

## Tìm đường đi ngắn nhất với Floyd

### 1. Thuật toán Floyd

Cho  $G=(X, E)$  là một đồ thị có trọng không âm gồm  $n$  đỉnh. Thuật toán Floyd được dùng để tìm đường đi ngắn nhất giữa tất cả cặp đỉnh bất kỳ của một đồ thị  $G$ .

Gọi  $L$  là ma trận trọng lượng (với qui ước  $L_{hk} = 0$  nếu không có cạnh nối từ đỉnh  $h$  đến đỉnh  $k$ ).

Ta sử dụng thêm một ma trận  $n \times n$  để lưu vết của quá trình tìm đường đi:

- $sau\_nut1[i,j]$  : lưu chỉ số của đỉnh **ngay sau**  $i$  trên đường đi từ  $i$  đến  $j$ .

**Bước 1:** (khởi tạo)  $\forall u, v \in X$  :

Nếu  $L_{uv} > 0$  thì  
     $sau\_nut1[u,v] = v$ ;  
Nếu không  
     $sau\_nut1[u,v] = -1$ ;  
Cuối nếu

**Bước 2:** Với mỗi **đỉnh trung gian**  $k$ , tìm cặp  $i, j$  nào thỏa mãn  $L[i,j] = 0$  hoặc  $L[i, j] > L[i, k] + L[k, j]$

Nếu thỏa mãn thì  
     $L[i,j] = L[i,k] + L[k,j]$ ;  
     $sau\_nut1[i,j] = sau\_nut1[i,k]$ ;  
Cuối nếu

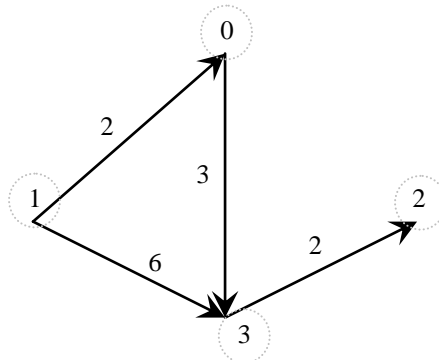
Cuối với mọi  $k, i, j$

**Chú ý:**

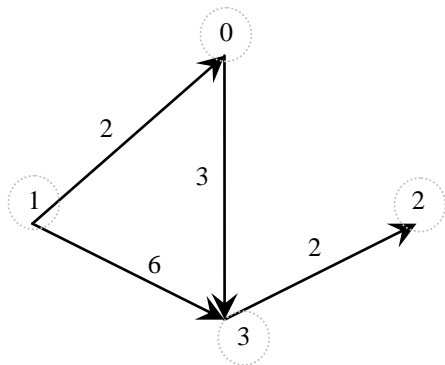
- Đỉnh  $k$  được gọi là trung gian của  $i, j$  nếu nó có đường đi từ  $i \rightarrow k$  và  $k \rightarrow j$ .
- Khi thuật toán kết thúc, nếu  $L_{ij} = 0$  thì không tồn tại đường đi từ  $i$  đến  $j$ , nếu ngược lại thì  $L_{ij}$  là độ dài đường đi ngắn nhất.
- Trong cách làm ở trên, việc xuất ra đường đi chỉ cần duyệt một khoảng từ  $j \rightarrow k$  có đỉnh trung gian hay không vì từ  $i \rightarrow k$  không có đỉnh trung gian nào.
- Ta có thể không cần khởi tạo ma trận lưu chỉ số  $sau\_nut1$  như ở trên (set bằng 0). Như vậy trong bước 2, chỉ lưu giá trị trung gian đơn giản như sau:  $sau\_nut1[i,j] = k$ . Tuy nhiên, việc xuất ra đường đi từ  $i$  đến  $j$  chúng ta phải tiến hành duyệt hai khoảng  $i \rightarrow k$  và  $k \rightarrow j$  để kiểm tra xem có đỉnh nào làm trung gian bên trong hai khoảng này nữa không.

## 2. Ví dụ Floyd

Cho đồ thị sau:



Tìm đường đi ngắn nhất của mọi cặp đỉnh trong đồ thị.



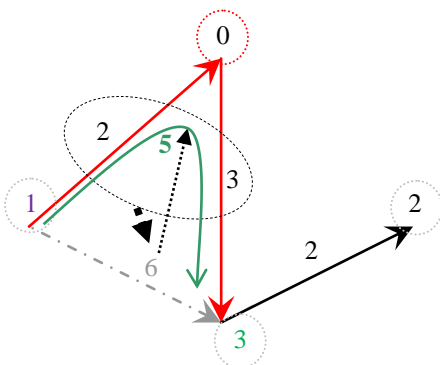
**Bước 1: khởi tạo**

Ma trận trọng số:

L[...,...]	0	1	2	3
0	0	0	0	3
1	2	0	0	6
2	0	0	0	0
3	0	0	2	0

sau\_nut1 sẽ được khởi tạo giá trị là j nếu có cạnh nối i đến j và được khởi tạo giá trị 0 nếu ngược lại

sau_nut1[...]	0	1	2	3
0	-1	-1	-1	3
1	0	-1	-1	3
2	-1	-1	-1	-1
3	-1	-1	2	-1



**Bước 2:**

**Xét đỉnh : k = 0**

- \* [i = 0, j = ?]: k không là trung gian.
- \* [i = 1, j = 0]: k không là trung gian.
- \* [i = 1, j = 1]: k không là trung gian.
- \* [i = 1, j = 2]: k không là trung gian.
- \* [i = 1, j = 3]:

N/x:  $L[i,j] > L[i,k] + L[k,j]$  hay

$L[1,3] = 6 > L[1,0] + L[0,3] = 5$  nên:

$L[1,3] = L[1,0] + L[0,3] = 5;$

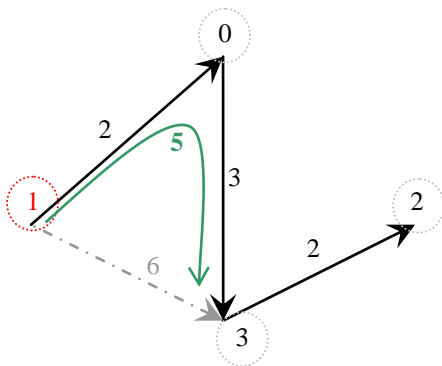
$sau\_nut1[1,3] = sau\_nut1[1,0] = 0;$

L[...,...]	0	1	2	3
0	0	0	0	3
1	2	0	0	6 → 5
2	0	0	0	0
3	0	0	2	0

sau_nut1[...]	0	1	2	3
0	-1	-1	-1	3
1	0	-1	-1	3 → 0
2	-1	-1	-1	-1
3	-1	-1	2	-1

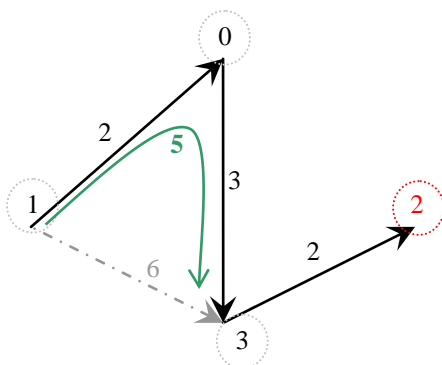
\*  $[i = 2, j = ?]$ : k không là trung gian.

\*  $[i = 3, j = ?]$ : k không là trung gian.



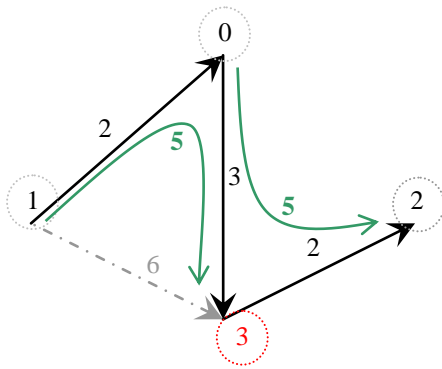
**Xét đỉnh : k = 1**

\*  $[i = ?, j = ?]$ : k không là trung gian.



**Xét đỉnh : k = 2**

\*  $[i = ?, j = ?]$ : k không là trung gian.



### Xét đỉnh : $k = 3$

\*  $[i = 0, j = 0]$ : k không là trung gian.

\*  $[i = 0, j = 1]$ : k không là trung gian.

\*  $[i = 0, j = 2]$ :

Vì  $L[i, j] = 0$  hay  $L[0, 2] = 0$  nên:

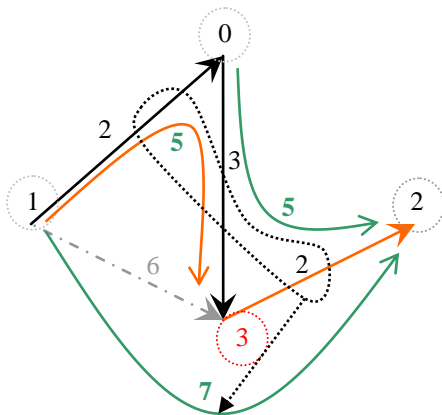
$$L[0, 2] = L[0, 3] + L[3, 2] = 5;$$

$$\text{sau\_nut1}[0, 2] = \text{sau\_nut1}[0, 3] = 3;$$

$L[...]$	0	1	2	3
0	0	0	0 → 5	3
1	2	0	0	5
2	0	0	0	0
3	0	0	2	0

$\text{sau\_nut1}[...]$	0	1	2	3
0	-1	-1	-1 → 3	3
1	0	-1	-1	0
2	-1	-1	-1	-1
3	-1	-1	2	-1

\*  $[i = 0, j = 3]$ : k không là trung gian.



( $k = 3$ : cont)

\*  $[i = 1, j = 0]$ : k không là trung gian.

\*  $[i = 1, j = 1]$ : k không là trung gian.

\*  $[i = 1, j = 2]$ :

N/x:  $L[1, 2] = 0$  nên:

$$L[1, 2] = L[1, 3] + L[3, 2] = 5 + 2 = 7;$$

$$\text{sau\_nut1}[1, 2] = \text{sau\_nut1}[1, 3] = 0;$$

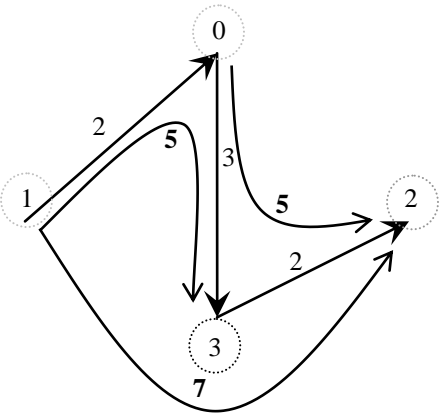
$L[...]$	0	1	2	3
0	0	0	5	3
1	2	0	0 → 7	5
2	0	0	0	0
3	0	0	2	0

$\text{sau\_nut1}[...]$	0	1	2	3
0	-1	-1	3	3
1	0	-1	-1 → 0	0
2	-1	-1	-1	-1
3	-1	-1	2	-1

\*  $[i = 1, j = 3]$ : k không là trung gian.

\*  $[i = 2, j = ?]$ : k không là trung gian.

\*  $[i = 3, j = ?]$ : k không là trung gian.

	Kết thúc thuật toán
“?”: đại diện cho bất kỳ đỉnh nào	

### 3. Hướng dẫn cài đặt Floyd

```
#define MAX 100
```

```
typedef struct {
    int n;
    int L[MAX][MAX];      // ma trận trọng số
}GRAPH;
```

```
int sau_nut1[MAX,MAX];    // lưu chỉ số của đỉnh ngay sau i trên đường đi từ i đến j.
```

```
/*Khởi tạo các thông số cho thuật toán*/
```

```
void Init(GRAPH g)
{
    for (i...)
        for (j ...)
            if (...) // có đường đi từ i,j
            {
                sau_nut1[i,j] = i;
            }
            else //gán giá trị -1
                ...
}
```

```
...
```

```
/*Thuật toán Floyd*/
```

```
void FloydAlg(GRAPH &g)
```

```
{
    //Khoi tao cac thong so cho thuat toan
    ....
    //Duyệt từng đỉnh k
    for(k...)
        // kiểm tra k có là trung gian nối từ đỉnh i đến j hay không
        for(i...)
            if(...[i][k]>0)
                for(j...)
                    if(...[k][j]>0)
```

```

// cập nhật độ dài đường đi nếu có đường ngắn hơn
if((...[i][j]==0 && i!=j) || ....[i][j] > ....[i][k] + ...[k][j])
{
    ...[i][j] = ...[i][k] + ...[k][j];
    sau_nut1[i][j] = sau_nut1[i][k];
}
}

void PrintScreen(Graph g)
{
    for(i...)
        for(j...)
            // Nếu có đường đi từ i đến j, xuất ra lộ trình đi
            if (...[i][j] > 0)
            {
                int s, t;
                printf("%d->%d=%d: ", i, j, ...L[i][j]);
                s = i;
                t = j;
                printf("%d->", s);
                do {
                    t = sau_nut1[s][t];
                    printf("%d->", t);
                } while (s!=t);
                printf("\n");
            }
}

```