Big-O Blue

LECTURE 11

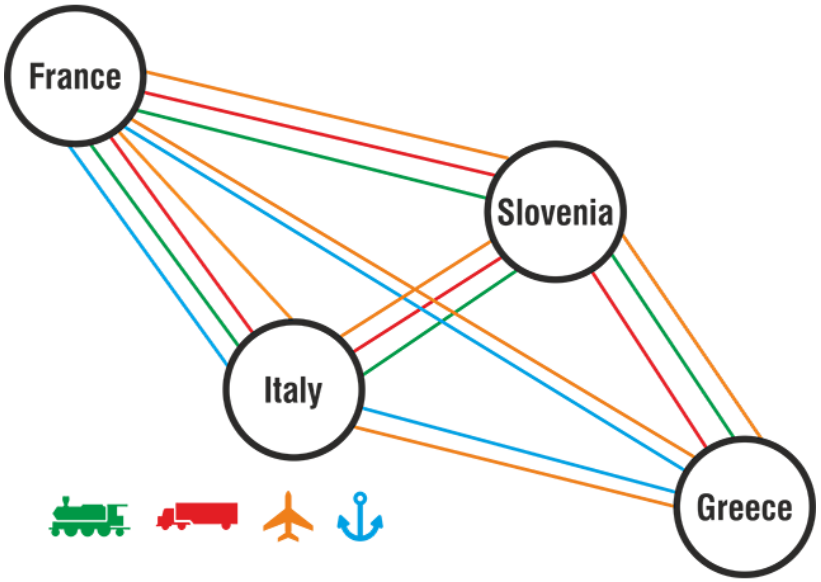
FLOYD-WARSHALL ALGORITHM

Phạm Nguyễn Sơn Tùng

Email: sontungtn@gmail.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.

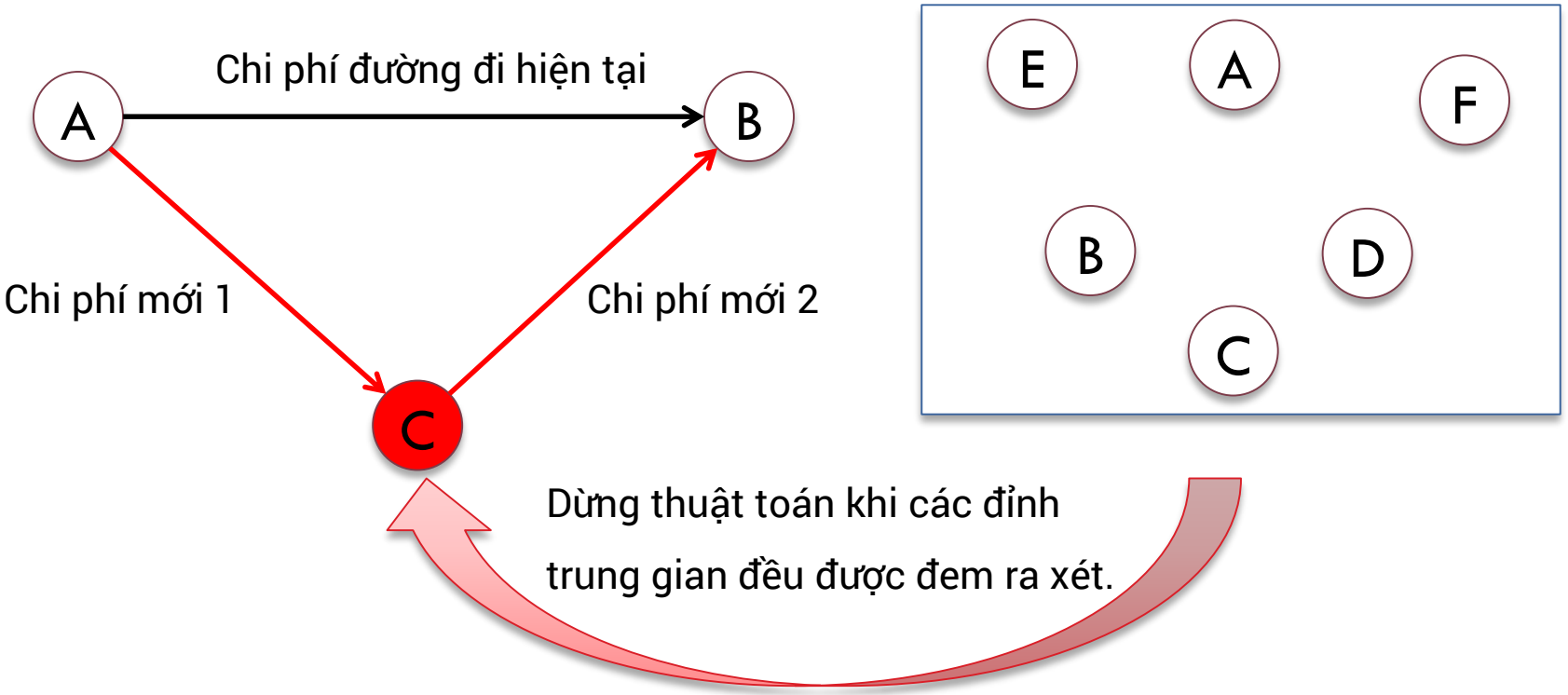


Độ phức tạp: O (V^3)

- 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.

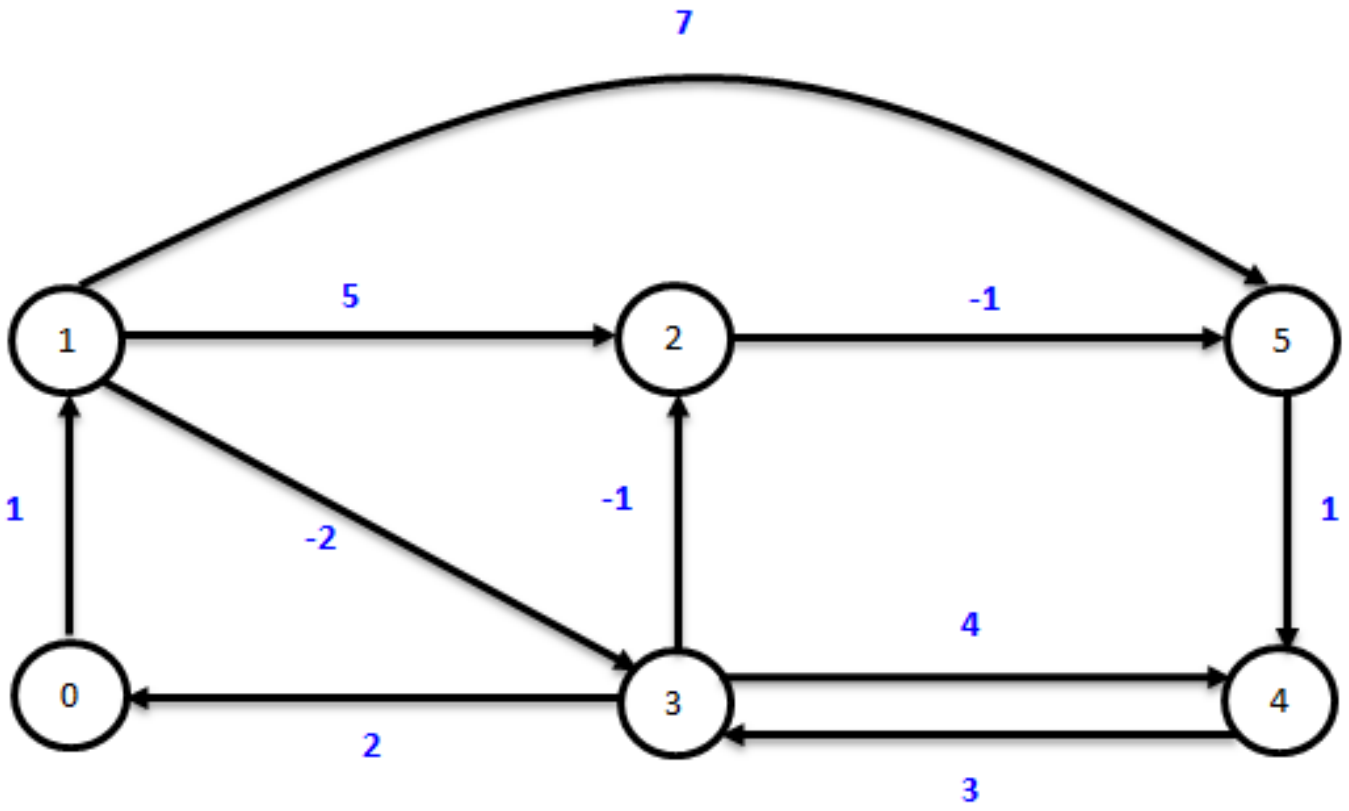


Nếu: Chi phí đường đi hiện tại > chi phí mới 1 + chi phí mới 2

➔ 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 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 KỀ LẦN 1

Bước 1: Chạy thuật toán (k=0)

Xét điều kiện dist[i][j] > dist[i][k] + dist[k][j]

dist[0][0] > dist[0][0] + dist[0][0] ✗

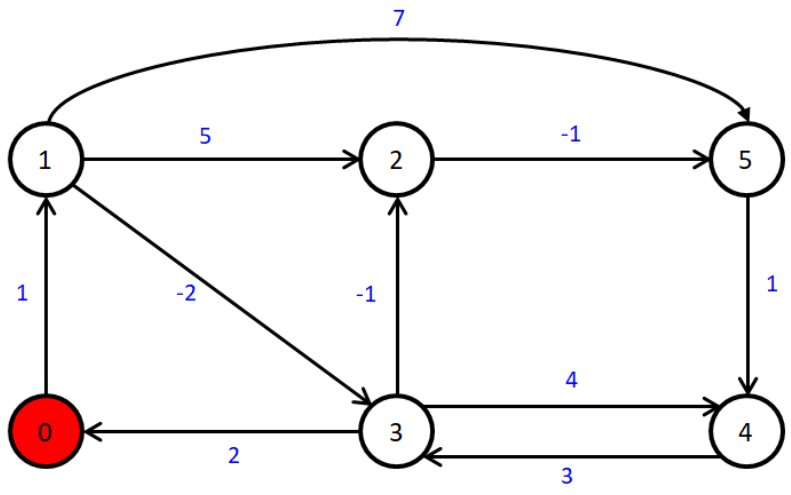
0 0 0

j=0

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 dist[i][j] > dist[i][k] + dist[k][j]

dist[0][1] > dist[0][0] + dist[0][1] ✗

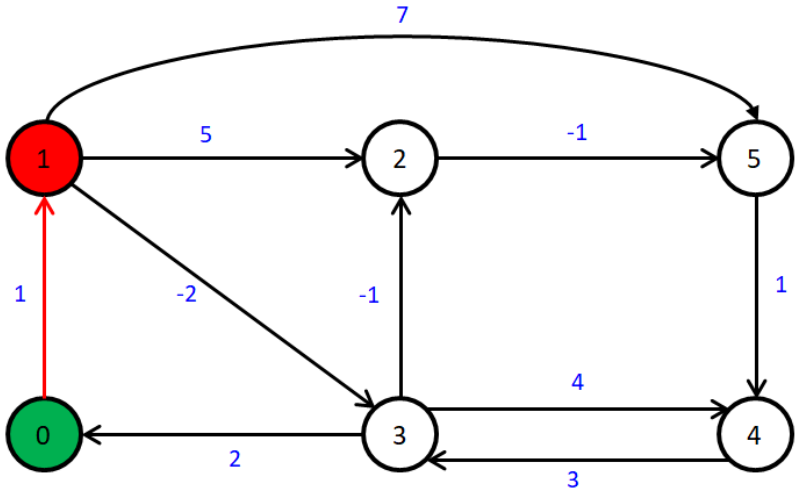
1 **0** **1**

j=1

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 `dist[i][j] > dist[i][k] + dist[k][j]`

`dist[0][2] > dist[0][0] + dist[0][2]` ✗

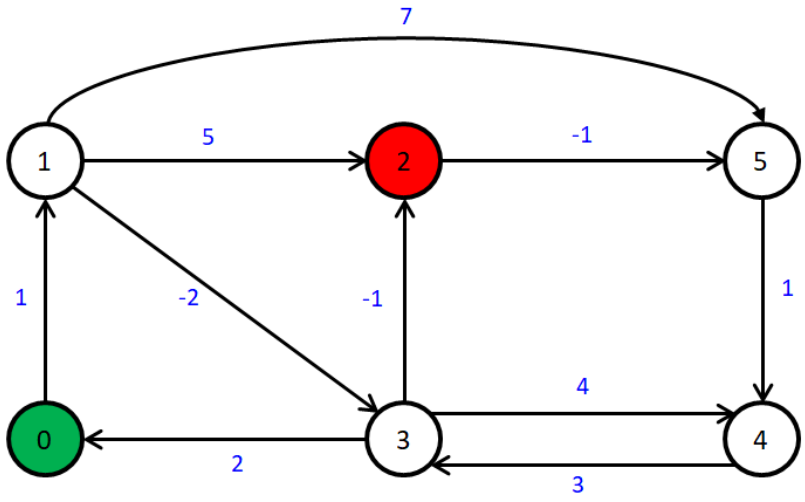
∞ 0 ∞

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 `dist[i][j] > dist[i][k] + dist[k][j]`

`dist[0][3] > dist[0][0] + dist[0][3]` ✗

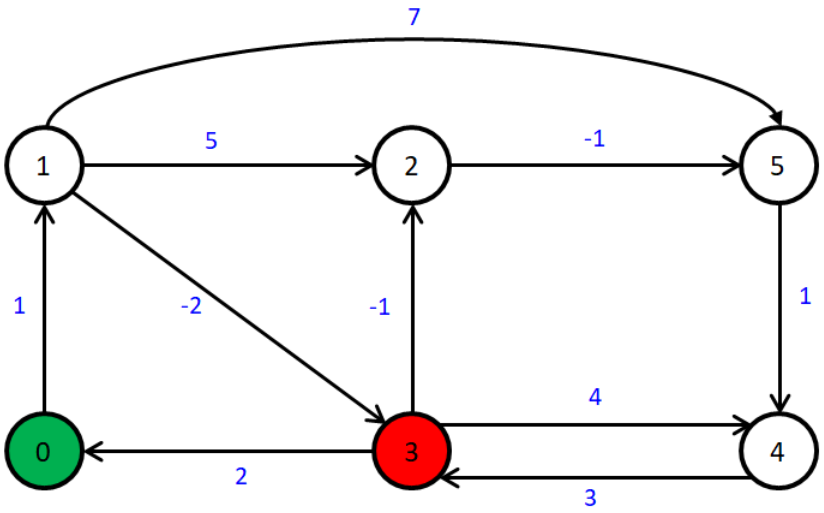
∞ 0 ∞

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 `dist[i][j] > dist[i][k] + dist[k][j]`

`dist[0][4] > dist[0][0] + dist[0][4]` ✗

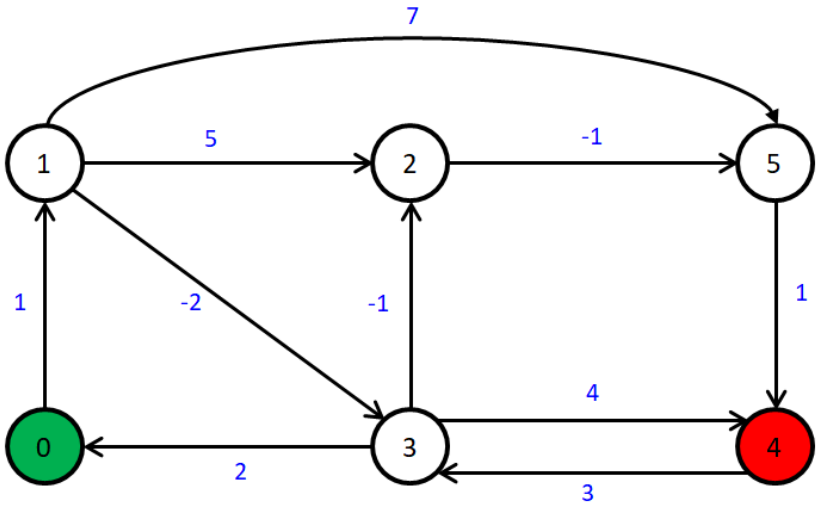
∞ 0 ∞

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 `dist[i][j] > dist[i][k] + dist[k][j]`

dist[0][5] > dist[0][0] + dist[0][5] ✗

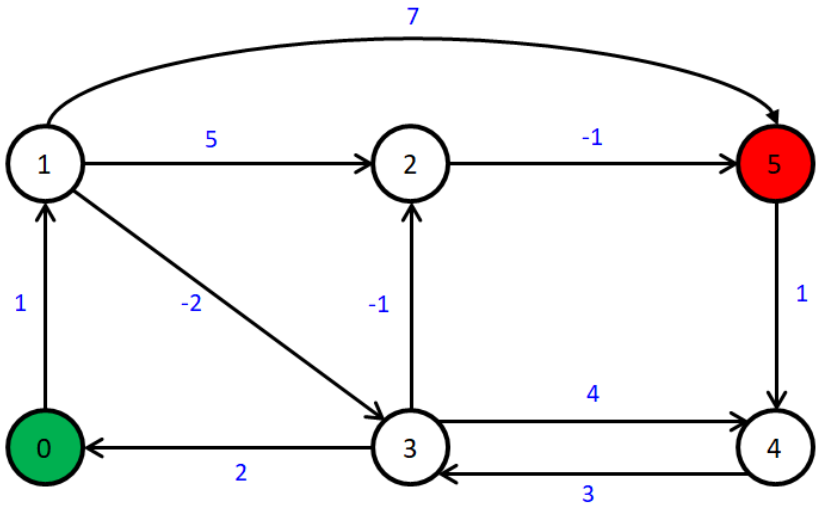
∞ 0 ∞

j=5

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



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

Bước 1: Chạy thuật toán (k=0)

Xét điều kiện `dist[i][j] > dist[i][k] + dist[k][j]`

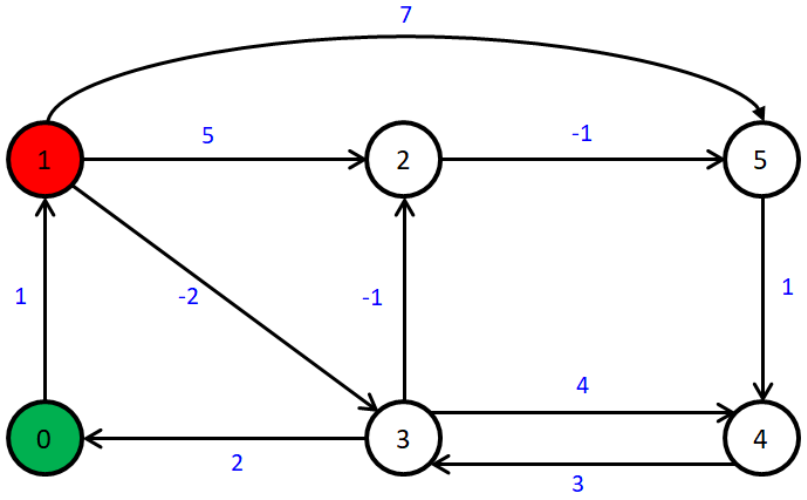
j = 0, j = 1, j = 2, j = 3, j = 4, j = 5 ✗

j=0 → 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 `dist[i][j] > dist[i][k] + dist[k][j]`

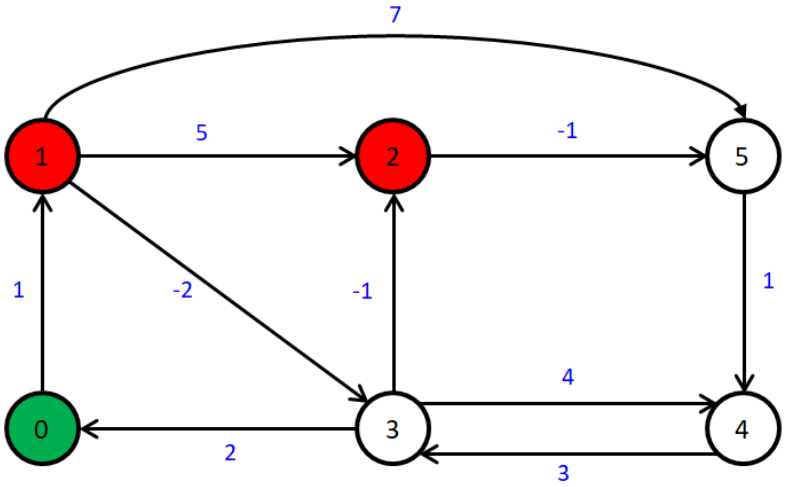
j = 0, j = 1, j = 2, j = 3, j = 4, j = 5 ✗

j=0 → j=5

i=2

	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 `dist[i][j] > dist[i][k] + dist[k][j]`

1 **dist[3][1] > dist[3][0] + dist[0][1]** ✓

∞ 2 1

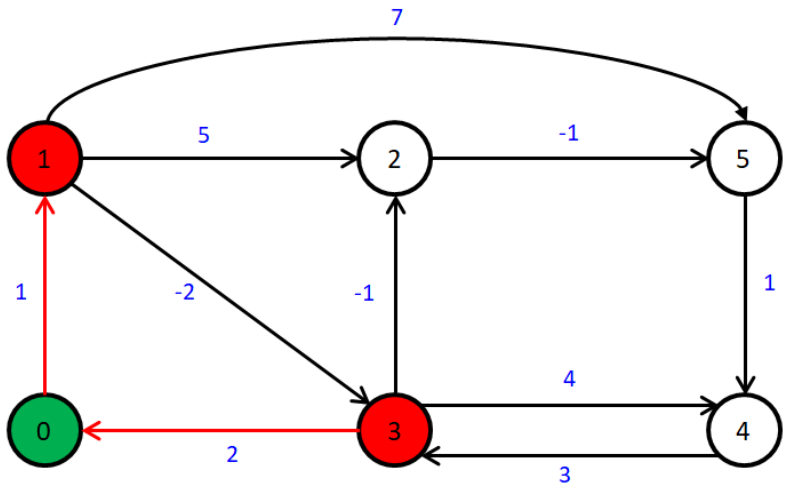
➔ **dist[3][1] = 2 + 1 = 3**

j=1

i=3

	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 path[i][j] = path[k][j]

path[3][1] = 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 `dist[i][j] > dist[i][k] + dist[k][j]`

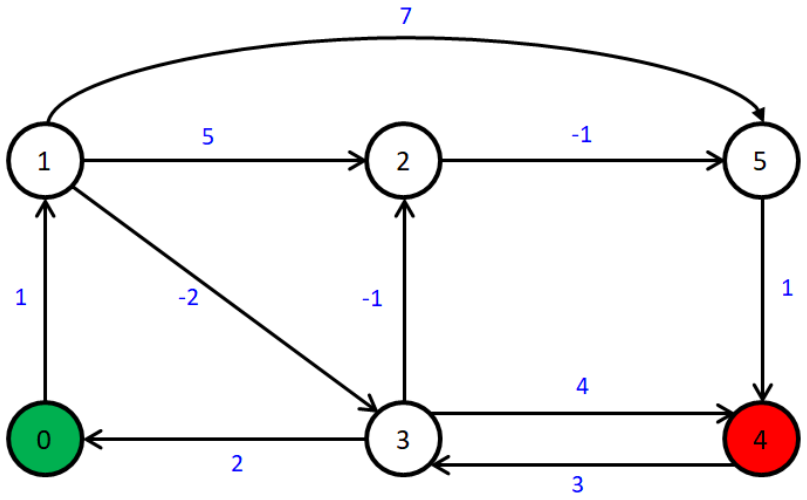
j = 0, j = 1, j = 2, j = 3, j = 4, j = 5 ✗

j=0 ⟶ j=5

i=4

	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



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

Bước 1: Chạy thuật toán (k=0)

Xét điều kiện `dist[i][j] > dist[i][k] + dist[k][j]`

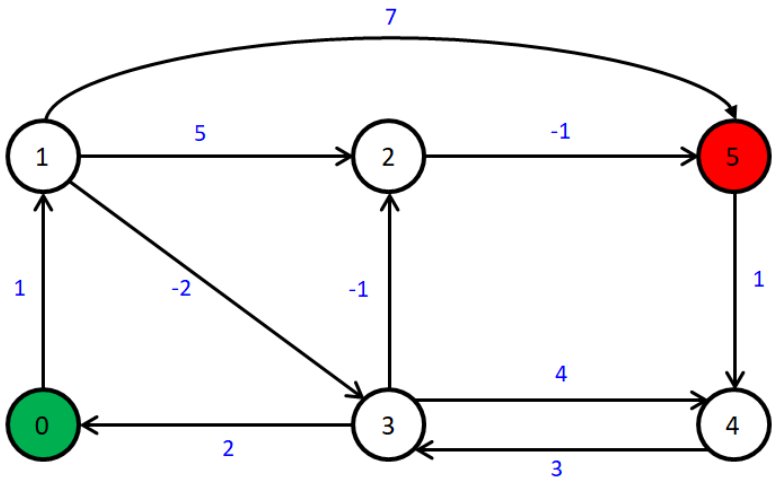
j = 0, j = 1, j = 2, j = 3, j = 4, j = 5 ✗

j=0 → 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 KỀ LẦN 2

Bước 2: Chạy thuật toán (k=1)

Xét điều kiện dist[i][j] > dist[i][k] + dist[k][j]

1 **dist[0][2] > dist[0][1] + dist[1][2]** ✓

∞ 1 5

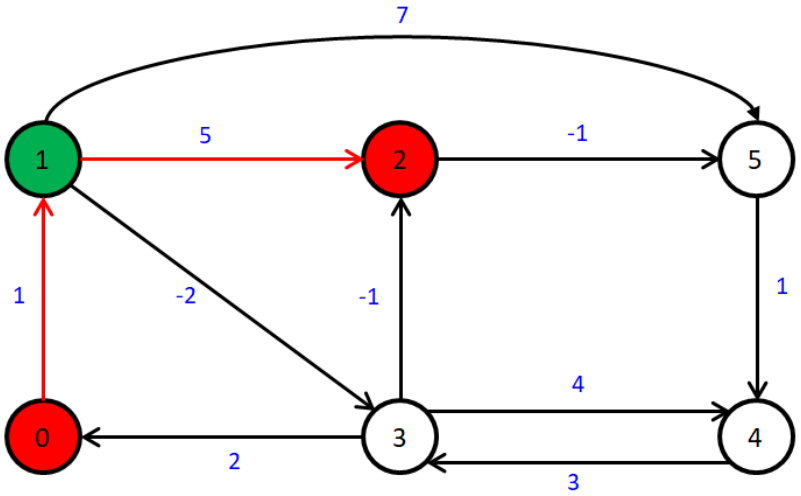
➔ **dist[0][2] = 1 + 5 = 6**

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 `path[i][j] = path[k][j]`

`path[0][2] = 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 dist[i][j] > dist[i][k] + dist[k][j]

2 **dist[0][3] > dist[0][1] + dist[1][3]** ✓

∞ 1 -2

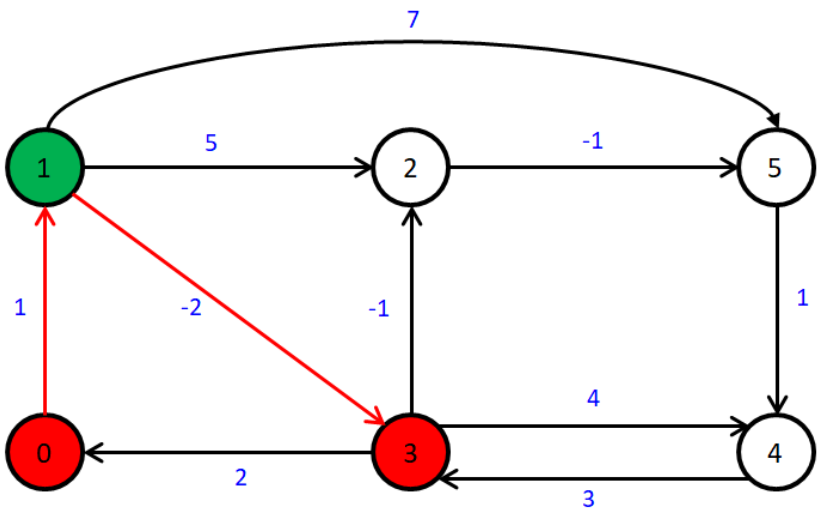
➔ **dist[0][3] = 1 + (-2) = -1**

j=3

i=0

	0	1	2	3	4	5
0	0	1	6	(∞) -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 path[i][j] = path[k][j]

path[0][3] = 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 dist[i][j] > dist[i][k] + dist[k][j]

3 `dist[0][5] > dist[0][1] + dist[1][5]` ✓

∞ 1 7

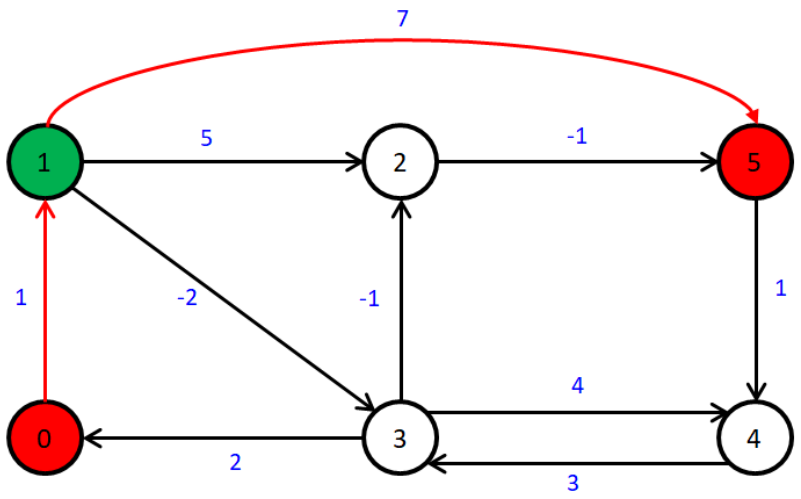
➔ `dist[0][5] = 1 + 7 = 8`

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 path[i][j] = path[k][j]

path[0][5] = 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 dist[i][j] > dist[i][k] + dist[k][j]

4 `dist[3][5] > dist[3][1] + dist[1][5]` ✓

∞ 3 7

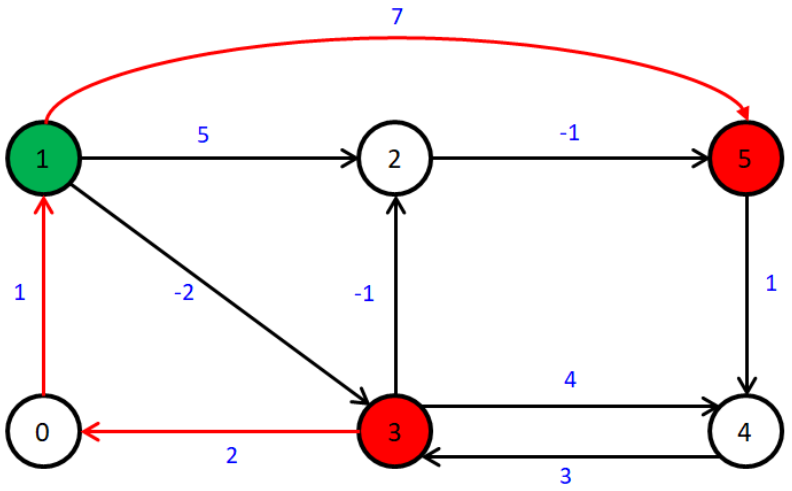
➔ `dist[3][5] = 3 + 7 = 10`

j=5

i=3

	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 path[i][j] = path[k][j]

path[3][5] = 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

BƯỚC 3

CHẠY VÒNG LẶP DUYỆT QUA MA TRẬN KỀ LẦN 3

Bước 3: Chạy thuật toán (k=2)

Xét điều kiện dist[i][j] > dist[i][k] + dist[k][j]

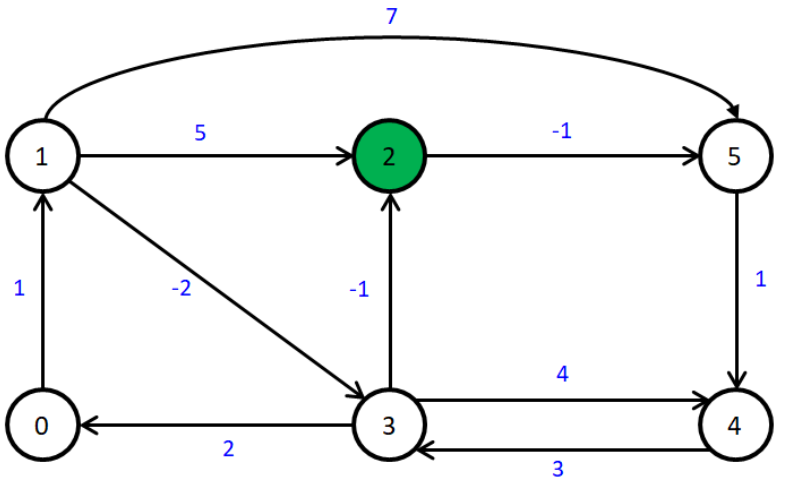
1. **dist[0][5] = dist[0][2] + dist[2][5] = 5**
2. **dist[1][5] = dist[1][2] + dist[2][5] = 4**
3. **dist[3][5] = dist[3][2] + dist[2][5] = -2**

j=0 → j=5

i=0 ↓ 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 path[i][j] = path[k][j]

1. **path[0][5] = path[2][5] = 2**
2. **path[1][5] = path[2][5] = 2**
3. **path[3][5] = path[2][5] = 2**

j=0 → j=5

i=0 ↓ 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 KỀ LẦN 4

Bước 4: Chạy thuật toán (k=3)

```
Xét điều kiện dist[i][j] > dist[i][k] + dist[k][j]
```

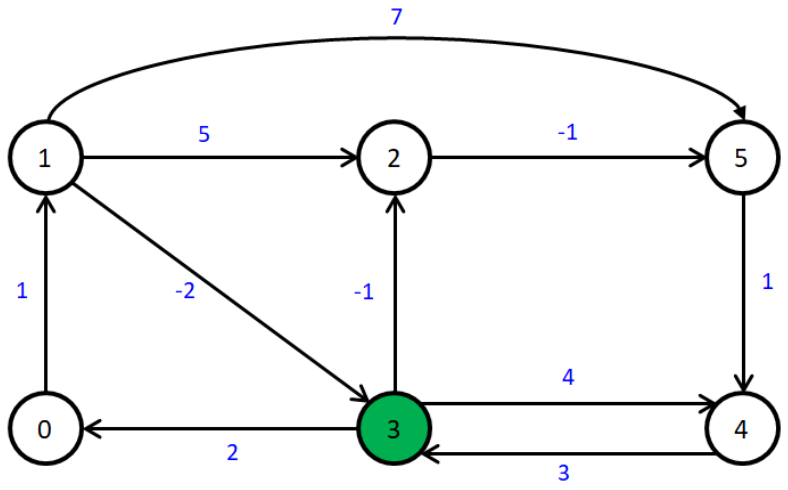
Có 11 giá trị trong ma trận **dist** được cập nhật khi **k = 3**.

j=0 → j=5

i=0 ↓ 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 path[i][j] = path[k][j]`

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

j=0 → j=5

i=0 ↓ i=5

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

path

BƯỚC 5

CHẠY VÒNG LẶP DUYỆT QUA CÁC DANH SÁCH KỀ LẦN 5

Bước 5: Chạy thuật toán (k=4)

Xét điều kiện dist[i][j] > dist[i][k] + dist[k][j]

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

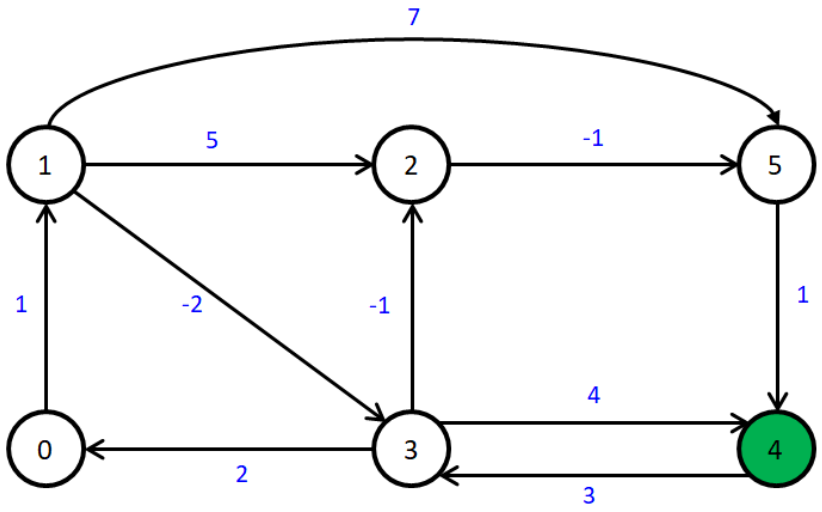
j=0 → j=5

i=0

i=5

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

dist



Bước 5: Chạy thuật toán (k=4)

`Cập nhật path[i][j] = 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

i=0 ↓ i=5

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

path

BƯỚC 6

CHẠY VÒNG LẶP DUYỆT QUA CÁC DANH SÁCH KỀ LẦN 6

Bước 6: Chạy thuật toán (k=5)

`Xét điều kiện dist[i][j] > dist[i][k] + dist[k][j]`

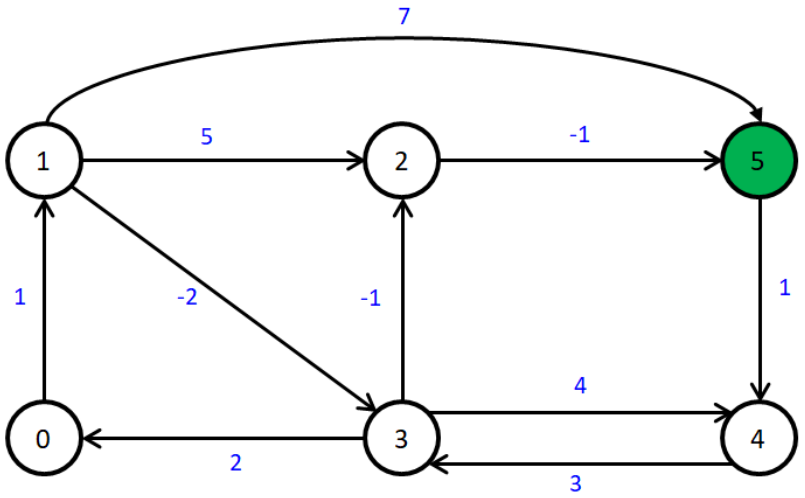
Có 7 giá trị trong ma trận **dist** được cập nhật khi **k = 5**.

j=0 → j=5

i=0 ↓ i=5

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

dist



Bước 6: Chạy thuật toán (k=5)

`Cập nhật path[i][j] = 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

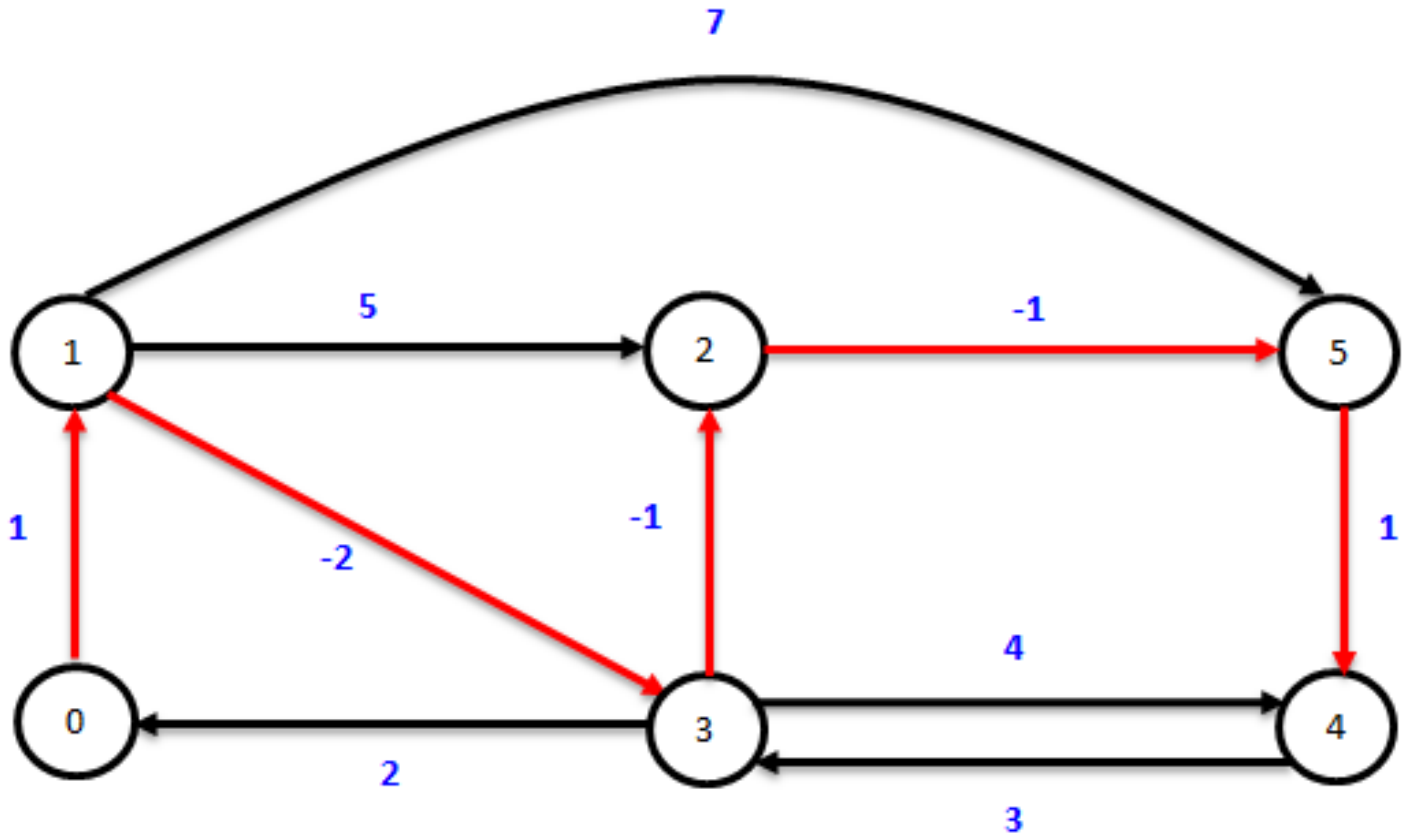
dist

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

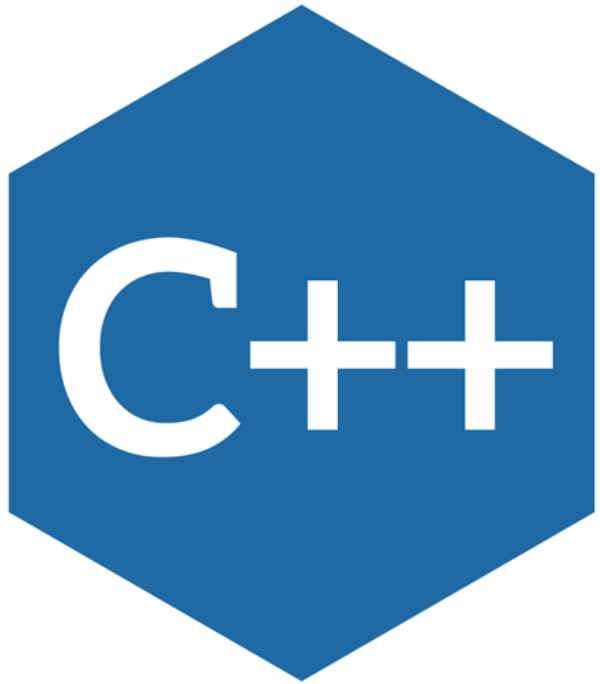
0 → 1 → 3 → 2 → 5 → 4
Chi phí: -2

Đường đi trên đồ thị

0 → 1 → 3 → 2 → 5 → 4
Chi phí: -2



MÃ NGUỒN MINH HỌA BẰNG C++



Source Code Floyd-Warshall

Khai báo thư viện và các biến toàn cục:

```cpp
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;
#define MAX 105
#define INF 1e9
vector<vector<int> > graph;
vector<vector<int> > dist;
vector<vector<int> > path;
int V;
```

Source Code Floyd-Warshall

```
11. void printPath(int s, int t)
12. {
13.     int b[MAX];
14.     int m = 0;
15.     while (s!=t)
16.     {
17.         b[m++] = t;
18.         t = path[s][t];
19.     }
20.     b[m++] = s;
21.     for (int i = m - 1; i >= 0; i--)
22.          cout << b[i] << " ";
23. }
```

Source Code Floyd-Warshall

```
24. bool FloydWarshall(vector<vector<int> > &graph, vector<vector<int> > &dist)
25. {
26.     for (int i = 0; i < V; i++)
27.         for (int j = 0; j < V; j++)
28.         {
29.             dist[i][j] = graph[i][j];
30.             if (graph[i][j] != INF && i != j)
31.                 path[i][j] = i;
32.             else
33.                 path[i][j] = -1;
34.         }
//to be continued
```

Source Code Floyd-Warshall

```
35.     for (int k = 0; k < V; k++)
36.     {
37.         for (int i = 0; i < V; i++)
38.         {
39.             for (int j = 0; j < V; j++)
40.             {
41.                 if (dist[i][j] > dist[i][k] + dist[k][j])
42.                 {
43.                     dist[i][j] = dist[i][k] + dist[k][j];
44.                     path[i][j] = path[k][j];
45.                 }
46.             }
47.         }
48.     }
49.     for (int i = 0; i < V; i++)
50.         if (dist[i][i] < 0)
51.             return false;
52.     return true;
53. }
```

Source Code Floyd-Warshall

Hàm main (part 1)

```
54. int main()
55. {
56.     int temp;
57.     cin >> V;
58.     graph = vector<vector<int> >(V, vector<int>(V));
59.     dist = vector<vector<int> >(V, vector<int>(V));
60.     path = vector<vector<int> >(V, vector<int>(V));
61.     for (int i = 0; i < V; i++)
62.         for (int j = 0; j < V; j++)
63.         {
64.             cin >> temp;
65.             if(temp==0 && i!=j)
66.                 graph[i][j] = INF;
67.             else
68.                 graph[i][j] = temp;
69.         }
//to be continued
```

Source Code Floyd-Warshall

Hàm main (part 2)

```
70.         FloydWarshall(graph, dist);
71.         int s = 0;
72.         int t = 4;
73.         int result = dist[s][t];
74.         printPath(s, t);
75.         cout << result;
76.         return 0;
77.     }
```

MÃ NGUỒN MINH HỌA BẰNG PYTHON

Source Code Floyd-Warshall

In đường đi:

```
1.  MAX = 100
2.  INF = int(1e9)
3.  def printPath(s, t):
4.      b = []
5.      while s != t:
6.          b.append(t)
7.          t = path[s][t]
8.      b.append(s)
9.      for i in range(len(b)-1,-1,-1):
10.         print(b[i], end=' ' if i > 0 else '\n')
```

Source Code Floyd-Warshall

```
12. def FloydWarshall(graph, dist):
13.     for i in range(V):
14.         for j in range(V):
15.             dist[i][j] = graph[i][j]
16.             if graph[i][j] != INF and i != j:
17.                 path[i][j] = i
18.             else:
19.                 path[i][j] = -1
20.     for k in range(V):
21.         for i in range(V):
22.             for j in range(V):
23.                 if dist[i][j] > dist[i][k] + dist[k][j]:
24.                     dist[i][j] = dist[i][k] + dist[k][j]
25.                     path[i][j] = path[k][j]
26.     for i in range(V):
27.         if dist[i][i] < 0:
28.             return False
29.     return True
```

Source Code Floyd-Warshall

Hàm main:

```
30. if __name__ == '__main__':
31.     V = int(input())
32.     graph = [[None for i in range(V)] for j in range(V)]
33.     dist = [[None for i in range(V)] for j in range(V)]
34.     path = [[None for i in range(V)] for j in range(V)]
35.     for i in range(V):
36.         line = list(map(int, input().split()))
37.         for j in range(V):
38.             graph[i][j] = INF if line[j] == 0 and i != j else line[j]
39.     FloydWarshall(graph, dist)
40.     s, t = 0, 4
41.     printPath(s, t)
42.     print(dist[s][t])
```

MÃ NGUỒN MINH HỌA BẰNG JAVA

Source Code Floyd-Warshall

Khai báo thư viện và các biến toàn cục:

```
// init inside class
1. final static int INF = 1000000000;
2. private static int path[][];
3. private static int dist[][];
4. private static int graph[][];
```

Source Code Floyd-Warshall

In đường đi:

```
5.  public static void printPath(int s, int t) {
6.      ArrayList<Integer> b = new ArrayList<Integer>();
7.      while (s != t) {
8.          b.add(t);
9.          t = path[s][t];
10.     }
11.     b.add(s);
12.     for (int i = b.size() - 1; i >= 0; i--) {
13.         System.out.print(b.get(i) + " ");
14.     }
15.     System.out.println();
16. }
```

Source Code Floyd-Warshall

Thuật toán chính Floyd-Warshall (part 1)

```
17. public static boolean floydWarshall(int graph[][], int dist[][]) {
18.     int V = graph.length;
19.     for (int i = 0; i < V; i++)
20.         for (int j = 0; j < V; j++) {
21.             dist[i][j] = graph[i][j];
22.             if (graph[i][j] != INF && i != j) {
23.                 path[i][j] = i;
24.             }
25.             else {
26.                 path[i][j] = -1;
27.             }
28.         }
    // to be continued
```

Source Code Floyd-Warshall

Thuật toán chính Floyd-Warshall (part 2)

```
29.     for (int k = 0; k < V; k++) {
30.         for (int i = 0; i < V; i++) {
31.             for (int j = 0; j < V; j++) {
32.                 if (dist[i][j] > dist[i][k] + dist[k][j]) {
33.                     dist[i][j] = dist[i][k] + dist[k][j];
34.                     path[i][j] = path[k][j];
35.                 }
36.             }
37.         }
38.     }
39.     for (int i = 0; i < V; i++)
40.         if (dist[i][i] < 0)
41.             return false;
42.     return true;
43. }
```

Source Code Floyd-Warshall

Hàm main (part 1)

```
44. public static void main (String[] args) {
45.     Scanner sc = new Scanner(System.in);
46.     int V = sc.nextInt();
47.     int temp;
48.     graph = new int[V][V];
49.     path = new int[V][V];
50.     dist = new int[V][V];
51.     for (int i = 0; i < V; i++) {
52.         for (int j = 0; j < V; j++) {
53.             temp = sc.nextInt();
54.             if (temp == 0 && i != j) {
55.                 graph[i][j] = INF;
56.             } else {
57.                 graph[i][j] = temp;
58.             }
59.         }
60.     } // to be continued
```

Source Code Floyd-Warshall

Hàm main (part 2)

```
61.         floydWarshall(graph, dist);
62.         int s = 0, t = 4;
63.         int result = dist[s][t];
64.         printPath(s, t);
65.         System.out.println(result);
66.     }
```

Hỏi đáp