TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Toán rời rạc

Nguyễn Khánh Phương

Bộ môn Khoa học máy tính
E-mail: phuongnk@soict.hust.edu.vn

PHẦN 1: LÝ THUYẾT TỔ HỢP

(Combinatorial Theory)

PHẦN 2: LÝ THUYẾT ĐỒ THỊ

(Graph Theory)

Phần thứ hai

LÝ THUYẾT ĐỒ THỊ

Graph Theory

Nguyễn Khánh Phương

Bộ môn Khoa học Máy tính,
Viện CNTT và Truyền thông,
Đại học Bách khoa Hà nội,
E-mail: phuongnk@soict.hust.edu.vn

Nội dung phần 2

Chương 1. Các khái niệm cơ bản

Chương 2. Biểu diễn đồ thị

Chương 3. Duyệt đồ thị

Chương 4. Cây và cây khung của đồ thị

Chương 5. Bài toán đường đi ngắn nhất

Chương 6. Bài toán luồng cực đại trong mạng

Nội dung chi tiết

5.1. Bài toán đường đi ngắn nhất (ĐĐNN)

5.2. Tính chất của ĐĐNN, Giảm cận trên

5.3. Thuật toán Bellman-Ford

5.4. Thuật toán Dijkstra

5.5. Đường đi ngắn nhất trong đồ thị không có chu trình

5.6. Thuật toán Floyd-Warshall

5.1. Bài toán đường đi ngắn nhất

- Cho đơn đồ thị có hướng $G = (V,E)$ với hàm trọng số w: $E \to R$ ($w(e)$ được gọi là độ dài hay trọng số của cạnh e)

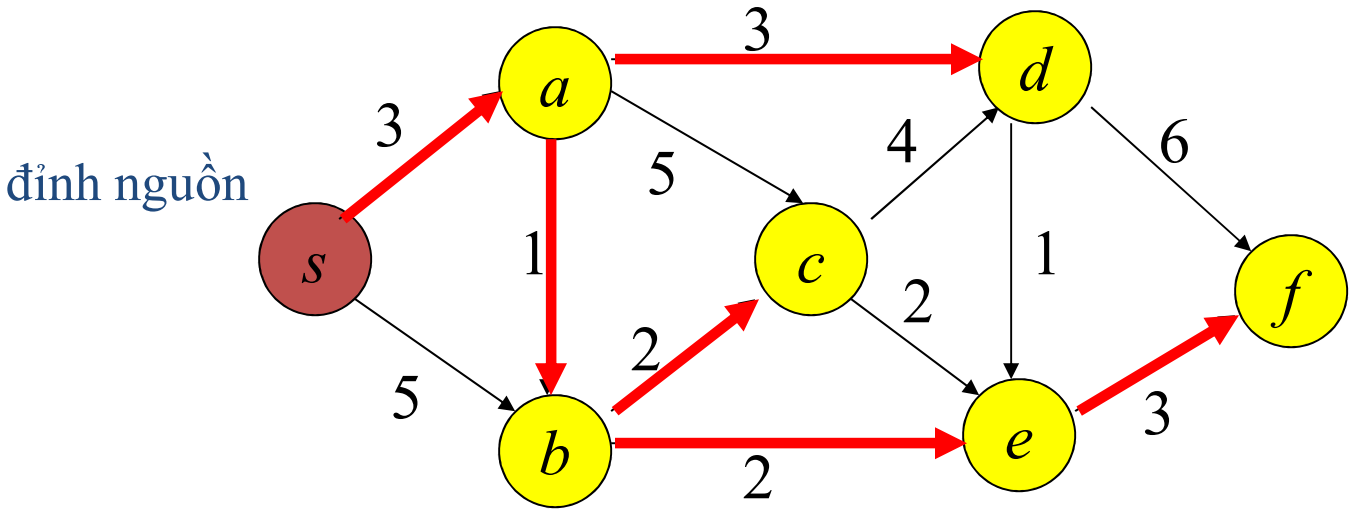
- Độ dài của đường đi $P = v_1 \xrightarrow{w(v_1,v_2)} v_2 \xrightarrow{w(v_2,v_3)} \dots \xrightarrow{w(v_{k-1},v_k)} v_k$ là số

$$w(P) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$$

- Đường đi ngắn nhất từ đỉnh u đến đỉnh v là đường đi có độ dài ngắn nhất trong số các đường đi nối u với v.
- Độ dài của đường đi ngắn nhất từ u đến v còn được gọi là khoảng cách từ u tới v và ký hiệu là $\delta(u,v)$.

Ví dụ

Cho đồ thị có trọng số $G = (V, E)$, và đỉnh nguồn $s \in V$, hãy tìm đường đi ngắn nhất từ s đến mỗi đỉnh còn lại.



	s	*a*	*b*	*c*	*d*	*e*	*f*
path	*s*	*s,a*	*s,a,b*	*s,a,b,c*	*s,a,d*	*s,a,b,e*	*s,a,b,e,f*
weight	0	3	4	6	6	6	9

Các ứng dụng thực tế

- Giao thông (Transportation)
- Truyền tin trên mạng (Network routing) (cần hướng các gói tin đến đích trên mạng theo đường nào?)
- Điều khiển robot (Robot path planning)
- Truyền thông (Telecommunications)
- Speech interpretation (best interpretation of a spoken sentence)
- Medical imaging
- ...

Các dạng bài toán ĐĐNN

1. **Bài toán một nguồn một đích:** Cho hai đỉnh *s* và *t*, cần tìm đường đi ngắn nhất từ *s* đến *t*.
2. **Bài toán một nguồn nhiều đích:** Cho *s* là đỉnh nguồn, cần tìm đường đi ngắn nhất từ s đến tất cả các đỉnh còn lại.
3. **Bài toán mọi cặp:** Tìm đường đi ngắn nhất giữa mọi cặp đỉnh của đồ thị.

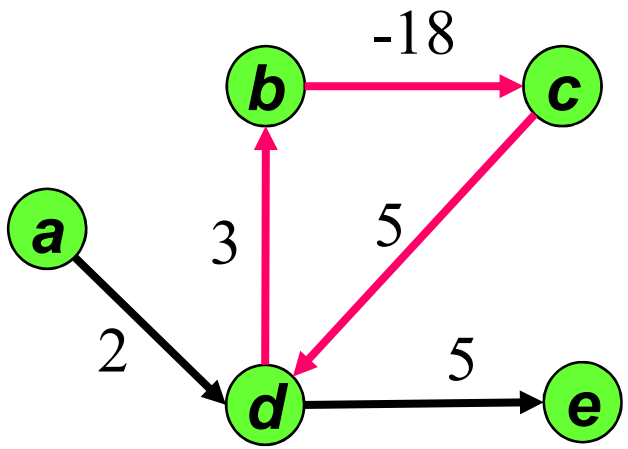
Đường đi ngắn nhất theo số cạnh - BFS.

Nhận xét:

- Các bài toán được xếp theo thứ tự từ đơn giản đến phức tạp
- Hễ có thuật toán hiệu quả để giải một trong ba bài toán thì thuật toán đó cũng có thể sử dụng để giải hai bài toán còn lại

Giả thiết cơ bản

- Nếu đồ thị có chu trình âm thì độ dài đường đi giữa hai đỉnh nào đó có thể làm nhỏ tuỳ ý:



Chu trình: $(d \to b \to c \to d)$

Độ dài = -10

Xét đường đi từ a đến e:

P: $a \to \sigma(d \to b \to c \to d) \to e$

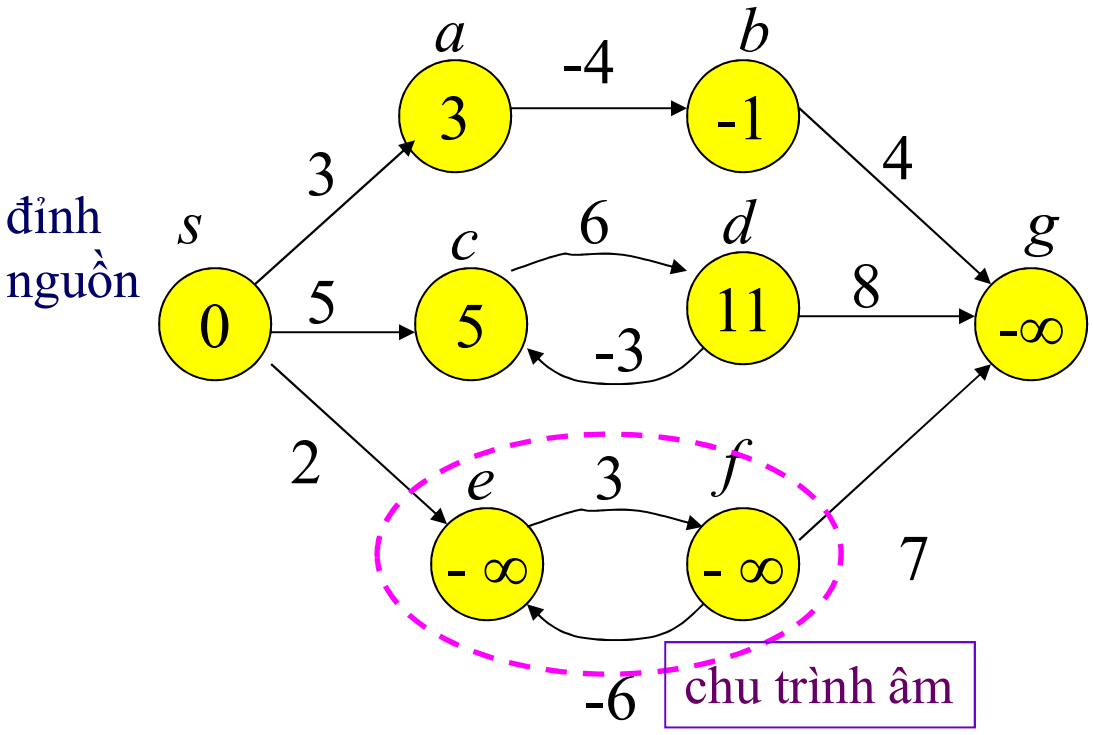
$w(P) = 7\text{-}10\sigma \to -\infty$, khi $\sigma \to +\infty$

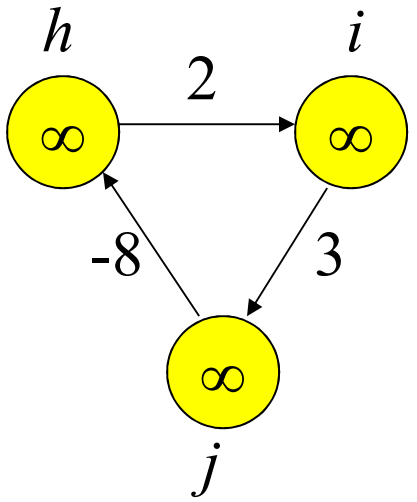
Giả thiết:

Đồ thị không chứa chu trình độ dài âm (gọi tắt là chu trình âm)

Trọng số âm

Độ dài của đường đi ngắn nhất có thể là -∞ hoặc ∞.





không đạt tới được từ *s*

Nội dung chi tiết

5.1. Bài toán đường đi ngắn nhất (ĐĐNN)

5.2. Tính chất của ĐĐNN, Giảm cận trên

5.3. Thuật toán Bellman-Ford

5.4. Thuật toán Dijkstra

5.5. Đường đi ngắn nhất trong đồ thị không có chu trình

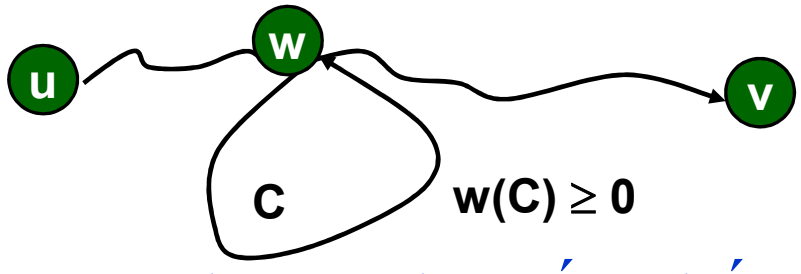
5.6. Thuật toán Floyd-Warshall

Các tính chất của đường đi ngắn nhất

- ***Tính chất 1.*** *Đường đi ngắn nhất luôn có thể tìm trong số các đường đi đơn.*

đường đi không có đỉnh nào bị lặp lại trên nó.

Chứng minh: Bởi vì việc loại bỏ chu trình độ dài không âm khỏi đường đi không làm tăng độ dài của nó.



- ***Tính chất 2.*** *Mọi đường đi ngắn nhất trong đồ thị G đều đi qua không quá n-1 cạnh, trong đó n là số đỉnh.*
  - *Như là hệ quả của tính chất 1*

>= *n* cạnh → không phải đường đi đơn

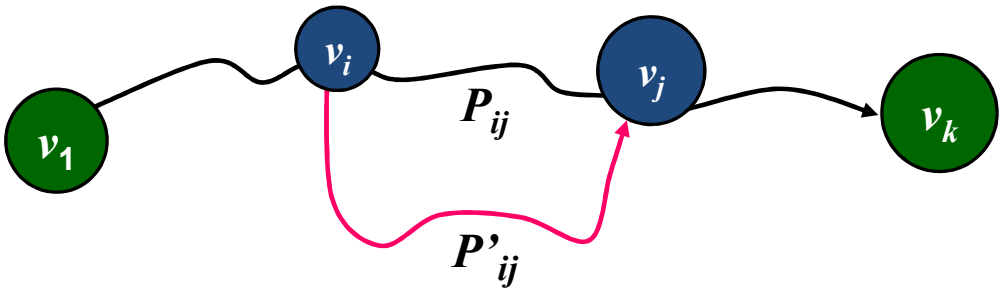
Các tính chất của đường đi ngắn nhất

Tính chất 3: Giả sử $P = \langle v_1, v_2, \ldots, v_k \rangle$ là đđnn từ v_1 đến v_k. Khi đó, $P_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ là đđnn từ v_i đến v_j, với $1 \le i \le j \le k$.

(Bằng lời: Mọi đoạn đường con của đường đi ngắn nhất đều là đường đi ngắn nhất)

Chứng minh. Phản chứng. Nếu P_{ij} không là đđnn từ v_i đến v_j, thì tìm được P'_{ij} là đường đi từ v_i đến v_j thoả mãn $w(P'_{ij}) < w(P_{ij})$. Khi đó gọi P' là đường đi thu được từ P bởi việc thay đoạn P_{ij} bởi P'_{ij}, ta có:

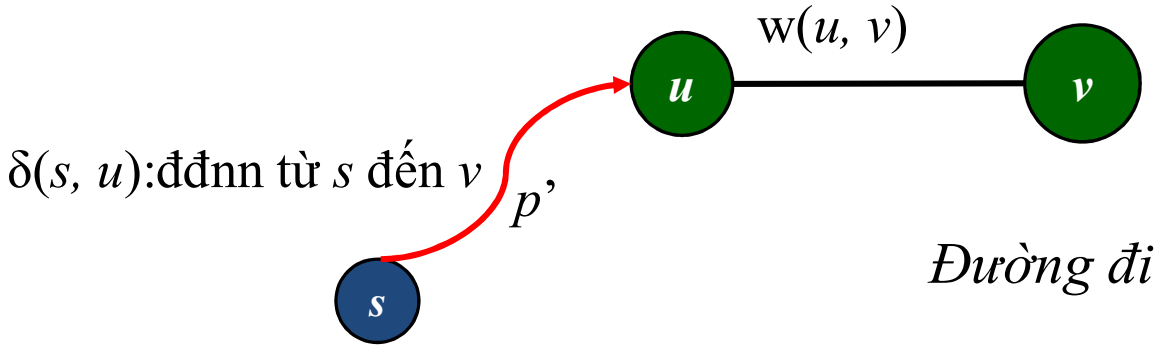
$$w(P') < w(P) \text{ ?!}$$



Các tính chất của đường đi ngắn nhất

Ký hiệu: $\delta(u, v)$ = độ dài đđnn từ u đến v (gọi là khoảng cách từ u đến v)

<u>Hệ quả:</u> Giả sử P là đđnn từ s tới v, trong đó $P = s \xrightarrow{p'} u \rightarrow v$.
Khi đó $\delta(s, v) = \delta(s, u) + w(u, v)$.



Đường đi ngắn nhất từ s đến v:

$P = s \xrightarrow{p'} u \rightarrow v$

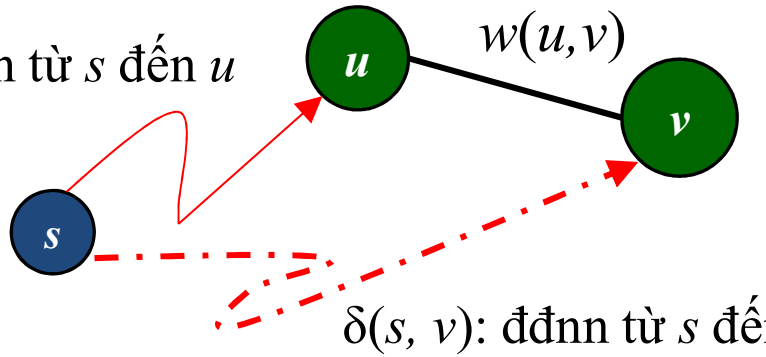
Các tính chất của đường đi ngắn nhất

Ký hiệu: $\delta(u, v)$ = độ dài đđnn từ u đến v (gọi là khoảng cách từ u đến v)

Hệ quả: Giả sử P là đđnn từ s tới v, trong đó $P = s \xrightarrow{p'} u \rightarrow v$.
Khi đó $\delta(s, v) = \delta(s, u) + w(u, v)$.

Tính chất 4: Giả sử $s \in V$. Đối với mỗi cạnh $(u,v) \in E$, ta có
$\delta(s, v) \leq \delta(s, u) + w(u,v)$.

$\delta(s, u)$:đđnn từ s đến u



$\delta(s, v)$: đđnn từ s đến v

~~Nếu $\delta(s, v) > \delta(s, u) + w(u,v)$???~~

Nếu $\delta(s, v) < \delta(s, u) + w(u,v)$???

Nếu $\delta(s, v) = \delta(s, u) + w(u,v)$???

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Đường đi ngắn nhất xuất phát từ một đỉnh

Biểu diễn đường đi ngắn nhất

Các thuật toán tìm đường đi ngắn nhất làm việc với hai mảng:

- $d(v)$ = độ dài đường đi từ s đến v ngắn nhất hiện biết
  (cận trên cho độ dài đường đi ngắn nhất thực sự).
- $p(v)$ = đỉnh đi trước v trong đường đi nói trên
  (sẽ sử dụng để truy ngược đường đi từ s đến v) .

$$\delta(s,v) \le d(v)$$

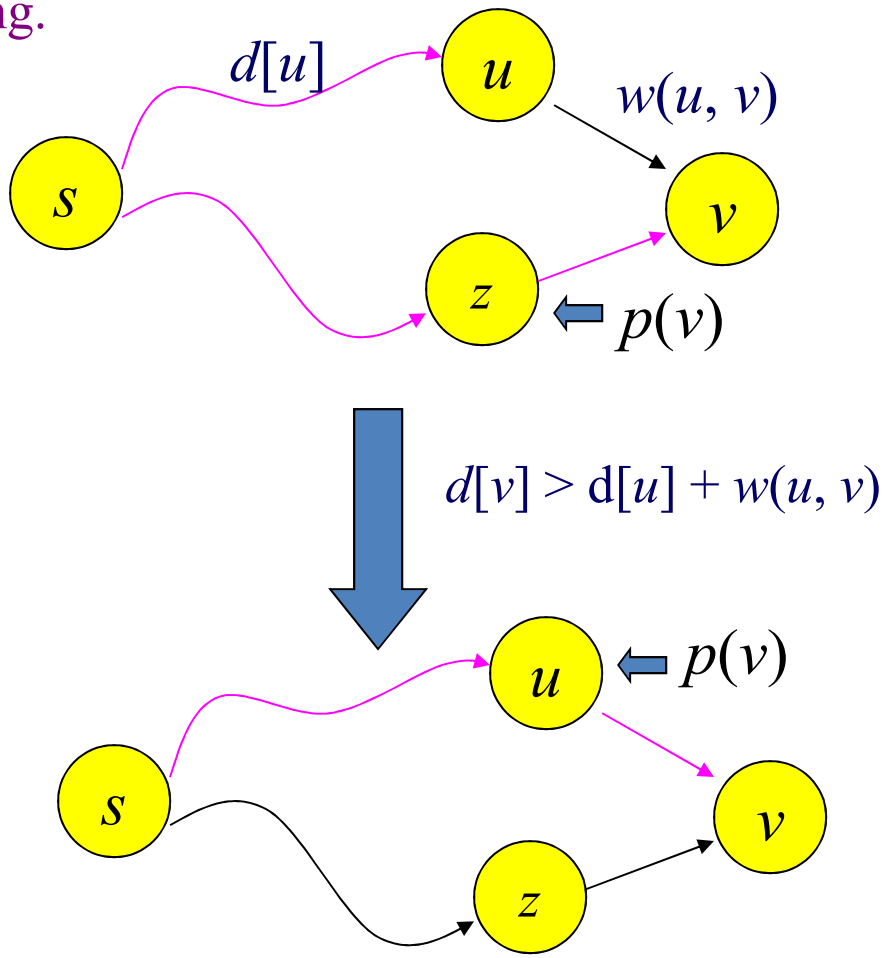
Khởi tạo (Initialization)

```
for v ∈ V(G)
      do d[v] ← ∞
         p[v] ← NULL
d[s] ← 0
```

Giảm cận trên

Đang xây dựng đường đi ngắn nhất từ s đến v:
Giả sử đường đi ngắn nhất hiện biết từ s đến v: $s \ldots z \rightarrow v$
Sử dụng cạnh (u, v) để kiểm tra xem đường đi đến v đã tìm được có thể làm ngắn hơn nhờ đi qua u hay không.

```
Relax(u, v)
    if (d[v] > d[u]  + w(u, v))
    {
        d[v] ← d[u] + w(u, v)
        p[v] ← u
    }
```



Các thuật toán tìm đđnn khác nhau ở số lần dùng các cạnh và trình tự duyệt chúng để thực hiện giảm cận

Giảm cận trên

```
Relax(u, v)
    if (d[v] > d[u]  + w(u, v))
    {
            d[v] ← d[u] + w(u, v)
            p[v] ← u
    }
```

Các thuật toán tìm đđnn khác nhau ở:

- Số lần mỗi cạnh được dùng để thực hiện giảm cận
- Trình tự duyệt cạnh để thực hiện giảm cận

Nhận xét chung

- Việc cài đặt các thuật toán được thể hiện nhờ ***thủ tục gán nhãn***:
  - Mỗi đỉnh v sẽ có nhãn gồm 2 thành phần ($d[v]$, $p[v]$). Nhãn sẽ biến đổi trong quá trình thực hiện thuật toán
- Nhận thấy rằng để tính khoảng cách từ s đến t, ở đây, ta phải tính khoảng cách từ s đến tất cả các đỉnh còn lại của đồ thị.
- Hiện nay vẫn chưa biết thuật toán nào cho phép tìm đđnn nhất giữa hai đỉnh làm việc thực sự hiệu quả hơn những thuật toán tìm đđnn từ một đỉnh đến tất cả các đỉnh còn lại.

Nội dung chi tiết

5.1. Bài toán đường đi ngắn nhất (ĐĐNN)

5.2. Tính chất của ĐĐNN, Giảm cận trên

5.3. Thuật toán Bellman-Ford

5.4. Thuật toán Dijkstra

5.5. Đường đi ngắn nhất trong đồ thị không có chu trình

5.6. Thuật toán Floyd-Warshall

Thuật toán Bellman-Ford

Richard Bellman
1920-1984

Lester R. Ford, Jr.
1927-2017

Thuật toán Bellman-Ford

- Thuật toán Bellman-Ford tìm đường đi ngắn nhất từ đỉnh *s* đến tất cả các đỉnh còn lại của đồ thị.
- Thuật toán làm việc trong trường hợp trọng số của các cung là tuỳ ý.
- Giả thiết rằng trong đồ thị không có chu trình âm.
- **Đầu vào:** *Đồ thị G=(V,E) với n đỉnh xác định bởi ma trận trọng số w[u,v], u,v ∈ V, đỉnh nguồn s ∈ V;*
- **Đầu ra:** *Với mỗi v ∈ V*

  $d[v] = \delta(s, v)$; Độ dài đường đi ngắn nhất từ *s* đến *v*

  p[v] - đỉnh đi trước v trong đđnn từ s đến v.

Thuật toán Bellman-Ford: Full version

```
Bellman-Ford(G, w, s)
// Bước 1: Khởi tạo mảng d và p là DDNN đi qua nhiều nhất 1 cạnh
1.   Initialize-Single-Source(G, s)
/* Bước 2: Tính DĐNN qua nhiều nhất i cạnh từ DDNN qua nhiều nhất (i-1)
   cạnh */
2.   for (k=1; k<=|V|-1;k++)
3.     for each edge (u, v) ∈ E
4.         Relax(u, v)
5.   for each edge (u, v) ∈ E
6.     if d[v] > d[u] + w(u, v)
7.         return False    //đồ thị có chu trình âm
8.   return True
```

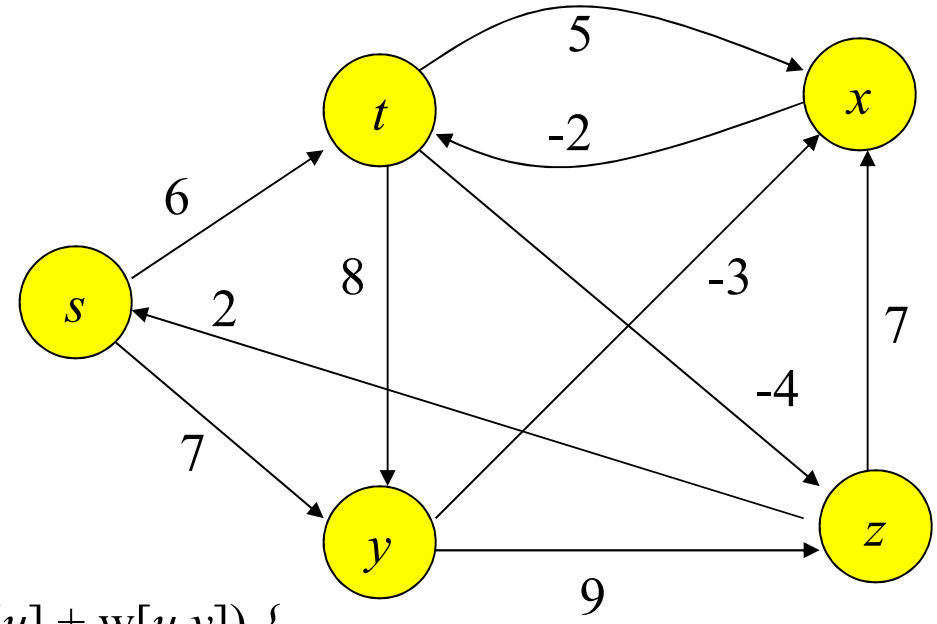
```
Relax(u, v)
 if d[v] > d[u] + w(u, v)
 {
    d[v] = d[u] + w[u,v] ;
    p[v] = u ;
 }
```

```
Initialize-Single-Source(G, s)
for v ∈ V\s
{
    d[v] = w[s,v];
    p[v] = s;
}
p[s]=s; d[s]=0;
```

Mô tả thuật toán

```
void  Bellman-Ford ( )
{
    for   v ∈ V\s  // Khởi tạo
    {
          d[v] = w[s,v];
          p[v] = s;
    }
    d[s]=0; p[s]=s;
    for  (k= 1; k<= |V| -1; k++)
          for   each (u, v) ∈ E
                RELAX(u, v);
}
```

	d	p
t	6	s
x	∞	s
y	7	s
z	∞	s
s	**0**	**s**



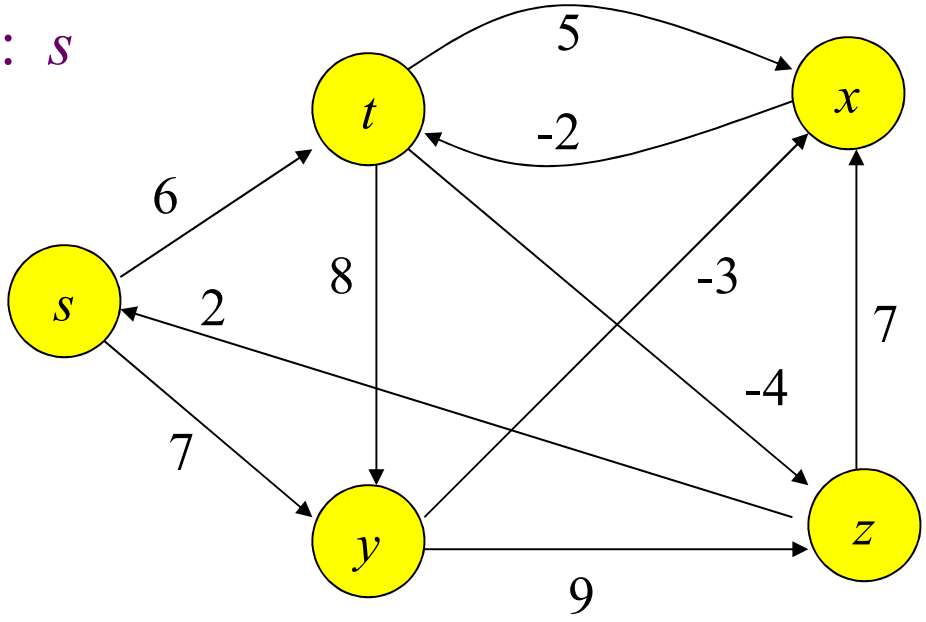

```
v != s
  if   (d[v] > d[u] + w[u,v]) {
         d[v] = d[u] + w[u,v] ;
         p[v] = u ;
       }
```

Độ phức tạp tính toán của thuật toán là $O(|V||E|)$.

Ví dụ

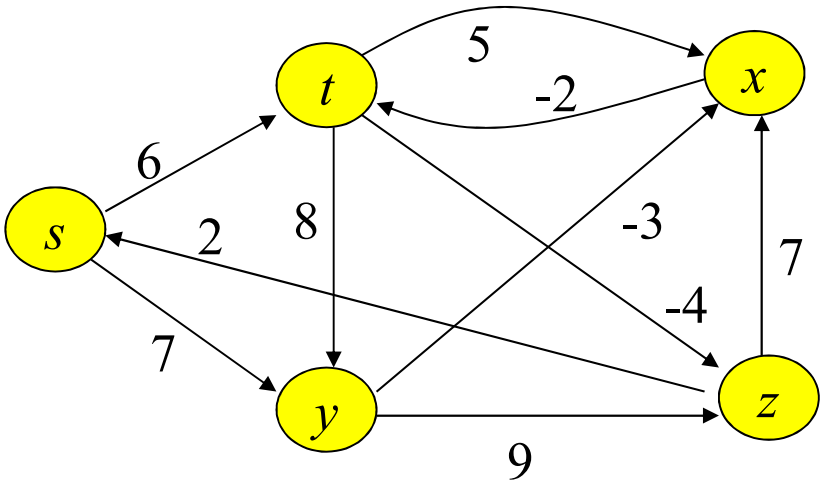
Áp dụng thuật toán Bellman-Ford tìm đường đi ngắn nhất từ *s* đến tất cả các đỉnh còn lại của đồ thị

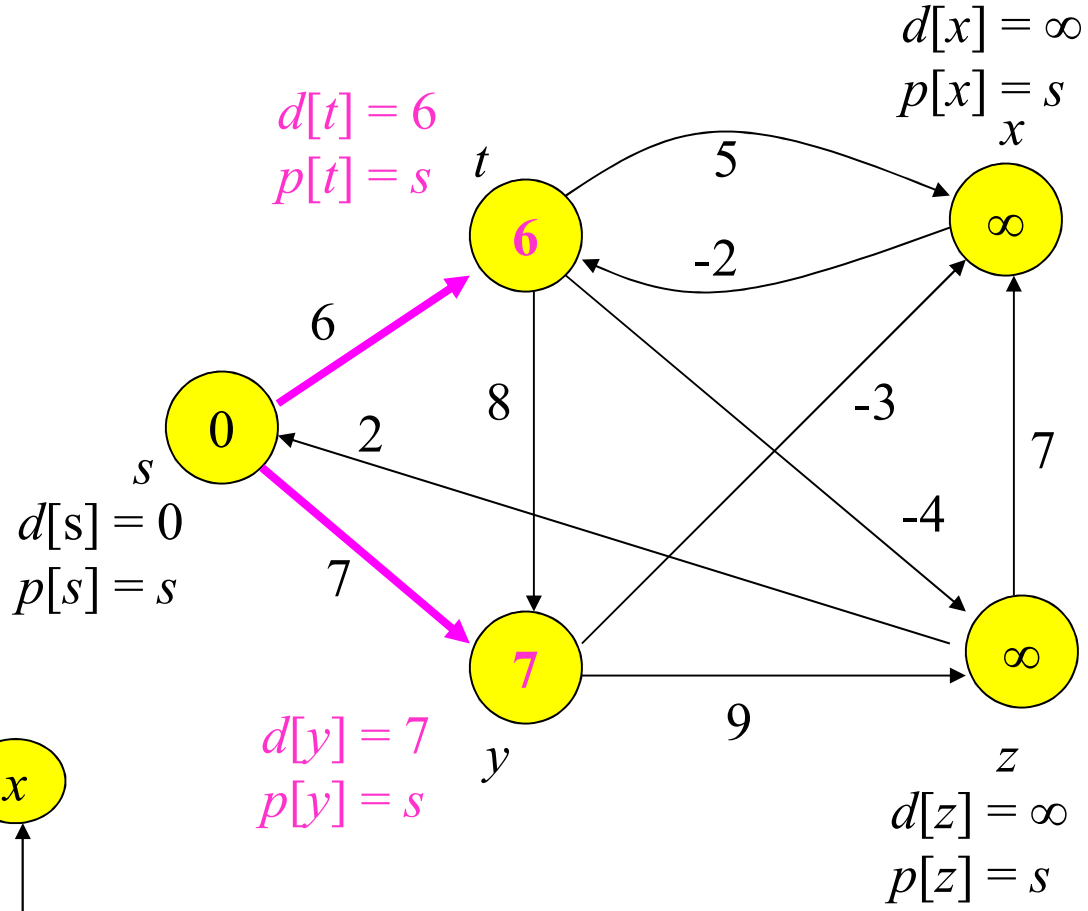
Source: *s*



Khởi tạo

```
void  Bellman_Ford ( )
{
    for   v ∈ V\s  // Khởi tạo
    {
          d[v] = w[s,v];
          p[v] = s;
    }
    d[s]=0; p[s]=s;
    for  (k= 1; k<= |V| -1; k++)
          for   each (u, v) ∈ E
                RELAX(u, v);
}
```





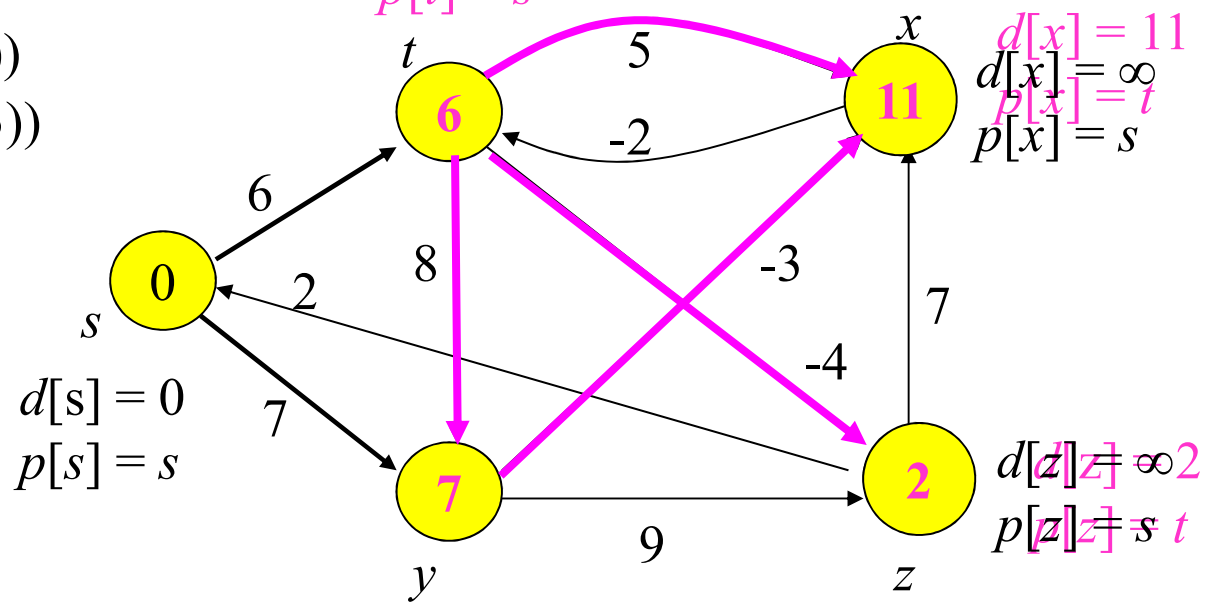
Lần 1: $k = 1$

```
for (k= 1; k<= |V| -1; k++)
    for each (u, v) ∈ E
        RELAX(u, v);
```

Trình tự duyệt cạnh để giảm cận:

(t, x), if (∞ > 6+5)
(t, y), if (7 > 6+8)
(t, z), if (∞ > 6+(-4))
(y, x), if (11 > 7+(-3))
(y, z),
(x, t),
(z, x),
(s, t),
(s, y)

```
RELAX(u, v);
if (d[v] > d[u] + w[u,v]) {
    d[v] = d[u] + w[u,v] ;
    p[v] = u ;
}
```



Lần 1: $k = 1$

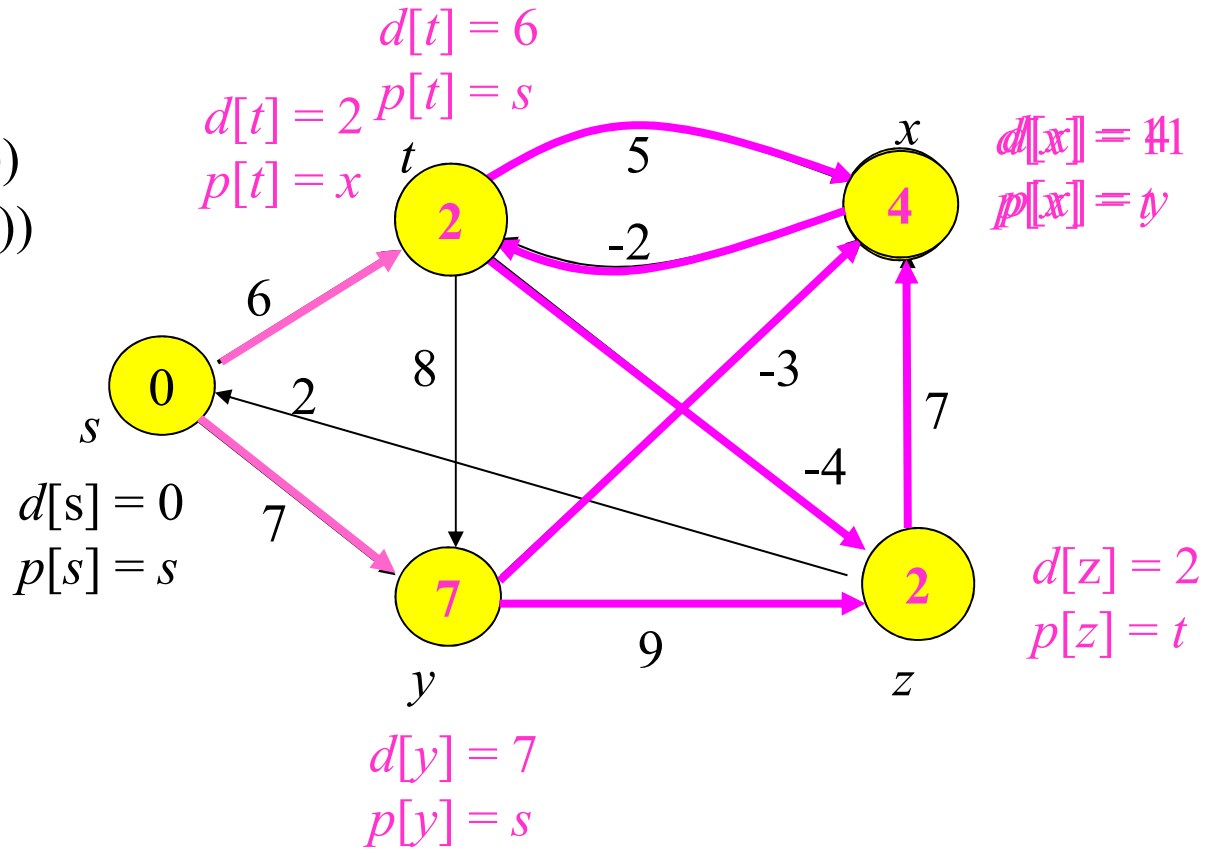
```
for (k= 1; k<= |V| -1; k++)
        for each (u, v) ∈ E
                RELAX(u, v);
```

RELAX(*u*, *v*);

```
if (d[v] > d[u] + w[u,v]) {
    d[v] = d[u] + w[u,v] ;
    p[v] = u ;
}
```

Trình tự duyệt cạnh để giảm cận:

(*t*, *x*), if (∞ > 6+5)
(*t*, *y*), if (7 > 6+8)
(*t*, *z*), if (∞ > 6+(-4))
(*y*, *x*), if (11 > 7+(-3))
(*y*, *z*), if (2 > 7+9)
(*x*, *t*), if (6 > 4-2)
(*z*, *x*), if (4 > 2+7)
(*s*, *t*), if (2 > 0+6)
(*s*, *y*) if (7 > 0+7)



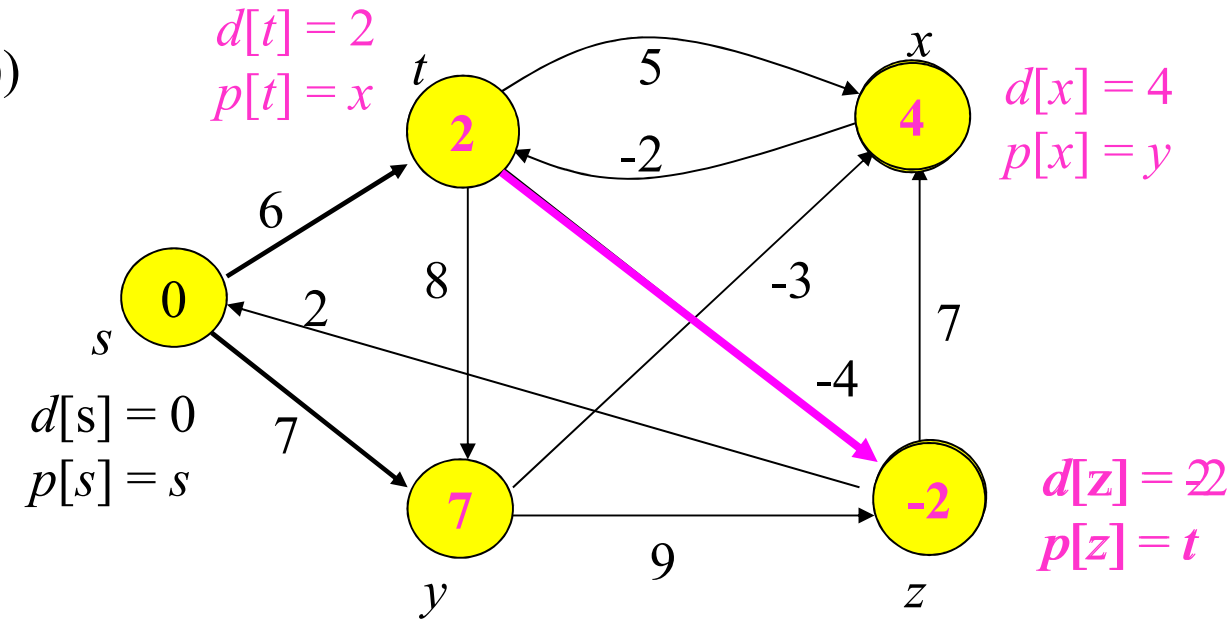
Lần 2: $k = 2$

```
for (k= 1; k<= |V| -1; k++)
        for each (u, v) ∈ E
                RELAX(u, v);
```

```
RELAX(u, v);
if (d[v] > d[u] + w[u,v]) {
    d[v] = d[u] + w[u,v] ;
    p[v] = u ;
}
```

Trình tự duyệt cạnh để giảm cận:

(*t*, *x*),
(*t*, *y*),
(*t*, *z*), if (2 > 2+(-4))
(*y*, *x*),
(*y*, *z*),
(*x*, *t*),
(*z*, *x*),
(*s*, *t*),
(*s*, *y*)



Lần 3: $k = 3$

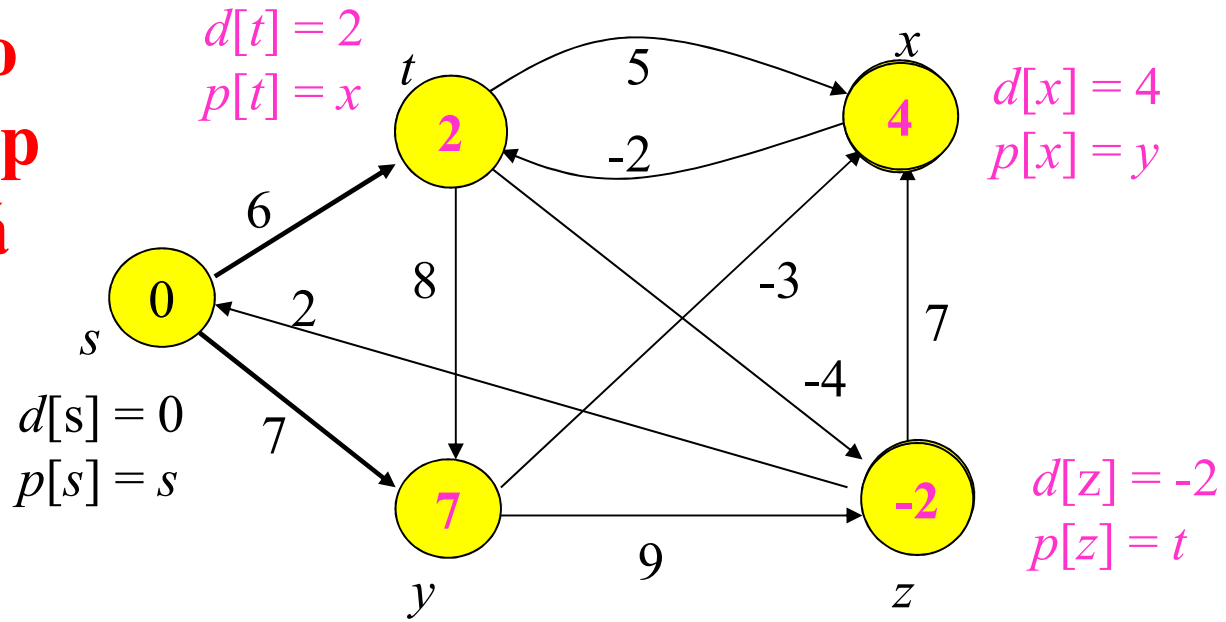
```
for (k= 1; k<= |V| -1; k++)
        for each (u, v) ∈ E
                RELAX(u, v);
```

```
RELAX(u, v);
if (d[v] > d[u] + w[u,v]) {
    d[v] = d[u] + w[u,v] ;
    p[v] = u ;
}
```

Trình tự duyệt cạnh để giảm cận:

(t, x),
(t, y),
(t, z),
(y, x),
(y, z),
(x, t),
(z, x),
(s, t),
(s, y)

Không d[u] nào được cập nhật giá trị



Nhận xét

- Thuật toán chấm dứt sau 3 vòng lặp
- Giá trị thu được ở mỗi vòng lặp và số vòng lặp phải thực hiện (tốc độ thuật toán) phụ thuộc vào thứ tự duyệt cạnh
- Giá trị *d* của một đỉnh có thể được cập nhật nhiều hơn 1 lần trong cùng 1 vòng lặp

Cải tiến

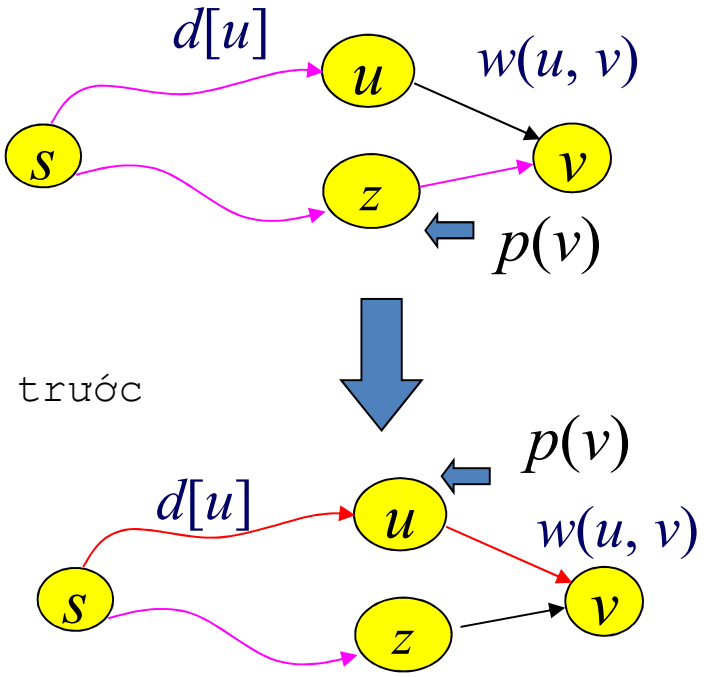
```
Bellman-Ford(G, w, s)
1.    Initialize-Single-Source(G, s)
2.    for (k=1; k<=|V|-1;k++)
3.       for each edge (u, v) ∈ E
4.           Relax(u, v)
```

Cải tiến:

- Không cần duyệt các cạnh (u, v) nếu $d[u]$ không được thay đổi giá trị ở bước lặp trước.
- Nếu không có một giá trị d[u] nào được cập nhật ở bước lặp k → thuật toán kết thúc

Cải tiến Bellman-Ford:

```
Bellman-Ford(G, w, s)
1.    Initialize-Single-Source(G, s)
2.    for (k=1; k<=|V|-1;k++) {
3.       for each node u ∈ V
4.       {
5.          if d[u] thay đổi giá trị ở bước lặp trước
6.             for each edge (u, v) ∈ E
7.                Relax(u, v)
8.       }
9.       if không có d[u] nào thay đổi giá trị
         ở bước lặp k, stop
1.    }
```



Nội dung chi tiết

5.1. Bài toán đường đi ngắn nhất (ĐĐNN)

5.2. Tính chất của ĐĐNN, Giảm cận trên

5.3. Thuật toán Bellman-Ford

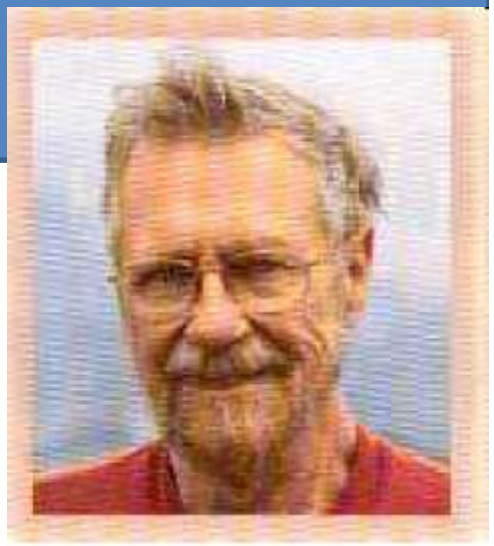
5.4. Thuật toán Dijkstra

5.5. Đường đi ngắn nhất trong đồ thị không có chu trình

5.6. Thuật toán Floyd-Warshall

Thuật toán Dijkstra

- Trong trường hợp trọng số trên các cung là không âm, thuật toán do Dijkstra đề nghị hiệu quả hơn rất nhiều so với thuật toán Bellman-Ford.
- Thuật toán được xây dựng dựa trên thủ tục gán nhãn. Thoạt tiên nhãn của các đỉnh là tạm thời. Ở mỗi một bước lặp có một nhãn tạm thời trở thành nhãn cố định. Nếu nhãn của một đỉnh u trở thành cố định thì $d[u]$ sẽ cho ta độ dài của đđnn từ đỉnh s đến u. Thuật toán kết thúc khi nhãn của tất cả các đỉnh trở thành cố định.



Edsger W.Dijkstra

(1930-2002)

Thuật toán Dijkstra

- **Đầu vào:** *Đồ thị có hướng $G=(V,E)$ với n đỉnh,*

  $s \in V$ là đỉnh xuất phát,

  $w[u,v]$, $u,v \in V$ - ma trận trọng số;

  Giả thiết: $\boldsymbol{w[u,v] \geq 0, \quad \forall\, u, v \in V.}$

- **Đầu ra:** *Với mỗi $v \in V$*

  $d[v] = \delta(s, v)$;

  $p[v]$ - *đỉnh đi trước v trong đđnn từ s đến v.*

Thuật toán Dijkstra

```
void Dijkstra ( )
{
    for   v ∈ V    // Khởi tạo
    {
        d[v] = w[s,v] ;
        p[v]=s;
    }
    d[s] = 0;   S = {s};                    // S: tập các đỉnh có nhãn cố định
    T = V \ {s};                            // T: tập các đỉnh có nhãn tạm thời
    while  (T ≠ ∅)                          // Bước lặp
    {
        Tìm đỉnh  u ∈ T thoả mãn  d[u] = min{ d[z] :  z  ∈  T};
        T =  T \ {u};  S= S  ∪ {u};      //Cố định nhãn của đỉnh u
        for   v ∈ T                         //Gãn nhãn lại cho các đỉnh trong T
            if   (d[v] > d[u] + w[u,v])
            {
                d[v] = d[u] + w[u,v] ;
                p[v] = u ;
            }
    }
}
```

Tập S: Chỉ cần cho chứng minh định lý

Thuật toán Dijkstra

- **Chú ý:** Nếu chỉ cần tìm đường đi ngắn nhất từ s đến t thì có thể chấm dứt thuật toán khi đỉnh t trở thành có nhãn cố định.

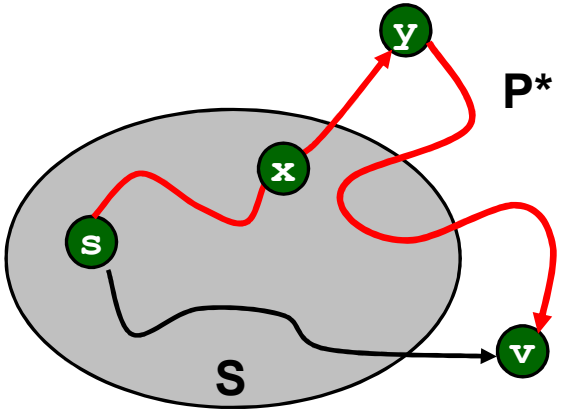
- **Định lý 1.** *Thuật toán Dijkstra tìm được đường đi ngắn nhất từ đỉnh s đến tất cả các đỉnh còn lại trên đồ thị sau thời gian* $O(n^2)$.

Chứng minh tính đúng đắn của Thuật toán Dijkstra

Ta sẽ CM với mỗi v ∈ S, d(v) = δ(*s, v*).

– Qui nạp theo |S|.

– Cơ sở qui nạp: Với |S| = 1, rõ ràng là đúng.



– Chuyển qui nạp:

• giả sử thuật toán Dijkstra bổ sung v vào S

→ d(v) = min{ d[z] : z ∉ S};

• nếu d(v) không là độ dài đđnn từ s đến v, thì gọi P* là đđnn từ s đến v

→ P* phải sử dụng cạnh ra khỏi S, chẳng hạn (x, y)

khi đó d(v) > δ(s, v) giả thiết

= δ(s, x) + w(x, y) + δ(y, v) tính chất 3:Mọi đoạn đường con của đường đi ngắn nhất đều là đường đi ngắn nhất

≥ δ(s, x) + w(x, y) δ(y, v) là không âm

= d(x) + w(x, y) giả thiết quy nạp

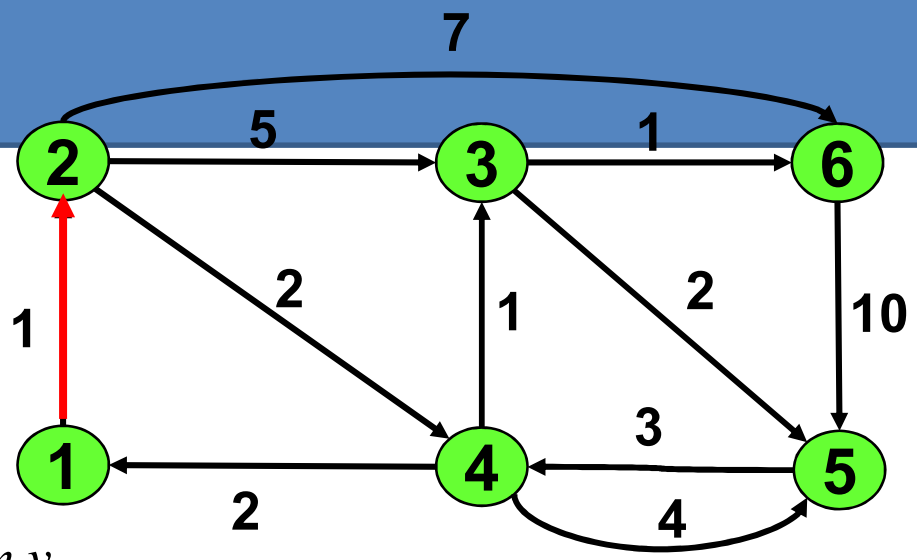
≥ d(y) theo thuật toán

→ $d(v) > d(y)$

vì thế thuật toán Dijkstra phải chọn y thay vì chọn v ?!

Ví dụ

Tìm đường đi ngắn nhất từ đỉnh 1 đến tất cả các đỉnh còn lại



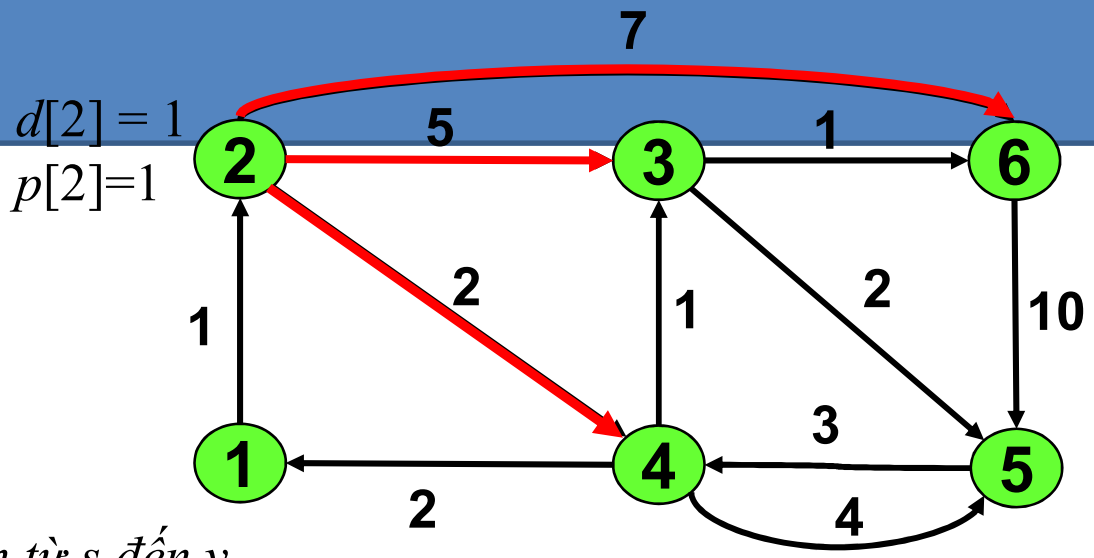
$d[v] = \delta(s, v)$;

$p[v]$ - *đỉnh đi trước v trong đđnn từ s đến v.*

	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	[0,1]	[1,1]	[∞,1]	[∞,1]	[∞,1]	[∞,1]
1						
2						
3						
4						
5						

Ví dụ

Tìm đường đi ngắn nhất từ đỉnh 1 đến tất cả các đỉnh còn lại



$d[v] = \delta(s, v)$;

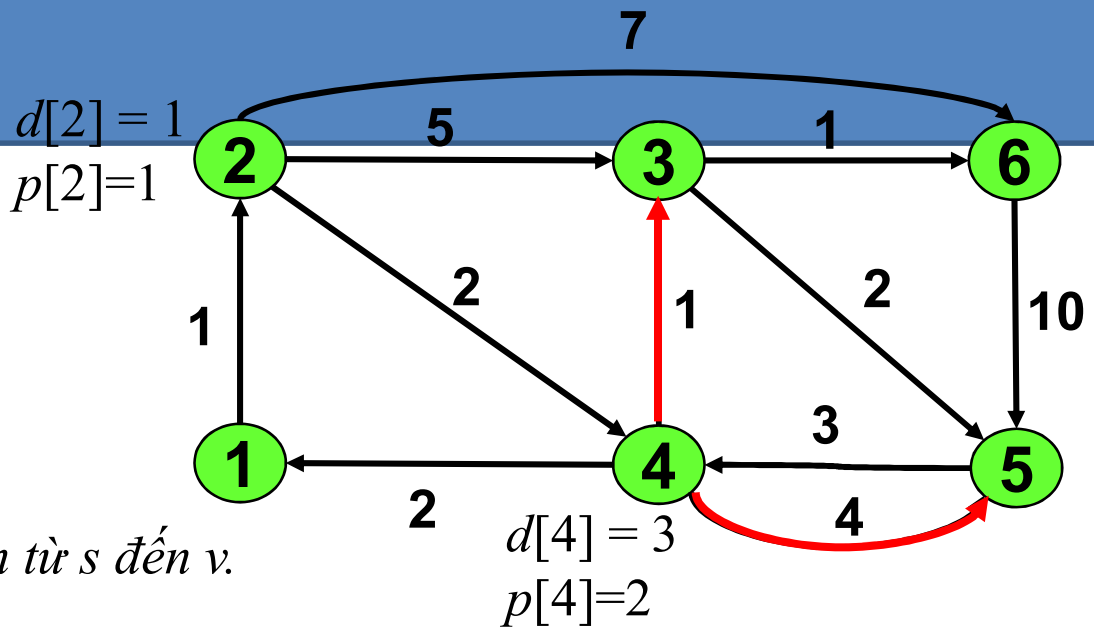
$p[v]$ - *đỉnh đi trước v trong đđnn từ s đến v.*

Cập nhật lại nhãn cho các đỉnh còn lại

	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	[0,1]	**[1,1]***	[∞,1]	[∞,1]	[∞,1]	[∞,1]
1	-	-	[6, 2]	[3, 2]	[∞,1]	[8,2]
2		Xét đỉnh 3: if (∞ > 1+5) → cập nhật lại nhãn cho đỉnh 3				
3		Xét đỉnh 4: if (∞ > 1+2) → cập nhật lại nhãn cho đỉnh 4				
4		Xét đỉnh 5: if (∞ > 1+ ∞)				
5		Xét đỉnh 6: if (∞ > 1+ 7) → cập nhật lại nhãn cho đỉnh 6				

Ví dụ

Tìm đường đi ngắn nhất từ đỉnh 1 đến tất cả các đỉnh còn lại



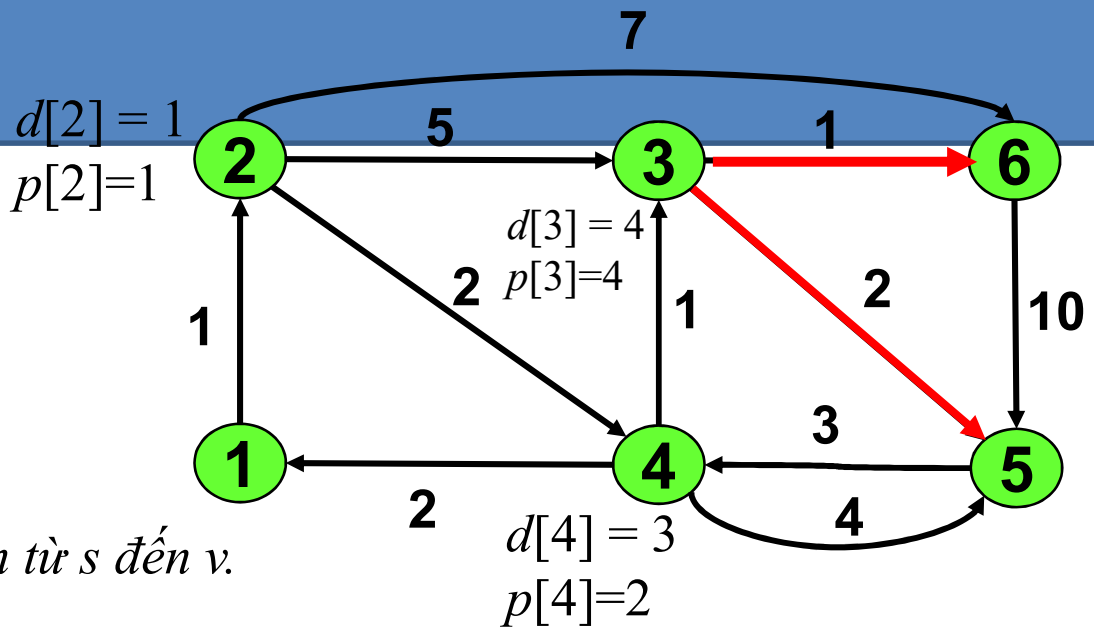
$d[v] = \delta(s, v)$;

$p[v]$ - *đỉnh đi trước v trong đđnn từ s đến v.*

	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	[0,1]	**[1,1]***	[∞,1]	[∞,1]	[∞,1]	[∞,1]
1	-	-	[6, 2]	**[3, 2]***	[∞,1]	[8,2]
2	-	-	[4, 4]	-	[7, 4]	[8,2]
3		Xét đỉnh 3: if (6 > 3+1) → cập nhật lại nhãn cho đỉnh 3				
4		Xét đỉnh 5: if (∞ > 3+ 4) → cập nhật lại nhãn cho đỉnh 5				
5		Xét đỉnh 6: if (8 > 3+ ∞)				

Ví dụ

Tìm đường đi ngắn nhất từ đỉnh 1 đến tất cả các đỉnh còn lại



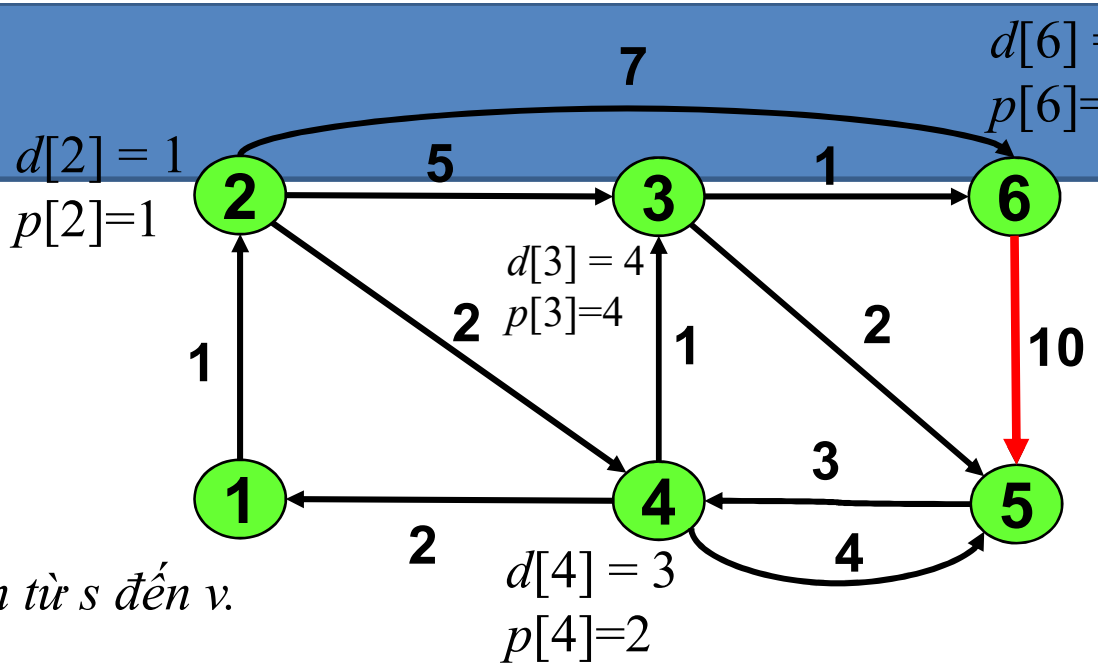
$d[v] = \delta(s, v)$;

$p[v]$ - *đỉnh đi trước v trong đđnn từ s đến v.*

	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	[0,1]	**[1,1]***	[∞,1]	[∞,1]	[∞,1]	[∞,1]
1	-	-	[6, 2]	**[3, 2]***	[∞,1]	[8,2]
2	-	-	**[4, 4]***	-	[7, 4]	[8,2]
3	-	-	-	-	[6, 3]	[5, 3]
4		Xét đỉnh 5: if (7 > 4+ 2)→ cập nhật lại nhãn cho đỉnh 5				
5		Xét đỉnh 6: if (8 > 4+ 1) → cập nhật lại nhãn đỉnh 6				

Ví dụ

Tìm đường đi ngắn nhất từ đỉnh 1 đến tất cả các đỉnh còn lại



$d[v] = \delta(s, v)$;

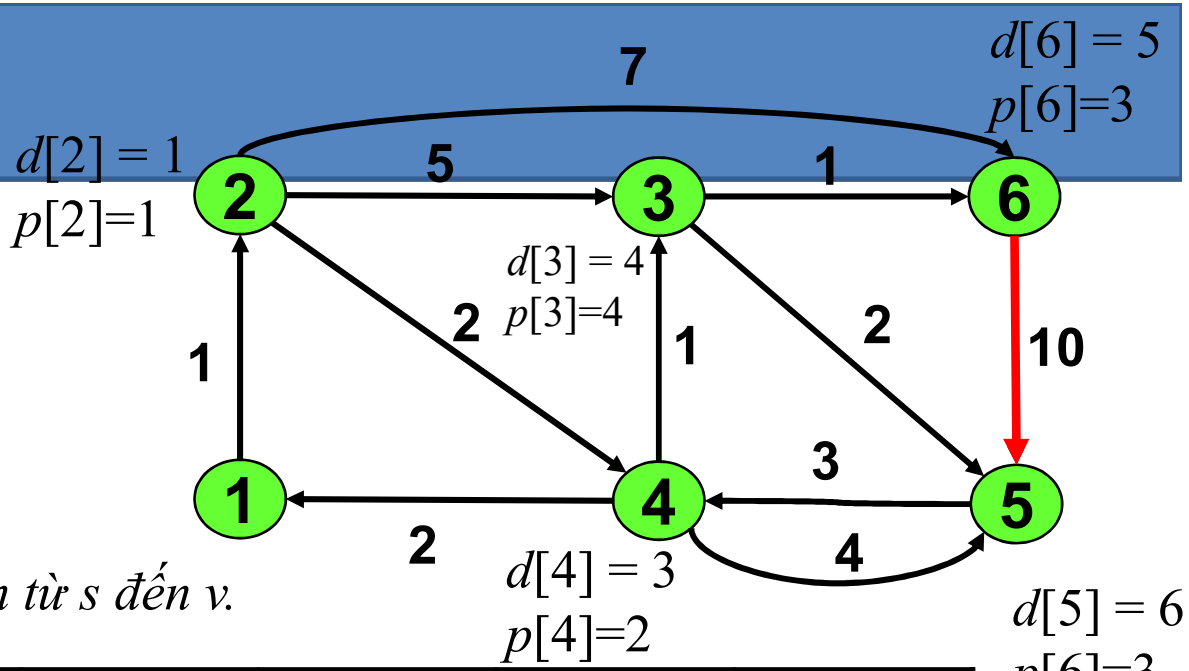
$p[v]$ - *đỉnh đi trước v trong đđnn từ s đến v.*

	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	[0,1]	**[1,1]***	[∞,1]	[∞,1]	[∞,1]	[∞,1]
1	-	-	[6, 2]	**[3, 2]***	[∞,1]	[8,2]
2	-	-	**[4, 4]***	-	[7, 4]	[8,2]
3	-	-	-	-	[6, 3]	**[5, 3]***
4	-	-	-	-	[6, 3]	-
5						

Xét đỉnh 5: if (6 > 5+ 10)

Ví dụ

Tìm đường đi ngắn nhất từ đỉnh 1 đến tất cả các đỉnh còn lại



$d[v] = \delta(s, v)$;

$p[v]$ - *đỉnh đi trước v trong đđnn từ s đến v.*

	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	[0,1]	**[1,1]***	[∞,1]	[∞,1]	[∞,1]	[∞,1]
1	-	-	[6, 2]	**[3, 2]***	[∞,1]	[8,2]
2	-	-	**[4, 4]***	-	[7, 4]	[8,2]
3	-	-	-	-	[6, 3]	**[5, 3]***
4	-	-	-	-	**[6, 3]***	-
5	-	-	-	-	-	-

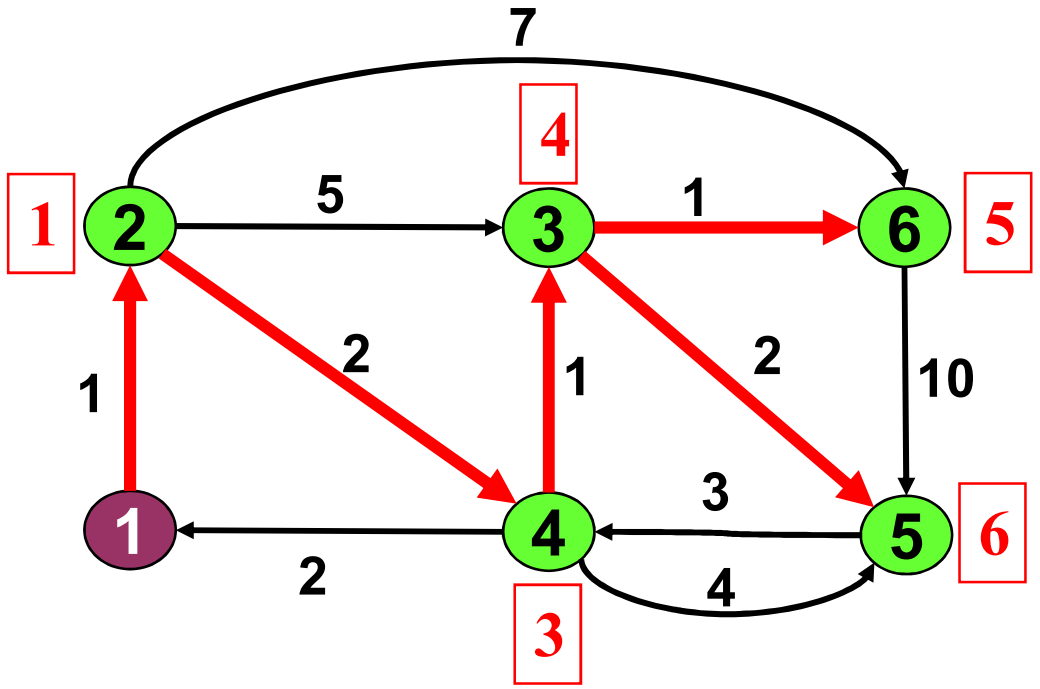
Kết luận: đường đi ngắn nhất từ đỉnh 1 đến các đỉnh còn lại của đồ thị ????

Cây đường đi ngắn nhất

- Tập cạnh $\{(p(v), v): v \in V \setminus \{s\}\}$ tạo thành cây có gốc tại đỉnh nguồn s được gọi là cây đđnn xuất phát từ đỉnh s.

• Các cạnh màu đỏ tạo thành cây đđnn xuất phát từ đỉnh 1

• Số màu đỏ viết bên cạnh mỗi đỉnh là độ dài đường đi ngắn nhất từ 1 đến nó.



Nội dung chi tiết

5.1. Bài toán đường đi ngắn nhất (ĐĐNN)

5.2. Tính chất của ĐĐNN, Giảm cận trên

5.3. Thuật toán Bellman-Ford → **Trọng số trên cạnh: số thực tùy ý**
Độ phức tạp: O(|V||E|)

Đường đi ngắn nhất trong đồ thị có chu trình
Chu trình không âm

5.4. Thuật toán Dijkstra → **Trọng số trên cạnh >=0**
Độ phức tạp: O($|V|^2$)

5.5. Đường đi ngắn nhất trong đồ thị không có chu trình

Trọng số trên cạnh: số thực tùy ý **Độ phức tạp: O(|E|)**

5.6. Thuật toán Floyd-Warshall

Nội dung chi tiết

5.1. Bài toán đường đi ngắn nhất (ĐĐNN)

5.2. Tính chất của ĐĐNN, Giảm cận trên

5.3. Thuật toán Bellman-Ford

5.4. Thuật toán Dijkstra

5.5. Đường đi ngắn nhất trong đồ thị không có chu trình

5.6. Thuật toán Floyd-Warshall

Shortest Paths In Directed Acyclic Graphs

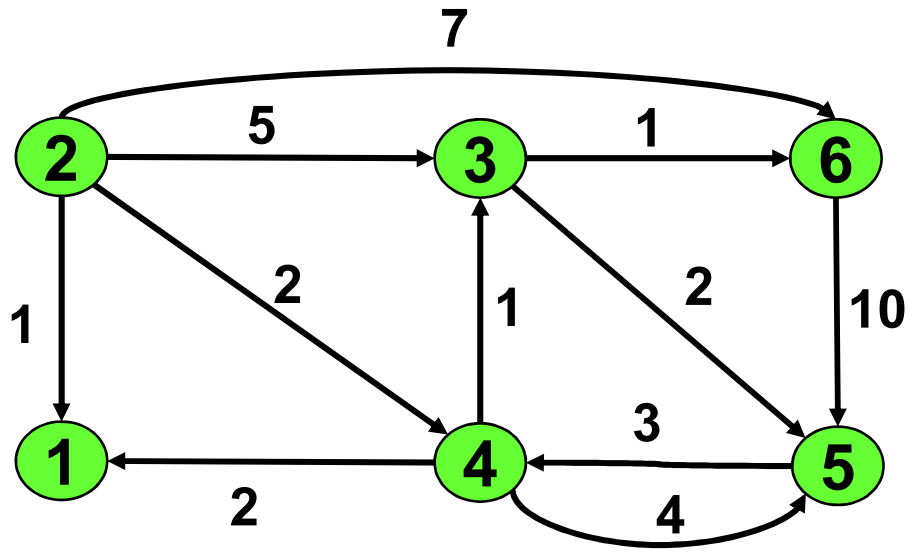
Đường đi trong đồ thị không có chu trình

Một trường hợp riêng của bài toán đường đi ngắn nhất giải được nhờ thuật toán với độ phức tạp tính toán $O(|E|)$, đó là bài toán trên đồ thị không có chu trình (còn trọng số trên các cung có thể là các số thực tuỳ ý). Kết quả sau đây là cơ sở để xây dựng thuật toán nói trên:

- **Định lý 2.** *Giả sử G là đồ thị không có chu trình. Khi đó các đỉnh của nó có thể đánh số sao cho mỗi cung của đồ thị chỉ hướng từ đỉnh có chỉ số nhỏ hơn đến đỉnh có chỉ số lớn hơn, nghĩa là mỗi cung của nó có thể biểu diễn dưới dạng* $(v[i], v[j])$, *trong đó* $i < j$.

Thuật toán đánh số đỉnh

- Trước hết nhận thấy rằng: *Trong đồ thị không có chu trình bao giờ cũng tìm được đỉnh có bán bậc vào bằng 0.*



Thuật toán đánh số đỉnh

- Trước hết nhận thấy rằng: *Trong đồ thị không có chu trình bao giờ cũng tìm được đỉnh có bán bậc vào bằng* 0.

Thực vậy, bắt đầu từ đỉnh v_1 nếu có cung đi vào nó từ v_2 thì ta lại chuyển sang xét đỉnh v_2. Nếu có cung từ v_3 đi vào v_2, thì ta lại chuyển sang xét v_3, ... Do đồ thị là không có chu trình nên sau một số hữu hạn lần chuyển như vậy ta phải đi đến đỉnh không có cung đi vào.

- **Thuật toán đánh số đỉnh:**

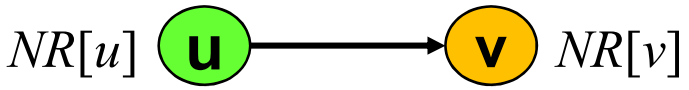
Thoạt tiên, tìm các đỉnh có bán bậc vào bằng 0. Ta đánh số những đỉnh này theo một thứ tự tuỳ ý bắt đầu từ 1.

Tiếp theo, loại bỏ khỏi đồ thị những đỉnh đã được đánh số cùng các cung đi ra khỏi chúng, ta thu được đồ thị mới cũng không có chu trình, và thủ tục được lặp lại với đồ thị mới này.

Quá trình đó sẽ được tiếp tục cho đến khi tất cả các đỉnh của đồ thị được đánh số.

Thuật toán đánh số đỉnh

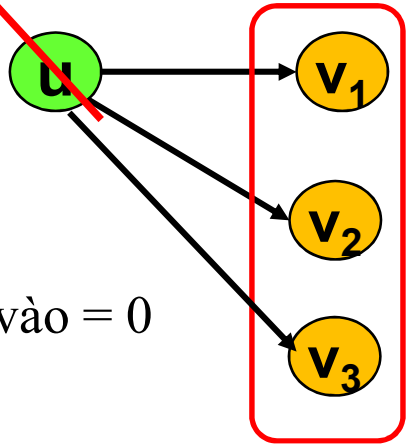
- **Đầu vào:** *Đồ thị có hướng G=(V,E) với n đỉnh không chứa chu trình được cho bởi danh sách kề Ke(v), v ∈ V.*
- **Đầu ra:** *Với mỗi đỉnh v ∈ V chỉ số NR [v] thoả mãn: Với mọi cung (u, v) của đồ thị ta đều có NR[u] < NR[v].*



Thuật toán đánh số đỉnh

```
void  Numbering ( )
{
    for   v ∈ V     Vao[v] := 0;
    for   u ∈ V          // Tính Vao[v] = bán bậc vào của v
        for   v ∈ Ke(u)   Vao[v] = Vao[v] + 1 ;

    QUEUE = ∅ ;
    for   v ∈ V
            if   (Vao[v] == 0) QUEUE = QUEUE ∪ { v } ;
    num = 0;
    while  (QUEUE ≠ ∅ )
    {
            u ⇐ QUEUE ; num = num + 1 ;   NR[u] = num ;
            for   v ∈ Ke(u)
            {
                    Vao[v] = Vao[v] - 1 ;
                    if   (Vao[v] == 0) QUEUE = QUEUE ∪ { v } ;
            }
    }
}
```

Tính bán bậc vào cho mỗi đỉnh của đồ thị



QUEUE: chứa các đỉnh có bán bậc vào = 0

Bổ sung vào QUEUE các đỉnh có bán bậc vào = 0

Đánh số đỉnh *u*

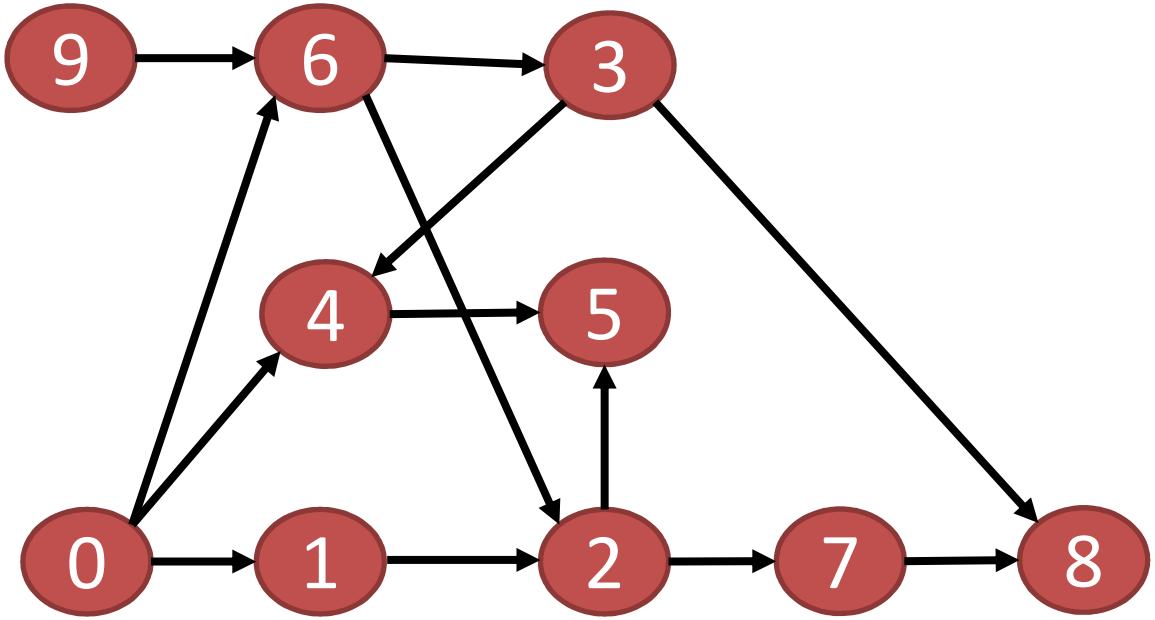
loại bỏ khỏi đồ thị đỉnh *u* đã được đánh số cùng các cung đi ra khỏi *u*

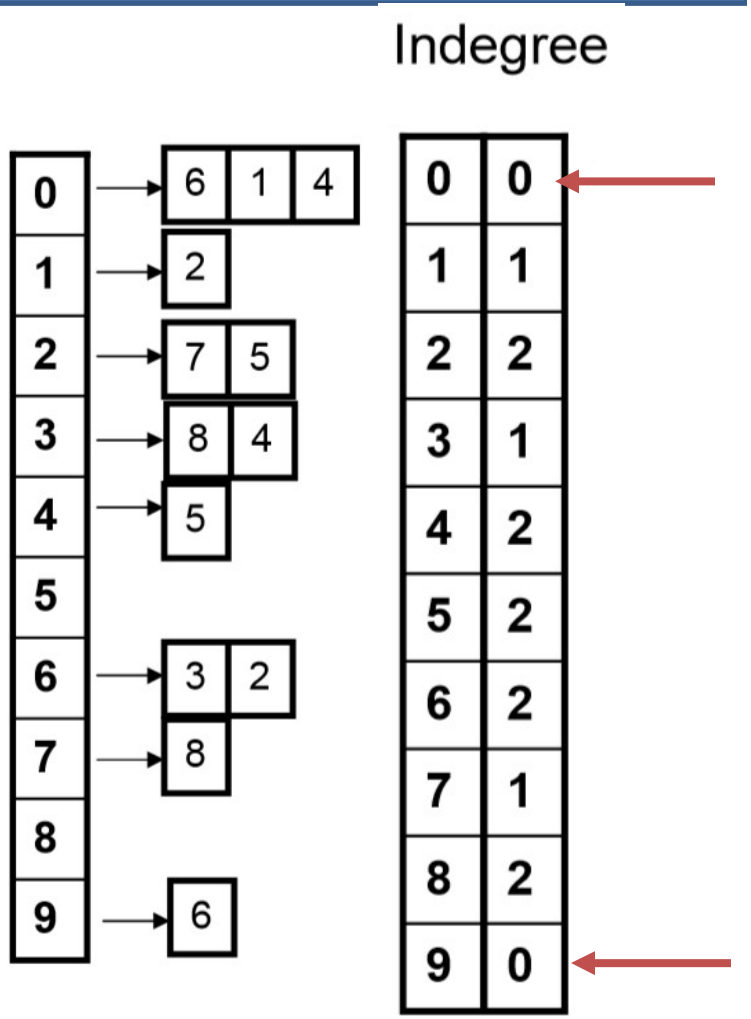
Thuật toán đánh số đỉnh

- Rõ ràng trong bước khởi tạo ta phải duyệt qua tất cả các cung của đồ thị khi tính bán bậc vào của các đỉnh, vì vậy ở đó ta tốn cỡ $O(|E|)$ phép toán. Tiếp theo, mỗi lần đánh số một đỉnh, để thực hiện việc loại bỏ đỉnh đã đánh số cùng với các cung đi ra khỏi nó, chúng ta lại duyệt qua tất cả các cung này. Suy ra để đánh số tất cả các đỉnh của đồ thị chúng ta sẽ phải duyệt qua tất cả các cung của đồ thị một lần nữa.
- Vậy độ phức tạp của thuật toán là $O(|E|)$.

```
void  Numbering ( )
{
    for   v ∈ V     Vao[v] := 0;
    for   u ∈ V         // Tính Vao[v] = bán bậc vào của v
        for   v ∈ Ke(u)   Vao[v] = Vao[v] + 1 ;

   QUEUE = ∅ ;
    for   v ∈ V
          if   (Vao[v] == 0) QUEUE = QUEUE ∪ { v } ;
    num = 0;
    while  (QUEUE ≠ ∅ )
    {
          u ⇐ QUEUE ;  num = num + 1 ;   NR[u] = num ;
          for   v ∈ Ke(u)
          {
                Vao[v] = Vao[v] - 1 ;
                if   (Vao[v] == 0) QUEUE = QUEUE ∪ { v } ;
          }
    }
}
```

Sắp xếp topo: Ví dụ

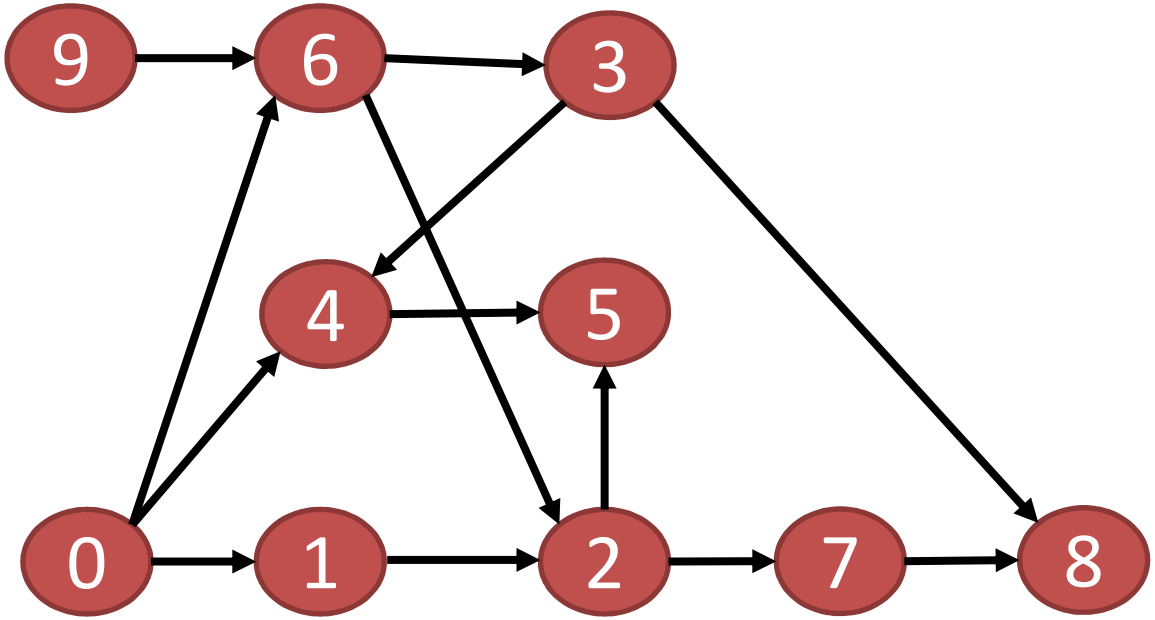


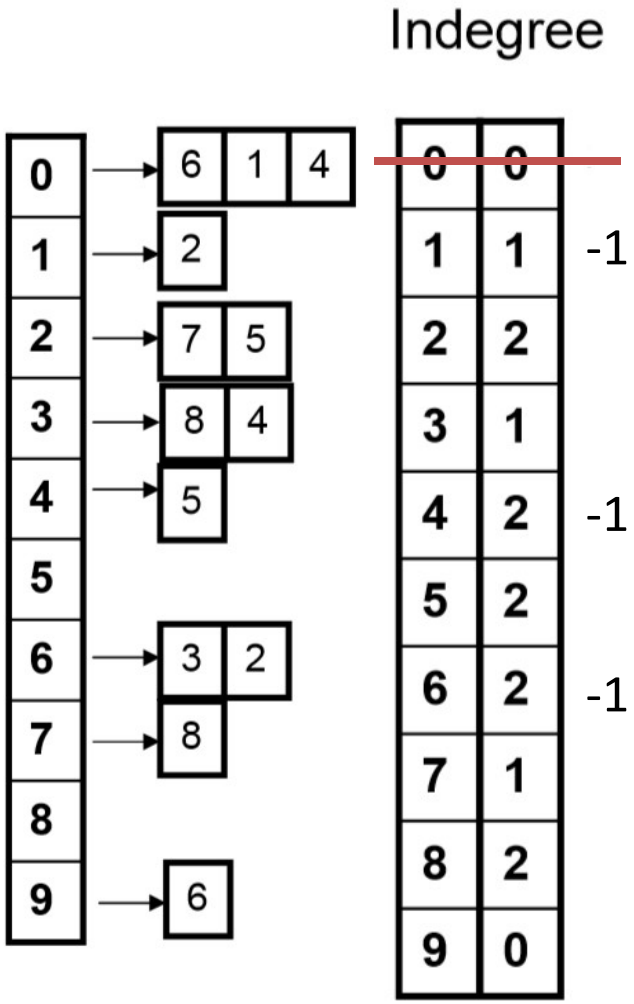


Q={0, 9}

Output:

Sắp xếp topo: Ví dụ



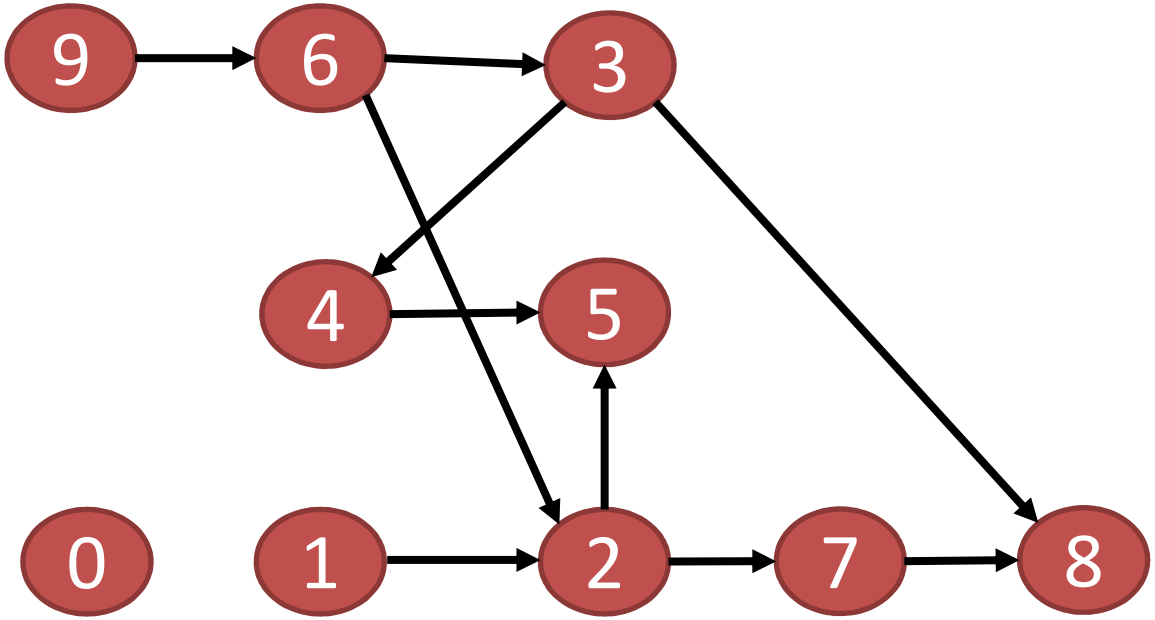


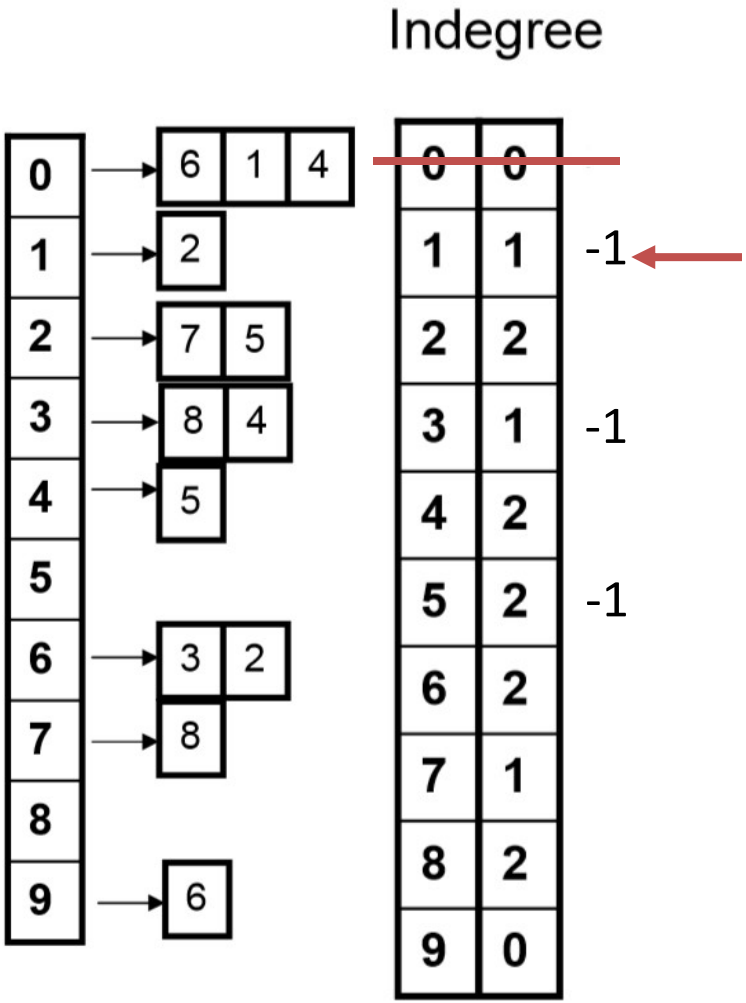
1

Dequeue 0; Q={9}

Output: 0

Sắp xếp topo: Ví dụ



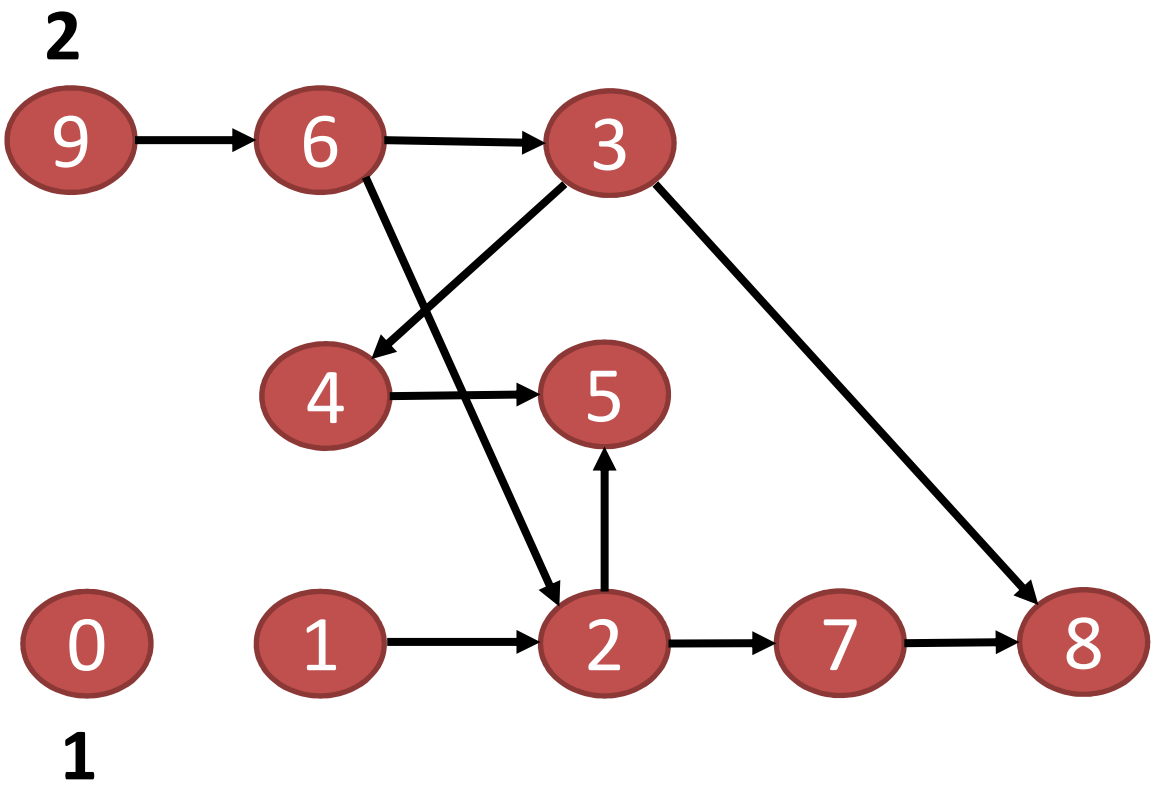


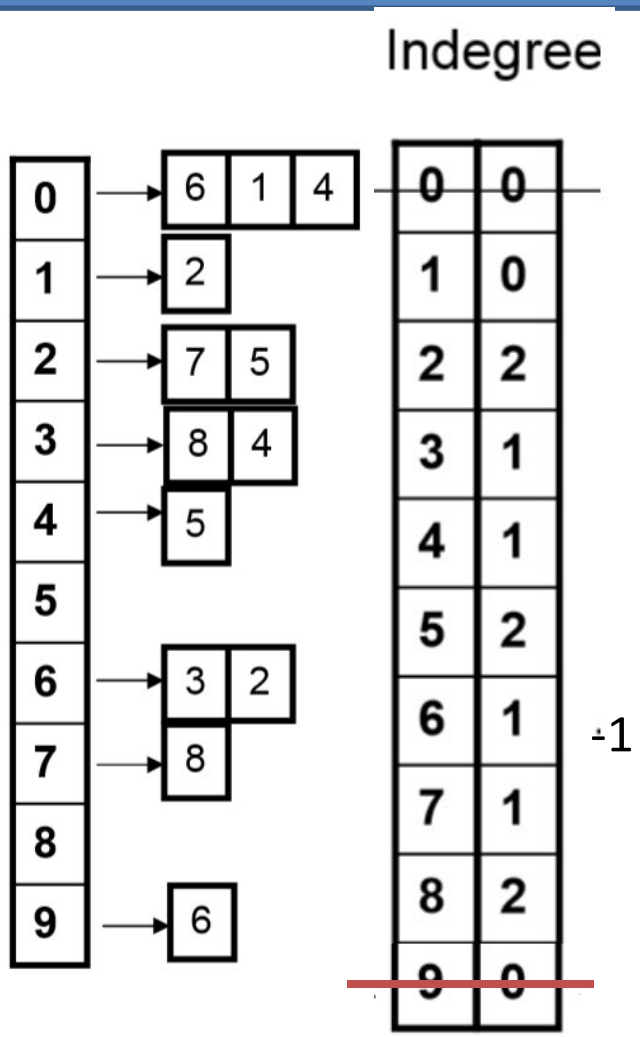
1

Dequeue 0; **Q={9} 1}**

Output: 0

Sắp xếp topo: Ví dụ

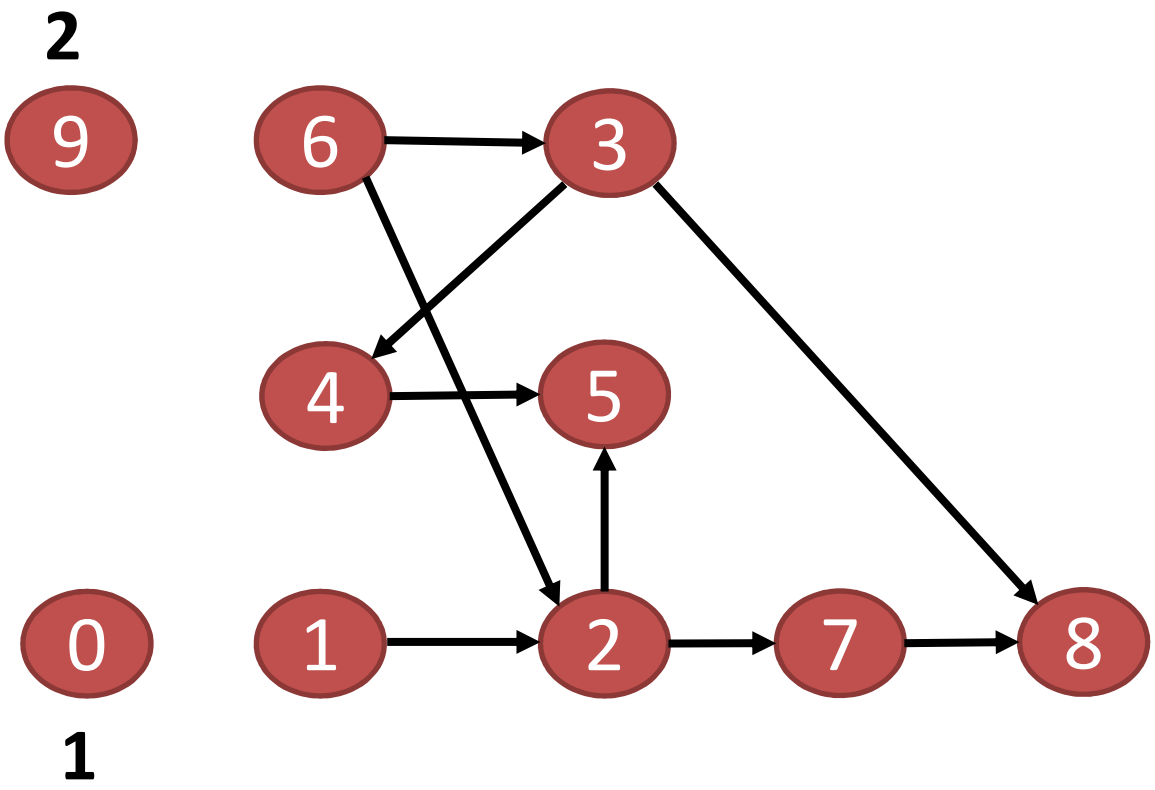


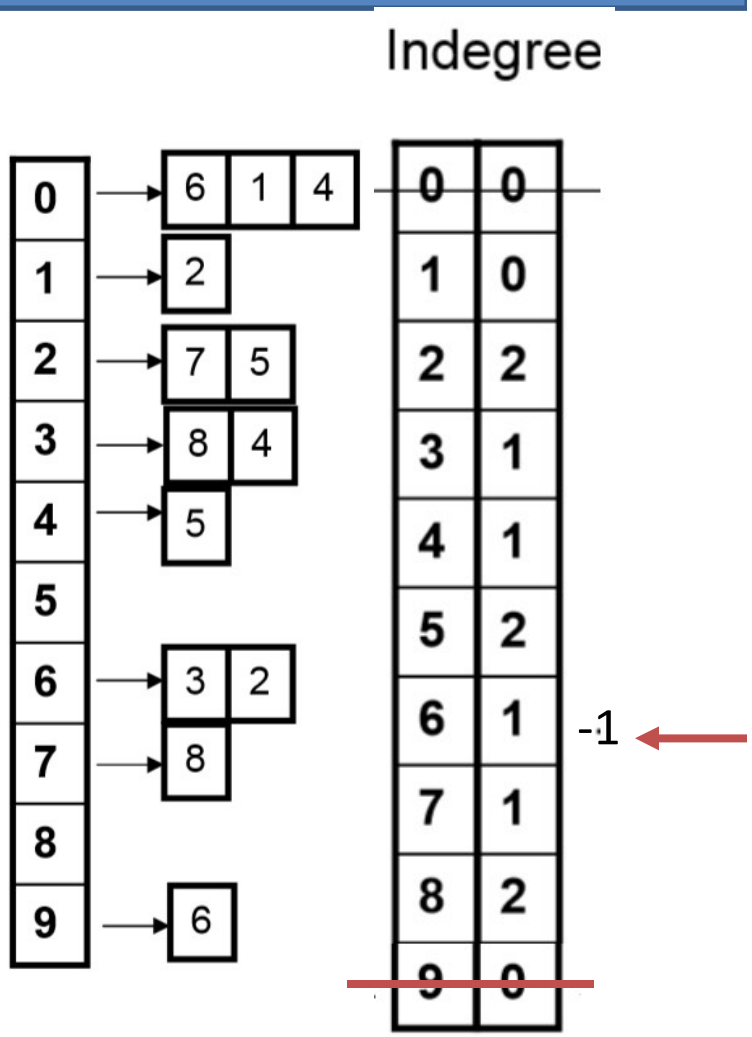


Dequeue 9; Q={1}

Output: 0 9

Sắp xếp topo: Ví dụ

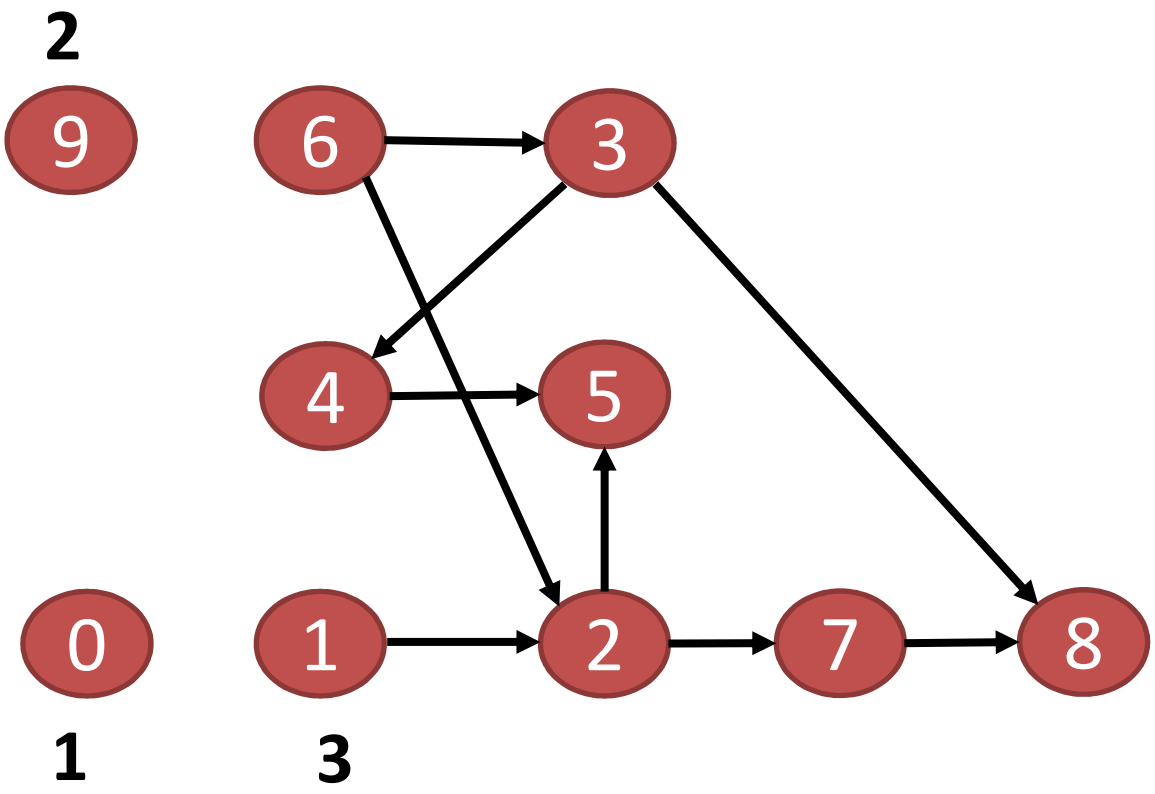


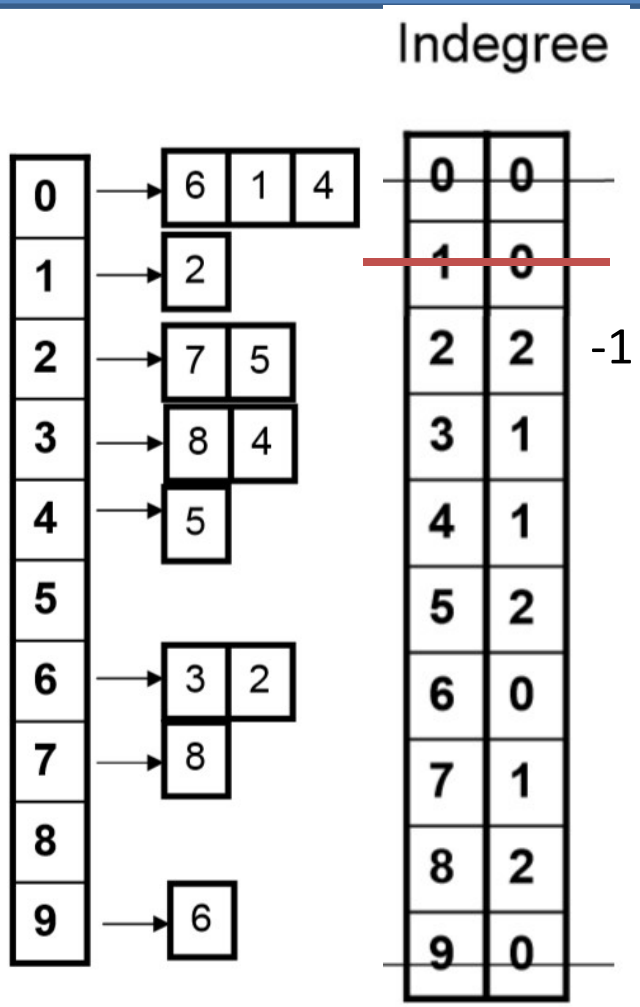


Dequeue 9; Q={1,6}

Output: 0 9

Sắp xếp topo: Ví dụ

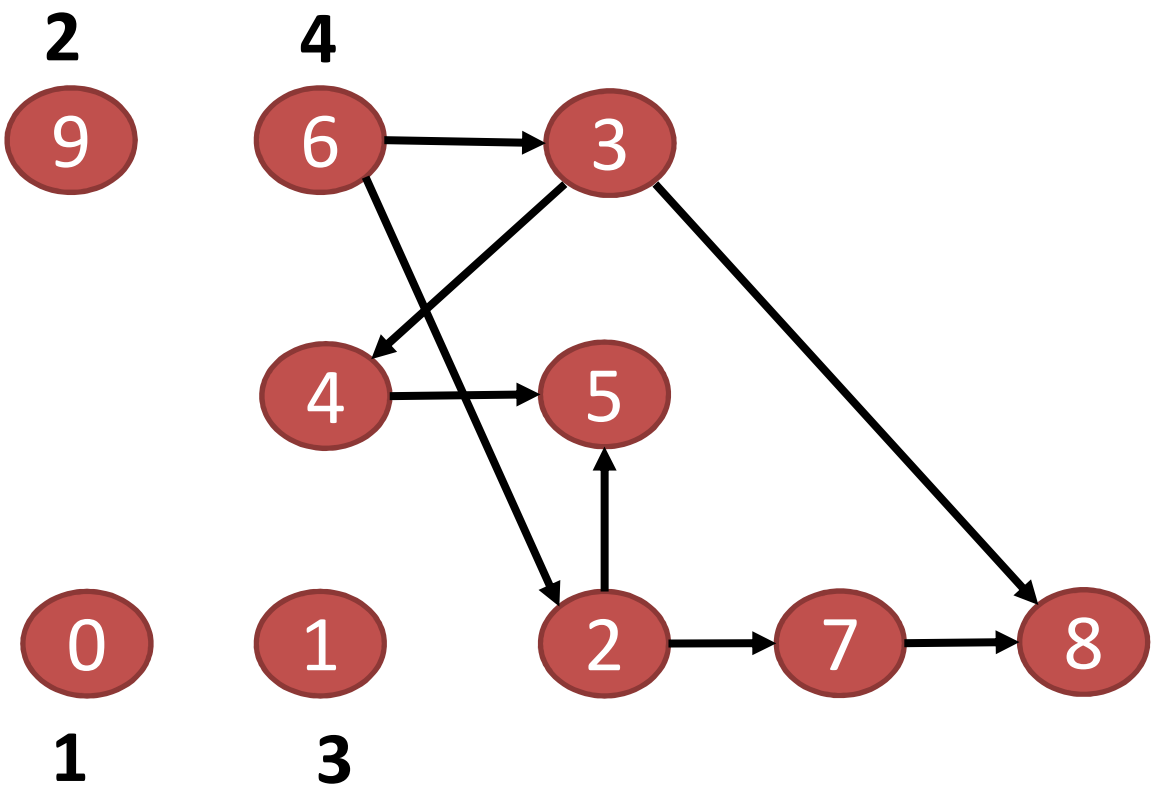


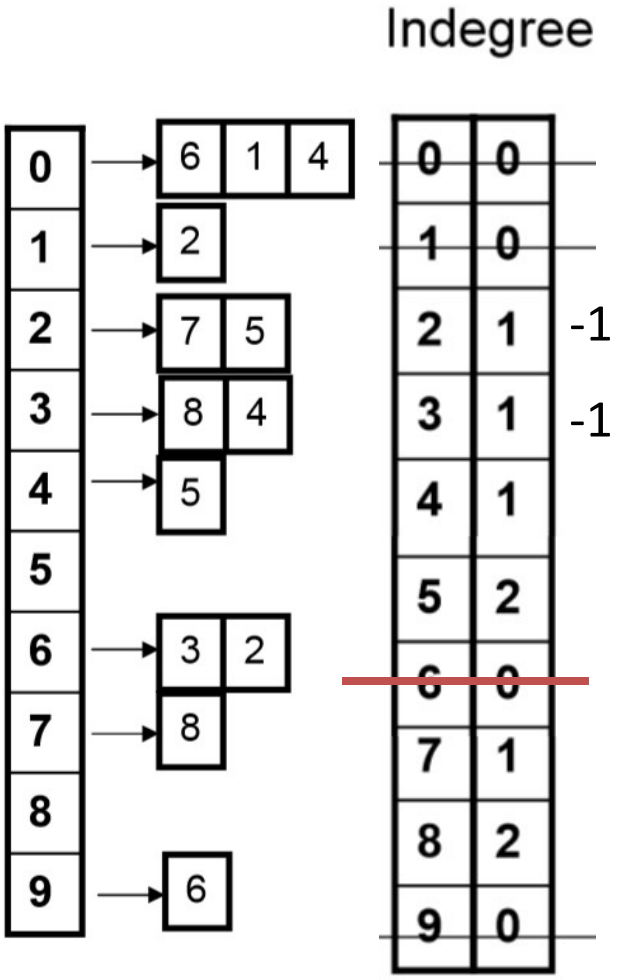


Dequeue 1; Q={6}

Output: 0 9 1

Sắp xếp topo: Ví dụ

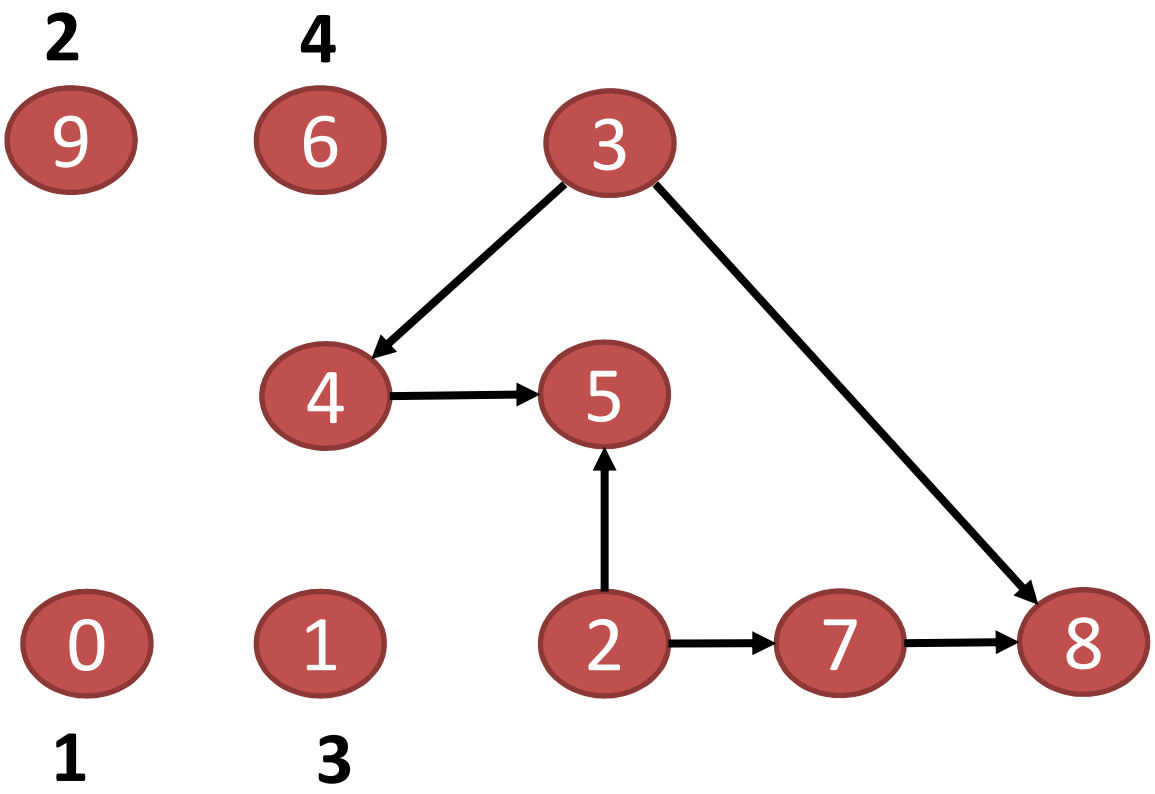


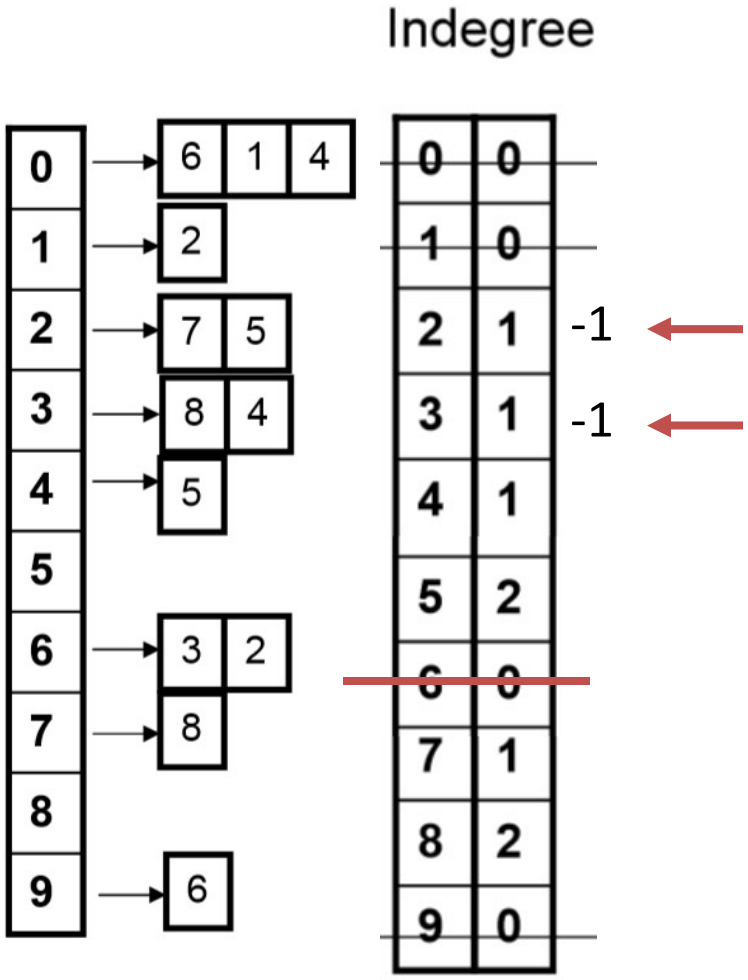


Dequeue 6; Q={}

Output: 0 9 1 6

Sắp xếp topo: Ví dụ

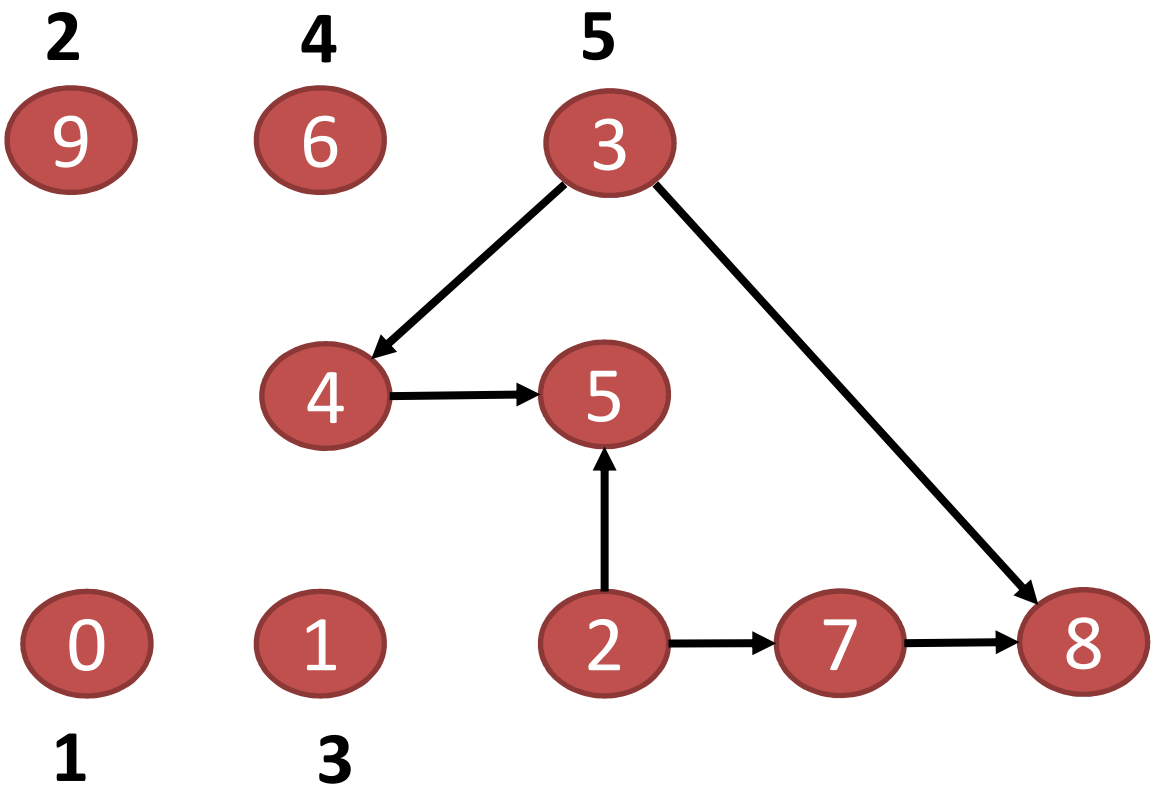


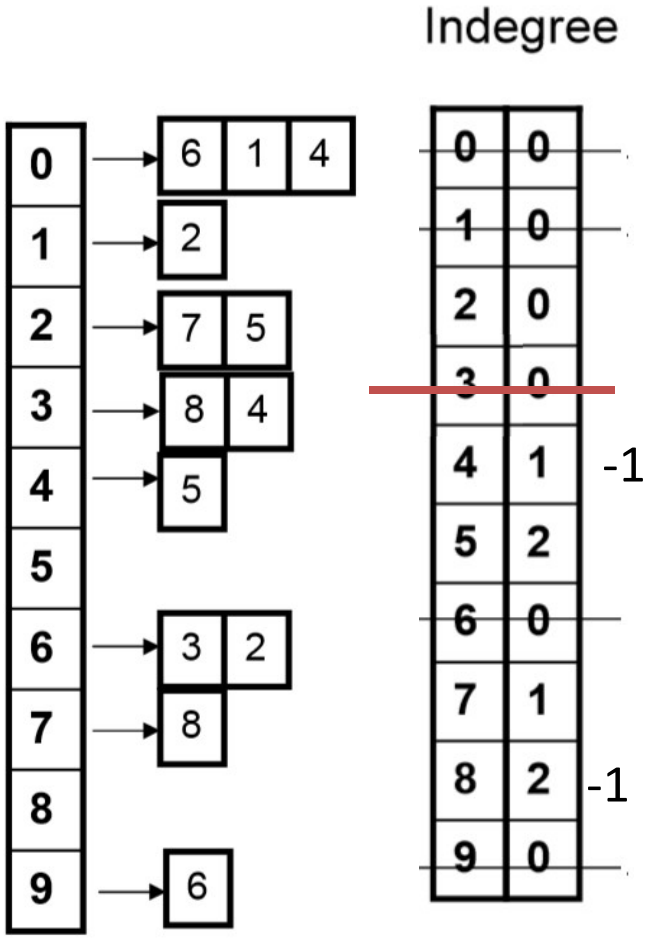


Dequeue 6; Q={3,2}

Output: 0 9 1 6

Sắp xếp topo: Ví dụ

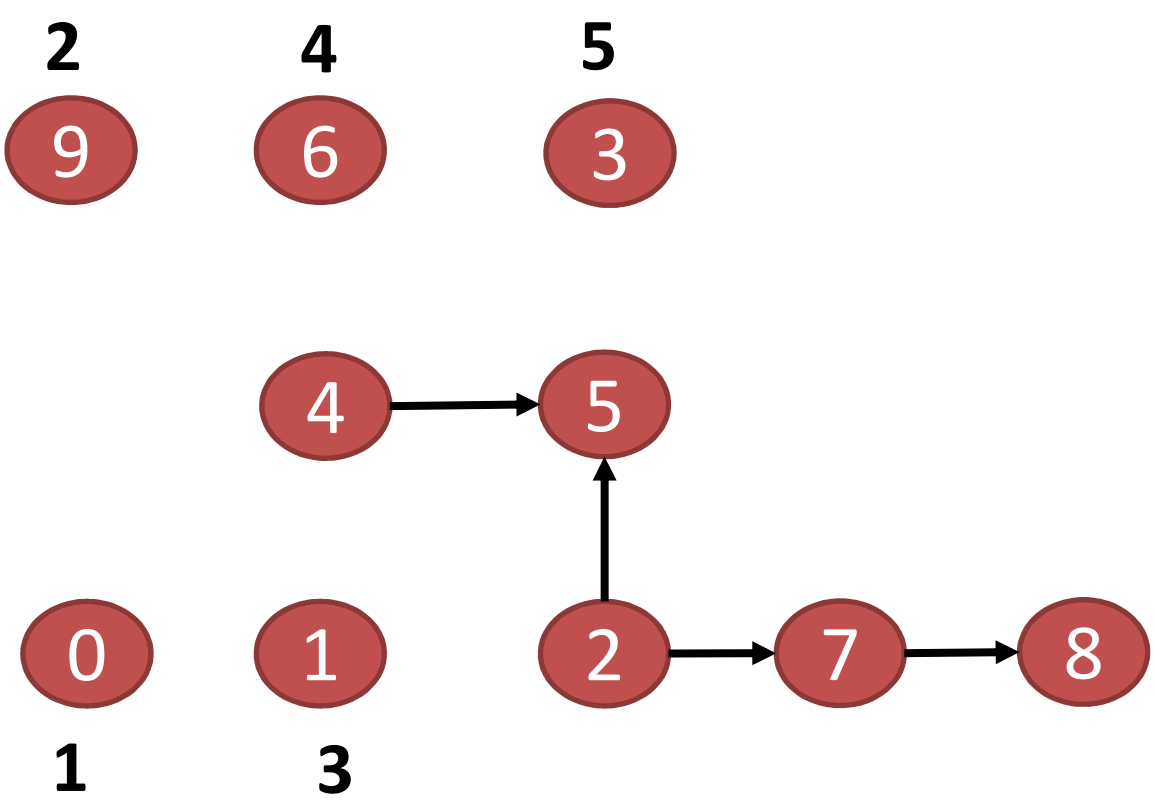


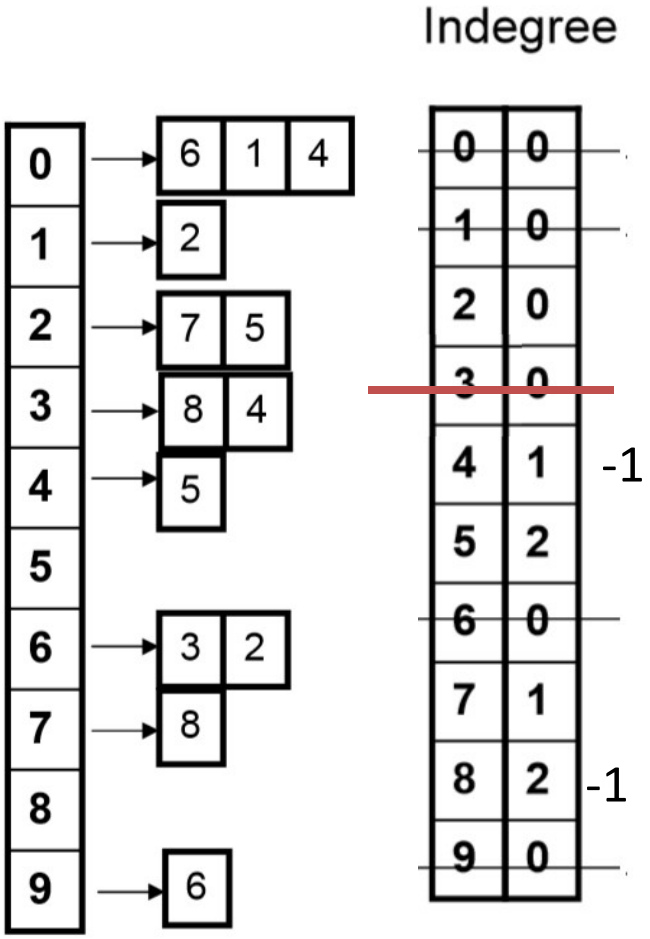


Dequeue 3; Q={2}

Output: 0 9 1 6 3

Sắp xếp topo: Ví dụ

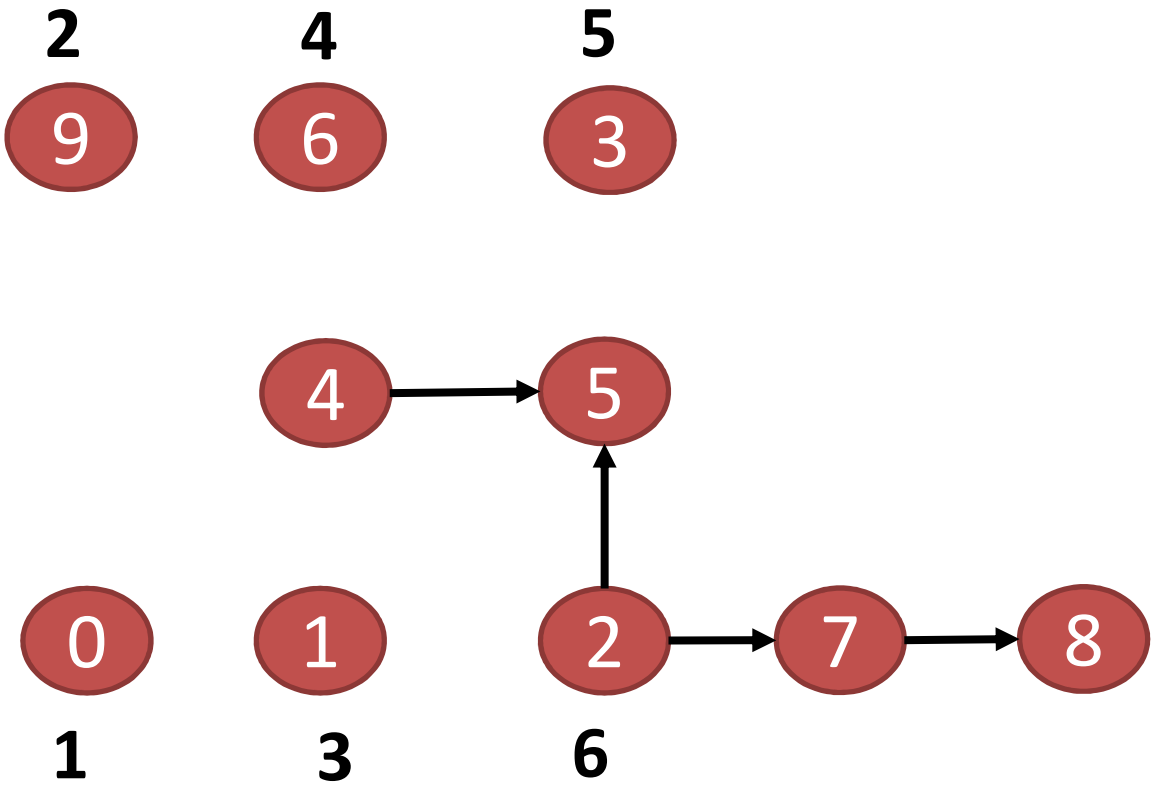


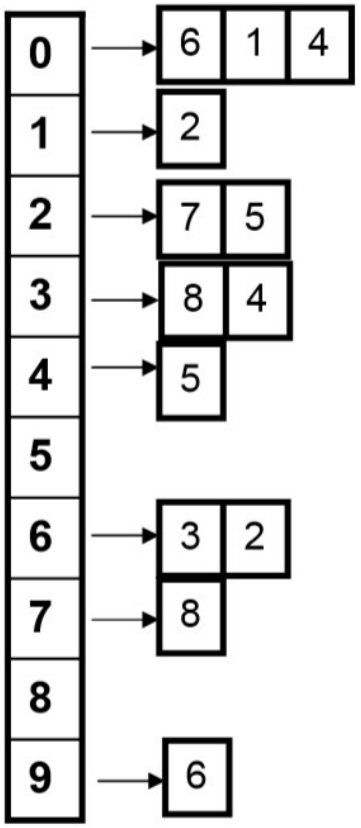


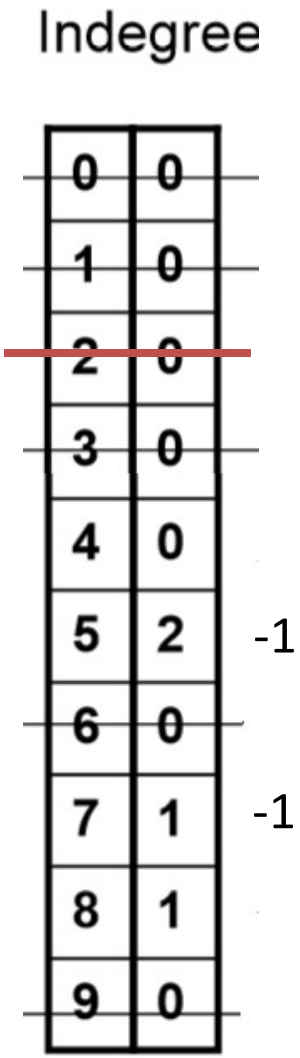
Dequeue 3; **Q={2}** 4}

Output: 0 9 1 6 3

Sắp xếp topo: Ví dụ



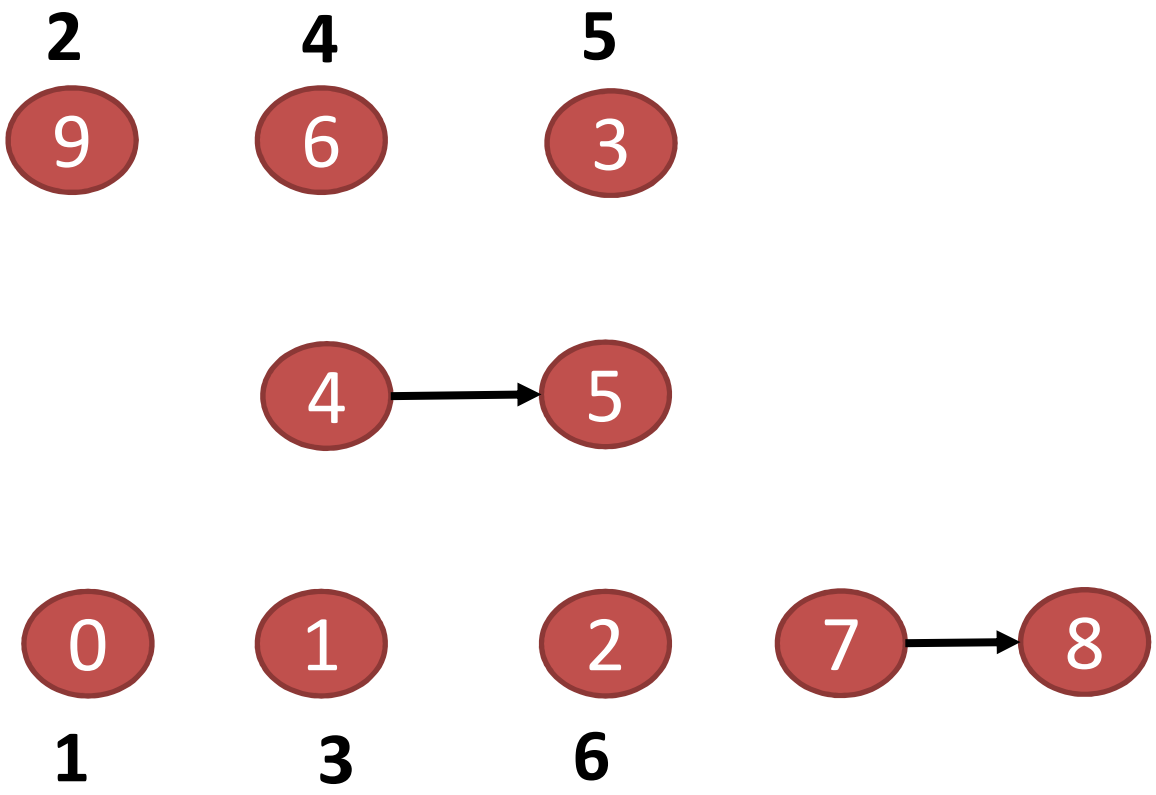




Dequeue 2; Q={4}

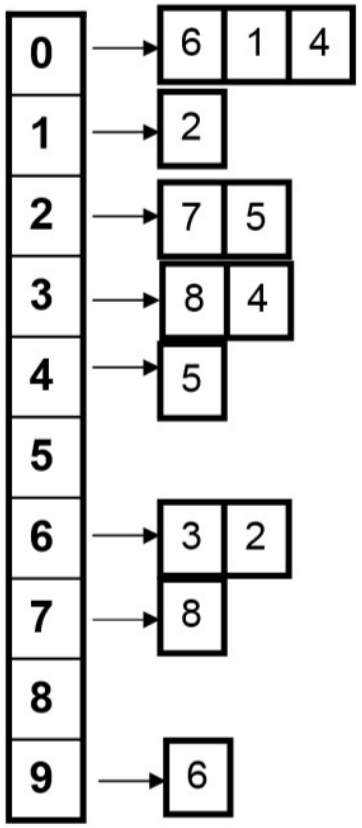
Output: 0 9 1 6 3 2

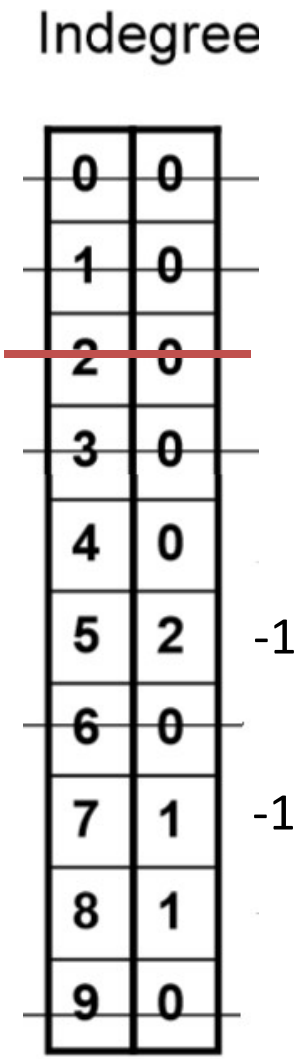
Sắp xếp topo: Ví dụ



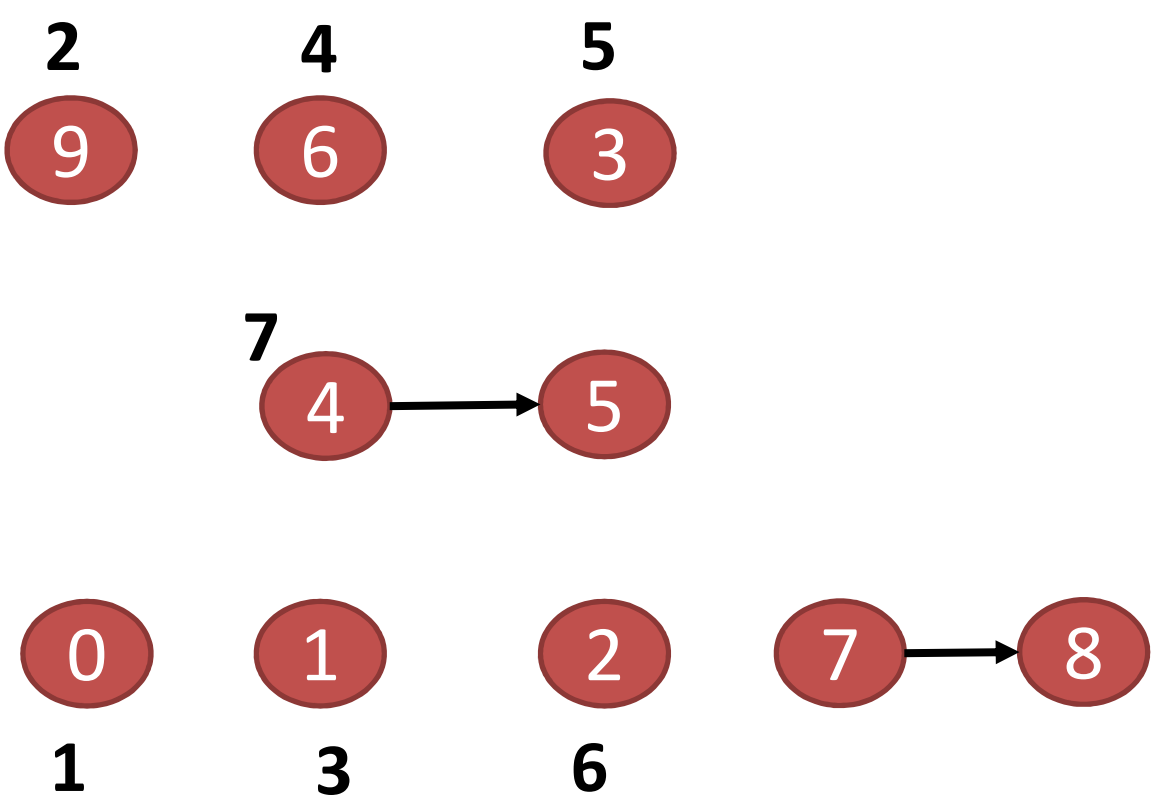
Dequeue 2; Q={4, 7}

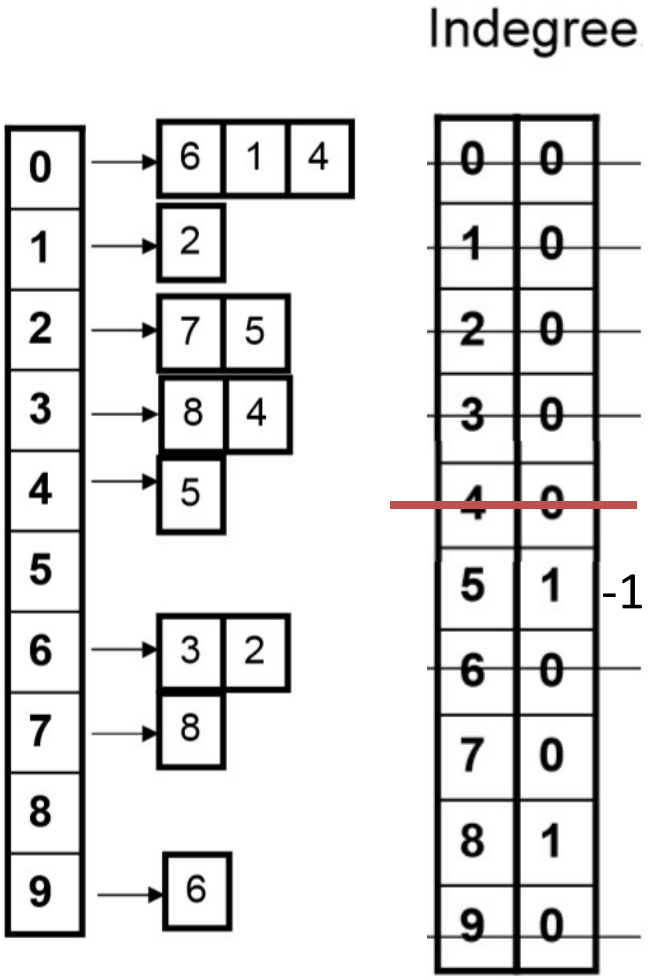
Output: 0 9 1 6 3 2





Sắp xếp topo: Ví dụ

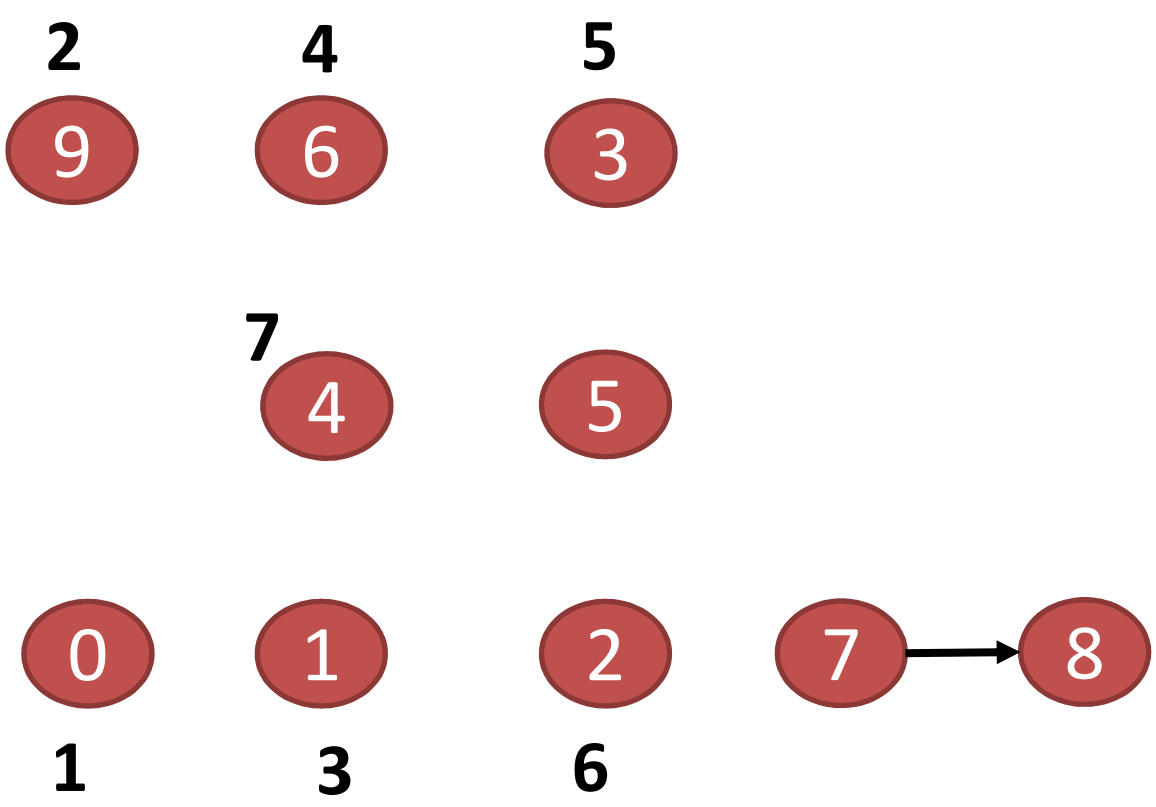


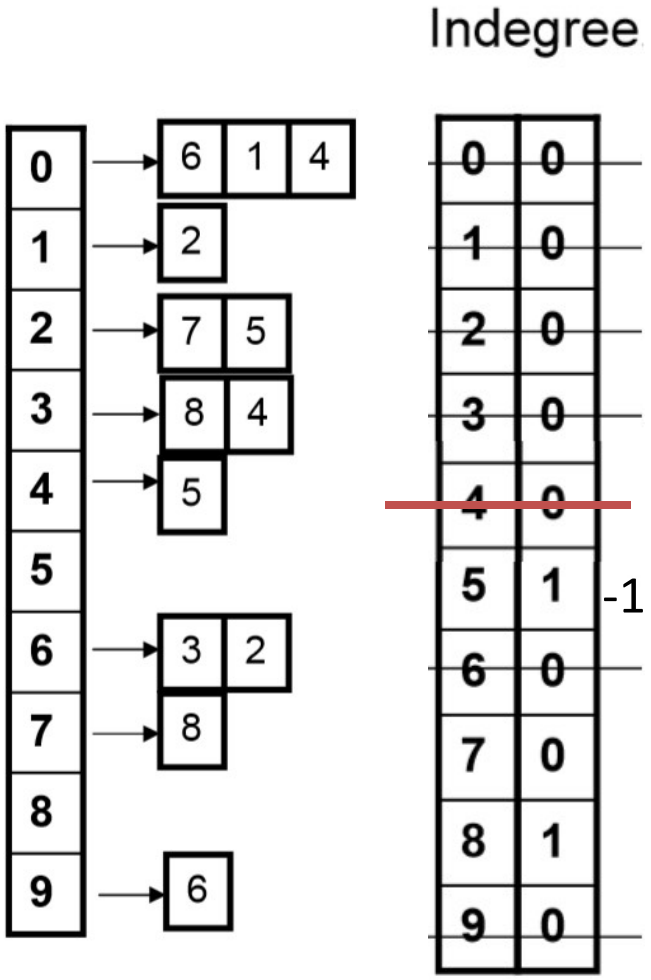


Dequeue 4; Q={7}

Output: 0 9 1 6 3 2 4

Sắp xếp topo: Ví dụ

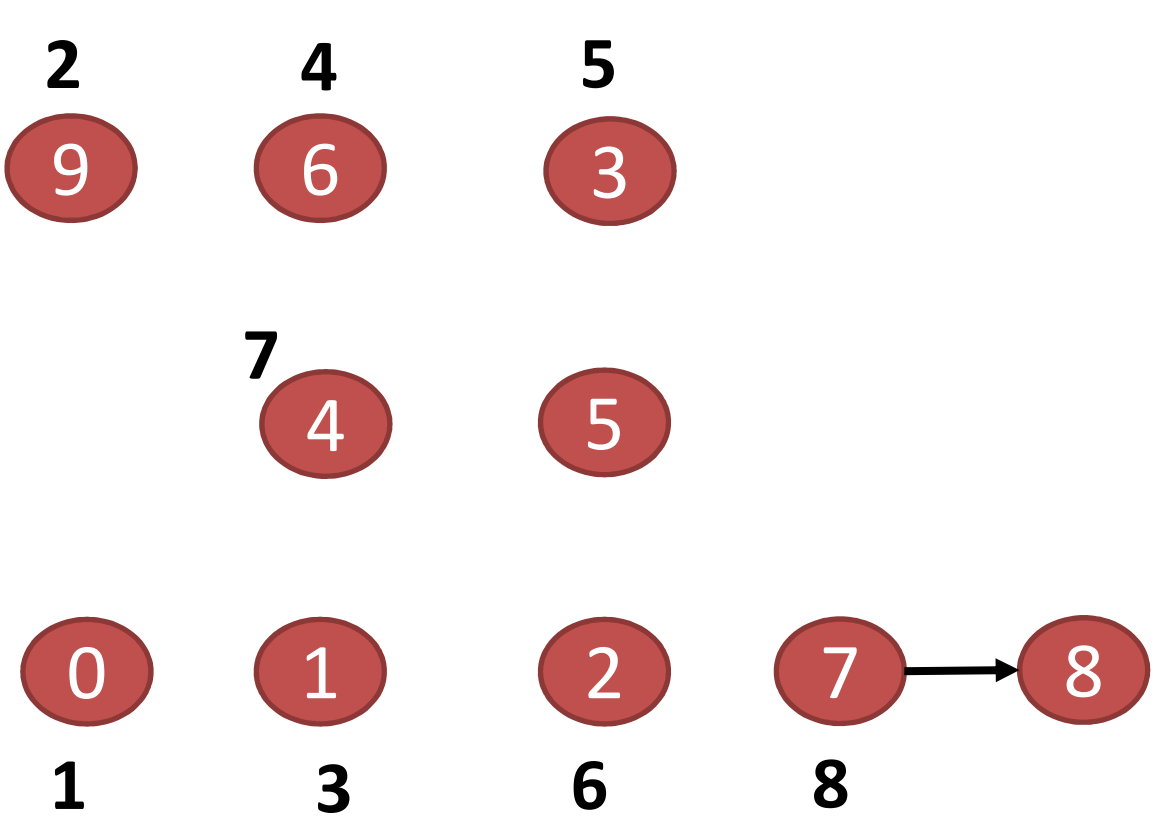


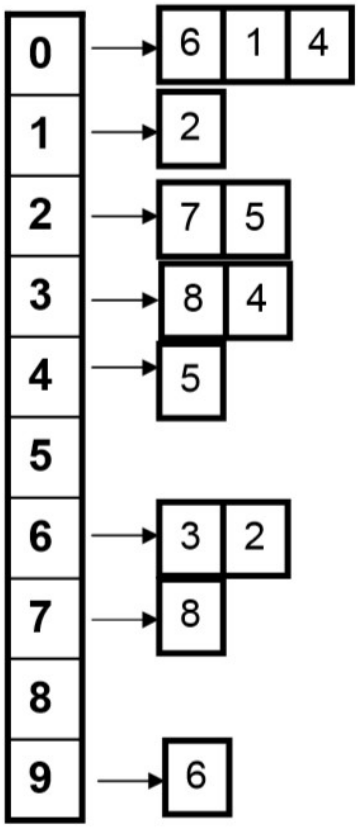


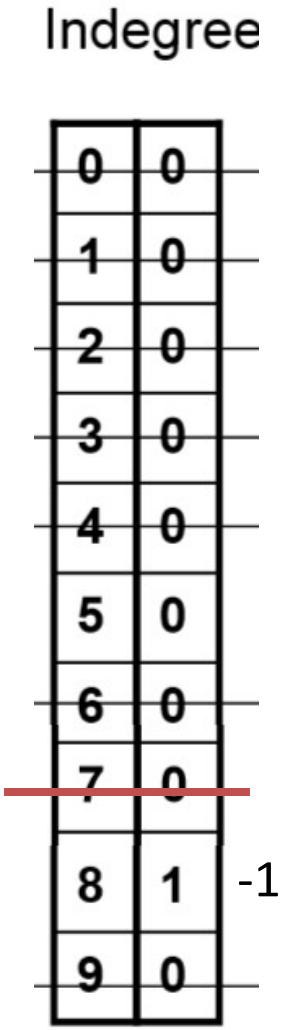
Dequeue 4; Q={7, 5}

Output: 0 9 1 6 3 2 4

Sắp xếp topo: Ví dụ



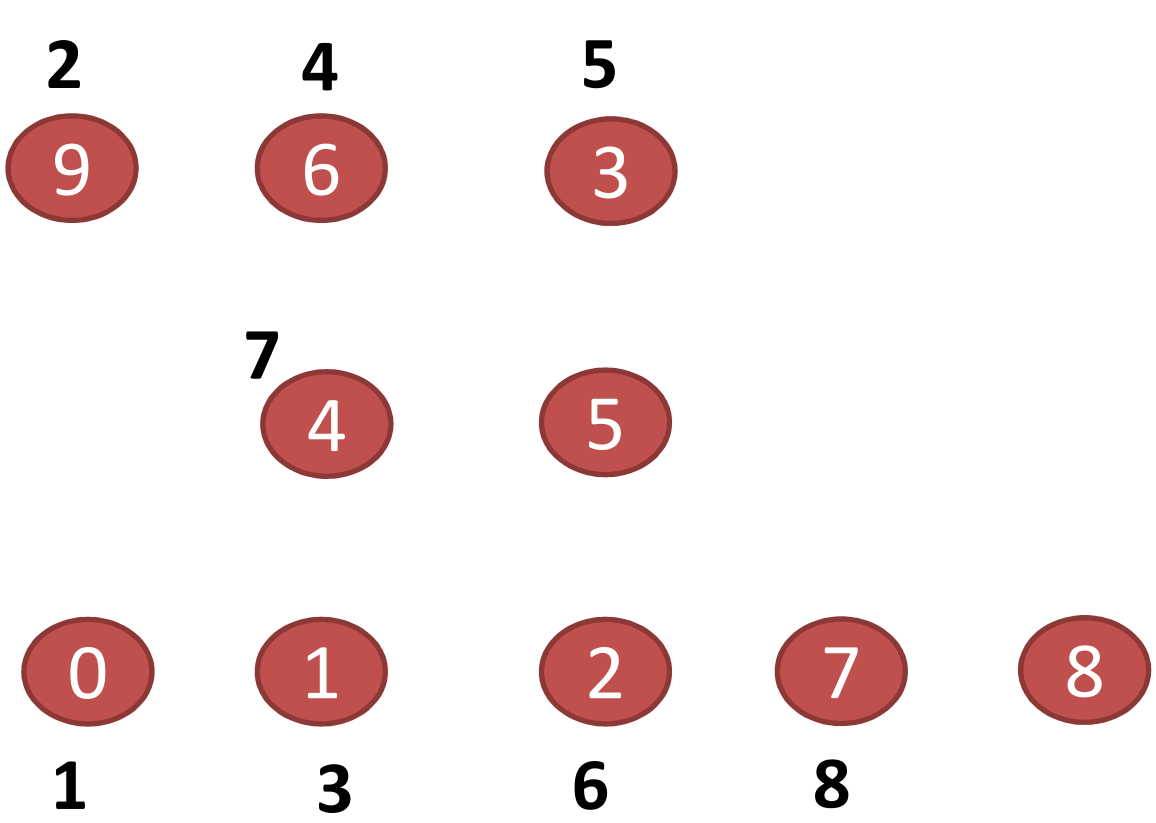


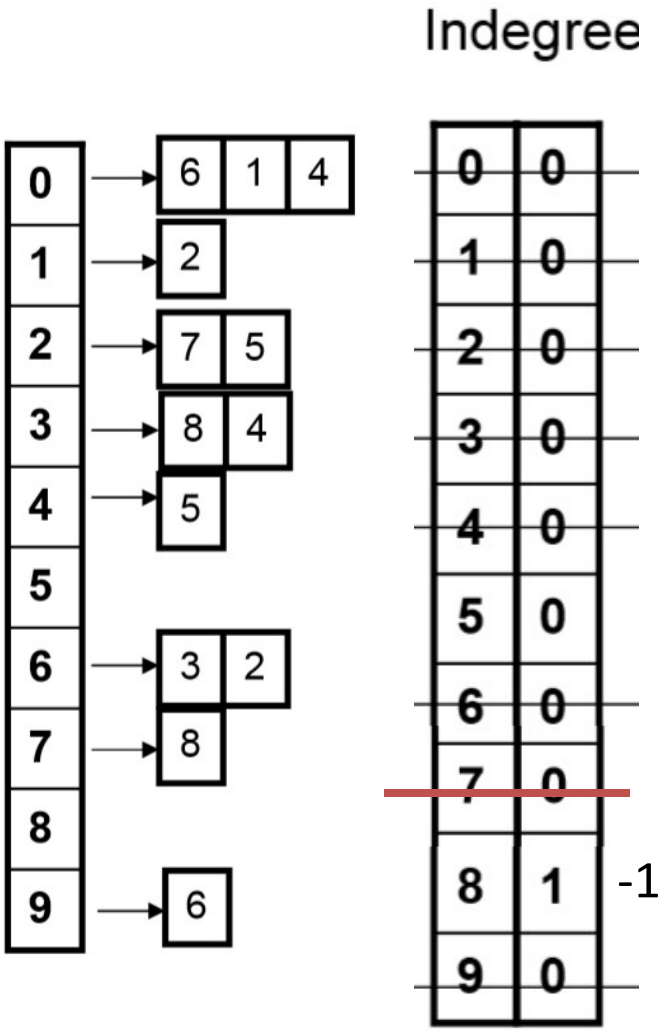


Dequeue 7; Q={5}

Output: 0 9 1 6 3 2 4 7

Sắp xếp topo: Ví dụ

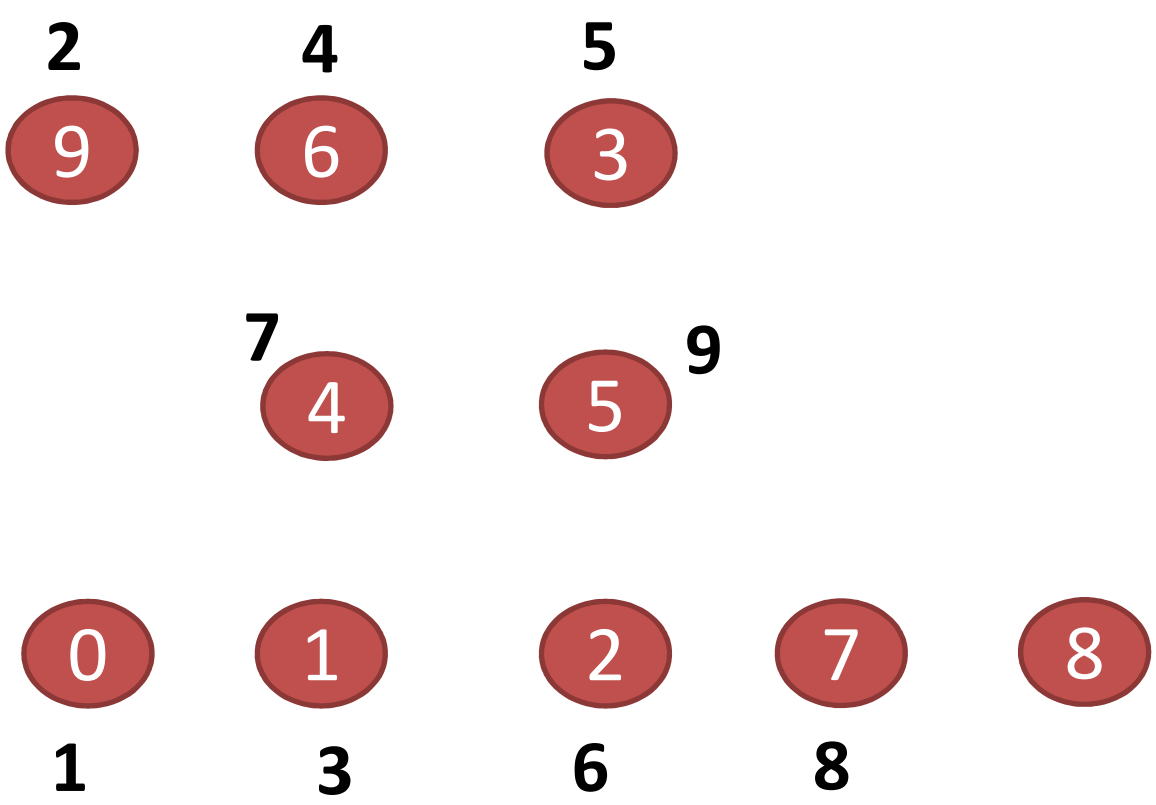


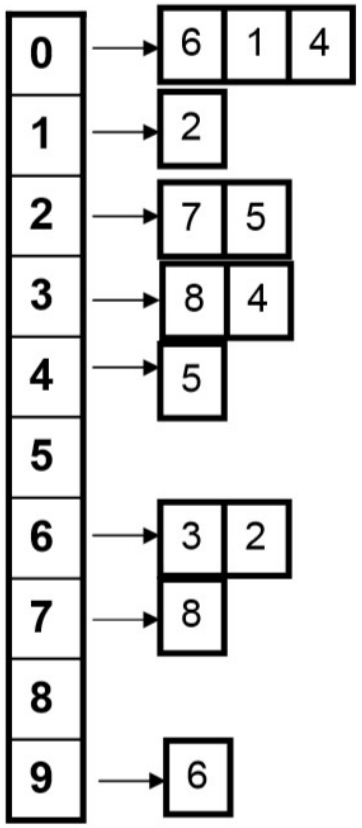


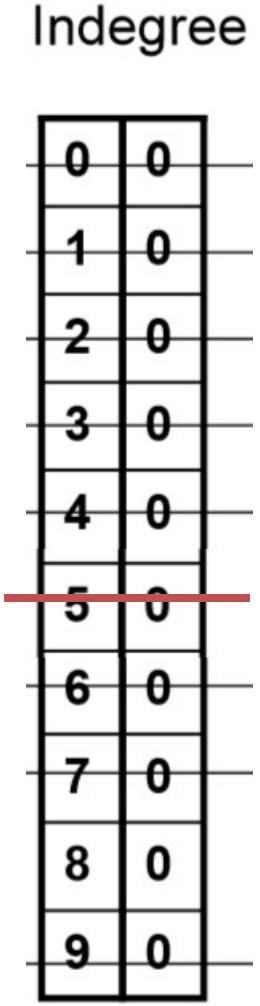
Dequeue 7; Q={5}8}

Output: 0 9 1 6 3 2 4 7

Sắp xếp topo: Ví dụ



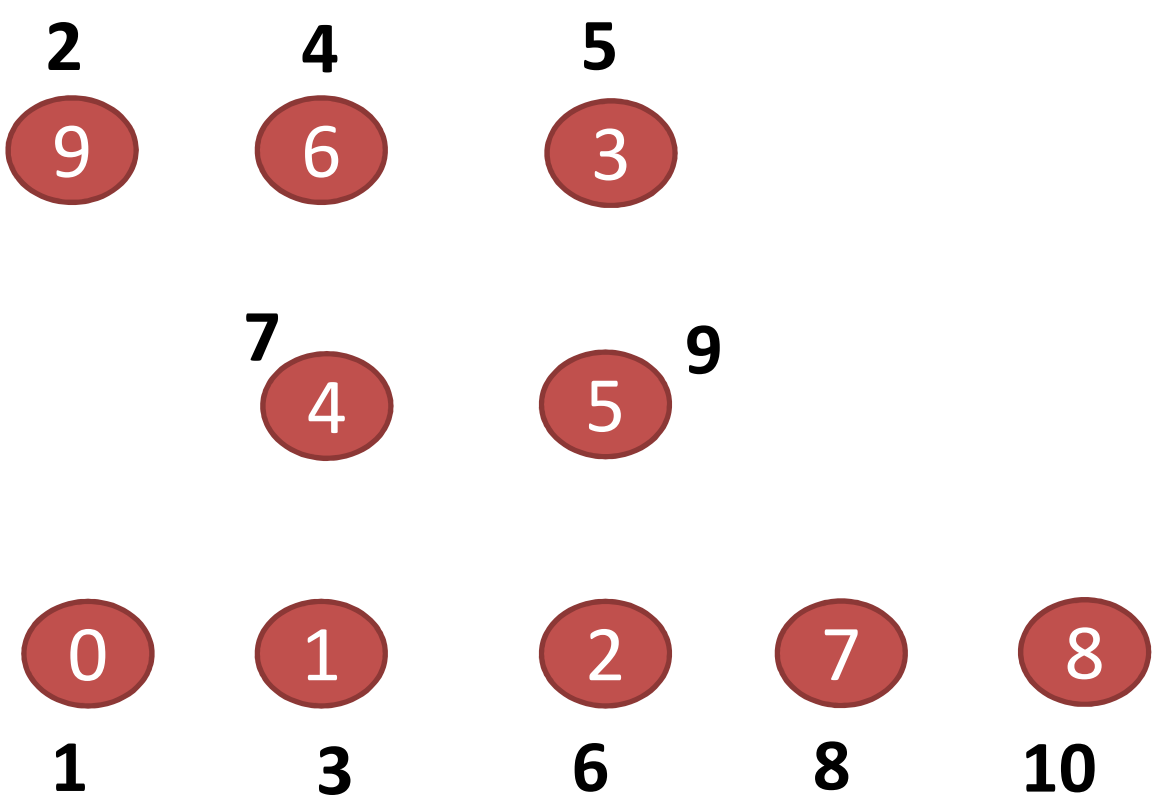


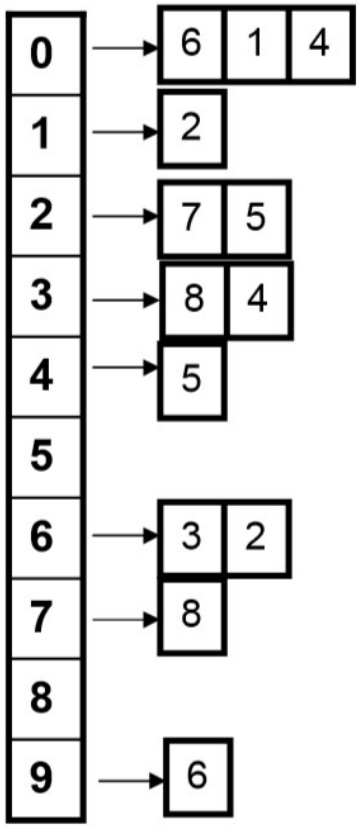


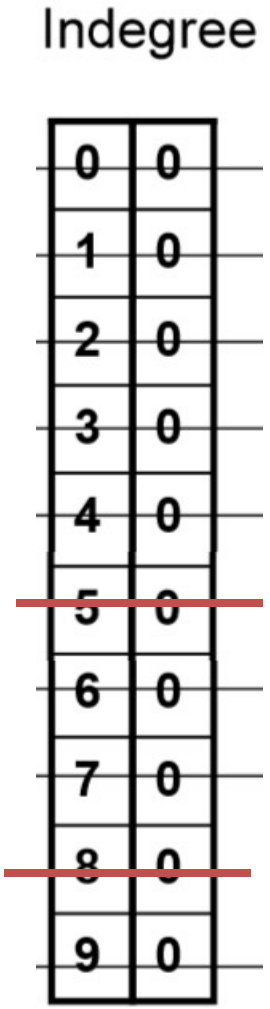
Dequeue 5; Q={8}

Output: 0 9 1 6 3 2 4 7 5

Sắp xếp topo: Ví dụ



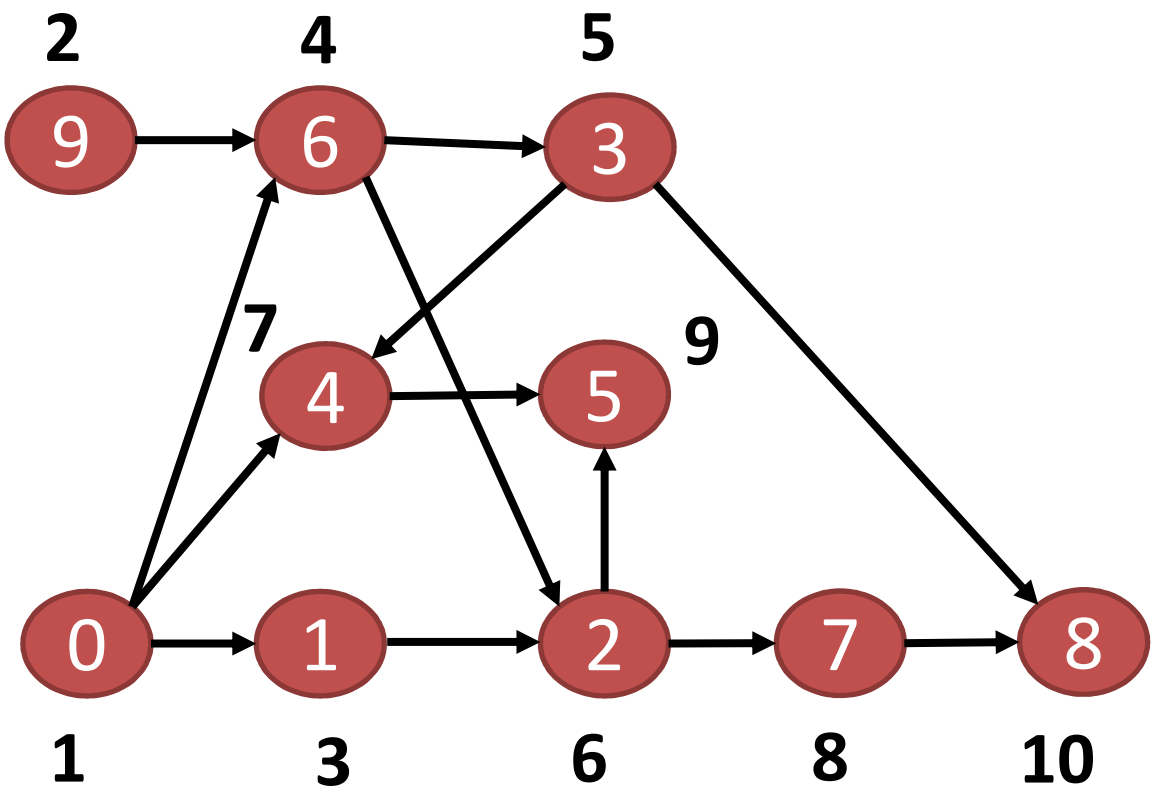




Dequeue 8; Q={}

Output: 0 9 1 6 3 2 4 7 5 8

Sắp xếp topo: Ví dụ



Ví dụ 2: Sắp xếp tôpô: đánh số đỉnh

Có 1 file mô tả các điều kiện tiên quyết giữa các lớp học như sau:

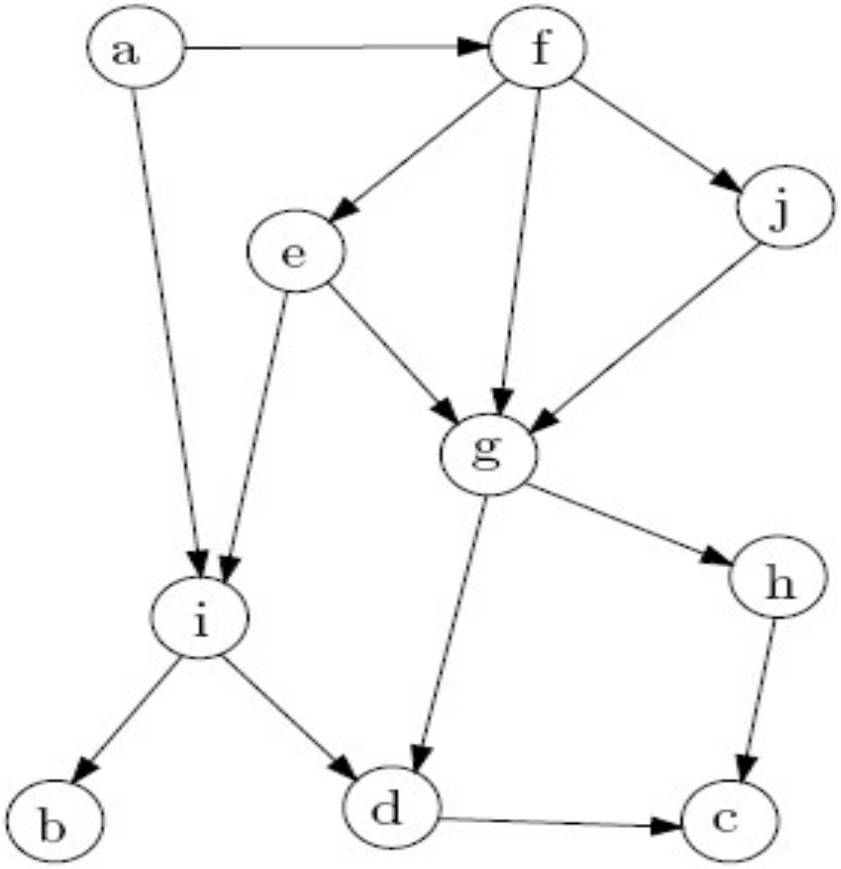
CLASS CS00b
PREREQ CS00i
CLASS CS00c
PREREQ CS00d
PREREQ CS00h
CLASS CS00d
PREREQ CS00i
PREREQ CS00g
CLASS CS00e
PREREQ CS00f
CLASS CS00f
PREREQ CS00a

CLASS CS00g
PREREQ CS00e
PREREQ CS00f
PREREQ CS00j
CLASS CS00h
PREREQ CS00g
CLASS CS00i
PREREQ CS00a
PREREQ CS00e
CLASS CS00j
PREREQ CS00f

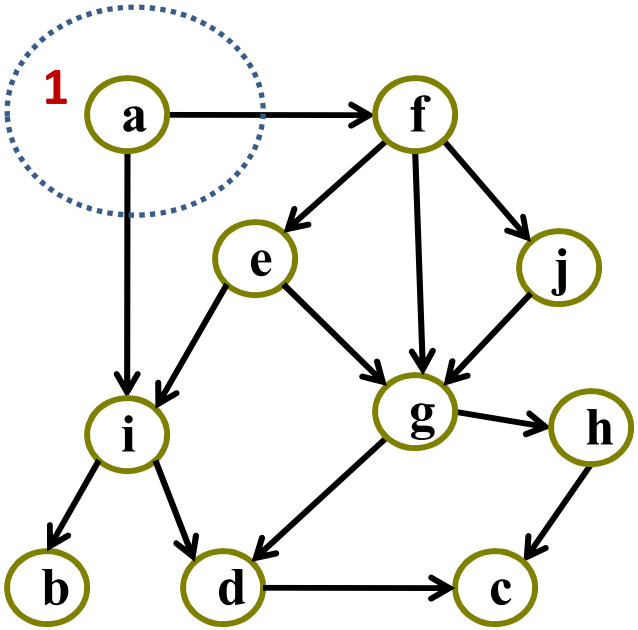
Hãy đưa ra thứ tự cần học các môn ở trên sao cho điều kiện tiên quyết luôn được thỏa mãn

Sắp xếp tôpô: đánh số đỉnh

Ví dụ. Thực hiện thuật toán đánh số đỉnh đối với đồ thị



Sắp xếp tôpô: đánh số đỉnh

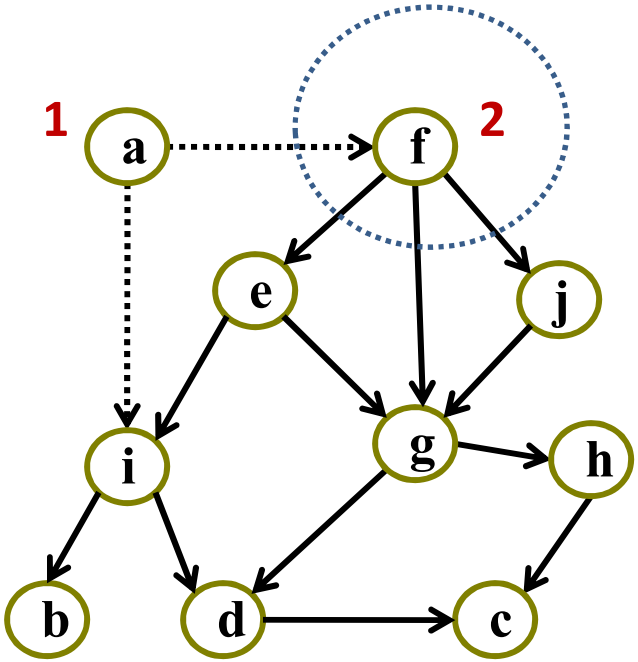


Đỉnh a có deg-(a)=0

Đánh số a bởi 1

Xoá các cung đi ra khỏi a

Sắp xếp tôpô: đánh số đỉnh

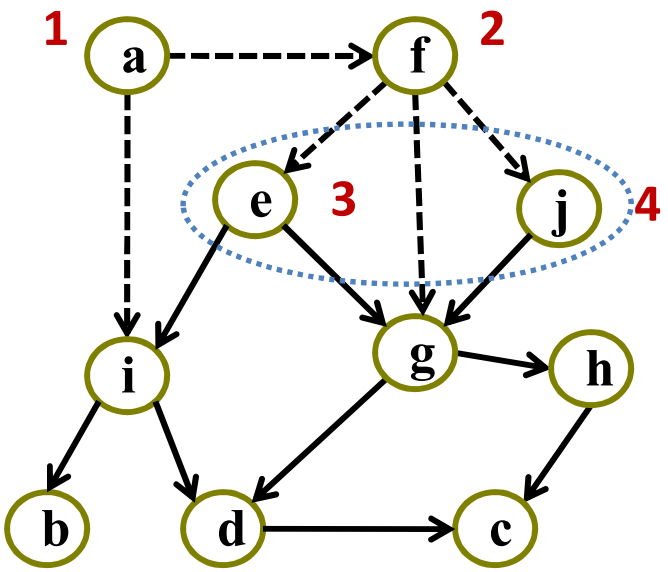


Đỉnh f có deg-(f)=0

Đánh số f bởi 2

Xoá các cung đi ra khỏi f

Sắp xếp tôpô: đánh số đỉnh

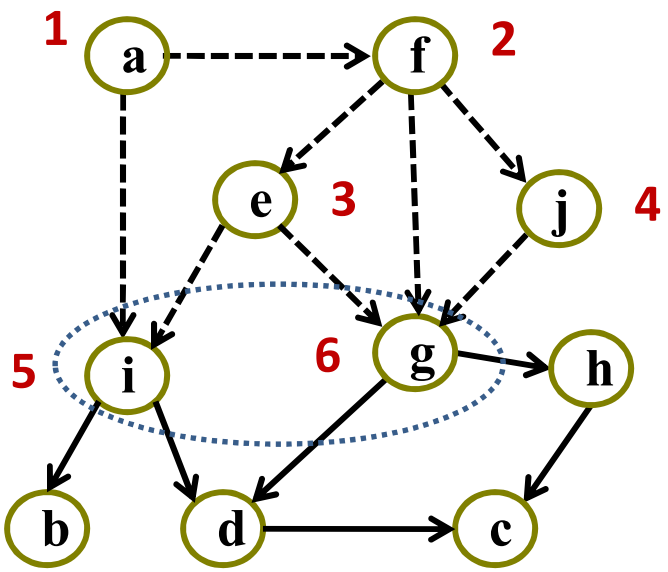


Đỉnh e và j có deg-(e)= deg-(j) = 0

Đánh số e và j theo thứ tự tuỳ ý bởi 3 và 4

Xoá các cung đi ra khỏi e và j

Sắp xếp tôpô: đánh số đỉnh

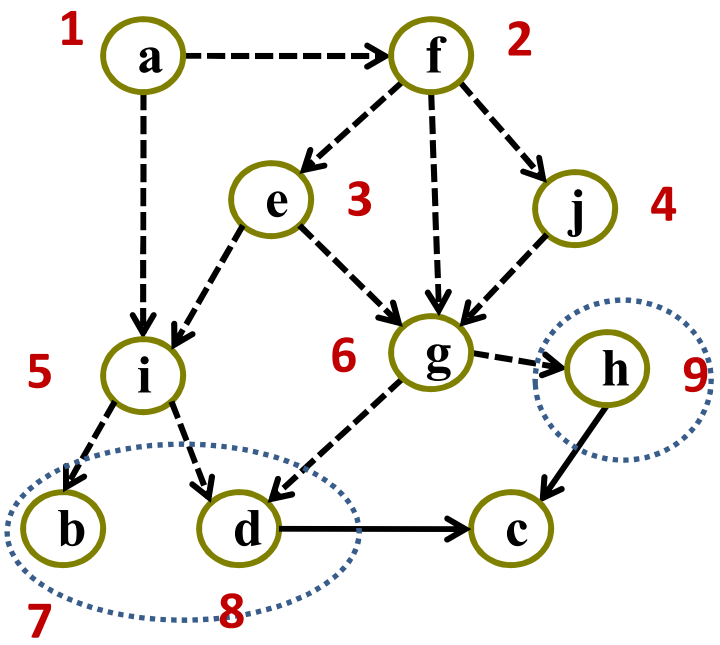


Đỉnh i và g có deg-(i)= deg-(g) = 0

Đánh số i và g theo thứ tự tuỳ ý bởi 5 và 6

Xoá các cung đi ra khỏi i và g

Sắp xếp tôpô: đánh số đỉnh

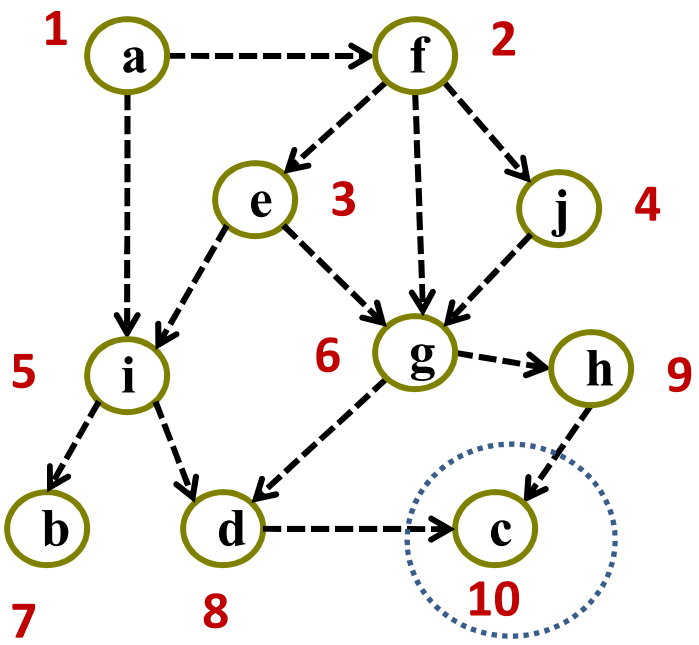


Đỉnh b, d và h có deg-(b)= deg-(d) = deg(h) = 0

Đánh số b, d và h theo thứ tự tuỳ ý bởi 7, 8 và 4

Xoá các cung đi ra khỏi b, d và h

Sắp xếp tôpô: đánh số đỉnh



Đỉnh c có deg-(c)= 0

Đánh số c bởi 10

Thuật toán kết thúc

Thuật toán tìm đđnn trên đồ thị không có chu trình

- Do có thuật toán đánh số trên, nên khi xét đồ thị không có chu trình ta có thể giả thiết là các đỉnh của nó đã được đánh số sao cho mỗi cung chỉ đi từ đỉnh có chỉ số nhỏ đến đỉnh có chỉ số lớn hơn.

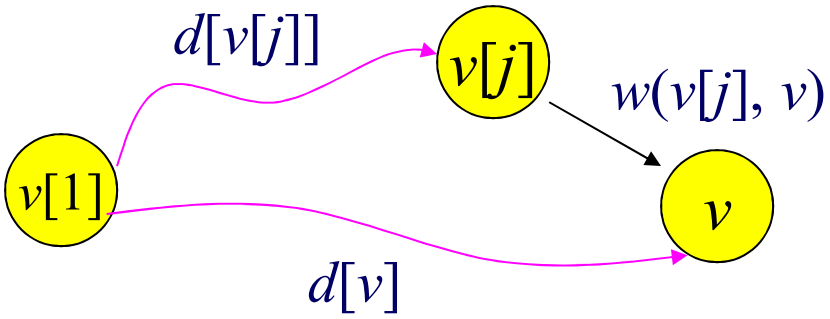
Thuật toán tìm đường đi ngắn nhất từ đỉnh nguồn v[1] đến tất cả các đỉnh còn lại trên đồ thị không có chu trình

- **Đầu vào:** *Đồ thị $G=(V, E)$, trong đó $V=\{ v[1], v[2], \dots , v[n] \}$.*
  Đối với mỗi cung $(v[i], v[j]) \in E$, ta có $i < j$.
  Đồ thị được cho bởi danh sách kề $Ke(v)$, $v \in V$.
- **Đầu ra:** *Khoảng cách từ $v[1]$ đến tất cả các đỉnh còn lại được ghi trong mảng $d[v[i]]$, $i = 2, 3, \dots, n$*

Thuật toán tìm đđnn trên đồ thị không có chu trình

```
void   Critical_Path ( )
{
     d[v[1]] = 0;
     for   (j=2; j<=n;j++) d[v[j]] =∞;
     for   v[j] ∈ Ke[v[1]]
              d[v[j]] := w(v[1], v[j]) ;

     for  (j= 2; j <= n;j++)
       for  v ∈ Ke[v[j]]
          d[v] =  min (d[v],  d[v[j]] + w(v[j], v));
}
```

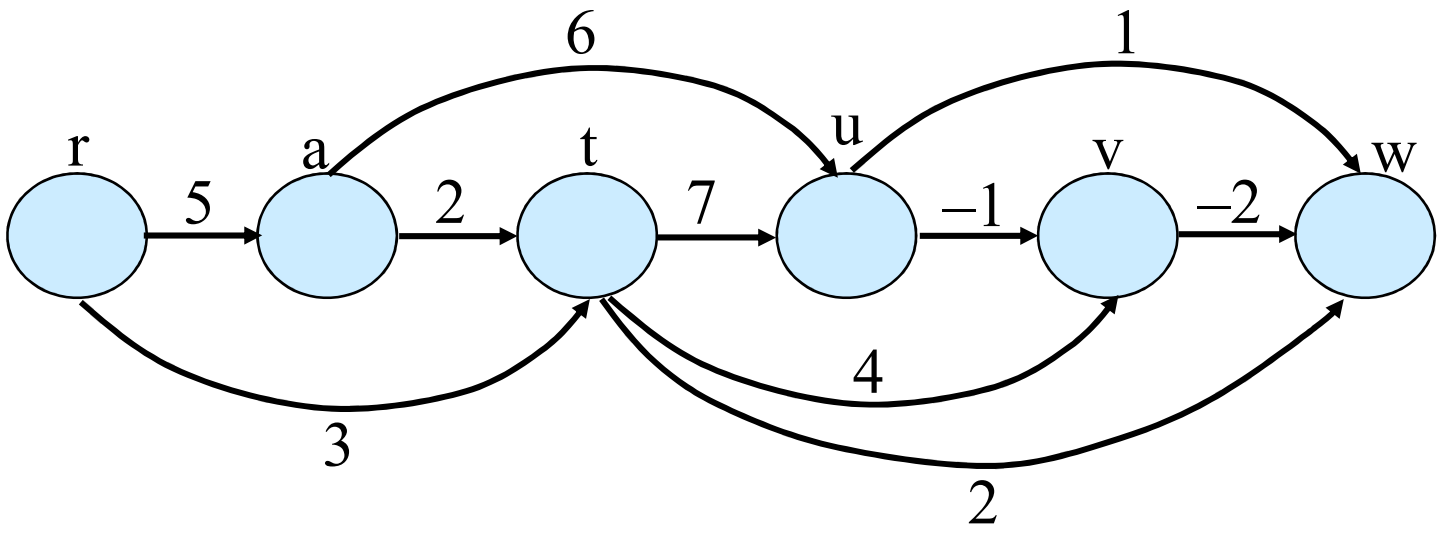
Khởi tạo



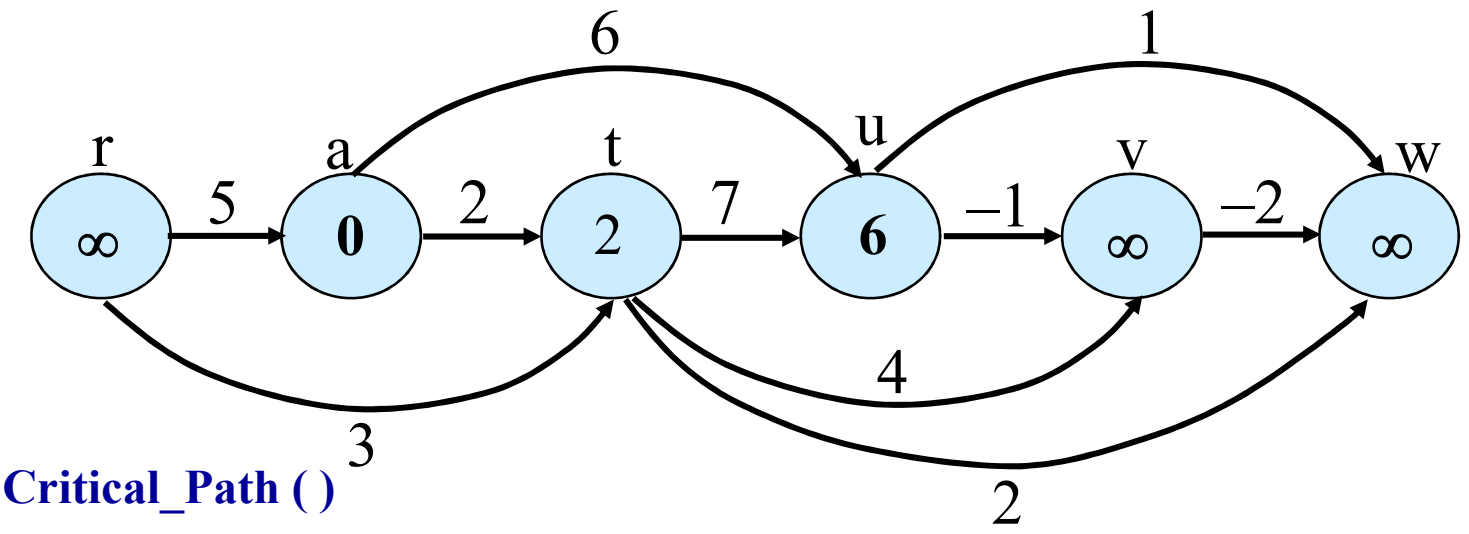
Độ phức tạp tính toán của thuật toán là O(|E|), do mỗi cung của đồ thị phải xét qua đúng 1 lần

Ví dụ

Cần tìm đường đi ngắn nhất từ ***a*** đến tất cả các đỉnh đạt đến được từ nó



Ví dụ




```
void   Critical_Path ( )
{
   d[v[1]] = 0;
   for   (j=2; j<=n;j++) d[v[j]] =∞;
  for   v[j] ∈ Ke[v[1]]
         d[v[j]] := w(v[1], v[j]) ;
   for  (j= 2; j <= n;j++)
          for  v ∈ Ke[v[j]]
                d[v] =  min (d[v],  d[v[j]] + w(v[j], v) ) ;
}
```

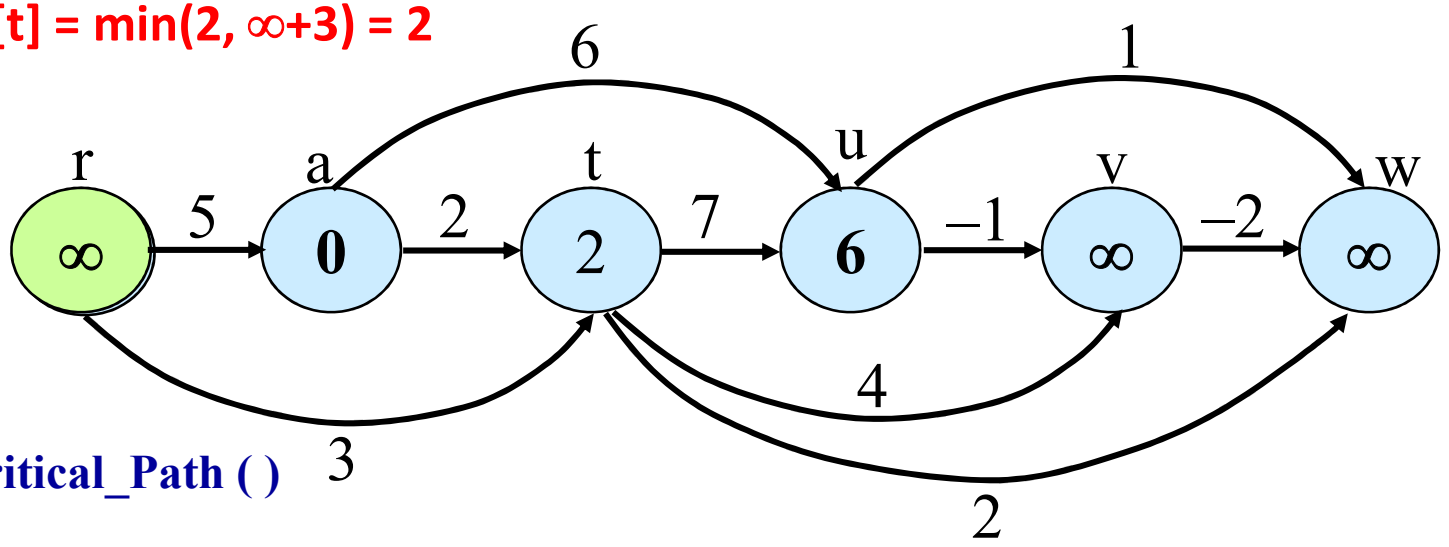
Khởi tạo

j = 2: *v*[2] = *r*

Ke[r] = {*a*, *t*}

d[a] = min(0, ∞+5) = 0

d[t] = min(2, ∞+3) = 2




```
void   Critical_Path ( )
{
    d[v[1]] = 0;
    for   (j=2; j<=n;j++) d[v[j]] =∞;
   for   v[j] ∈ Ke[v[1]]
         d[v[j]] := w(v[1], v[j]) ;
   for  (j= 2; j <= n;j++)
         for  v ∈ Ke[v[j]]
               d[v] =  min (d[v],  d[v[j]] + w(v[j], v) ) ;
}
```

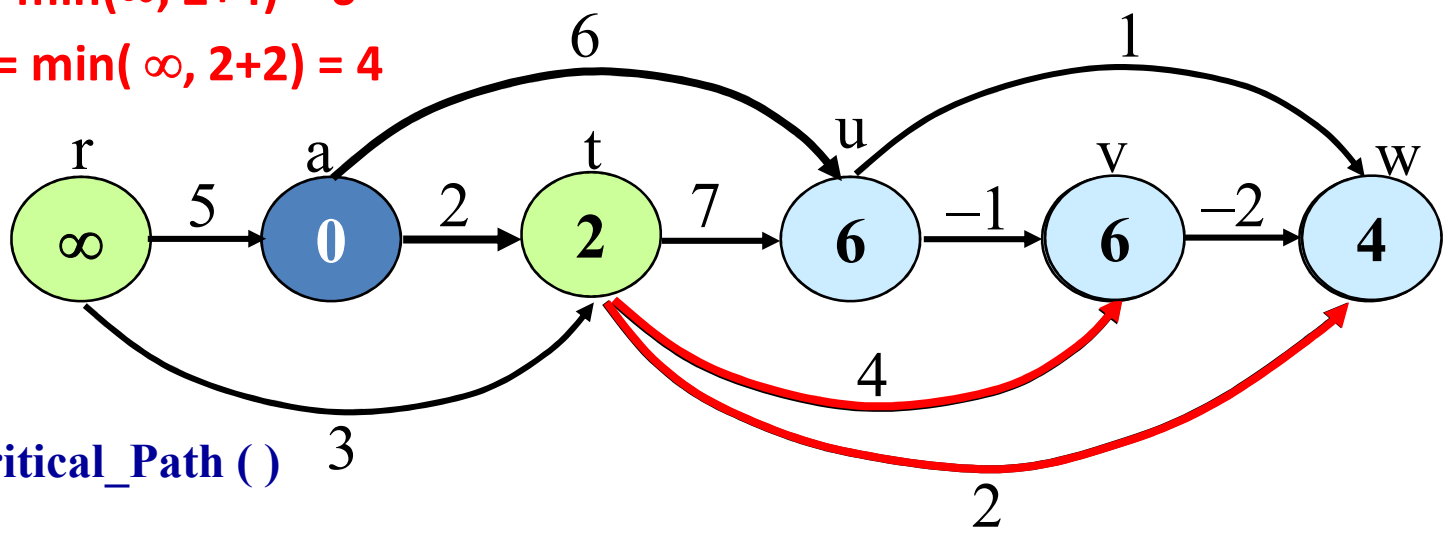
Duyệt qua các đỉnh: r, t, u, v, w

j = 3: *v*[3] = *t*

Ke[t] = {u, *v*, *w*}

d[v] = min(∞, 2+4) = 6

d[w] = min(∞, 2+2) = 4




```
void   Critical_Path ( )
{
    d[v[1]] = 0;
    for   (j=2; j<=n;j++) d[v[j]] =∞;
   for   v[j] ∈ Ke[v[1]]
         d[v[j]] := w(v[1], v[j]) ;
   for  (j= 2; j <= n;j++)
         for  v ∈ Ke[v[j]]
               d[v] =  min (d[v],  d[v[j]] + w(v[j], v) ) ;
}
```

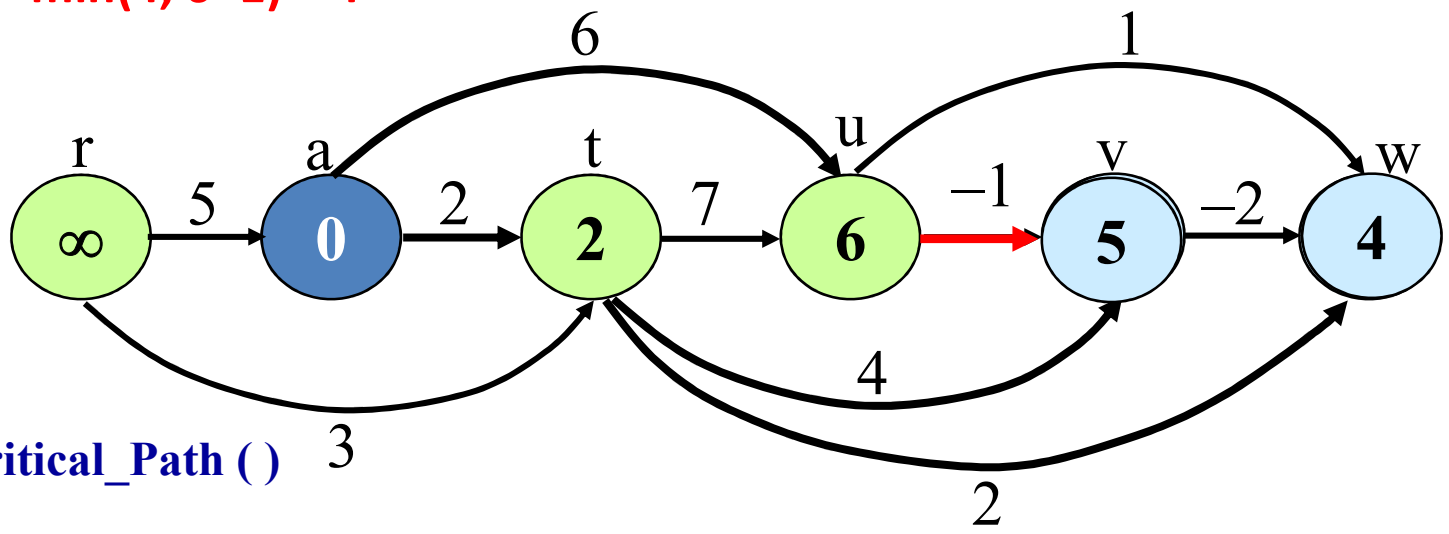
Duyệt qua các đỉnh: r, t, u, v, w

j = 4: *v*[4] = *u*

Ke[u] = {*v*, *w*}

d[v] = min(6, 6-1) = 5

d[w] = min(4, 6+1) = 4



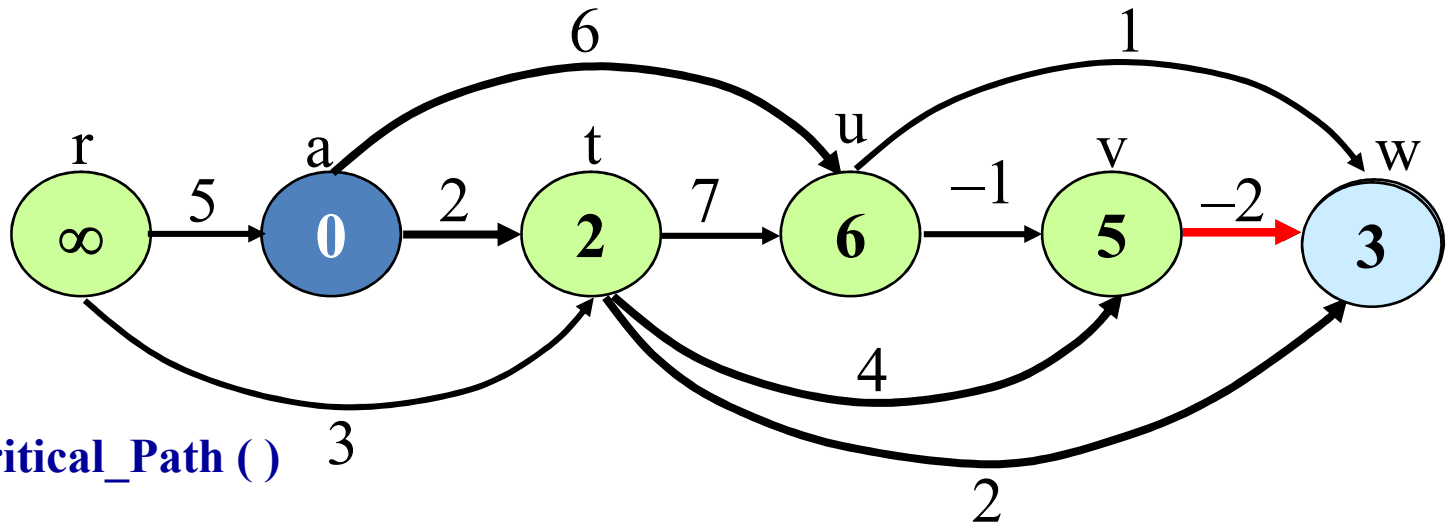

```
void   Critical_Path ( )
{
    d[v[1]] = 0;
    for   (j=2; j<=n;j++) d[v[j]] =∞;
   for   v[j] ∈ Ke[v[1]]
         d[v[j]] := w(v[1], v[j]) ;
   for  (j= 2; j <= n;j++)
         for  v ∈ Ke[v[j]]
               d[v] =  min (d[v],  d[v[j]] + w(v[j], v) ) ;
}
```

Duyệt qua các đỉnh: r, t, u, v, w

j = 5: *v*[5] = *v*

Ke[v] = {*w*}

d[w] = min(4, 5-2) = 3

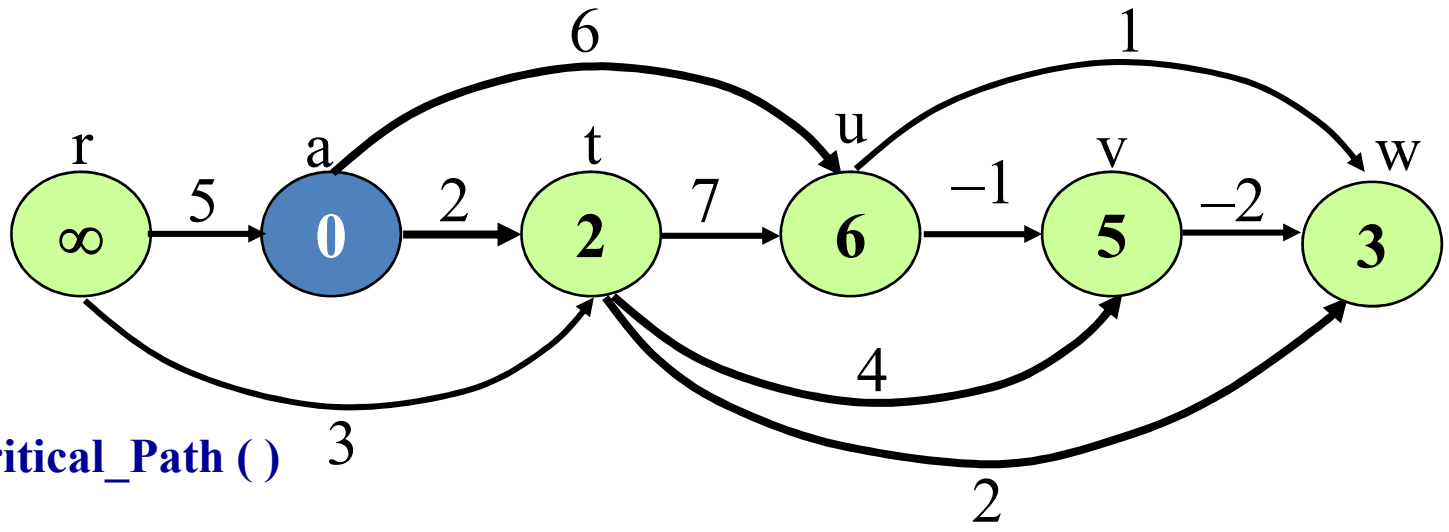



```
void   Critical_Path ( )
{
    d[v[1]] = 0;
    for   (j=2; j<=n;j++) d[v[j]] =∞;
   for   v[j] ∈ Ke[v[1]]
         d[v[j]] := w(v[1], v[j]) ;
   for  (j= 2; j <= n;j++)
         for  v ∈ Ke[v[j]]
               d[v] =  min (d[v],  d[v[j]] + w(v[j], v) ) ;
}
```

Duyệt qua các đỉnh: r, t, u, v, w

j = 6: *v*[6] = *w*

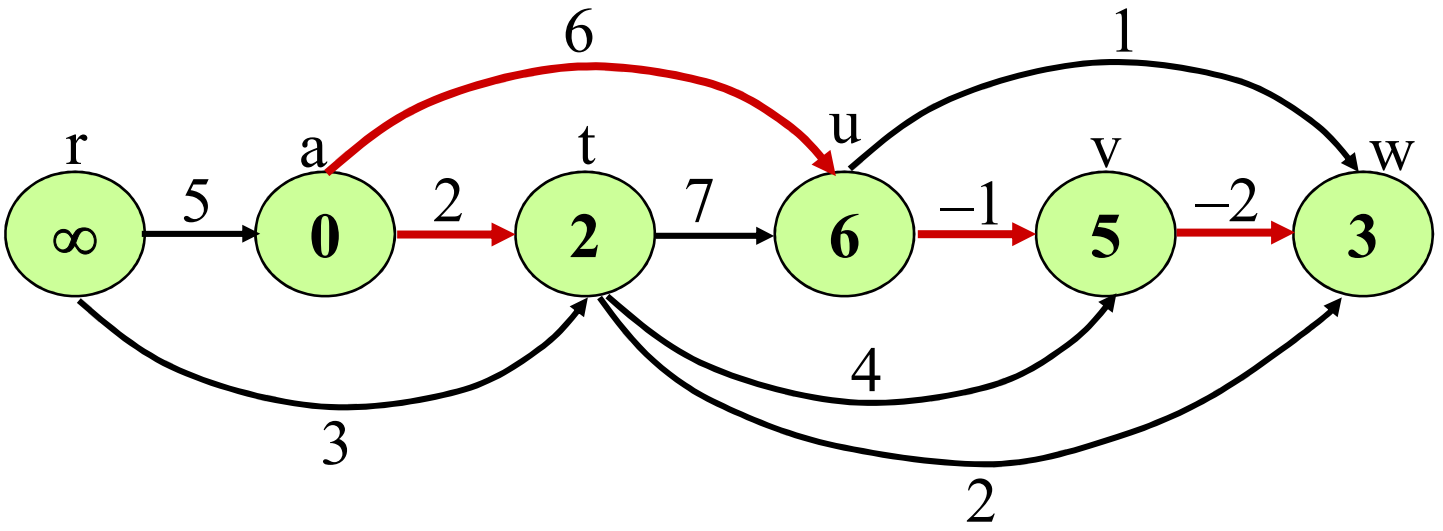
Ke[w] = ∅




```
void   Critical_Path ( )
{
    d[v[1]] = 0;
    for   (j=2; j<=n;j++) d[v[j]] =∞;
   for   v[j] ∈ Ke[v[1]]
         d[v[j]] := w(v[1], v[j]) ;
   for  (j= 2; j <= n;j++)
         for  v ∈ Ke[v[j]]
               d[v] =  min (d[v],  d[v[j]] + w(v[j], v) ) ;
}
```

Duyệt qua các đỉnh: r, t, u, v, w

Ví dụ



Kết quả: Cây đường đi ngắn nhất từ *a* thể hiện bởi các cung màu đỏ

Ứng dụng: PERT

- Xây dựng phương pháp giải bài toán điều khiển việc thực hiện những dự án lớn, gọi tắt là PERT (*Project Evaluation and Review Technique*) hay CDM (*Critical path Method*).

- Việc thi công một công trình lớn được chia ra làm n công đoạn, đánh số từ 1 đến n. Có một số công đoạn mà việc thực hiện nó chỉ được tiến hành sau khi một số công đoạn nào đó đã hoàn thành. Đối với mỗi công đoạn i biết $t[i]$ là thời gian cần thiết để hoàn thành nó ($i = 1, 2,..., n$).

Ứng dụng: PERT

- Các dữ liệu với $n = 8$ được cho trong bảng sau đây

Công đoạn	t[i] (tuần)	Các công đoạn phải hoàn thành trước nó
1	15	Không có
2	30	1
3	80	Không có
4	45	2, 3
5	4	4
6	15	2, 3
7	15	5, 6
8	19	5

Ứng dụng: PERT

Bài toán PERT: Giả sử thời điểm bắt đầu tiến hành thi công công trình là 0. Hãy tìm tiến độ thi công công trình (chỉ rõ mỗi công đoạn phải được bắt đầu thực hiện vào thời điểm nào) để cho công trình được hoàn thành xong trong thời điểm sớm nhất có thể được.

Thuật toán PERT

Ta có thể xây dựng đồ thị có hướng n đỉnh biểu diễn ràng buộc về trình tự thực hiệc các công đoạn như sau:

- Mỗi đỉnh của đồ thị tương ứng với một công đoạn.
- Nếu công đoạn i phải được thực hiện trước công đoạn j thì trên đồ thị có cung (i,j), trọng số trên cung này được gán bằng $t[i]$
- Thêm vào đồ thị 2 đỉnh 0 và n+1 tương ứng với hai sự kiện đặc biệt:
  - đỉnh số 0 tương ứng với công đoạn *Lễ khởi công*, nó phải được thực hiện trước tất cả các công đoạn khác, và
  - đỉnh n+1 tương ứng với công đoạn *Cắt băng khánh thành công trình,* nó phải thực hiện sau tất cả các công đoạn,
  - với $t[0] = t[n+1] = 0$ (trên thực tế chỉ cần nối đỉnh 0 với tất cả các đỉnh có bán bậc vào bằng 0 và nối tất cả các đỉnh có bán bậc ra bằng 0 với đỉnh n+1).

Gọi đồ thị thu được là G.

Công đoạn	1	2	3	4	5	6	7	8
t[i]	15	30	80	45	4	15	15	19
Các công đoạn phải hoàn thành trước nó	Ko có	1	Ko có	2, 3	4	2, 3	5, 6	5

Mỗi đỉnh của đồ thị tương ứng với một công đoạn.

Nếu công đoạn *i* phải được thực hiện trước công đoạn *j* thì trên đồ thị có cung (*i,j*), trọng số trên cung này được gán bằng *t*[*i*]

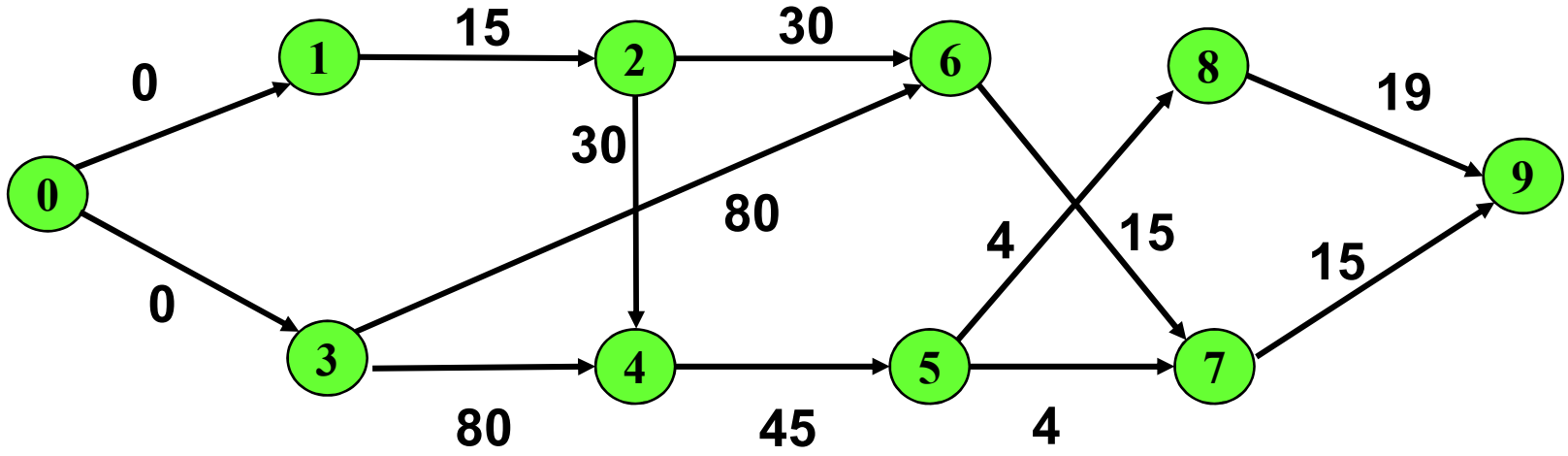
Thêm vào đồ thị 2 đỉnh 0 và *n*+1 tương ứng với hai sự kiện đặc biệt:

đỉnh số 0 tương ứng với công đoạn *Lễ khởi công*, nó phải được thực hiện trước tất cả các công đoạn khác, và

đỉnh *n*+1 tương ứng với công đoạn *Cắt băng khánh thành công trình,* nó phải thực hiện sau tất cả các công đoạn,

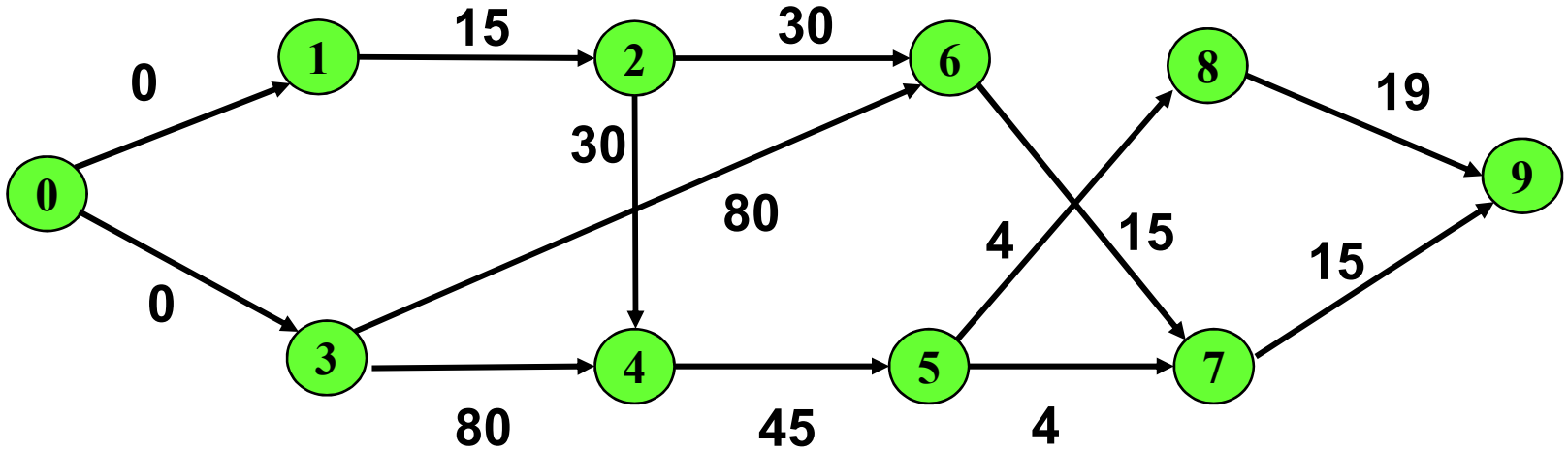
với *t*[0] = *t*[*n*+1] = 0 (tức là: nối đỉnh 0 với tất cả các đỉnh có bán bậc vào bằng 0 và nối tất cả các đỉnh có bán bậc ra bằng 0 với đỉnh *n*+1).

Các đỉnh không có công đoạn nào phải hoàn thành trước nó



Bài toán PERT: Giả sử thời điểm bắt đầu tiến hành thi công công trình là 0. Hãy tìm tiến độ thi công công trình (chỉ rõ mỗi công đoạn phải được bắt đầu thực hiện vào thời điểm nào) để cho công trình được hoàn thành xong trong thời điểm sớm nhất có thể

Công đoạn	1	2	3	4	5	6	7	8
t[i]	15	30	80	45	4	15	15	19
Các công đoạn phải hoàn thành trước nó	Ko có	1	Ko có	2, 3	4	2, 3	5, 6	5



Bài toán đặt ra dẫn về bài toán tìm đường đi dài nhất từ đỉnh 0 đến tất cả các đỉnh còn lại trên đồ thị G.

Thuật toán PERT

- Do đồ thị G không chứa chu trình, nên để giải bài toán đặt ra có thể áp dụng thuật toán Critical_Path trong đó chỉ cần đổi toán tử ***min*** thành toán tử ***max***.
- Kết thúc thuật toán, ta thu được $d[v]$ là độ dài đường đi dài nhất từ đỉnh 0 đến đỉnh v.
- Khi đó $d[v]$ cho ta thời điểm sớm nhất có thể bắt đầu thực hiện công đoạn v

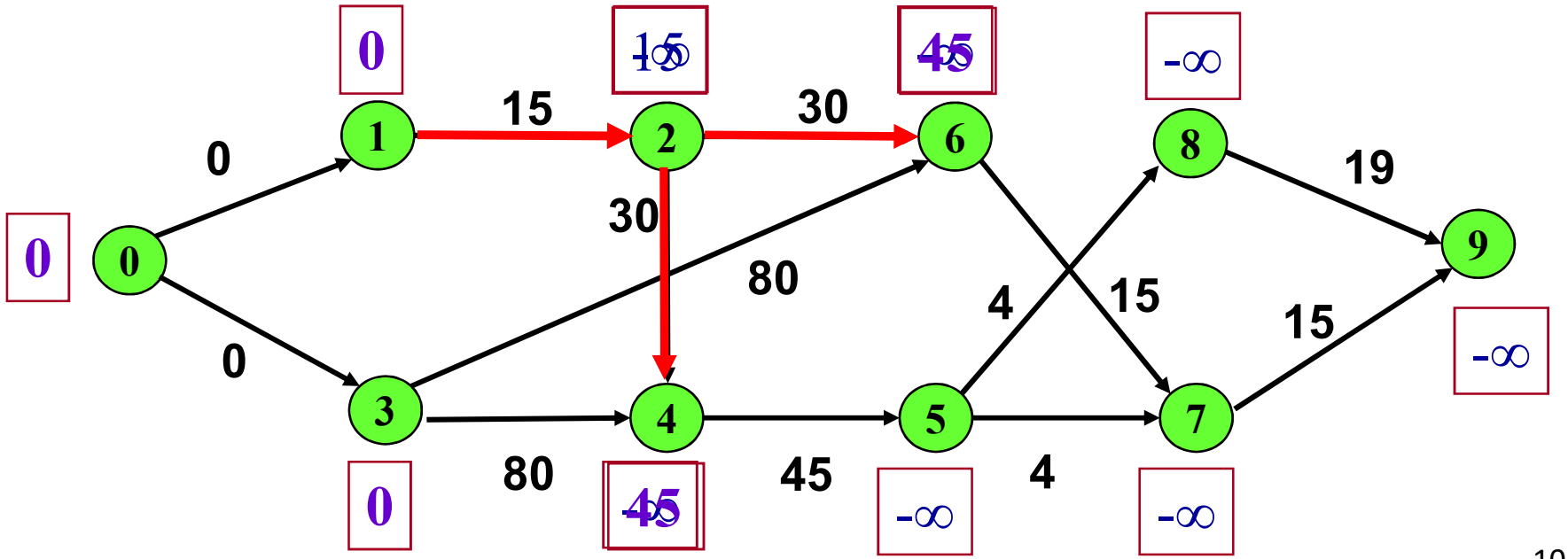
→$d[n+1]$ là thời điểm sớm nhất có thể cắt băng khánh thành, tức là thời điểm sớm nhất có thể hoàn thành toàn bộ công trình.

```
void  Critical_Path ( )
{
   d[v[1]] = 0;
   for  (j=2; j<=n;j++) d[v[j]] = -∞;
   for  v[j] ∈ Ke[v[1]]
        d[v[j]] := w(v[1], v[j]) ;

   for  (j= 2; j <= n;j++)
        for  v ∈ Ke[v[j]]
             d[v] = max (d[v],  d[v[j]] + w(v[j], v) ) ;
}
```

Duyệt qua các đỉnh: 1 2 3 4 5 6 7 8 9

d[2] = max {-∞, 0+15} = 15

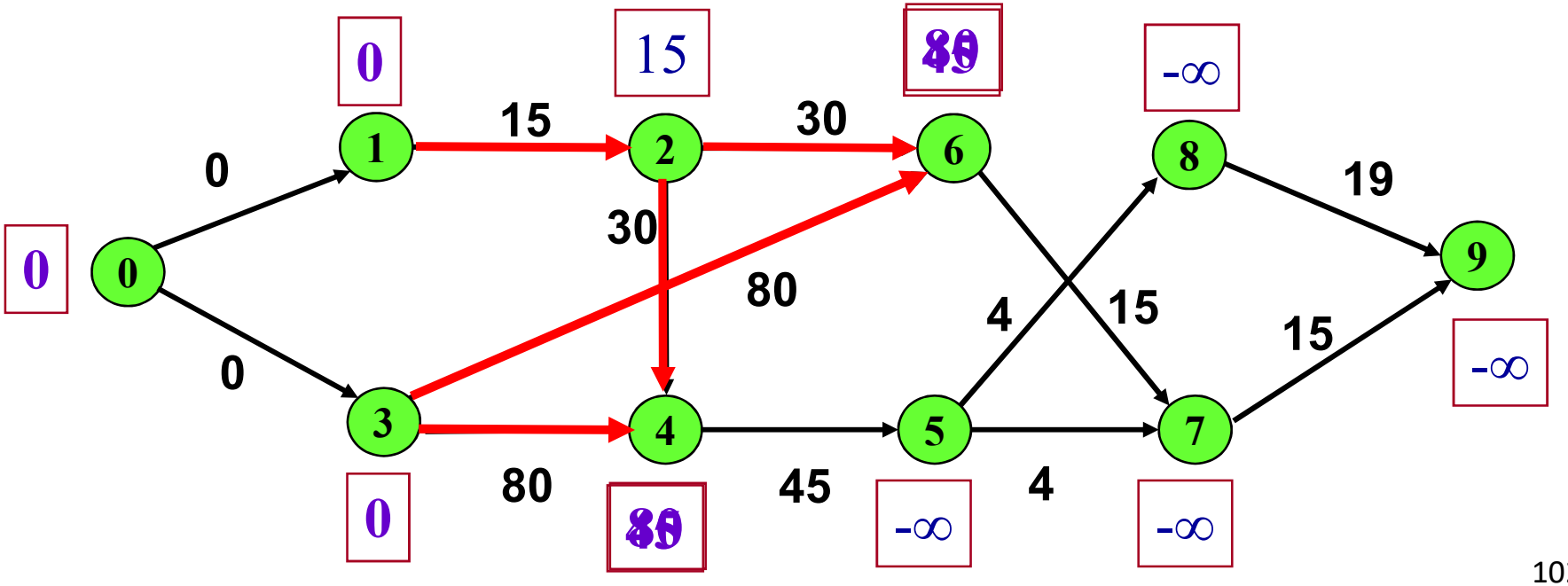
d[4] = max {-∞, 15+30} =45 d[6] = max {-∞, 15+30} =45



```
void  Critical_Path ( )
{
    d[v[1]] = 0;
    for  (j=2; j<=n;j++) d[v[j]] = -∞;
    for  v[j] ∈ Ke[v[1]]
         d[v[j]] := w(v[1], v[j]) ;

    for  (j= 2; j <= n;j++)
        for  v ∈ Ke[v[j]]
             d[v] =  max (d[v],  d[v[j]] + w(v[j], v) ) ;
}
```

Duyệt qua các đỉnh: 1 2 3 4 5 6 7 8 9

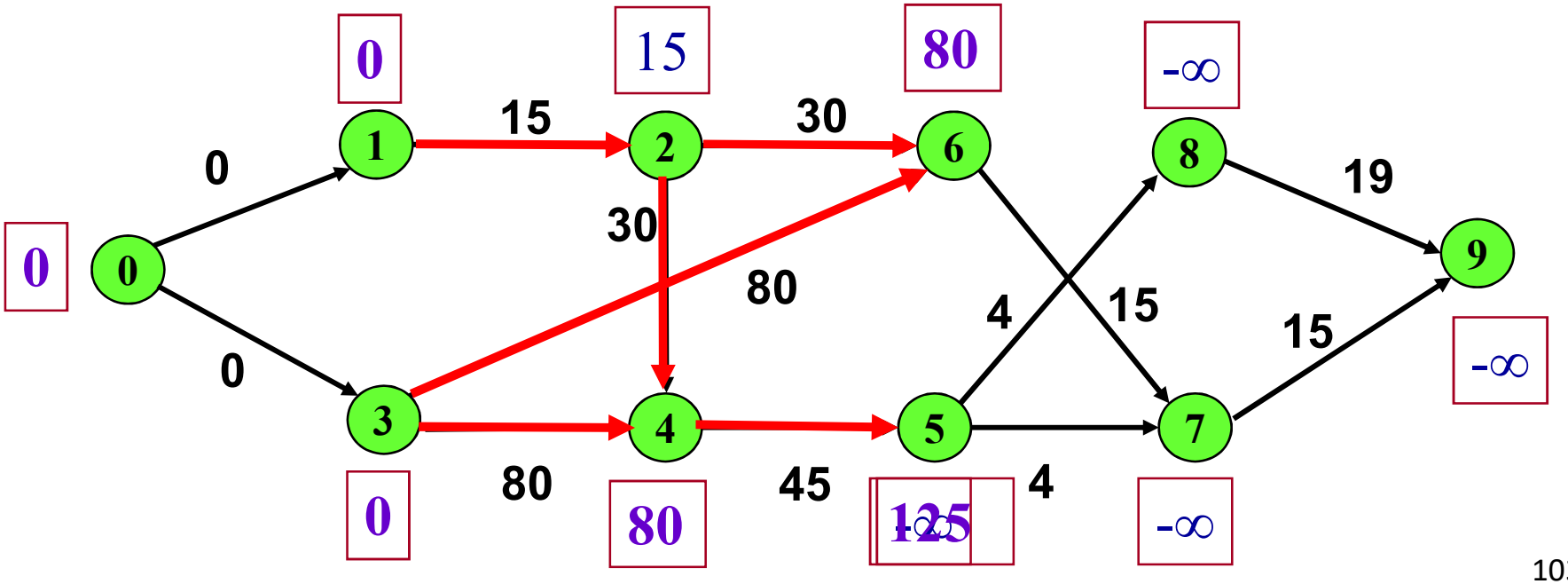
d[4] = max {45, 0+80} = 80 d[6] = max {45, 0+80} = 80



```
void   Critical_Path ( )
{
    d[v[1]] = 0;
    for   (j=2; j<=n;j++) d[v[j]] = -∞;
    for   v[j] ∈ Ke[v[1]]
          d[v[j]] := w(v[1], v[j]) ;

    for  (j= 2; j <= n;j++)
          for  v ∈ Ke[v[j]]
                d[v] =  max (d[v],  d[v[j]] + w(v[j], v) ) ;
}
```

Duyệt qua các đỉnh: 1 2 3 4 5 6 7 8 9

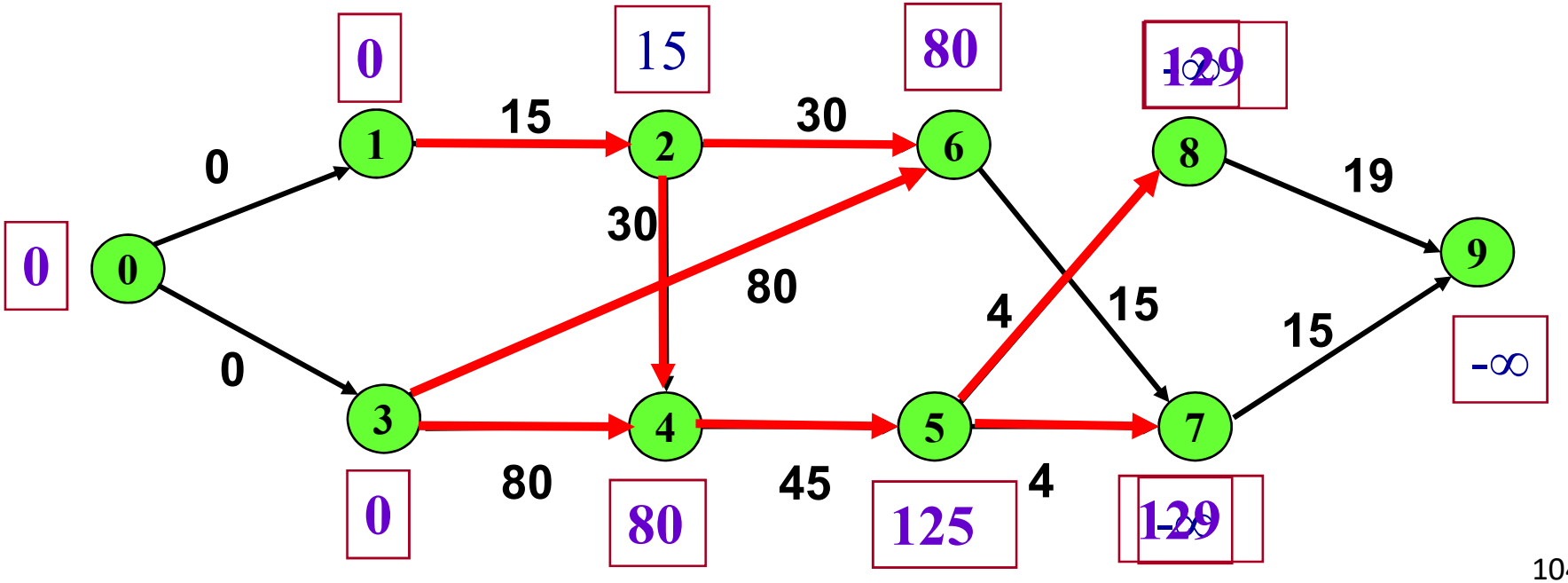
d[5] = max {-∞, 80+45} = 125



```
void   Critical_Path ( )
{
    d[v[1]] = 0;
    for   (j=2; j<=n;j++) d[v[j]] = -∞;
    for   v[j] ∈ Ke[v[1]]
          d[v[j]] := w(v[1], v[j]) ;

    for  (j= 2; j <= n;j++)
         for  v ∈ Ke[v[j]]
              d[v] =  max (d[v],  d[v[j]] + w(v[j], v) ) ;
}
```

Duyệt qua các đỉnh: 1 2 3 4 5 6 7 8 9

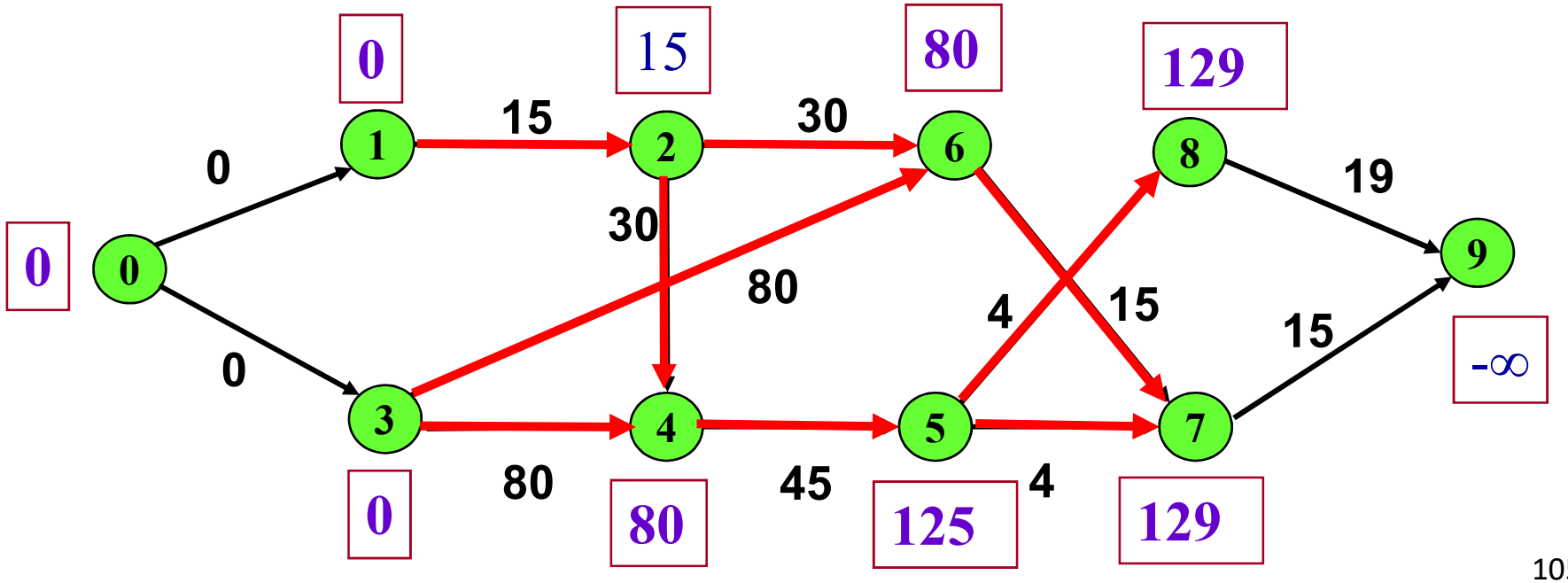
d[7] = max {-∞, 125+4} = 129 d[8] = max {-∞, 125+4} = 129



```
void   Critical_Path ( )
{
    d[v[1]] = 0;
    for   (j=2; j<=n;j++) d[v[j]] = -∞;
    for   v[j] ∈ Ke[v[1]]
          d[v[j]] := w(v[1], v[j]) ;

    for  (j= 2; j <= n;j++)
          for  v ∈ Ke[v[j]]
                d[v] =  max (d[v],  d[v[j]] + w(v[j], v) ) ;
}
```

Duyệt qua các đỉnh: 1 2 3 4 5 6 7 8 9

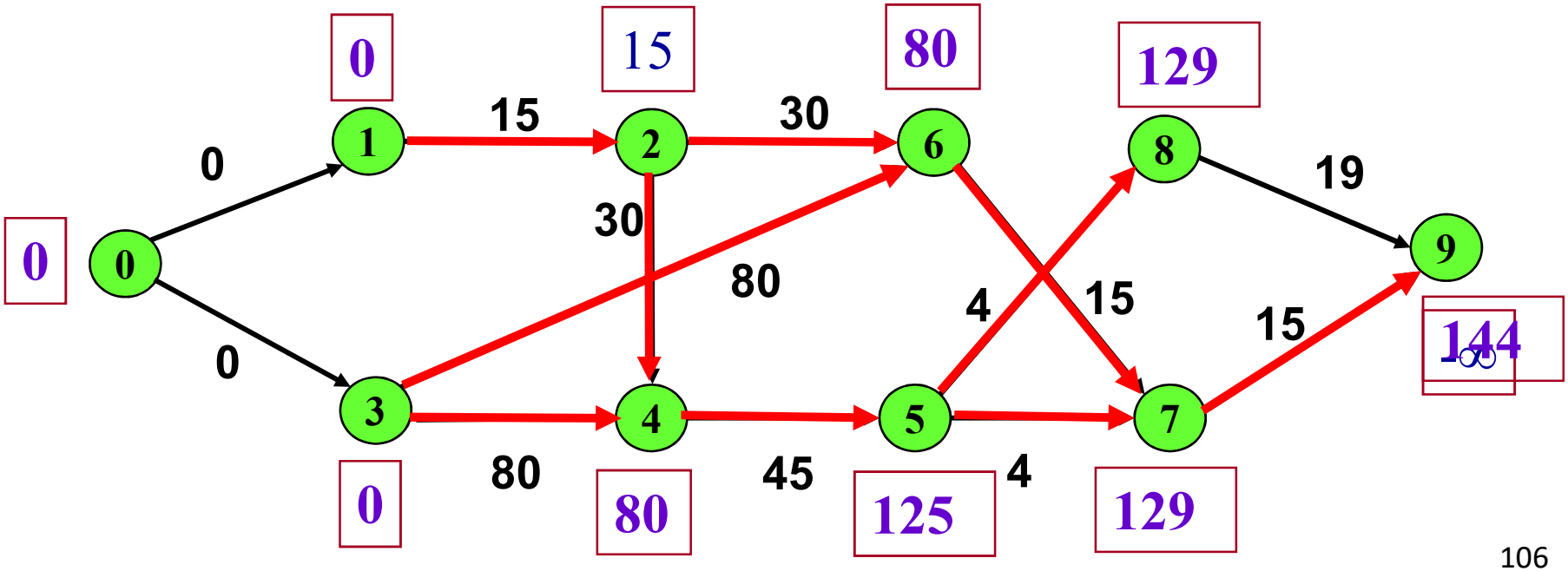
d[7] = max {129, 80+15} = 129



```
void   Critical_Path ( )
{
    d[v[1]] = 0;
    for   (j=2; j<=n;j++) d[v[j]] = -∞;
    for   v[j] ∈ Ke[v[1]]
          d[v[j]] := w(v[1], v[j]) ;

    for  (j= 2; j <= n;j++)
         for  v ∈ Ke[v[j]]
              d[v] =  max (d[v],  d[v[j]] + w(v[j], v) ) ;
}
```

Duyệt qua các đỉnh: 1 2 3 4 5 6 7 8 9

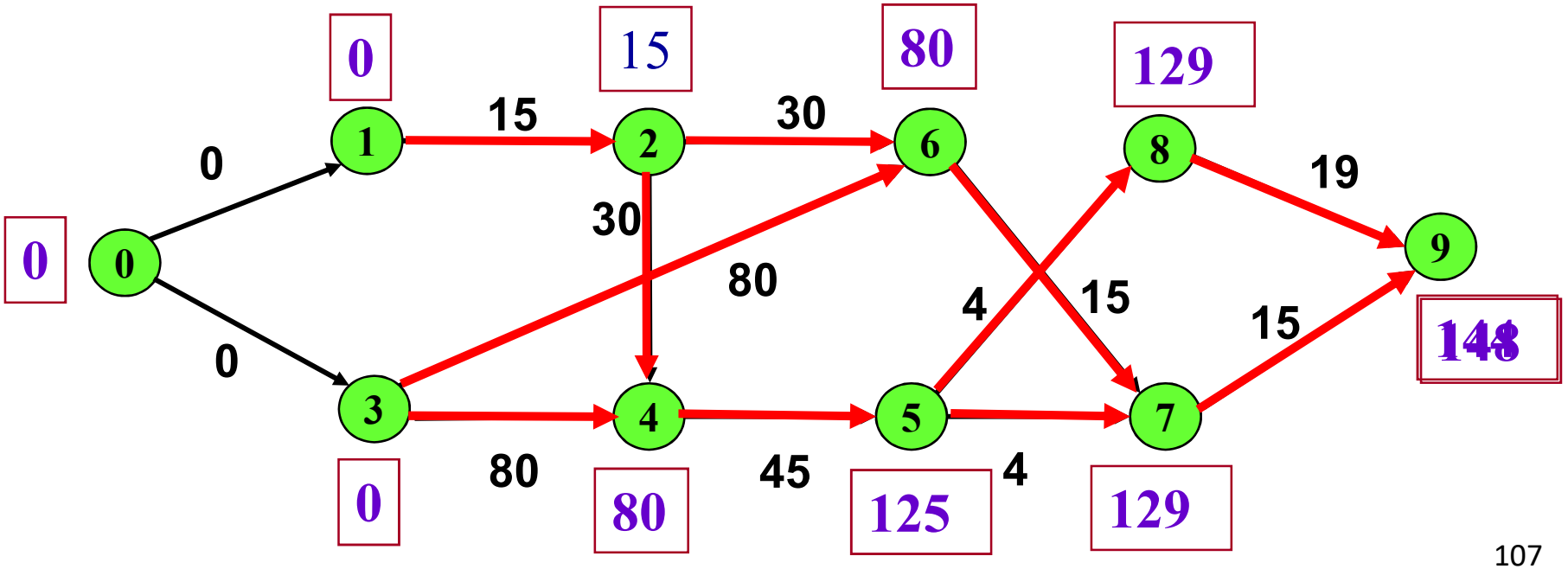
d[9] = max {-∞, 129+15} = 144



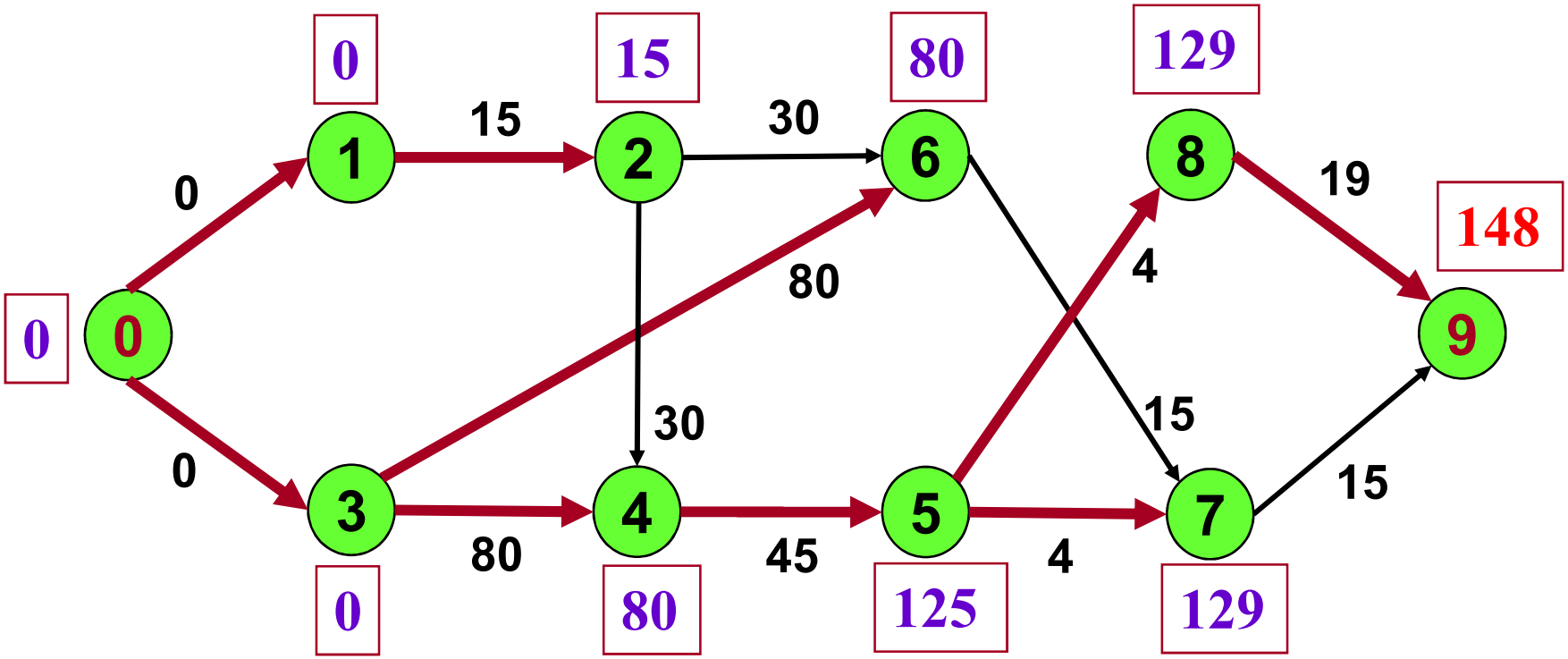
```
void   Critical_Path ( )
{
    d[v[1]] = 0;
    for   (j=2; j<=n;j++) d[v[j]] = -∞;
    for   v[j] ∈ Ke[v[1]]
          d[v[j]] := w(v[1], v[j]) ;

    for  (j= 2; j <= n;j++)
         for  v ∈ Ke[v[j]]
              d[v] =  max (d[v],  d[v[j]] + w(v[j], v) ) ;
}
```

Duyệt qua các đỉnh: 1 2 3 4 5 6 7 8 9

d[9] = max {144, 129+19} = 148



Bài toán PERT: Giả sử thời điểm bắt đầu tiến hành thi công công trình là 0. Hãy tìm tiến độ thi công công trình (chỉ rõ mỗi công đoạn phải được bắt đầu thực hiện vào thời điểm nào) để cho công trình được hoàn thành xong trong thời điểm sớm nhất có thể được.



Kết luận: Công trình hoàn thành sớm nhất là sau 148 tuần

Tiến độ thi công công trình:???

Nội dung chi tiết

5.1. Bài toán đường đi ngắn nhất (ĐĐNN)

5.2. Tính chất của ĐĐNN, Giảm cận trên

5.3. Thuật toán Bellman-Ford

5.4. Thuật toán Dijkstra

5.5. Đường đi ngắn nhất trong đồ thị không có chu trình

Đường đi ngắn nhất từ 1 đỉnh đến mọi đỉnh

5.6. Đường đi ngắn nhất giữa mọi cặp đỉnh - Thuật toán Floyd-Warshall

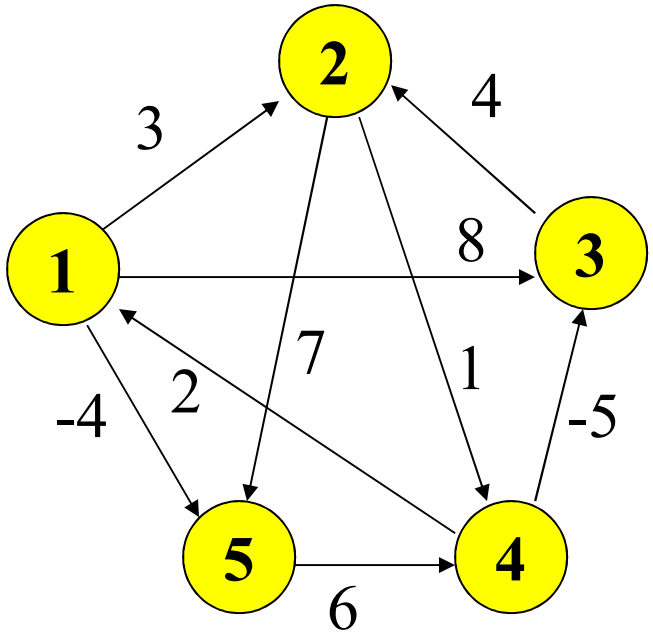
ĐƯỜNG ĐI NGẮN NHẤT GIỮA MỌI CẶP ĐỈNH
All-Pairs Shortest Paths

Đường đi ngắn nhất giữa mọi cặp đỉnh

Bài toán Cho đồ thị $G = (V, E)$, với trọng số trên cạnh e là $w(e)$, đối với mỗi cặp đỉnh u, v trong V, tìm đường đi ngắn nhất từ u đến v.

- Đầu vào: *ma trận trọng số*.
- Đầu ra *ma trận*: phần tử ở dòng u cột v là độ dài đường đi ngắn nhất từ u đến v.
- Cho phép có trọng số âm
- **Giả thiết: Đồ thị không có chu trình âm.**

Ví dụ



Đầu vào

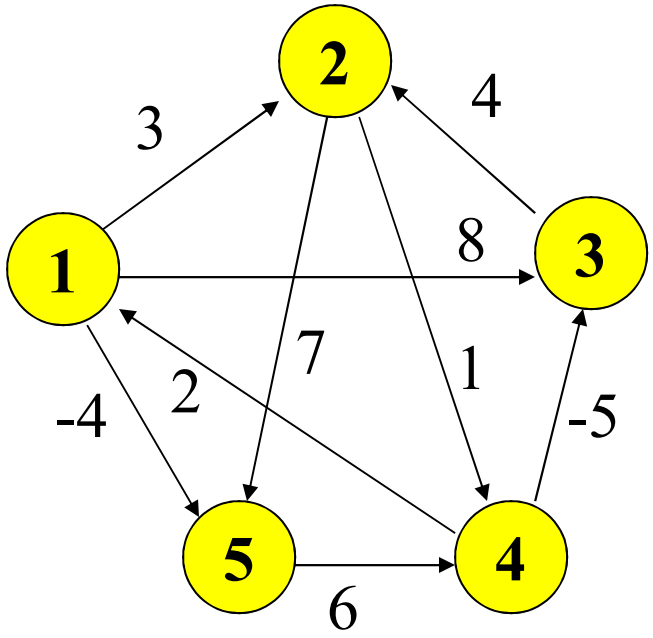
Ma trận trọng số $W_{nxn} = (w)_{ij}$ với

$$w_{ij} = \begin{cases} 0 & \text{nếu } i = j \\ w(i,j) & \text{nếu } i \neq \text{j } \& \ (i,j) \in E \\ \infty & \text{còn lại} \end{cases}$$

$$W_{5x5} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Đầu ra

Ma trận: phần tử ở dòng *u* cột *v* là độ dài đường đi ngắn nhất từ *u* đến *v*.



Đường đi từ 1 đến 2 : 1- 5 - 4 - 3 - 2

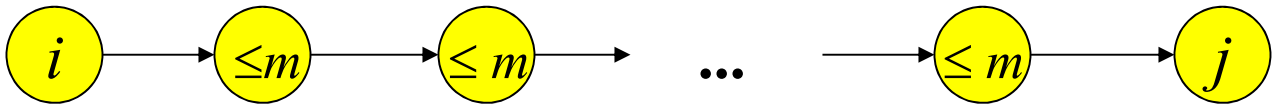
$= -4 + 6 - 5 + 4$

$$\begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Đường đi từ 5 đến 1 : 5 - 4 - 1

Thuật toán Floyd-Warshall

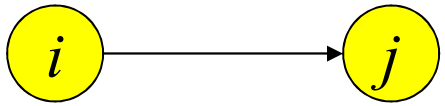
$d_{ij}^{(m)}$ = độ dài đường đi ngắn nhất từ i đến j sử dụng các đỉnh trung gian trong tập đỉnh $\{1, 2, \ldots, m\}$.



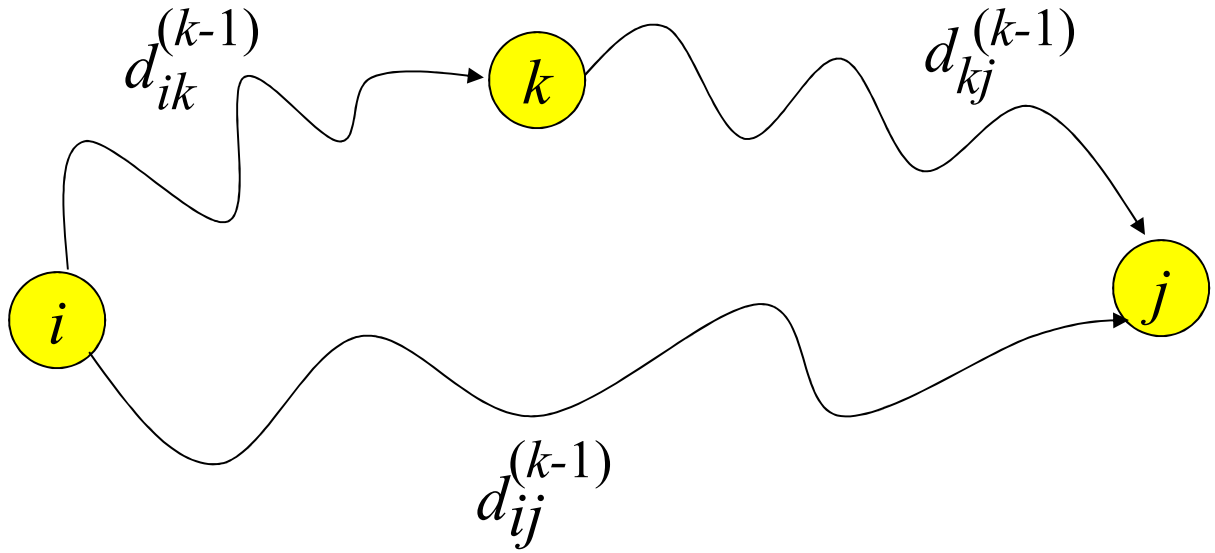
Đồ thị có n đỉnh $\{1,2,..,n\}$ → độ dài đường đi ngắn nhất từ i đến j là $d_{ij}^{(n)}$

Công thức đệ qui tính $d^{(h)}$

- $d_{ij}^{(0)} = w_{ij}$



- $d_{ij}^{(k)} = \min (d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$ nếu $k \geq 1$



Thuật toán Floyd-Warshall

void Floyd-Warshall(n, W)
{
    $D^{(0)} \leftarrow W$
    for (k=1; k<=n; k++)
        for (i=1; i<=n; i++)
            for (j =1; j<=n; j++)
                $d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
    return $D^{(n)}$;
}

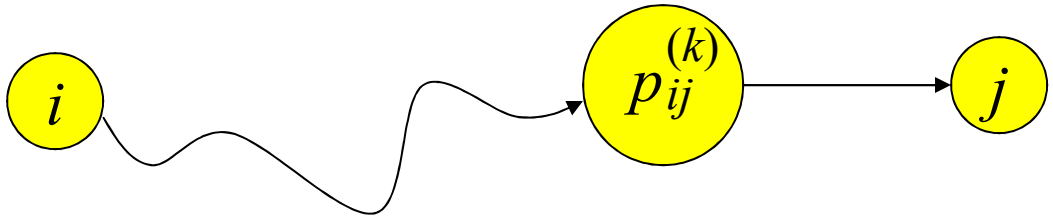
Đường đi chỉ qua các đỉnh trung gian trong {1,.., k}

Mọi cặp đỉnh (i, j)

Thời gian tính $\Theta(n^3)$!

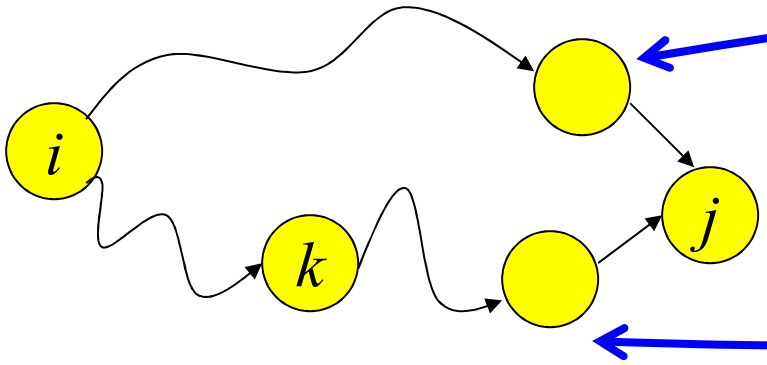
Xây dựng đường đi ngắn nhất

Predecessor matrix $P^{(k)} = (p_{ij}^{(k)})$:



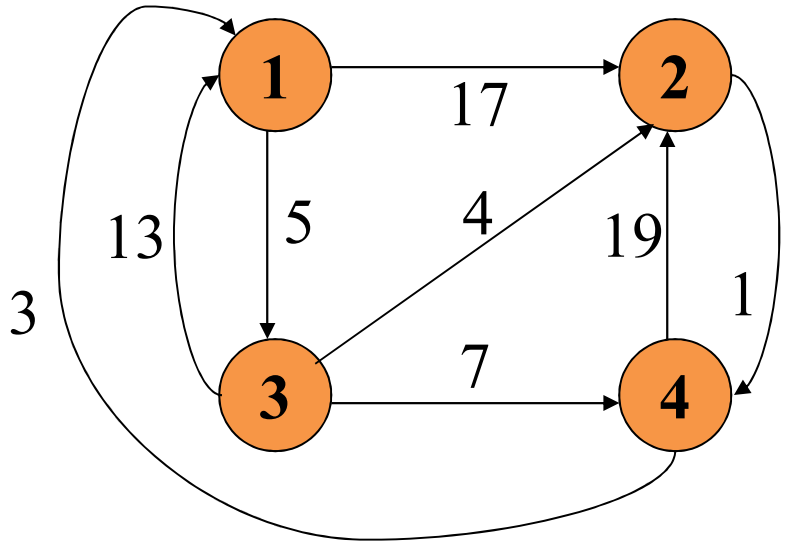
đường đi ngắn nhất từ i đến j chỉ qua các đỉnh trung gian trong $\{1, 2, \ldots, k\}$.

$$p_{ij}^{(0)} = \begin{cases} i, & \text{nếu } (i,j) \in E \\ \text{Nil}, & \text{nếu } (i,j) \notin E \end{cases}$$



$$p_{ij}^{(k)} = \begin{cases} p_{ij}^{(k-1)} & \text{nếu } d_{ij}^{(k-1)} \le d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ p_{kj}^{(k-1)} & \text{trái lại} \end{cases}$$

Ví dụ 1: Tìm đđnn giữa mọi cặp đỉnh

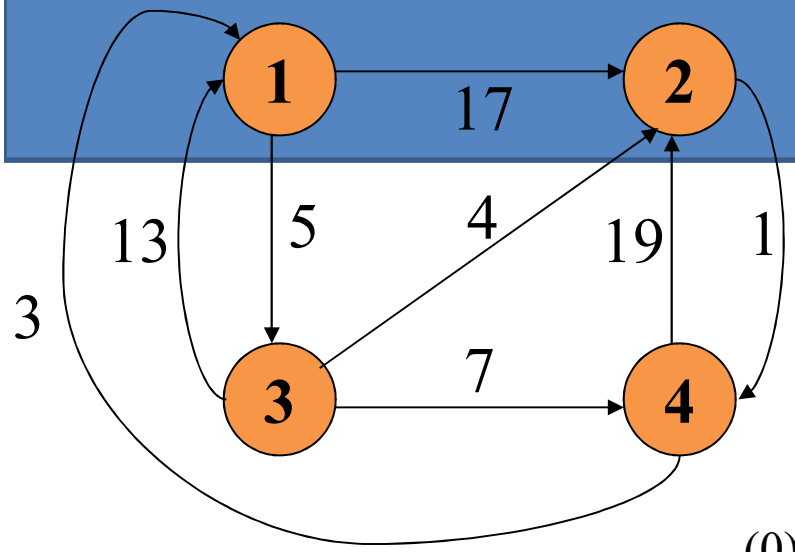


Đầu vào

Ma trận trọng số $W_{nxn} = (w)_{ij}$ với

$$w_{ij} = \begin{cases} 0 & \text{nếu } i = j \\ w(i,j) & \text{nếu } i \neq \text{j } \& \ (i,j) \in E \\ \infty & \text{còn lại} \end{cases}$$

$$W_{4x4} = \begin{pmatrix} 0 & 17 & 5 & \infty \\ \infty & 0 & \infty & 1 \\ 13 & 4 & 0 & 7 \\ 3 & 19 & \infty & 0 \end{pmatrix}$$



$D^{(0)} \leftarrow W$

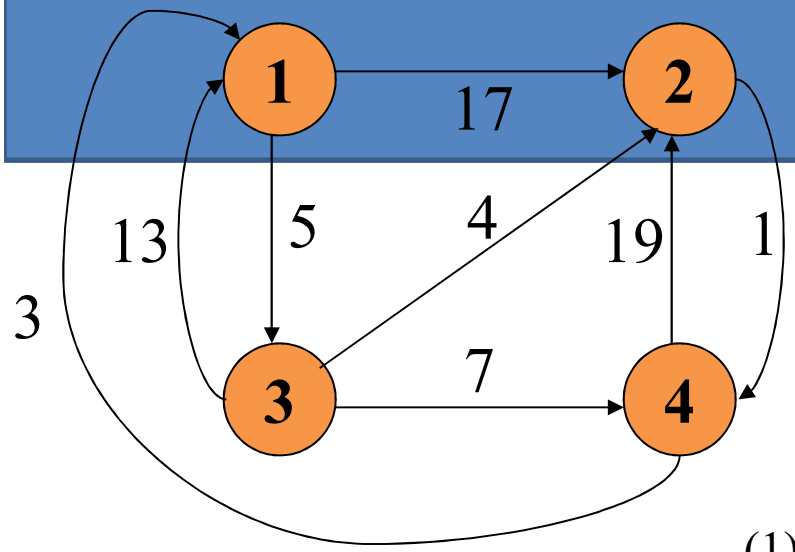
$$p_{ij}^{(0)} = \begin{cases} i, & \text{nếu } (i,j) \in E \\ \text{Nil}, & \text{nếu } (i,j) \notin E \end{cases}$$

$$D^{(0)} \begin{pmatrix} 0 & 17 & 5 & \infty \\ \infty & 0 & \infty & 1 \\ 13 & 4 & 0 & 7 \\ 3 & 19 & \infty & 0 \end{pmatrix} \quad P^{(0)} \begin{pmatrix} \text{Nil} & 1 & 1 & \text{Nil} \\ \text{Nil} & \text{Nil} & \text{Nil} & 2 \\ 3 & 3 & \text{Nil} & 3 \\ 4 & 4 & \text{Nil} & \text{Nil} \end{pmatrix}$$

$$d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$$

Có thể sử dụng **1** là đỉnh trung gian:

$$D^{(1)} \begin{pmatrix} 0 & 17 & 5 & \infty \\ \infty & 0 & \infty & 1 \\ 13 & 4 & 0 & 7 \\ 3 & 19 & 8 & 0 \end{pmatrix} \quad P^{(1)} \begin{pmatrix} \text{Nil} & 1 & 1 & \text{Nil} \\ \text{Nil} & \text{Nil} & \text{Nil} & 2 \\ 3 & 3 & \text{Nil} & 3 \\ 4 & 4 & 1 & \text{Nil} \end{pmatrix}$$



$D^{(0)} \leftarrow W$

$$p_{ij}^{(0)} = \begin{cases} i, & \text{nếu } (i,j) \in E \\ \text{Nil}, & \text{nếu } (i,j) \notin E \end{cases}$$

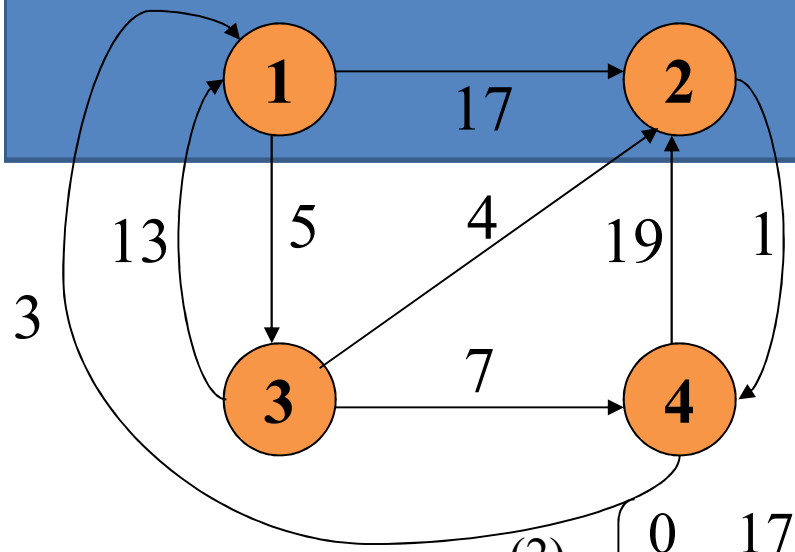
$$D^{(1)} \begin{pmatrix} 0 & 17 & 5 & \infty \\ \infty & 0 & \infty & 1 \\ 13 & 4 & 0 & 7 \\ 3 & 19 & \mathbf{8} & 0 \end{pmatrix} \quad P^{(1)} \begin{pmatrix} \text{Nil} & 1 & 1 & \text{Nil} \\ \text{Nil} & \text{Nil} & \text{Nil} & 2 \\ 3 & 3 & \text{Nil} & 3 \\ 4 & 4 & \mathbf{1} & \text{Nil} \end{pmatrix}$$

$$d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$$

Có thể sử dụng 2 là đỉnh trung gian:

$$D^{(2)} \begin{pmatrix} 0 & 17 & 5 & \mathbf{18} \\ \infty & 0 & \infty & 1 \\ 13 & 4 & 0 & \mathbf{5} \\ 3 & 19 & 8 & 0 \end{pmatrix} \quad P^{(2)} \begin{pmatrix} \text{Nil} & 1 & 1 & 2 \\ \text{Nil} & \text{Nil} & \text{Nil} & 2 \\ 3 & 3 & \text{Nil} & 2 \\ 4 & 4 & \mathbf{1} & \text{Nil} \end{pmatrix}$$

Ddnn từ 3 đến 4 đi qua đỉnh 2: 3….2 …….4



$D^{(0)} \leftarrow W$

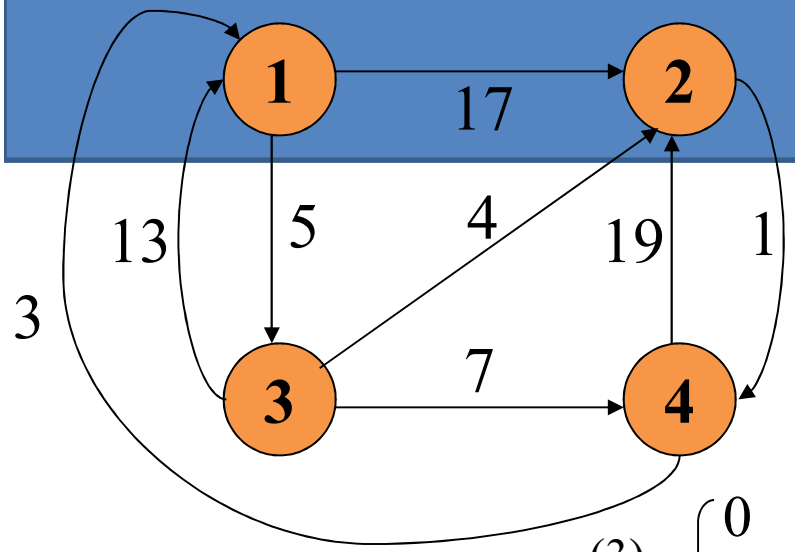
$$p_{ij}^{(0)} = \begin{cases} i, & \text{nếu } (i,j) \in E \\ \text{Nil}, & \text{nếu } (i,j) \notin E \end{cases}$$

$$D^{(2)} \begin{pmatrix} 0 & 17 & 5 & 18 \\ \infty & 0 & \infty & 1 \\ 13 & 4 & 0 & 5 \\ 3 & 19 & 8 & 0 \end{pmatrix} \qquad P^{(2)} \begin{pmatrix} \text{Nil} & 1 & 1 & \text{Nil} \\ \text{Nil} & \text{Nil} & \text{Nil} & 2 \\ 3 & 3 & \text{Nil} & 2 \\ 4 & 4 & 1 & \text{Nil} \end{pmatrix}$$

$$d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$$

Có thể sử dụng 3 là đỉnh trung gian:

$$D^{(3)} \begin{pmatrix} 0 & \mathbf{9} & 5 & \mathbf{10} \\ \infty & 0 & \infty & 1 \\ 13 & 4 & 0 & 5 \\ 3 & \mathbf{12} & 8 & 0 \end{pmatrix} \qquad P^{(3)} \begin{pmatrix} \text{Nil} & \mathbf{3} & 1 & \mathbf{2} \\ \text{Nil} & \text{Nil} & \text{Nil} & 2 \\ 3 & 3 & \text{Nil} & 2 \\ 4 & \mathbf{3} & 1 & \text{Nil} \end{pmatrix}$$



$D^{(0)} \leftarrow W$

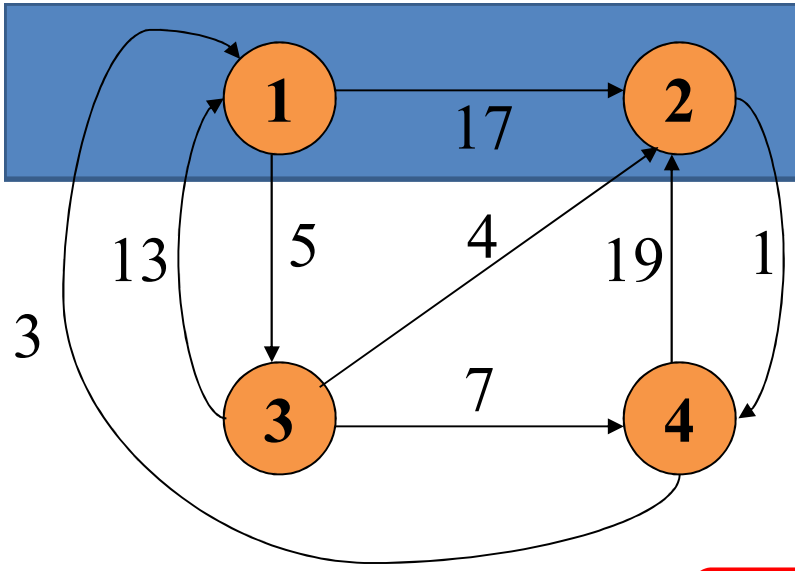
$$p_{ij}^{(0)} = \begin{cases} i, & \text{nếu } (i,j) \in E \\ \text{Nil}, & \text{nếu } (i,j) \notin E \end{cases}$$

$$D^{(3)} \begin{pmatrix} 0 & 9 & 5 & 10 \\ \infty & 0 & \infty & 1 \\ 13 & 4 & 0 & 5 \\ 3 & 12 & 8 & 0 \end{pmatrix} \qquad P^{(3)} \begin{pmatrix} \text{Nil} & 3 & 1 & 2 \\ \text{Nil} & \text{Nil} & \text{Nil} & 2 \\ 3 & 3 & \text{Nil} & 2 \\ 4 & 3 & 1 & \text{Nil} \end{pmatrix}$$

$$d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$$

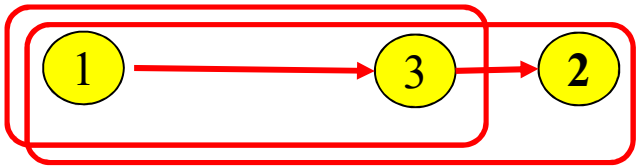
Có thể sử dụng (4) là đỉnh trung gian:

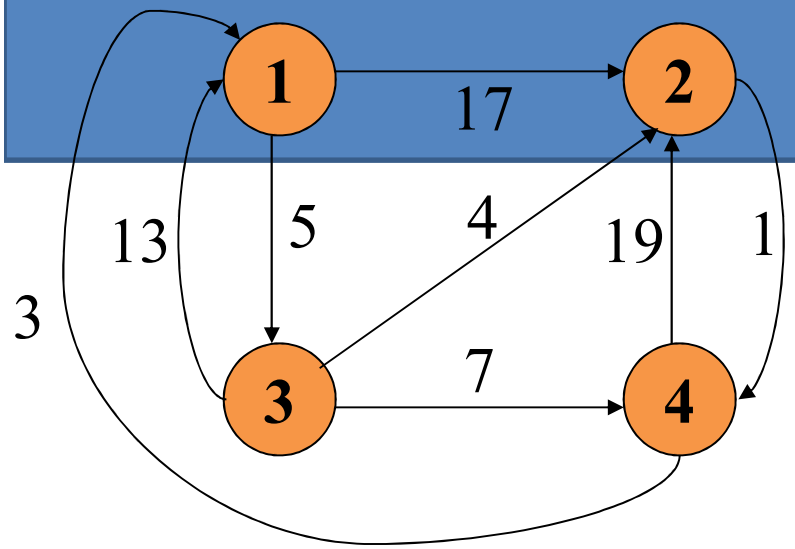
$$D^{(4)} \begin{pmatrix} 0 & 9 & 5 & 10 \\ 4 & 0 & 9 & 1 \\ 8 & 4 & 0 & 5 \\ 3 & 12 & 8 & 0 \end{pmatrix} \qquad P^{(4)} \begin{pmatrix} \text{Nil} & 3 & 1 & 2 \\ 4 & \text{Nil} & 1 & 2 \\ 4 & 3 & \text{Nil} & 2 \\ 4 & 3 & 1 & \text{Nil} \end{pmatrix}$$



$$D^{(4)} = \begin{pmatrix} 0 & 9 & 5 & 10 \\ 4 & 0 & 9 & 1 \\ 8 & 4 & 0 & 5 \\ 3 & 12 & 8 & 0 \end{pmatrix} \qquad P^{(4)} = \begin{pmatrix} \text{Nil} & 3 & 1 & 2 \\ 4 & \text{Nil} & 1 & 2 \\ 4 & 3 & \text{Nil} & 2 \\ 4 & 3 & 1 & \text{Nil} \end{pmatrix}$$

Đđnn từ đỉnh 1 đến đỉnh 2:
Độ dài = 9

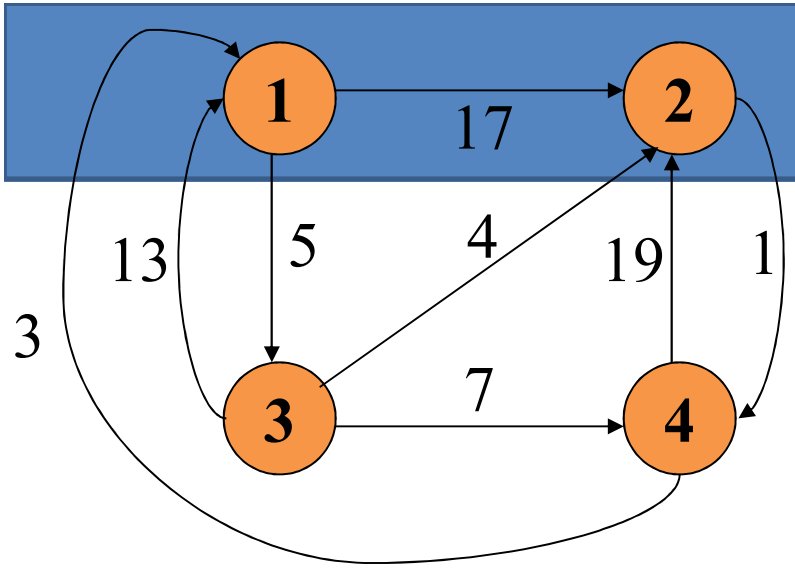




$$D^{(4)} = \begin{pmatrix} 0 & 9 & 5 & 10 \\ 4 & 0 & 9 & 1 \\ 8 & 4 & 0 & 5 \\ 3 & 12 & 8 & 0 \end{pmatrix} \qquad P^{(4)} = \begin{pmatrix} Nil & 3 & 1 & 2 \\ 4 & Nil & 1 & 2 \\ 4 & 3 & Nil & 2 \\ 4 & 3 & 1 & Nil \end{pmatrix}$$

Đđnn từ đỉnh 1 đến đỉnh 2:
Độ dài = 9

Đđnn từ đỉnh 1 đến đỉnh 3:
Độ dài = 5

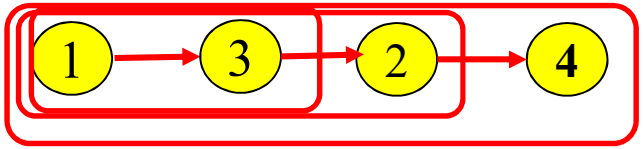


$$D^{(4)} = \begin{pmatrix} 0 & 9 & 5 & 10 \\ 4 & 0 & 9 & 1 \\ 8 & 4 & 0 & 5 \\ 3 & 12 & 8 & 0 \end{pmatrix} \qquad P^{(4)} = \begin{pmatrix} \text{Nil} & 3 & 1 & 2 \\ 4 & \text{Nil} & 1 & 2 \\ 4 & 3 & \text{Nil} & 2 \\ 4 & 3 & 1 & \text{Nil} \end{pmatrix}$$

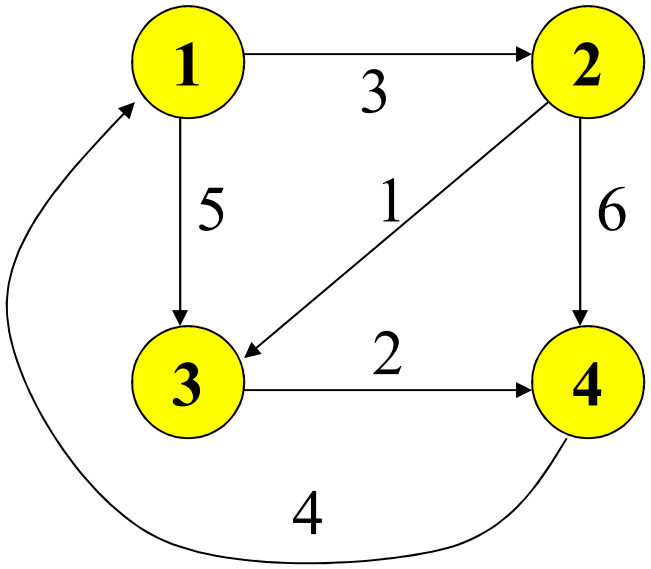
Đđnn từ đỉnh 1 đến đỉnh 2:
Độ dài = 9

Đđnn từ đỉnh 1 đến đỉnh 3:
Độ dài = 5

Đđnn từ đỉnh 1 đến đỉnh 4:
Độ dài = 10



Ví dụ 2: Tìm đđnn giữa mọi cặp đỉnh



$$D^{(0)} \leftarrow W$$

$$p_{ij}^{(0)} = \begin{cases} i, & \text{nếu } (i,j) \in E \\ \text{Nil}, & \text{nếu } (i,j) \notin E \end{cases}$$

$$D^{(0)} \begin{pmatrix} 0 & 3 & 5 & \infty \\ \infty & 0 & 1 & 6 \\ \infty & \infty & 0 & 2 \\ 4 & \infty & \infty & 0 \end{pmatrix} \qquad P^{(0)} \begin{pmatrix} \text{Nil} & 1 & 1 & \text{Nil} \\ \text{Nil} & \text{Nil} & 2 & 2 \\ \text{Nil} & \text{Nil} & \text{Nil} & 3 \\ 4 & \text{Nil} & \text{Nil} & \text{Nil} \end{pmatrix}$$

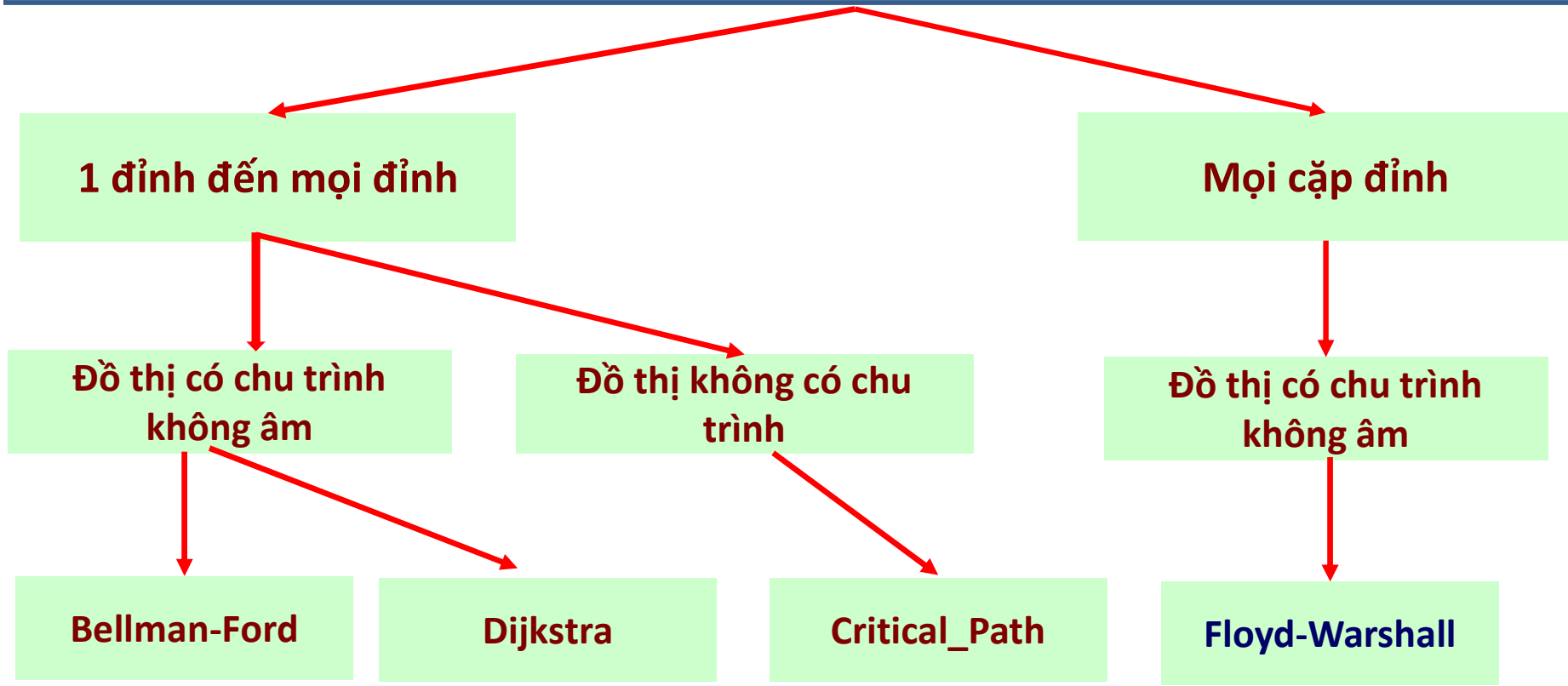
$$d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$$

Có thể sử dụng **1** là đỉnh trung gian:

$$D^{(1)} \begin{pmatrix} 0 & 3 & 5 & \infty \\ \infty & 0 & 1 & 6 \\ \infty & \infty & 0 & 2 \\ 4 & 7 & 9 & 0 \end{pmatrix} \qquad P^{(1)} \begin{pmatrix} \text{Nil} & 1 & 1 & \text{Nil} \\ \text{Nil} & \text{Nil} & 2 & 2 \\ \text{Nil} & \text{Nil} & \text{Nil} & 3 \\ 4 & 1 & 1 & \text{Nil} \end{pmatrix}$$

$= \min(\infty, 4+3)$ $\qquad$ $= \min(\infty, 4+5)$

Đường đi ngắn nhất



Trọng số trên cạnh: số thực tùy ý

Độ phức tạp: O(|V||E|)

Trọng số trên cạnh >=0

Độ phức tạp: $O(|V|^2)$

Trọng số trên cạnh: số thực tùy ý

Độ phức tạp: O(|E|)

Trọng số trên cạnh: số thực tùy ý

Độ phức tạp: $O(|V|^3)$