; k < V; k++)
54.     {
55.         for (int i = 0; i < V; i++)
56.         {
57.             for (int j = 0; j < V; j++)
58.             {
59.                 if (dist[i][k] != INF && dist[k][j] != INF
60.                     && dist[i][k] + dist[k][j] < dist[i][j])
61.                 {
62.                     dist[i][j] = dist[i][k] + dist[k][j];
63.                     path[i][j] = path[k][j];
64.                 }
65.             }
66.             if (dist[i][i] < 0)
67.                 return false;
68.         }
69.     }
70.     return true;
71. }
```

Source Code Floyd-Warshall



```
72. int main()
73. {
74.     int n, temp;
75.     cin >> V;
76.     graph = vector<vector<int> >(V, vector<int>(V));
77.     dist = vector<vector<int> >(V, vector<int>(V));
78.     path = vector<vector<int> >(V, vector<int>(V));
79.     for (int i = 0; i < V; i++)
80.     {
81.         for (int j = 0; j < V; j++)
82.         {
83.             cin >> temp;
84.             if (temp == 0 && i != j)
85.                 graph[i][j] = INF;
86.             else
87.                 graph[i][j] = temp;
88.         }
89.     }
    //to be continued
```

Source Code Floyd-Warshall

```
90.     if (FloydWarshall(graph, dist) == true)
91.         PrintSolution(path, dist);
92.     else
93.         cout << "Graph contains negative weight cycle";
94.     return 0;
95. }
```



MÃ NGUỒN MINH HỌA BẰNG PYTHON



Source Code Floyd-Warshall

In đường đi:

```
1. INF = 10**9
2. def PrintPath(path, s, f):
3.     if (path[s][f] == s):
4.         return
5.     PrintPath(path, s, path[s][f])
6.     print(path[s][f], end = ' ')
```



Source Code Floyd-Warshall

In đường đi:

```
7. def PrintSolution(path, dist, v):
8.     for i in range(v):
9.         for j in range(v):
10.            if (j != i and path[i][j] != -1):
11.                print("{} -> {}: {}".format(i, j, i), end = ' ')
12.                PrintPath(path, i, j)
13.                print(j)
14.            elif (j != i and path[i][j] == -1):
15.                print("{} -> {}: No path".format(i, j))
```



Source Code Floyd-Warshall

```
16. def FloydWarshall(graph, dist, path, v):
17.     for i in range(v):
18.         for j in range(v):
19.             dist[i][j] = graph[i][j]
20.             if (i == j):
21.                 path[i][j] = 0
22.             elif dist[i][j] != INF:
23.                 path[i][j] = i
24.             else:
25.                 path[i][j] = -1
```



Source Code Floyd-Warshall

```
26.     for k in range(v):
27.         for i in range(v):
28.             for j in range(v):
29.                 if (dist[i][k] != INF and dist[k][j] != INF
30.                     and dist[i][k] + dist[k][j] < dist[i][j]):
31.                     dist[i][j] = dist[i][k] + dist[k][j]
32.                     path[i][j] = path[k][j]
33.             if (dist[i][i] < 0):
34.                 return False
35.     return True
```



Source Code Floyd-Warshall

Hàm main:

```
36. v = int(input())
37. graph = []
38. dist = [[0 for i in range(v)] for j in range(v)]
39. path = [[0 for i in range(v)] for j in range(v)]
40. for i in range(v):
41.     graph.append(list(map(int, input().split())))
42. for i in range(v):
43.     for j in range(v):
44.         if (i != j and graph[i][j] == 0):
45.             graph[i][j] = INF
46. if (FloydWarshall(graph, dist, path, v) == True):
47.     PrintSolution(path, dist, v)
48. else:
49.     print("Graph contains negative weight cycle")
```



MÃ NGUỒN MINH HỌA BẰNG JAVA



Source Code Floyd-Warshall

```
1. import java.util.*;
2. public class MyClass {
3.     static final int MAX = 105;
4.     static final int INF = (int)1e9;
5.     static void PrintPath(int s, int f) {
6.         if (path[s][f] == s)
7.             return;
8.         PrintPath(s, path[s][f]);
9.         System.out.print(path[s][f] + " ");
10.    }
11.    static Integer graph[][] = new Integer[MAX][MAX], dist[][] = new
12.        Integer[MAX][MAX], path[][] = new Integer[MAX][MAX];
13.    static int V;
```



Source Code Floyd-Warshall



```
14. static void PrintSolution() {
15.     for (int i = 0; i < V; i++) {
16.         for (int j = 0; j < V; j++) {
17.             if (j != i && path[i][j] != -1) {
18.                 System.out.print(i + " -> " + j + ": " + i + " ");
19.                 PrintPath(i, j);
20.                 System.out.println(j);
21.             }
22.             else if (j != i && path[i][j] == -1) {
23.                 System.out.println(i + " -> " + j + ": No path");
24.             }
25.         }
26.     }
27. }
```

Source Code Floyd-Warshall


Thuật toán chính Floyd-Warshall (part 1)

```
28. static boolean FloydWarshall() {  
29.     for (int i = 0; i < V; i++) {  
30.         for (int j = 0; j < V; j++) {  
31.             dist[i][j] = graph[i][j];  
32.             if (i == j)  
33.                 path[i][j] = 0;  
34.             else if (dist[i][j] != INF)  
35.                 path[i][j] = i;  
36.             else  
37.                 path[i][j] = -1;  
38.         }  
39.     }  
  
    // to be continued
```



Source Code Floyd-Warshall

Thuật toán chính Floyd-Warshall (part 2)



```
40.     for (int k = 0; k < V; k++) {
41.         for (int i = 0; i < V; i++) {
42.             for (int j = 0; j < V; j++) {
43.                 if (dist[i][k] != INF && dist[k][j] != INF
44.                     && dist[i][k] + dist[k][j] < dist[i][j]) {
45.                     dist[i][j] = dist[i][k] + dist[k][j];
46.                     path[i][j] = path[k][j];
47.                 }
48.             }
49.             if (dist[i][i] < 0) {
50.                 return false;
51.             }
52.         }
53.     }
54.     return true;
55. }
```

Source Code Floyd-Warshall



```
56.     public static void main(String args[]) {
57.         Scanner sc = new Scanner(System.in);
58.         int n, temp;
59.         V = sc.nextInt();
60.         for (int i = 0; i < V; i++) {
61.             for (int j = 0; j < V; j++) {
62.                 temp = sc.nextInt();
63.                 if (temp == 0 && i != j)
64.                     graph[i][j] = INF;
65.                 else
66.                     graph[i][j] = temp;
67.             }
68.         }
69.         if (FloydWarshall() == true)
70.             PrintSolution();
71.         else
72.             System.out.print("Graph contains negative weight cycle");
73.     }
74. }
```

Hỏi đáp

