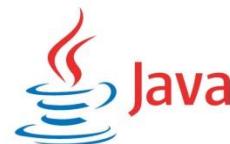


LECTURE 10

BELLMAN-FORD ALGORITHM



Phạm Nguyễn Sơn Tùng

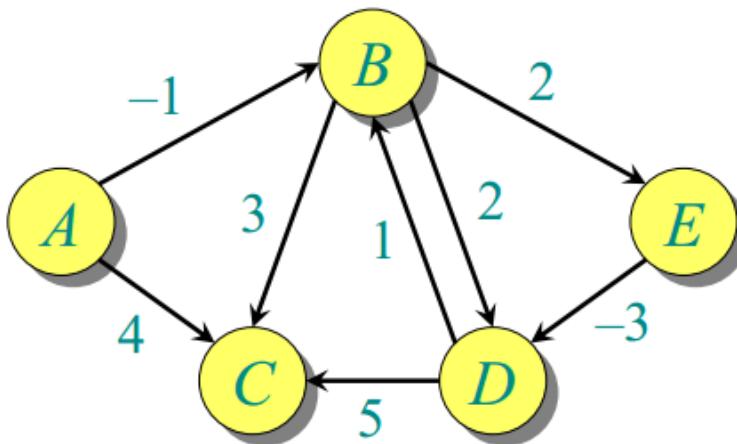
Email: sontungtn@gmail.com

Bellman-Ford

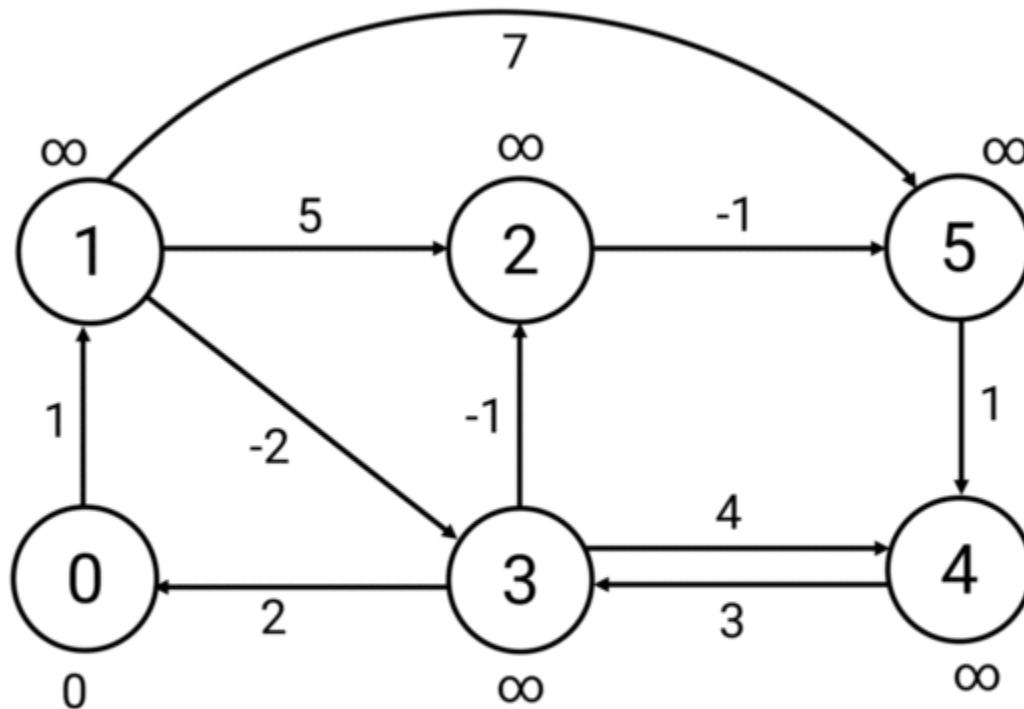
Thuật toán **Bellman-Ford** là thuật toán **tìm đường đi có chi phí nhỏ nhất từ một đỉnh đến tất cả các đỉnh** còn lại trong đồ thị có hướng hoặc vô hướng, có trọng số (trọng số có thể dương **hoặc âm**).

Độ phức tạp: **O (E . V)**

- **E (Edges)** là số lượng cạnh của đồ thị.
- **V (Vertices)** là số lượng đỉnh của đồ thị.

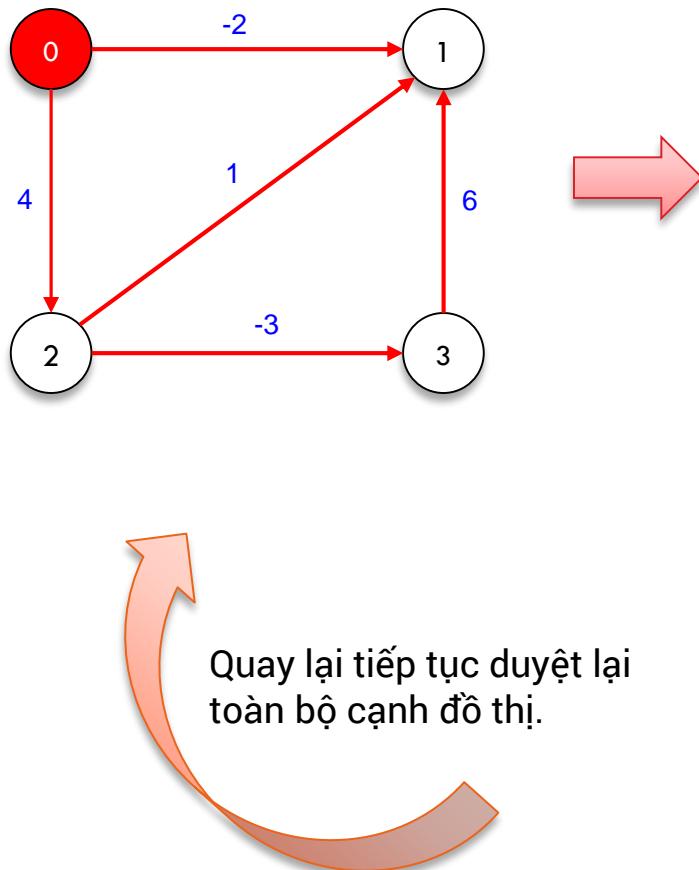


Mô phỏng cách chạy thuật toán



Ý tưởng của thuật toán

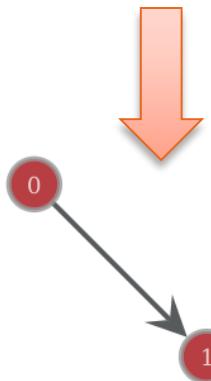
Xuất phát từ một đỉnh bất kỳ. Duyệt qua **toàn bộ cạnh đồ thị**.



So sánh chi phí đường đi hiện tại với đường đi trong bảng chi phí (nếu nhỏ hơn thì cập nhật)

Đỉnh	0	1	2	3
Chi phí	0	∞	∞	∞

Lưu vết.

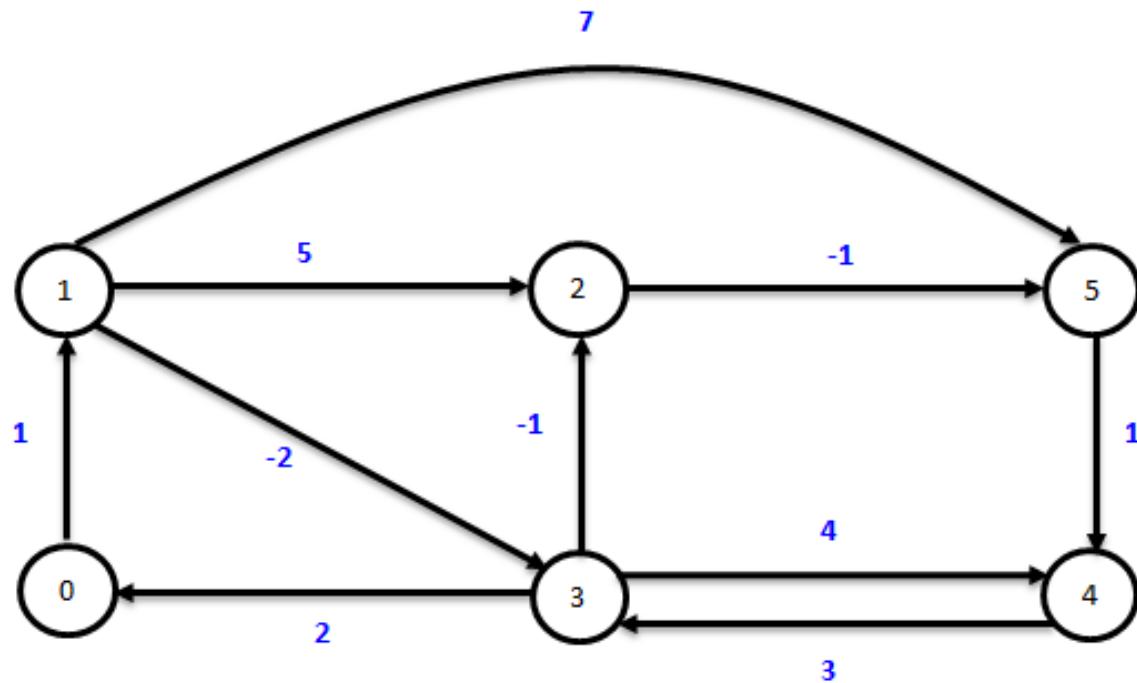


Đỉnh cha	0	1	2	3
Lưu vết	-1	0	-1	-1

→ Duyệt qua $V - 1$ lần thì dừng. Xuất kết quả bài toán.

Bài toán minh họa

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



Edge List

6 10		
0	1	1
1	2	5
1	3	-2
1	5	7
2	5	-1
3	0	2
3	2	-1
3	4	4
4	3	3
5	4	1

Bước 0: Chuẩn bị dữ liệu (1)

Chuyển danh sách cạnh vào **graph**.

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

Trong đó, mỗi phần tử bao gồm:

- **source**: đỉnh đầu.
- **target**: đỉnh đích.
- **weight**: trọng số.

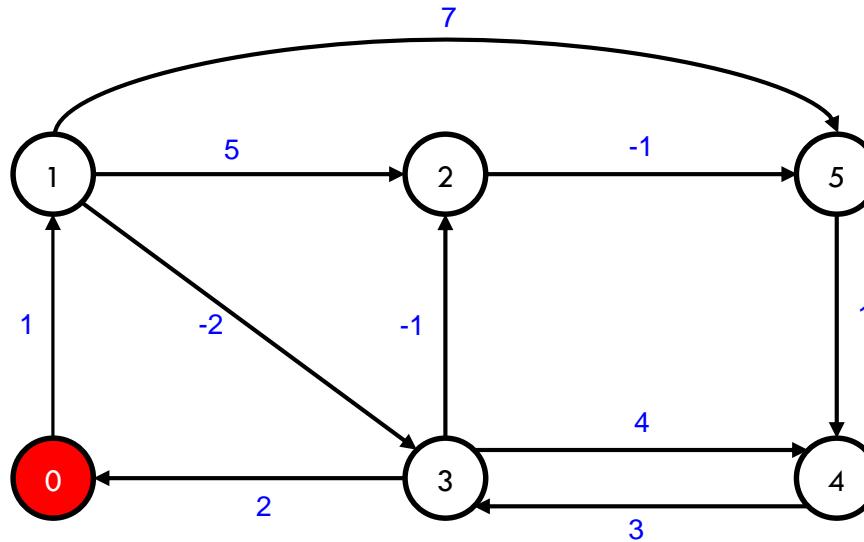
Mảng chứa chi phí đường đi **dist**.

Đỉnh	0	1	2	3	4	5
Chi phí	∞	∞	∞	∞	∞	∞

Mảng lưu vết đường đi **path**.

Đỉnh cha	0	1	2	3	4	5
Lưu vết	-1	-1	-1	-1	-1	-1

Bước 0: Chuẩn bị dữ liệu (2)



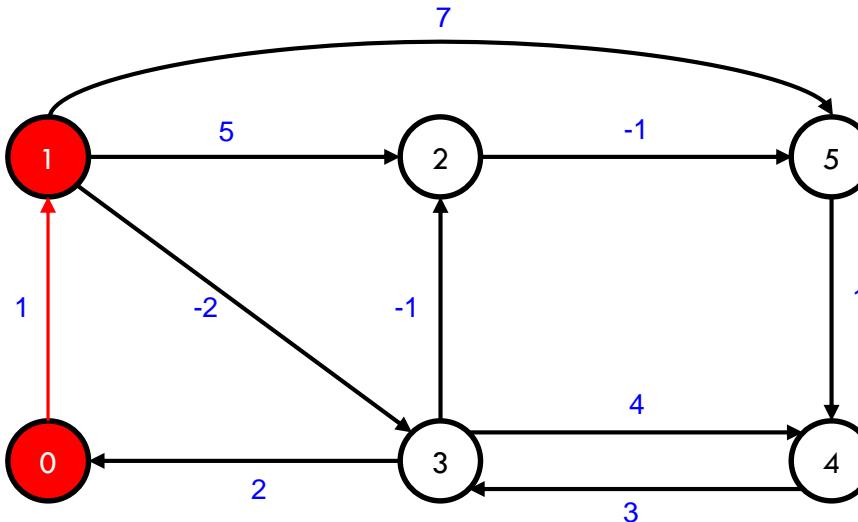
Gán chi phí cho đỉnh bắt đầu đi (đỉnh 0): **dist[0] = 0**

Đỉnh	0	1	2	3	4	5
Chi phí	0	∞	∞	∞	∞	∞

BƯỚC 1

**CHẠY VÒNG LẶP DUYỆT QUA DANH
SÁCH CẠNH LẦN 1**

Bước 1: Chạy thuật toán ($j=0$)



0 1 2 3 4 5 6 7 8 9

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

dist	Định	0	1	2	3	4	5
	Chi phí	0	∞	∞	∞	∞	∞

Lấy cạnh đầu tiên của **graph** ($u = 0, v = 1, w = 1$) để xem xét:

- Nếu chi phí tại **dist[u]** khác ∞ (0 khác ∞) ✓
- Và chi phí **dist[u] + w < dist[v]** ($0 + 1 < \infty$) ✓

→ Cập nhật **dist[v] = dist[u] + w = 1**.

Bước 1: Chạy thuật toán ($j=0$)

Cập nhật $\text{dist}[v] = \text{dist}[u] + w = 0 + 1 = 1.$

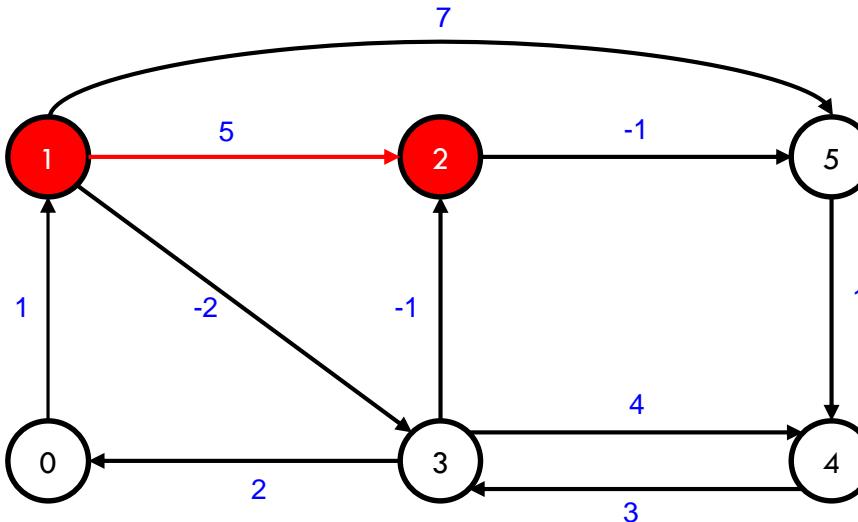
Cập nhật chi phí đỉnh đang xét (đỉnh 1) $\text{dist}[1] = 1.$

Đỉnh	0	1	2	3	4	5
Chi phí	0	1	∞	∞	∞	∞

Xét cạnh $(0, 1) \rightarrow$ cập nhật giá trị mảng lưu vết $\text{path}[1] = 0.$

Đỉnh cha	0	1	2	3	4	5
Lưu vết	-1	0	-1	-1	-1	-1

Bước 1: Chạy thuật toán ($j=1$)



0 1 2 3 4 5 6 7 8 9

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

dist	Định	0	1	2	3	4	5
	Chi phí	0	1	∞	∞	∞	∞

Lấy cạnh tiếp theo của **graph** ($u = 1, v = 2, w = 5$) để xem xét:

- Nếu chi phí tại **dist[u]** khác ∞ (1 khác ∞) ✓
- Và chi phí **dist[u] + w < dist[v]** ($1 + 5 < \infty$) ✓

→ Cập nhật **dist[v] = dist[u] + w = 6**.

Bước 1: Chạy thuật toán ($j=1$)

Cập nhật $\text{dist}[v] = \text{dist}[u] + w = 1 + 5 = 6$.

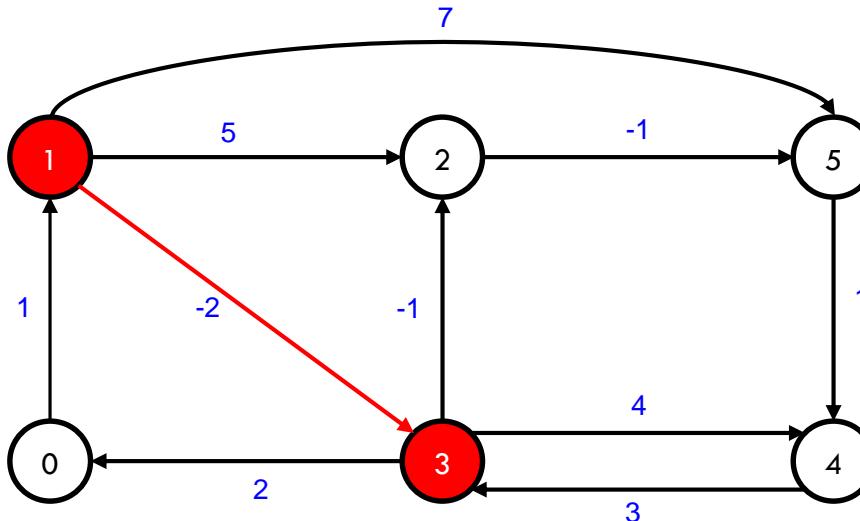
Cập nhật chi phí đỉnh đang xét (đỉnh 2) $\text{dist}[2] = 6$.

Đỉnh	0	1	2	3	4	5
Chi phí	0	1	6	∞	∞	∞

Xét cạnh (1, 2) \rightarrow cập nhật giá trị mảng lưu vết $\text{path}[2] = 1$.

Đỉnh cha	0	1	2	3	4	5
Lưu vết	-1	0	1	-1	-1	-1

Bước 1: Chạy thuật toán ($j=2$)



0 1 2 3 4 5 6 7 8 9

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

dist	Định	0	1	2	3	4	5
	Chi phí	0	1	6	∞	∞	∞

Lấy cạnh tiếp theo của **graph** ($u = 1, v = 3, w = -2$) để xem xét:

- Nếu chi phí tại **dist[u]** khác ∞ (1 khác ∞) ✓
- Và chi phí **dist[u] + w < dist[v]** ($1 + (-2) < \infty$) ✓

➔ Cập nhật **dist[v] = dist[u] + w = -1**.

Bước 1: Chạy thuật toán ($j=2$)

Cập nhật $\text{dist}[v] = \text{dist}[u] + w = 1 + (-2) = -1$.

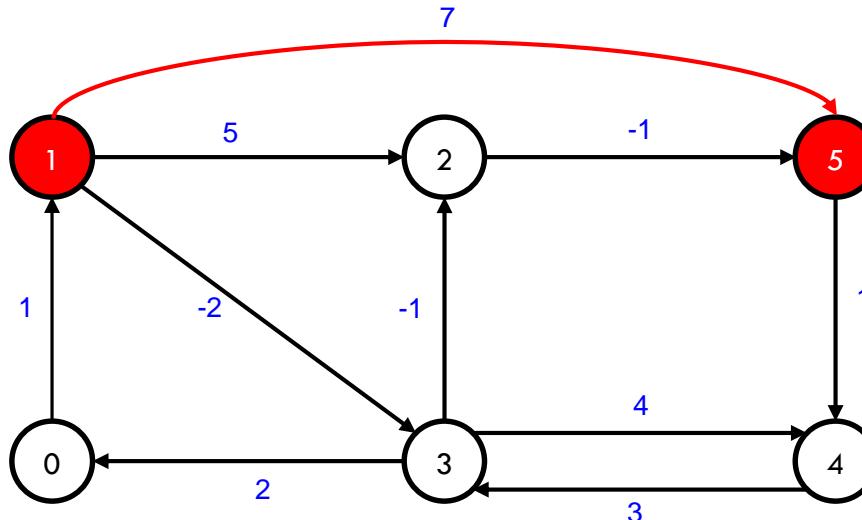
Cập nhật chi phí đỉnh đang xét (đỉnh 3) $\text{dist}[3] = -1$.

Đỉnh	0	1	2	3	4	5
Chi phí	0	1	6	-1	∞	∞

Xét cạnh (1, 3) \rightarrow cập nhật giá trị mảng lưu vết $\text{path}[3] = 1$.

Đỉnh cha	0	1	2	3	4	5
Lưu vết	-1	0	1	1	-1	-1

Bước 1: Chạy thuật toán ($j=3$)



0 1 2 3 4 5 6 7 8 9

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

dist	Định	0	1	2	3	4	5
	Chi phí	0	1	6	-1	∞	∞

Lấy cạnh tiếp theo của **graph** ($u = 1$, $v = 5$, $w = 7$) để xem xét:

- Nếu chi phí tại **dist[u]** khác ∞ (1 khác ∞) ✓
- Và chi phí **dist[u] + w < dist[v]** ($1 + 7 < \infty$) ✓

→ Cập nhật **dist[v] = dist[u] + w = 8**.

Bước 1: Chạy thuật toán ($j=3$)

Cập nhật $\text{dist}[v] = \text{dist}[u] + w = 1 + 7 = 8.$

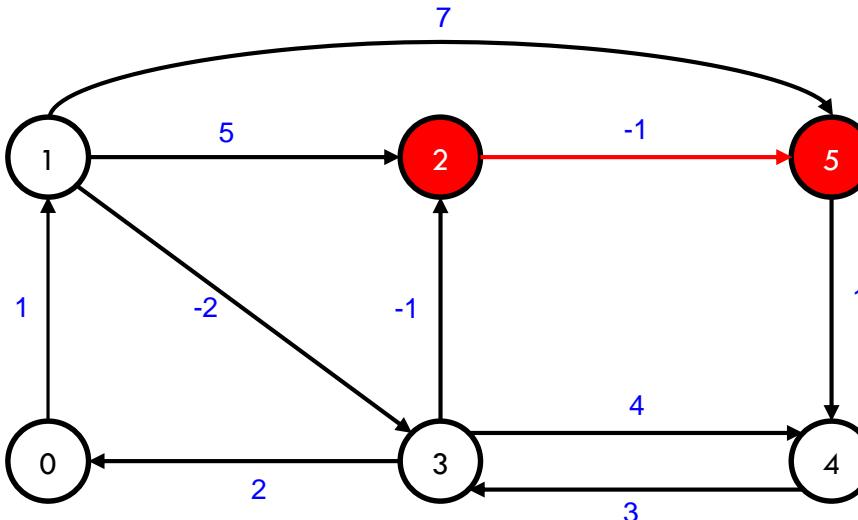
Cập nhật chi phí đỉnh đang xét (đỉnh 5) $\text{dist}[5] = 8.$

Đỉnh	0	1	2	3	4	5
Chi phí	0	1	6	-1	∞	8

Xét cạnh (1, 5) \rightarrow cập nhật giá trị mảng lưu vết $\text{path}[5] = 1.$

Đỉnh cha	0	1	2	3	4	5
Lưu vết	-1	0	1	1	-1	1

Bước 1: Chạy thuật toán ($j=4$)



0 1 2 3 4 5 6 7 8 9

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

dist	Định	0	1	2	3	4	5
	Chi phí	0	1	6	-1	∞	8

Lấy cạnh tiếp theo của **graph** ($u = 2, v = 5, w = -1$) để xem xét:

- Nếu chi phí tại **dist[u]** khác ∞ (6 khác ∞) ✓
- Và chi phí **dist[u] + w < dist[v]** ($6 + (-1) < 8$) ✓

→ Cập nhật **dist[v] = dist[u] + w = 5**.

Bước 1: Chạy thuật toán ($j=4$)

Cập nhật $\text{dist}[v] = \text{dist}[u] + w = 6 + -1 = 5$.

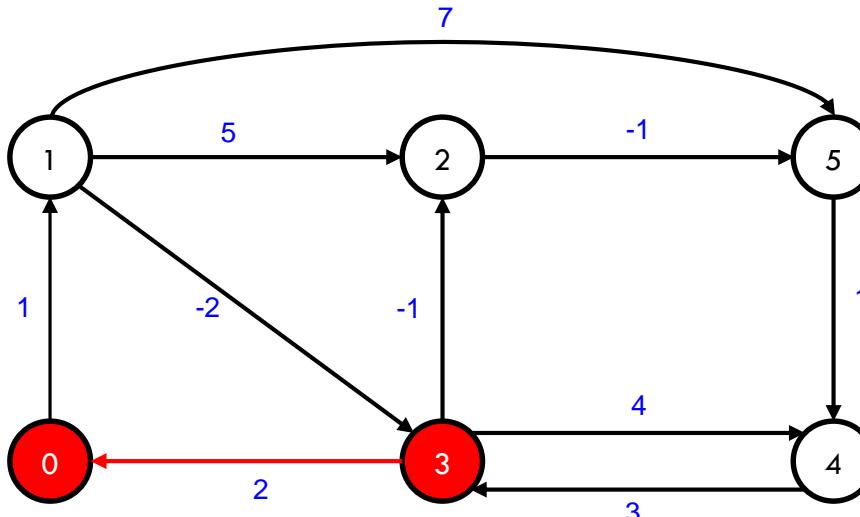
Cập nhật chi phí đỉnh đang xét (đỉnh 5) $\text{dist}[5] = 5$.

Đỉnh	0	1	2	3	4	5
Chi phí	0	1	6	-1	∞	5

Xét cạnh (2, 5) \rightarrow cập nhật giá trị mảng lưu vết $\text{path}[5] = 2$.

Đỉnh cha	0	1	2	3	4	5
Lưu vết	-1	0	1	1	-1	2

Bước 1: Chạy thuật toán ($j=5$)



0 1 2 3 4 5 6 7 8 9

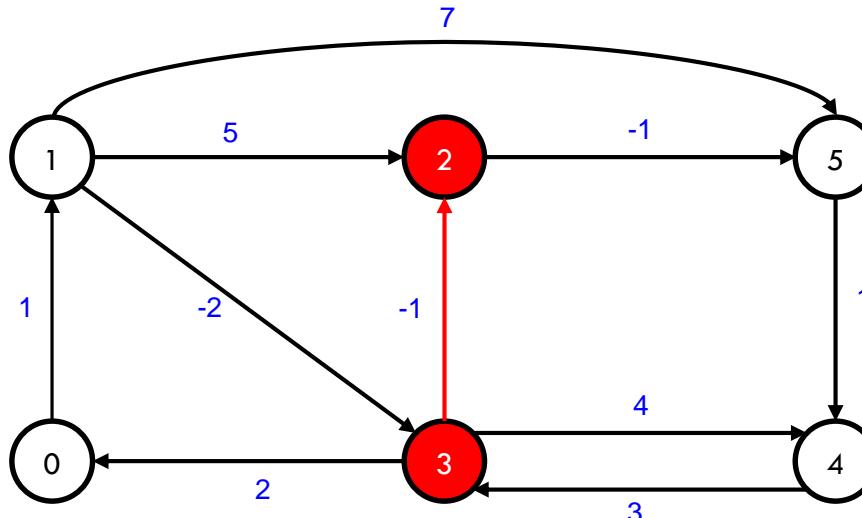
(0, 1, 1)	(1, 2, 5)	(1, 3, -2)	(1, 5, 7)	(2, 5, -1)	(3, 0, 2)	(3, 2, -1)	(3, 4, 4)	(4, 3, 3)	(5, 4, 1)
-----------	-----------	------------	-----------	------------	------------------	------------	-----------	-----------	-----------

dist	Định	0	1	2	3	4	5
	Chi phí	0	1	6	-1	∞	5

Lấy cạnh tiếp theo của **graph** ($u = 3, v = 0, w = 2$) để xem xét:

- Nếu chi phí tại **dist[u]** khác ∞ (-1 khác ∞) ✓
 - Và chi phí **dist[u] + w < dist[v]** ($(-1) + 2 > 0$) ✗
- Không cập nhật.

Bước 1: Chạy thuật toán ($j=6$)



0 1 2 3 4 5 6 7 8 9

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

dist	Định	0	1	2	3	4	5
	Chi phí	0	1	6	-1	∞	5

Lấy cạnh tiếp theo của **graph** ($u = 3, v = 2, w = -1$) để xem xét:

- Nếu chi phí tại **dist[u]** khác ∞ (-1 khác ∞) ✓
- Và chi phí **dist[u] + w < dist[v]** ($(-1) + (-1) < 6$) ✓

→ Cập nhật **dist[v] = dist[u] + w = -2**.

Bước 1: Chạy thuật toán ($j=6$)

Cập nhật $\text{dist}[v] = \text{dist}[u] + w = -1 + -1 = -2.$

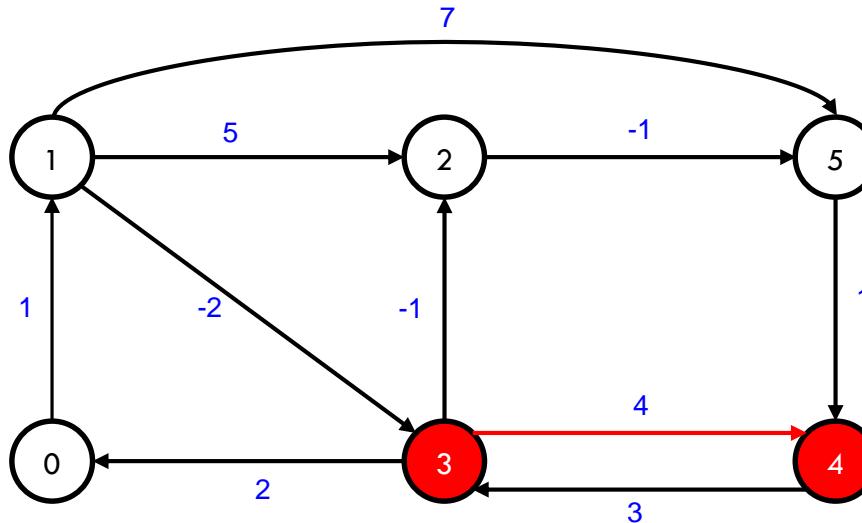
Cập nhật chi phí đỉnh đang xét (đỉnh 2) $\text{dist}[2] = -2.$

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

Xét cạnh (3, 2) \rightarrow cập nhật giá trị mảng lưu vết $\text{path}[2] = 3.$

Đỉnh cha	0	1	2	3	4	5
Lưu vết	-1	0	3	1	-1	2

Bước 1: Chạy thuật toán ($j=7$)



0 1 2 3 4 5 6 7 8 9

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

dist	Định	0	1	2	3	4	5
	Chi phí	0	1	6	-1	∞	5

Lấy cạnh tiếp theo của **graph** ($u = 3, v = 4, w = 4$) để xem xét:

- Nếu chi phí tại **dist[u]** khác ∞ (-1 khác ∞) ✓
- Và chi phí **dist[u] + w < dist[v]** ((-1) + 4 < ∞) ✓

→ Cập nhật **dist[v] = dist[u] + w = 3**.

Bước 1: Chạy thuật toán ($j=7$)

Cập nhật $\text{dist}[v] = \text{dist}[u] + w = -1 + 4 = 3$.

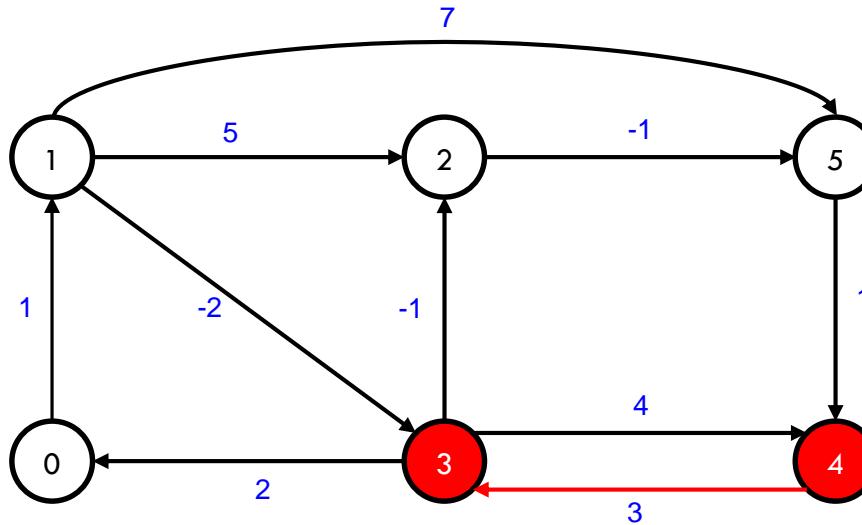
Cập nhật chi phí đỉnh đang xét (đỉnh 4) $\text{dist}[4] = 3$.

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

Xét cạnh (3, 4) \rightarrow cập nhật giá trị mảng lưu vết $\text{path}[4] = 3$.

Đỉnh cha	0	1	2	3	4	5
Lưu vết	-1	0	3	1	3	2

Bước 1: Chạy thuật toán ($j=8$)



0 1 2 3 4 5 6 7 8 9

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

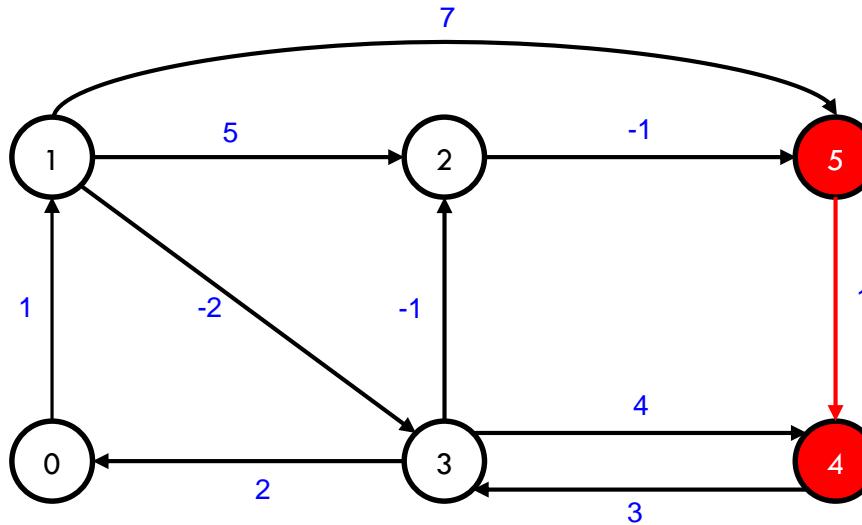
dist	Định	0	1	2	3	4	5
	Chi phí	0	1	-2	-1	3	5

Lấy cạnh tiếp theo của **graph** ($u = 4$, $v = 3$, $w = 3$) để xem xét:

- Nếu chi phí tại **dist[u]** khác ∞ (3 khác ∞) ✓
- Và chi phí **dist[u] + w < dist[v]** ($3 + 3 > -1$) ✗

→ Không cập nhật.

Bước 1: Chạy thuật toán ($j=9$)



0 1 2 3 4 5 6 7 8 9

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

dist	Định	0	1	2	3	4	5
	Chi phí	0	1	-2	-1	3	5

Lấy cạnh tiếp theo của **graph** ($u = 5$, $v = 4$, $w = 1$) để xem xét:

- Nếu chi phí tại **dist[u]** khác ∞ (5 khác ∞) ✓
- Và chi phí **dist[u] + w < dist[v]** ($5 + 1 > 3$) ✗

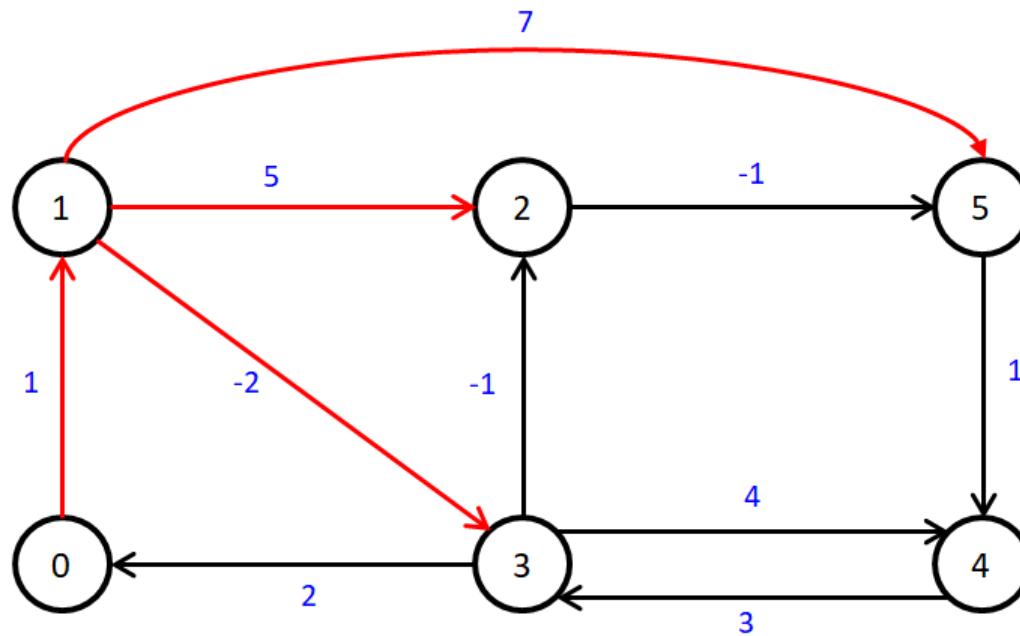
→ Không cập nhật.

BƯỚC 2

**CHẠY VÒNG LẶP DUYỆT QUA DANH
SÁCH CẠNH LẦN 2**

Bước 2: Chạy thuật toán

Tương tự như bước 1. Chạy vòng lặp lần lượt với $j=0, j=1, j=2, j=3$.

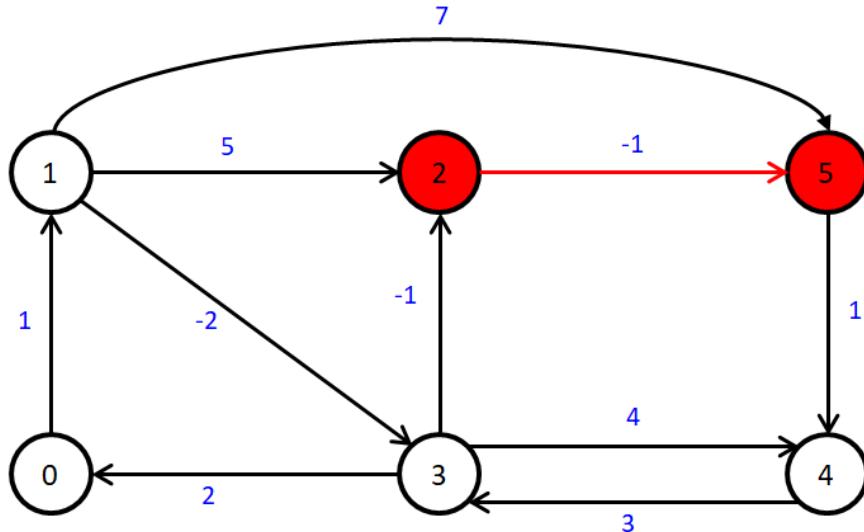


0 1 2 3 4 5 6 7 8 9

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

→ Không cập nhật.

Bước 2: Chạy thuật toán ($j=4$)



0 1 2 3 4 5 6 7 8 9

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

dist	Định	0	1	2	3	4	5
	Chi phí	0	1	-2	-1	3	5

Lấy cạnh tiếp theo của **graph** ($u = 2, v = 5, w = -1$) để xem xét:

- Nếu chi phí tại **dist[u]** khác ∞ (-2 khác ∞) ✓
 - Và chi phí **dist[u] + w < dist[v]** ($(-2) + (-1) < 5$) ✓
- Cập nhật **dist[v] = dist[u] + w = -3**.

Bước 2: Chạy thuật toán ($j=4$)

Cập nhật $\text{dist}[v] = \text{dist}[u] + w = (-2) + (-1) = -3.$

Cập nhật chi phí đỉnh đang xét (đỉnh 5) $\text{dist}[5] = -3.$

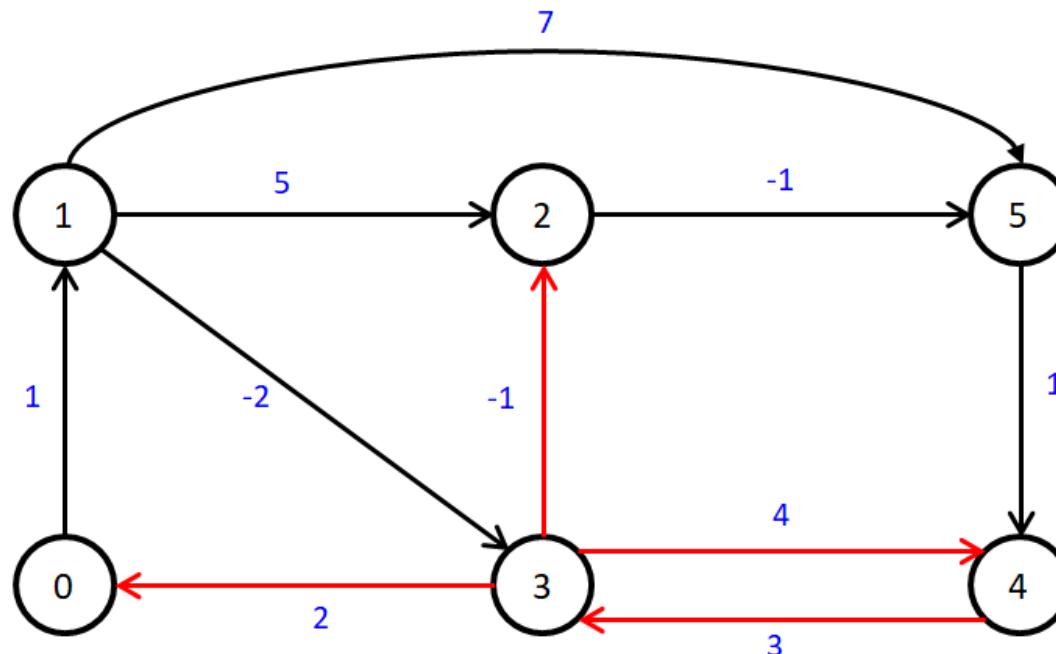
Đỉnh	0	1	2	3	4	5
Chi phí	0	1	-2	-1	3	-3

Xét cạnh (2, 5) \rightarrow cập nhật giá trị mảng lưu vết $\text{path}[5] = 2.$

Đỉnh cha	0	1	2	3	4	5
Lưu vết	-1	0	3	1	3	2

Bước 2: Chạy thuật toán

Tương tự như bước 1. Chạy vòng lặp lần lượt với $j=5, j=6, j=7, j=8$.

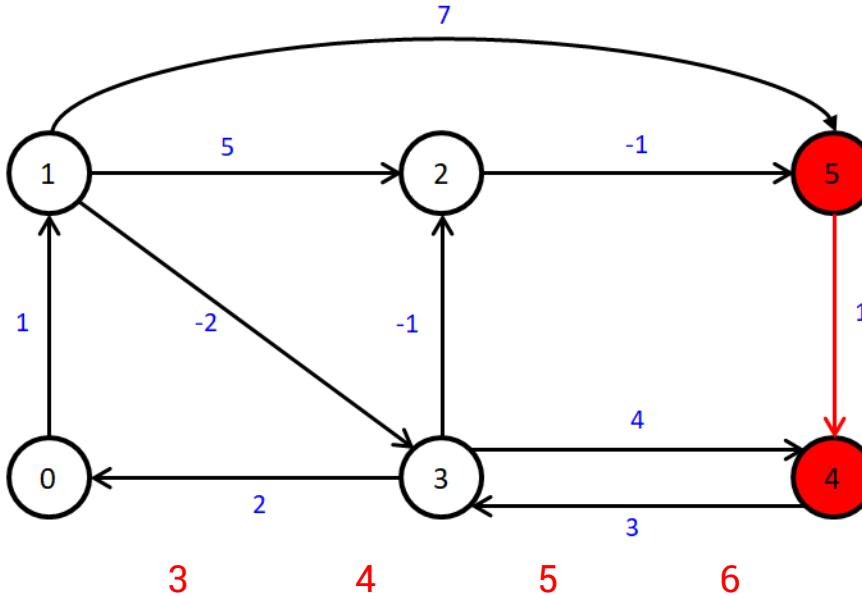


0 1 2 3 4 5 6 7 8 9

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

→ Không cập nhật.

Bước 2: Chạy thuật toán ($j=9$)



0 1 2 3 4 5 6 7 8 9

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

dist	Định	0	1	2	3	4	5
	Chi phí	0	1	-2	-1	3	-3

Lấy cạnh tiếp theo của **graph** ($u = 5$, $v = 4$, $w = 1$) để xem xét:

- Nếu chi phí tại **dist[u]** khác ∞ (-3 khác ∞) ✓
- Và chi phí **dist[u] + w < dist[v]** ((-3) + 1 < 3) ✓

→ Cập nhật **dist[v] = dist[u] + w = -2**.

Bước 2: Chạy thuật toán ($j=9$)

Cập nhật $\text{dist}[v] = \text{dist}[u] + w = (-3) + 1 = -2$.

Cập nhật chi phí đỉnh đang xét (đỉnh 4) $\text{dist}[4] = -2$.

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

Xét cạnh (5, 4) \rightarrow cập nhật giá trị mảng lưu vết $\text{path}[4] = 5$.

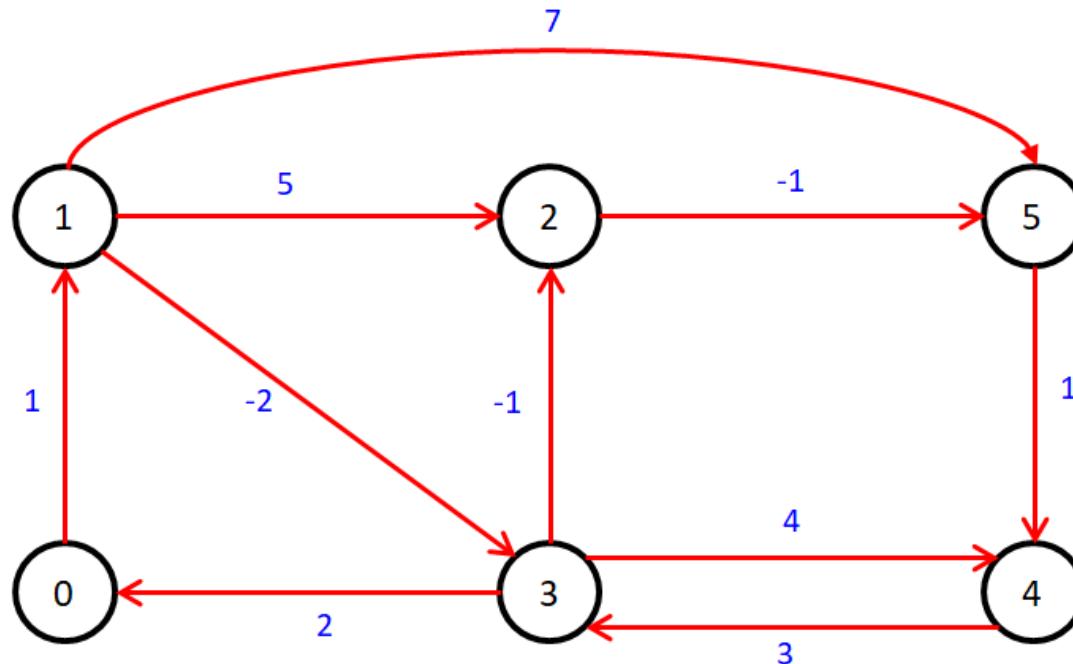
Đỉnh cha	0	1	2	3	4	5
Lưu vết	-1	0	3	1	5	2

BƯỚC 3

**CHẠY VÒNG LẶP DUYỆT QUA DANH
SÁCH CẠNH LẦN 3**

Bước 3: Chạy thuật toán

Tương tự như bước 1. Chạy vòng lặp lần lượt từ $j=0$ đến $j=9$.



0 1 2 3 4 5 6 7 8 9

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

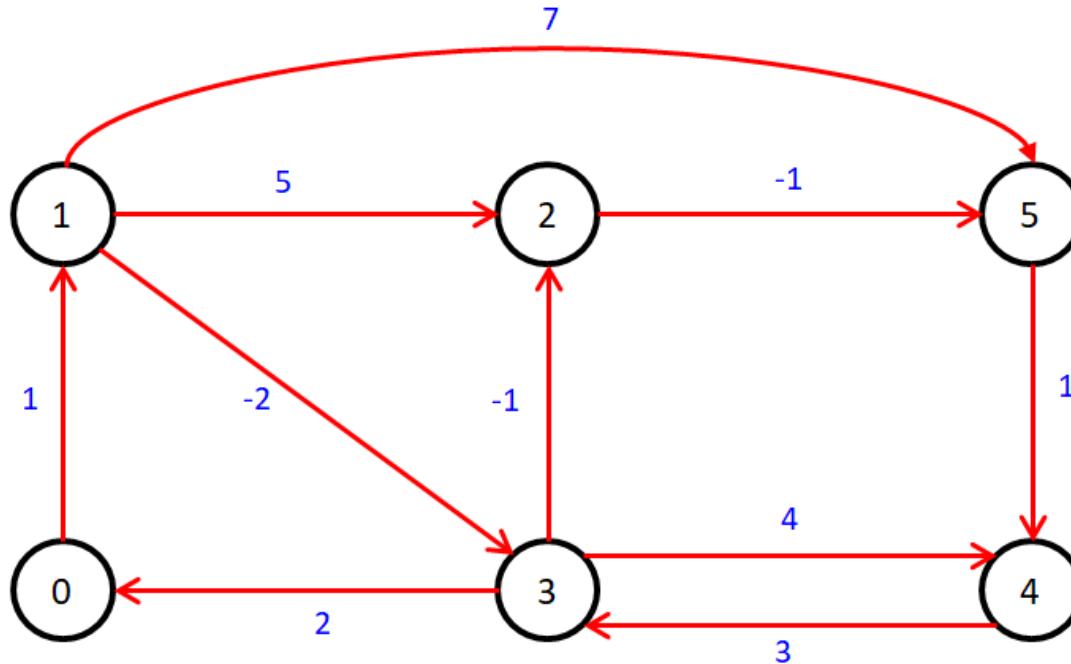
→ Không cập nhật.

BƯỚC 4

**CHẠY VÒNG LẶP DUYỆT QUA DANH
SÁCH CẠNH LẦN 4**

Bước 4: Chạy thuật toán

Tương tự như bước 1. Chạy vòng lặp lần lượt từ $j=0$ đến $j=9$.



0 1 2 3 4 5 6 7 8 9

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

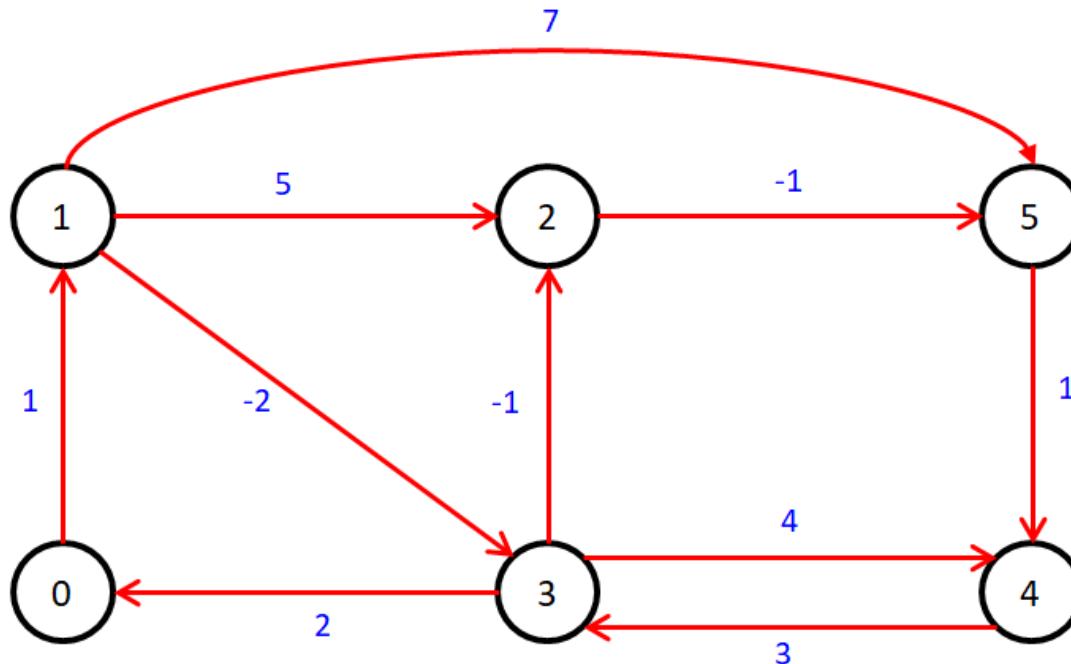
→ Không cập nhật.

BƯỚC 5

**CHẠY VÒNG LẶP DUYỆT QUA DANH
SÁCH CẠNH LẦN 5**

Bước 5: Chạy thuật toán

Tương tự như bước 1. Chạy vòng lặp lần lượt từ $j=0$ đến $j=9$.



0 1 2 3 4 5 6 7 8 9

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

→ Không cập nhật.

Dùng thuật toán và in ra đường đi

Tìm đường đi ngắn nhất từ 0 đến 4.



path

Đỉnh cha	0	1	2	3	4	5
Lưu vết	-1	0	3	1	5	2

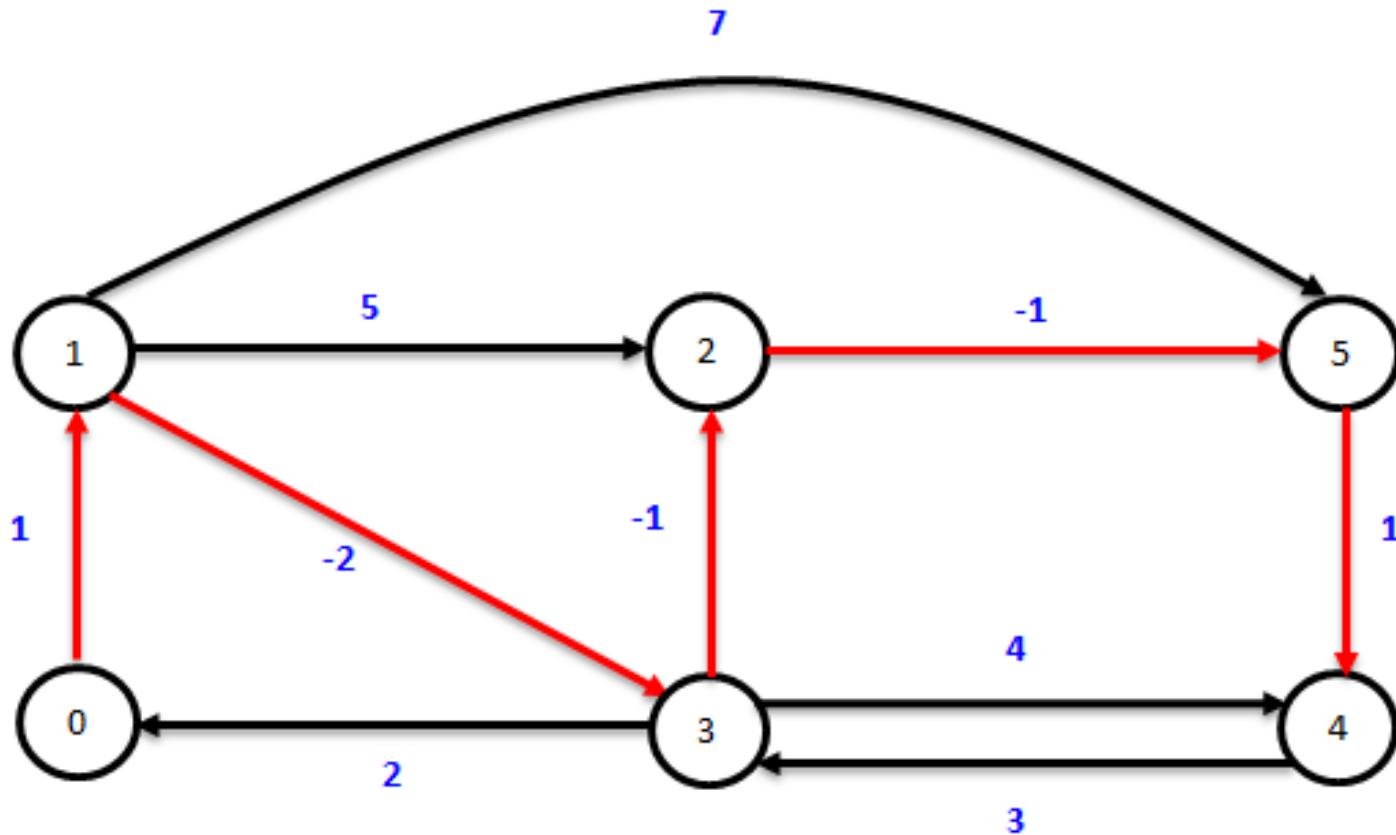
dist

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

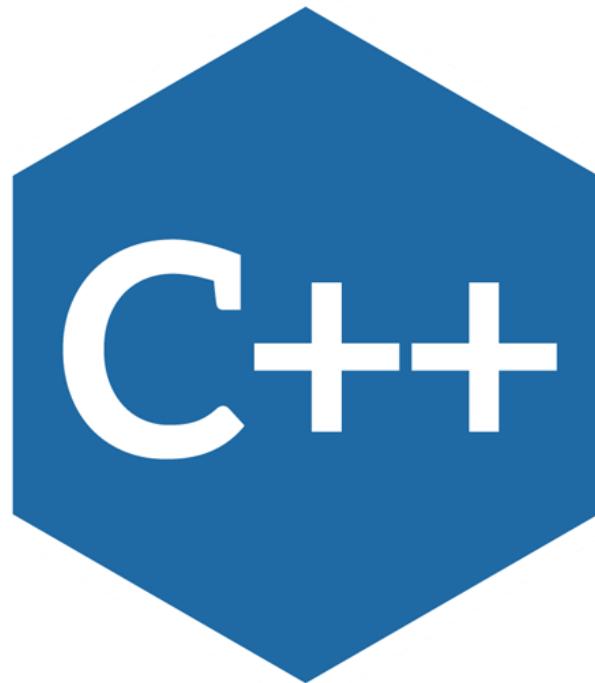
**0 → 1 → 3 → 2 → 5 → 4
Chi phí: -2**

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 Bellman-Ford

Khai báo thư viện và các biến toàn cục:

```
1. #include <iostream>
2. #include <vector>
3. using namespace std;
4. #define MAX 105
5. const int INF = 1e9;
6. struct Edge
7. {
8.     int source;
9.     int target;
10.    int weight;
11. };
12. vector<int> dist(MAX, INF);
13. vector<Edge> graph;
14. int n, m;
15. vector<int> path(MAX, -1);
```



Source Code Bellman-Ford

Thuật toán chính Bellman-Ford (part 1)

```
16. bool BellmanFord(int s)
17. {
18.     int u, v, w;
19.     dist[s] = 0;
20.     for (int i = 1; i <= n - 1; i++)
21.     {
22.         for (int j = 0; j < m; j++)
23.         {
24.             u = graph[j].source;
25.             v = graph[j].target;
26.             w = graph[j].weight;
27.             if (dist[u] != INF && (dist[u] + w < dist[v])) {
28.                 dist[v] = dist[u] + w;
29.                 path[v] = u;
30.             }
31.         }
32.     }
// to be continued
```



Source Code Bellman-Ford

Thuật toán chính Bellman-Ford (part 2)

```
//Để đảm bảo không tồn tại chu trình âm thì bellman-ford mới tìm được đường đi.  
33.     for (int i = 0; i < m; i++)  
34.     {  
35.         u = graph[i].source;  
36.         v = graph[i].target;  
37.         w = graph[i].weight;  
38.         if (dist[u] != INF && (dist[u] + w < dist[v]))  
39.             return false;  
40.     }  
41.     return true;  
42. }
```



Source Code Bellman-Ford

```
43. int main()
44. {
45.     int s, t, u, v, w;
46.     cin >> n >> m;
47.     for (int i = 0; i < m; i++)
48.     {
49.         cin >> u >> v >> w;
50.         graph.push_back(Edge(u, v, w));
51.     }
52.     s = 0; t = 4;
53.     bool res = BellmanFord(s);
54.     if (res == false)
55.         cout << "Graph contains negative weight cycle" << endl;
56.     else
57.         cout << dist[t] << endl;
58.     return 0;
59. }
```



MÃ NGUỒN MINH HỌA BẰNG PYTHON



Source Code Bellman-Ford

Khai báo thư viện và các biến toàn cục:

```
1. INF = 10**9
2. MAX = 105
3.
4. class Edge:
5.     def __init__(self, source, target, weight):
6.         self.source = source
7.         self.target = target
8.         self.weight = weight
9.
10. dist = [INF for _ in range(MAX)]
11. path = [-1 for _ in range(MAX)]
12. graph = []
```



Source Code Bellman-Ford

```
13. def BellmanFord(s):
14.     dist[s] = 0
15.     for i in range(1, n):
16.         for j in range(m):
17.             u = graph[j].source
18.             v = graph[j].target
19.             w = graph[j].weight
20.             if (dist[u] != INF) and (dist[u] + w < dist[v]):
21.                 dist[v] = dist[u] + w
22.                 path[v] = u
23.             for i in range(m):
24.                 u = graph[i].source
25.                 v = graph[i].target
26.                 w = graph[i].weight
27.                 if (dist[u] != INF) and (dist[u] + w < dist[v]):
28.                     return False
29.     return True
```



Source Code Bellman-Ford

Hàm main.

```
30. if __name__ == '__main__':
31.     n, m = map(int, input().split())
32.     for i in range(m):
33.         u, v, w = map(int, input().split())
34.         graph.append(Edge(u, v, w))
35.     s, t = 0, 4
36.     res = BellmanFord(s)
37.     if not res:
38.         print("Graph contains negative weight cycle")
39.     else:
40.         print(dist[t])
```



MÃ NGUỒN MINH HỌA BẰNG JAVA



Source Code Bellman-Ford

Class Edge

```
1. import java.util.Arrays;
2. import java.util.Scanner;
3. class Edge {
4.     public int source;
5.     public int target;
6.     public int weight;
7.
8.     public Edge(int source, int target, int weight) {
9.         this.source = source;
10.        this.target = target;
11.        this.weight = weight;
12.    }
13. }
```



Source Code Bellman-Ford

```
14. public class Main {  
15.     private static final int INF = (int)1e9;  
16.     private static final int MAX = 105;  
17.     private static int[] dist = new int[MAX];  
18.     private static Edge[] graph;  
19.     private static int n, m;  
20.     private static int[] path = new int[MAX];  
21.     private static boolean BellmanFord(int s) {  
22.         int u, v, w;  
23.         dist[s] = 0;  
24.         for (int i = 1; i <= n - 1; i++) {  
25.             for (int j = 0; j < m; j++) {  
26.                 u = graph[j].source;  
27.                 v = graph[j].target;  
28.                 w = graph[j].weight;  
29.                 if (dist[u] != INF && dist[u] + w < dist[v]) {  
30.                     dist[v] = dist[u] + w;  
31.                     path[v] = u;  
32.                 }  
33.             }  
34.         }  
// to be continued
```



Source Code Bellman-Ford

Thuật toán chính Bellman-Ford (part 2)

```
// kiểm tra chu trình âm  
35.         for (int i = 0; i < m; i++) {  
36.             u = graph[i].source;  
37.             v = graph[i].target;  
38.             w = graph[i].weight;  
39.             if (dist[u] != INF && dist[u] + w < dist[v]) {  
40.                 return false;  
41.             }  
42.         }  
43.         return true;  
44.     }
```



Source Code Bellman-Ford

Hàm main (part 1)

```
45.     public static void main(String[] args) {  
46.         Scanner sc = new Scanner(System.in);  
47.         n = sc.nextInt();  
48.         m = sc.nextInt();  
49.         graph = new Edge[m];  
50.         Arrays.fill(dist, INF);  
51.         Arrays.fill(path, -1);  
52.         int u, v, w;  
53.         for (int i = 0; i < m; i++) {  
54.             u = sc.nextInt();  
55.             v = sc.nextInt();  
56.             w = sc.nextInt();  
57.             graph[i] = new Edge(u, v, w);  
58.         }  
// to be continued
```



Source Code Bellman-Ford

Hàm main (part 2)

```
59.         int s = 0, t = 4;
60.         boolean res = BellmanFord(s);
61.         if (!res) {
62.             System.out.println("Graph contains negative weight cycle");
63.         }
64.         else {
65.             System.out.println(dist[t]);
66.         }
67.     }
68. }
```



Hỏi đáp

