BÀI TOÁN ĐƯỜNG ĐI NGẮN NHẤT

Nội dung

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. Thuật toán Floyd-Warshal

5.1. Bài toán đường đi ngắn nhất

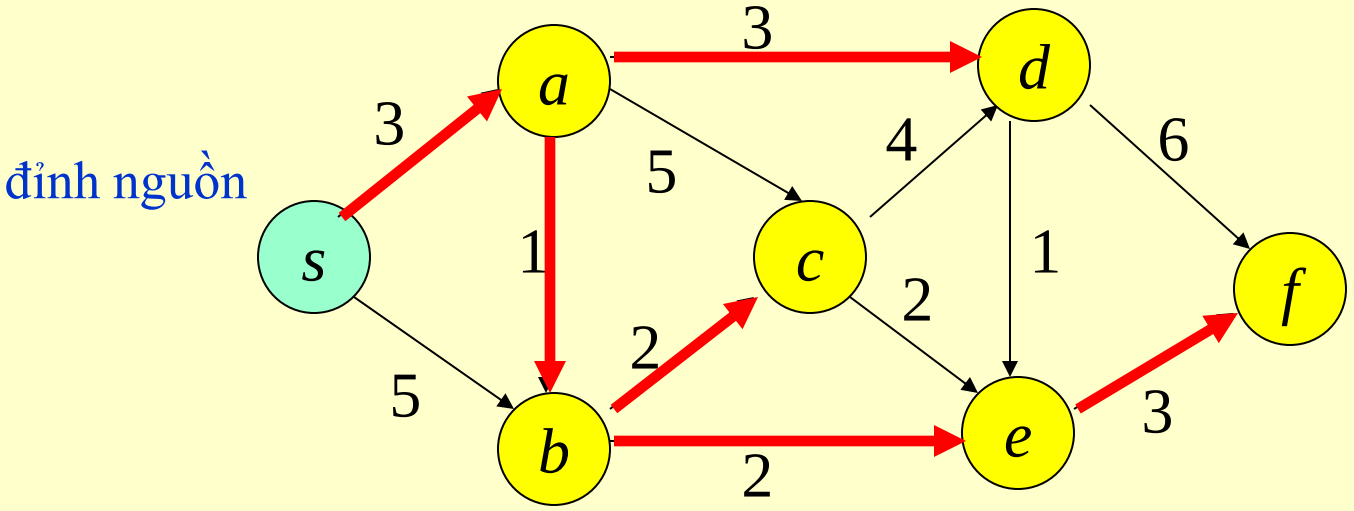
- Cho đơn đồ thị có hướng $G = (V,E)$ với hàm trọng số w: $E \to R$ ($w(e)$ được gọi là độ dài hay trọng số của cạnh e)
- Độ dài của đường đi $P = v_1 \to v_2 \to \ldots \to v_k$ là số

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

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

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?)
- Truyền thông (Telecommunications)
- Speech interpretation (best interpretation of a spoken sentence)
- Điều khiển robot (Robot path planning)
- Medical imaging
- Giải các bài toán phức tạp hơn trên mạng
- ...

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

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

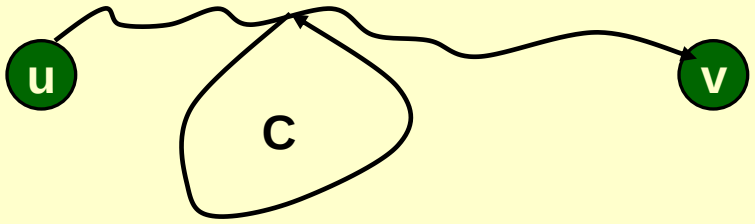
5.5. Đường đi ngắn nhất trong đồ thị không có chu trình

5.6. Thuật toán Floyd-Warshal

Các tính chất của ĐĐNN

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

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



w(C) ≥ 0

❖ ***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.*

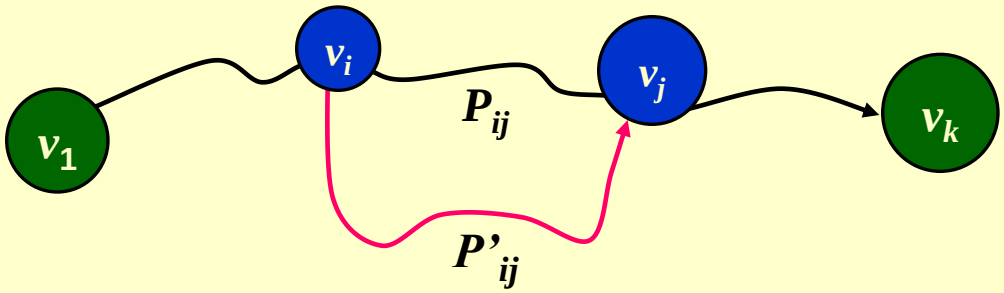
- *Như là hệ quả của tính chất 1*

Các tính chất của ĐĐNN

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)

CM. 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 $w(P') < w(P)$?!



Các tính chất của ĐĐNN

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 $\delta(s, v) \leq \delta(s, u) + w(u,v)$.

Đường đi ngắn nhất xuất phát từ một đỉnh
Single-Source Shortest Paths

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) .

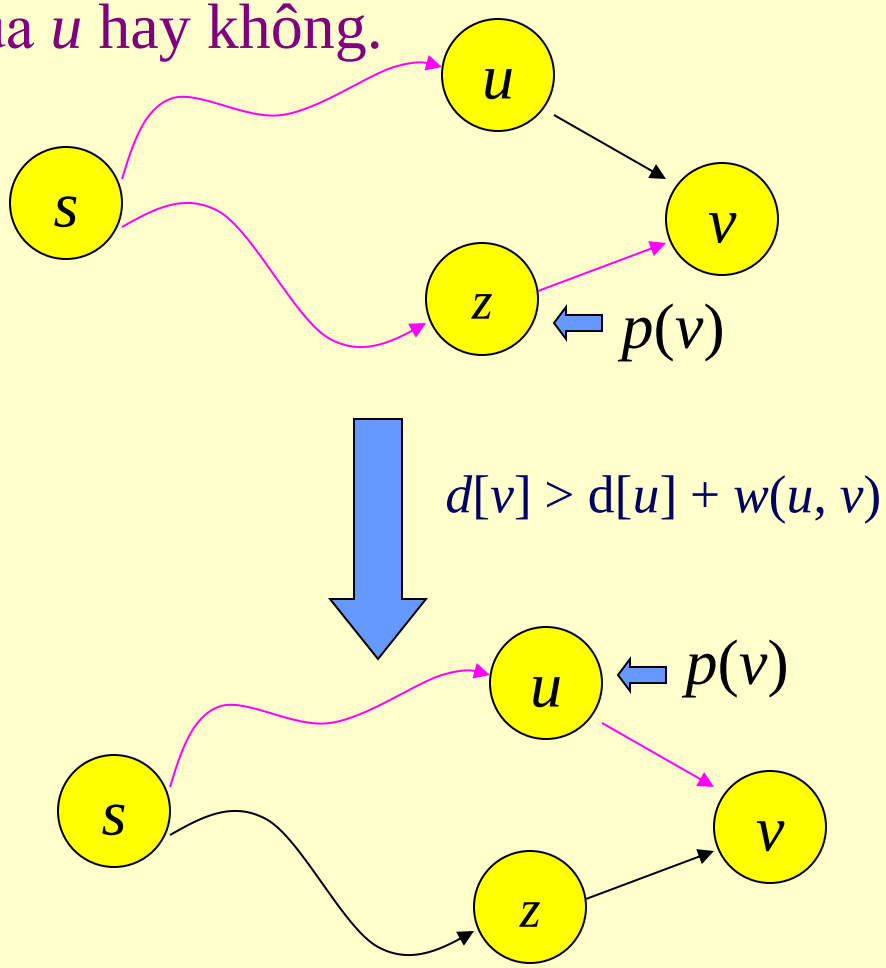
Khởi tạo (Initialization)

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

Giảm cận trên (Relaxation)

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)
        then 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

.

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

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-Warshal

Thuật toán Ford-Bellman

Richard Bellman
1920-1984

Lester R. Ford, Jr.
1927~

Thuật toán Ford-Bellman

- Thuật toán Ford - Bellman 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***
- $d[v] = \delta(s, v)$;
- *p[v] - đỉnh đi trước v trong đđnn từ s đến v.*

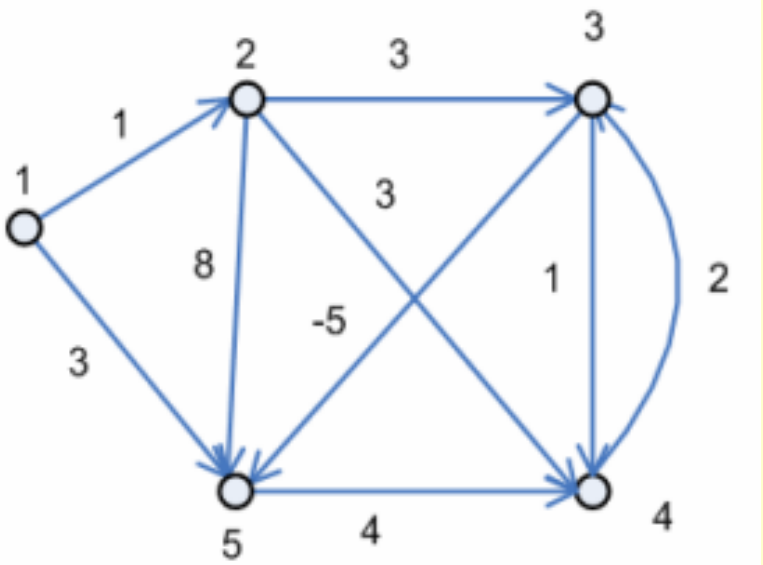
Mô tả thuật toán

```
procedure  Ford_Bellman;
begin
    for   v ∈ V  do  begin  (*  khởi tạo  *)
          d[v] := w[s,v] ; p[v]:=s;
    end;
    d[s]:=0; p[s]:=s;
    for  k := 1  to  n-2  do        (* n = |V|  *)
          for   v ∈ V \ {s}   do
                for   u ∈ V do
                      if   d[v] > d[u] + w[u,v]   then
                      begin  d[v] := d[u] + w[u,v] ;
                             p[v] := u ;
                      end;
end;
```

Nhận xét

- Tính đúng đắn của thuật toán có thể chứng minh trên cơ sở nguyên lý tối ưu của quy hoạch động.
- Độ phức tạp tính toán của thuật toán là $O(n^3)$.
- Có thể chấm dứt vòng lặp theo *k khi phát hiện trong quá trình thực hiện hai vòng lặp trong không có biến d[v] nào bị đổi giá trị. Việc này có thể xảy ra đối với* $k < n-2$, *và điều đó làm tăng hiệu quả của thuật toán trong việc giải các bài toán thực tế.*

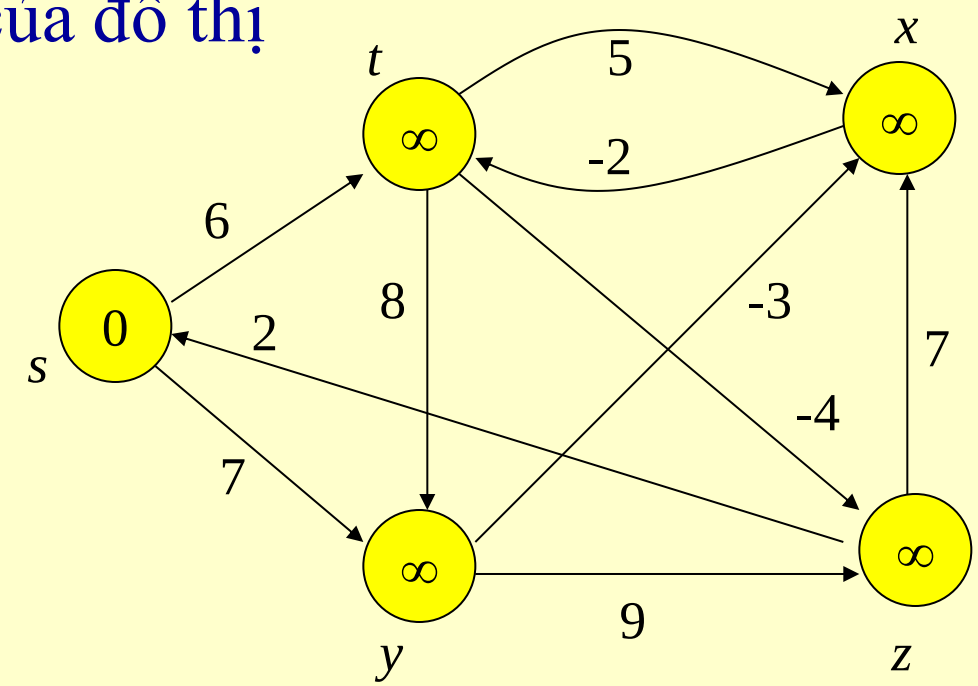
Ví du:

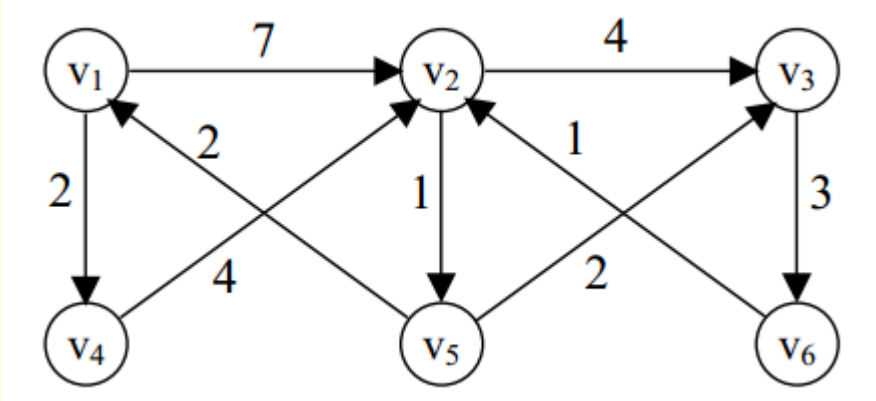


	1	2	3	4	5
1	∞	1	∞	∞	3
2	∞	∞	3	3	8
3	∞	∞	∞	1	-5
4	∞	∞	2	∞	∞
5	∞	∞	∞	4	∞

k	d[5], Truoc[5]	d[4], Truoc[4]	d[3], Truoc[3]	d[2], Truoc[2]
1	3, 1	∞, 1	∞, 1	1, 1
2	3, 1	4, 2	4, 2	1, 1
3	-1, 3	4, 2	4, 2	1, 1
4	-1, 3	3, 5	4, 2	1, 1
5	-1, 3	3, 5	4, 2	1, 1

Ví dụ: Tìm đường đi ngắn nhất từ đỉnh s đến tất cả các đỉnh còn lại của đồ thị





Nội dung

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-Warshal

Thuật toán Dijkstra

Edsger W.Dijkstra

(1930-2002)

- Trong trường hợp trọng số trên các cung là không âm, thuật toán do Dijkstra đề nghị hữu hiệu hơn rất nhiều so với thuật toán Ford-Bellman.
- Thuật toán dựa trên thủ tục gán nhãn. Thoạt đầu 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.*

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: ***w[u,v] > 0, u, v $\in$ V.***

Đầu ra: ***Với mỗi v $\in$ V***

d[v] = δ(s, v);

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

Thuật toán Dijkstra

Tập S: Chỉ cần cho chứng minh định lý

```
procedure Dijkstra;
begin
    for  v ∈ V  do  begin     (*  Khởi tạo  *)
                      d[v] := w[s,v] ;  p[v]:=s;
                    end;
    d[s] := 0;   S := {s};                (*  S – tập đỉnh có nhãn cố định *)
    T := V \ {s};                         (*  T là tập các đỉnh có nhãn tạm thời
*)
    while  T  ≠ ∅  do                     (*     Bước lặp   *)
    begin
        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  do                    (*  Gán nhãn lại cho các đỉnh trong T *)
            if  d[v] > d[u] + w[u,v]  then begin
                  d[v] := d[u] + w[u,v] ;  p[v] := u ;
            end;
    end;
end;
```

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.***

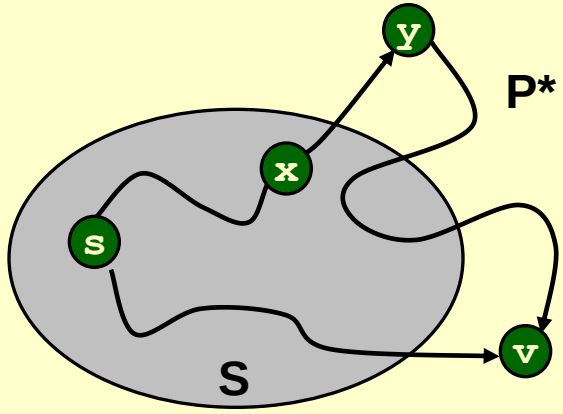
- **Định lý 1.** ***Thuật toán Dijkstra tìm được đường đi ngắn nhất từ đỉnh s đến tất cả các đỉnh còn lại trên đồ thị sau thời gian $O(n^2)$.***

- CM: Rõ ràng thời gian tính là $O(n^2)$

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.

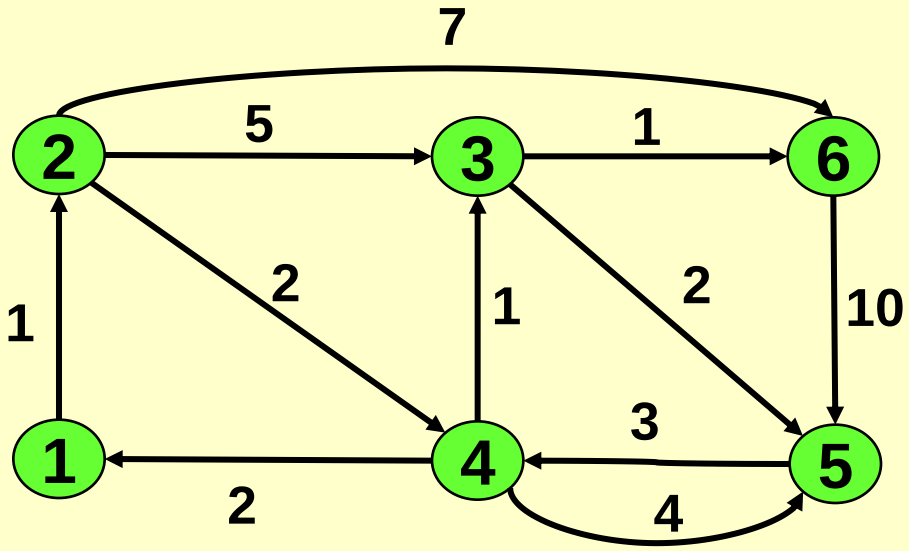


- Chuyển qui nạp:
  - giả sử thuật toán Dijkstra bổ sung v vào S
  - d(v) là độ dài của một đường đi từ s đến v
  - nếu d(v) không là độ dài đđnn từ s đến v, thì gọi P* là đđnn từ s đến v
  - P* phải sử dụng cạnh ra khỏi S, chẳng hạn (x, y)
  - khi đó d(v)> δ(s, v) giả thiết
    = δ(s, x) + w(x, y) + δ(y, v) tính chất 3
    ≥ 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

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

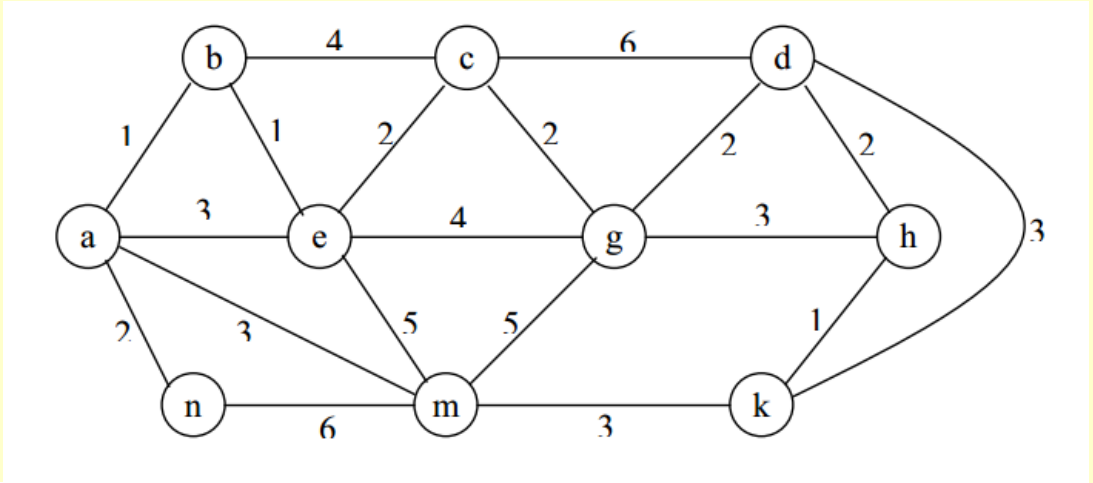
Ví dụ

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



	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	-	-	-	-	-	-

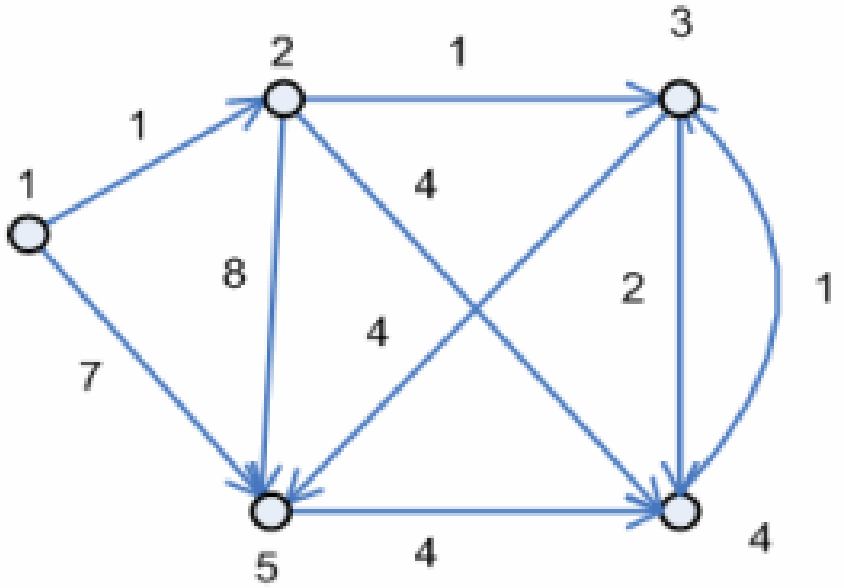
Ví dụ: Tìm đường đi ngắn nhất từ đỉnh a đến tất cả các đỉnh còn lại?



L(a)	L(b)	L(c)	L(d)	L(e)	L(g)	L(h)	L(k)	L(m)	L(n)
0	∞	∞	∞	∞	∞	∞	∞	∞	∞
–	1	∞	∞	3	∞	∞	∞	3	2
–	–	5	∞	2	∞	∞	∞	3	2
–	–	5	∞	2	∞	∞	∞	3	–
–	–	4	∞	–	6	∞	∞	3	–
–	–	4	∞	–	6	∞	6	–	–
–	–	–	10	–	6	∞	6	–	–
–	–	–	8	–	–	9	6	–	–
–	–	–	8	–	–	7	–	–	–
–	–	–	8	–	–	–	–	–	–

Bài tập

Tìm đường đi ngắn nhất từ đỉnh 1 đến các đỉnh còn lại trong đồ thị sau:

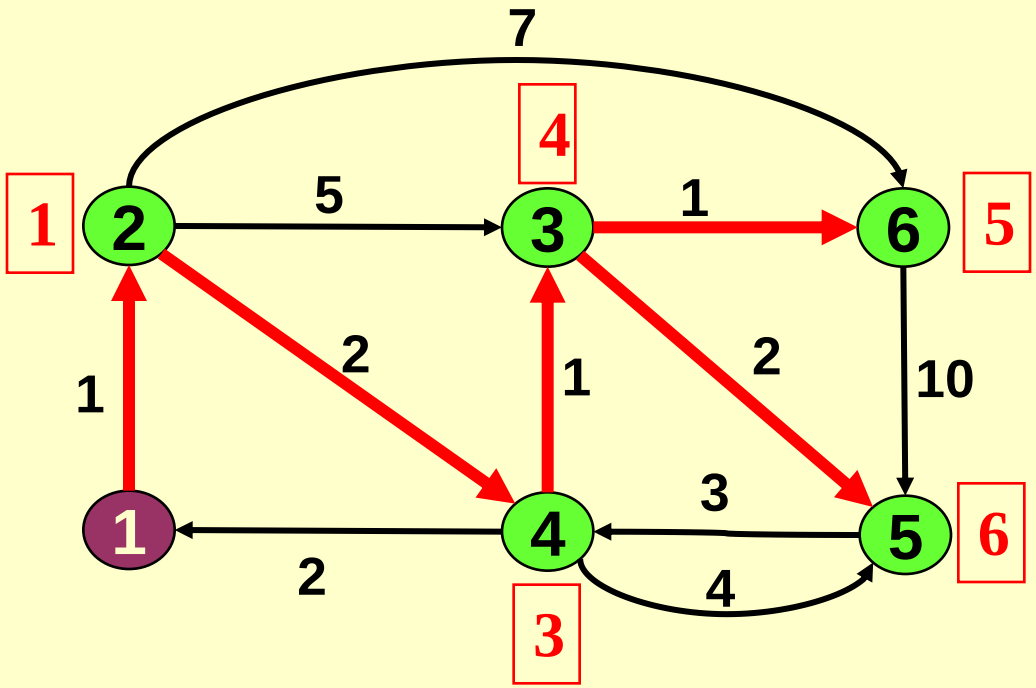


	1	2	3	4	5
1	∞	1	∞	∞	7
2	∞	∞	1	4	8
3	∞	∞	∞	2	4
4	∞	∞	1	∞	∞
5	∞	∞	∞	4	∞

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

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-Warshal

Nội dung

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-Warshal

ĐƯỜNG ĐI NGẮN NHẤT GIỮA MỌI CẶP ĐỈNH

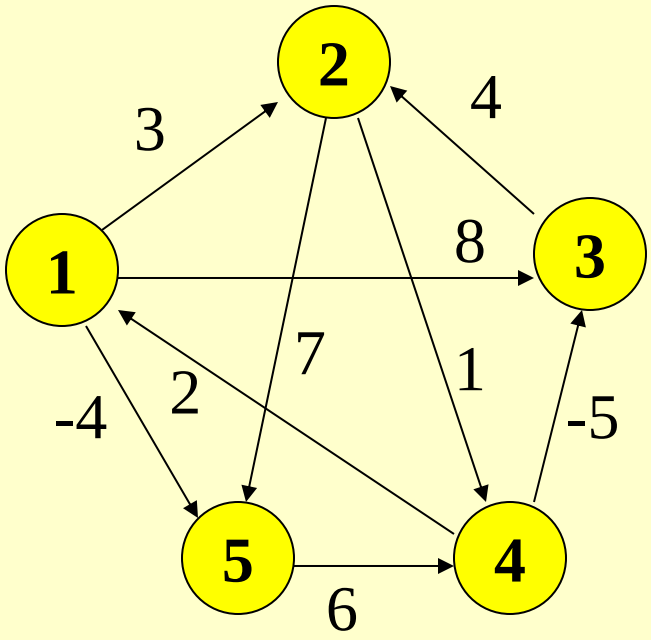
All-Pairs Shortest Paths

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



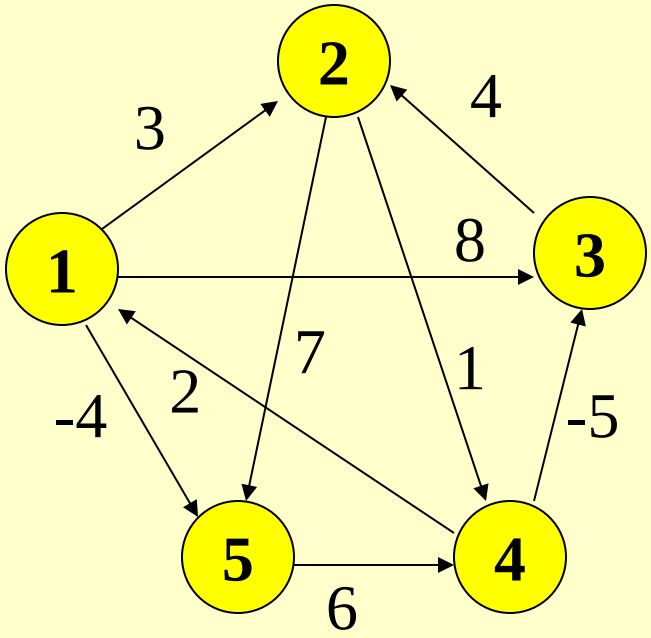
Đầu vào

$n \times n$ ma trận $W = (w_{ij})$ với

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

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

Tiếp



Đầu ra

Đường đi: 1- 5 - 4 - 3 - 2

= – 4 + 6 – 5 + 4

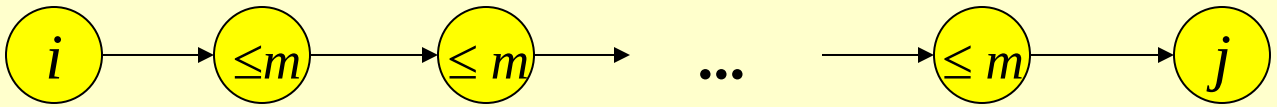
0	**1**	-3	2	-4
3	0	-4	1	-1
7	4	0	5	3
2	-1	-5	0	**-2**
8	5	1	6	0

4 - 1- 5

5 - 4 - 1

Thuật toán Floyd-Warshall

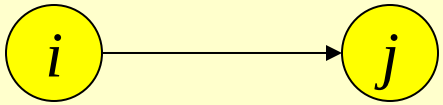
$d_{ij}^{(m)}$ = độ dài đường đi ngắn nhất từ i đến j sử dụng các đỉnh trung gian trong tập đỉnh { 1, 2, …, m }.



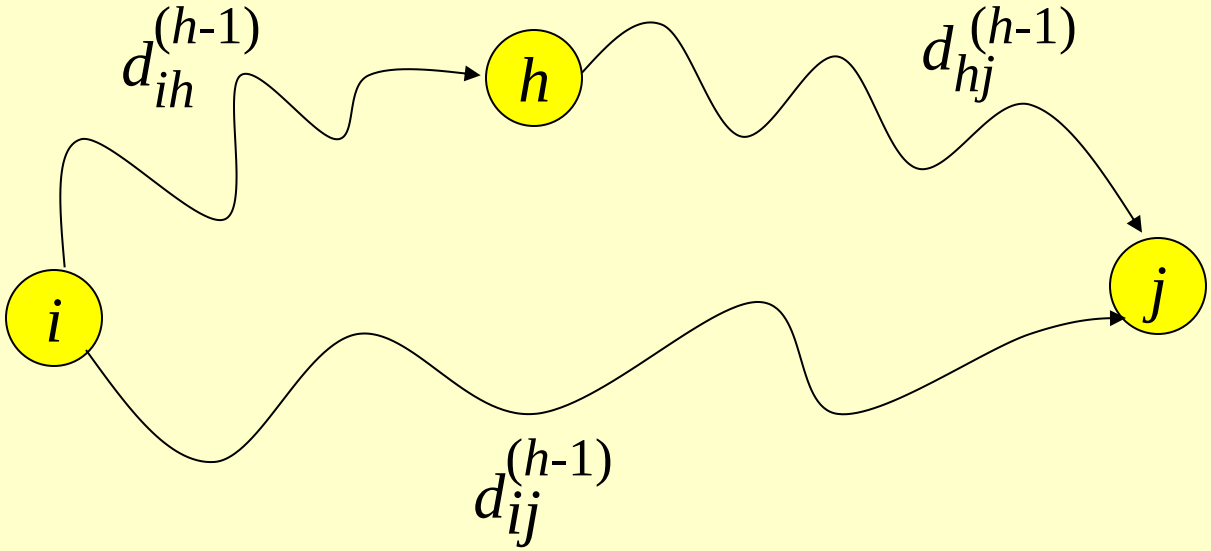
Khi đó độ dài đường đi ngắn nhất từ i đến j là $d_{ij}^{(n)}$

Công thức đệ qui tính $d^{(h)}$

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



- $d_{ij}^{(h)} = \min (d_{ij}^{(h-1)}, d_{ih}^{(h-1)} + d_{hj}^{(h-1)})$ nếu $h \geq 1$



Thuật toán Floyd-Warshall

Floyd-Warshall(n, W)

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

for $k \leftarrow 1$ to n do

for $i \leftarrow 1$ to n do

for $j \leftarrow 1$ to n do

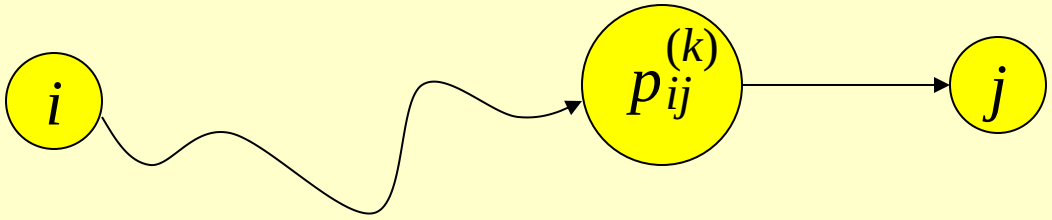
$d_{ij}^{(k)} \leftarrow \min (d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$

return $D^{(n)}$

Thời gian tính $\Theta(n^3)$!

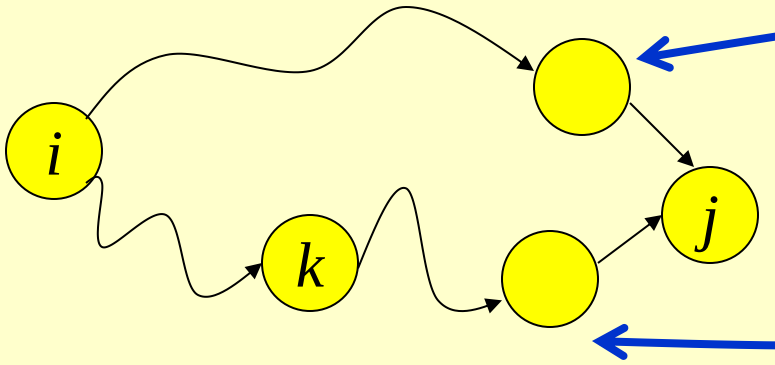
Xây dựng đường đi ngắn nhất

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



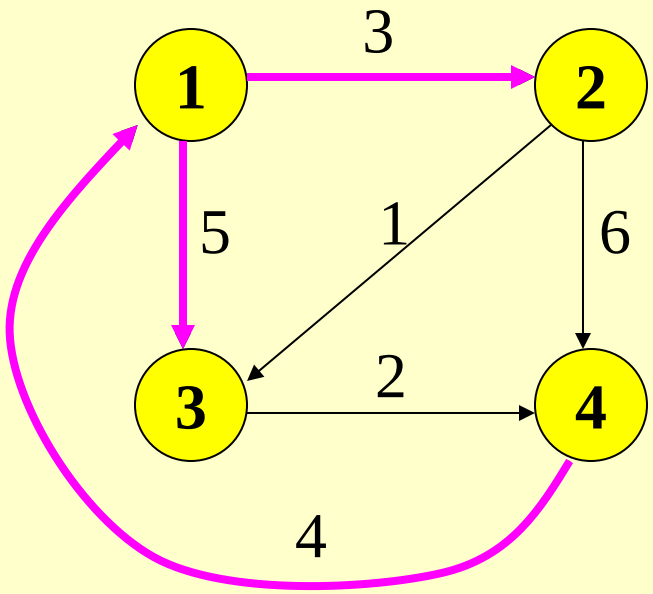
đường đi ngắn nhất từ i đến j chỉ qua các đỉnh trung gian trong $\{1, 2, \ldots, k\}$.

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



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

Ví dụ



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

$$P^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} \\ \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & \text{NIL} & \text{NIL} & 3 \\ 4 & \text{NIL} & \text{NIL} & \text{NIL} \end{pmatrix}$$

Có thể sử dụng 1 là đỉnh trung gian:

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

$$P^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} \\ \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & \text{NIL} & \text{NIL} & 3 \\ 4 & 1 & 1 & \text{NIL} \end{pmatrix}$$

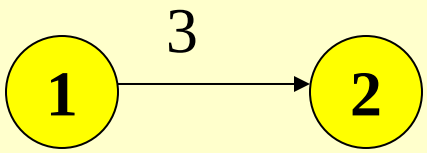
Ví dụ (tiếp)

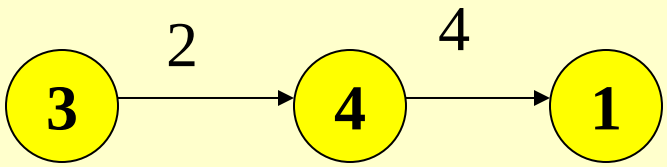
$$D^{(2)} = \begin{pmatrix} 0 & 3 & \mathbf{4} & \mathbf{9} \\ \infty & 0 & 1 & 6 \\ \infty & \infty & 0 & 2 \\ 4 & 7 & \mathbf{8} & 0 \end{pmatrix} \qquad P^{(2)} = \begin{pmatrix} \text{NIL} & 1 & \mathbf{2} & \mathbf{2} \\ \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & \text{NIL} & \text{NIL} & 3 \\ 4 & 1 & \mathbf{2} & \text{NIL} \end{pmatrix}$$

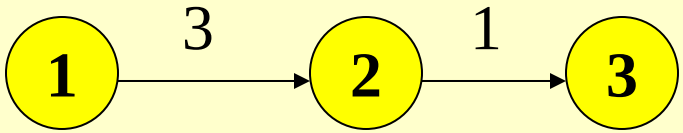
$$D^{(3)} = \begin{pmatrix} 0 & 3 & 4 & \mathbf{6} \\ \infty & 0 & 1 & \mathbf{3} \\ \infty & \infty & 0 & 2 \\ 4 & 7 & 8 & 0 \end{pmatrix} \qquad P^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 2 & \mathbf{3} \\ \text{NIL} & \text{NIL} & 2 & \mathbf{3} \\ \text{NIL} & \text{NIL} & \text{NIL} & 3 \\ 4 & 1 & 2 & \text{NIL} \end{pmatrix}$$

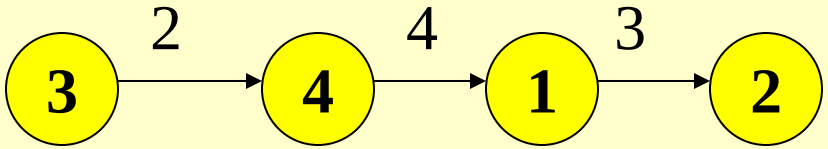
$$D^{(4)} = \begin{pmatrix} 0 & 3 & 4 & 6 \\ \mathbf{7} & 0 & 1 & 3 \\ \mathbf{6} & \mathbf{9} & 0 & 2 \\ 4 & 7 & 8 & 0 \end{pmatrix} \qquad P^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 2 & 3 \\ \mathbf{4} & \text{NIL} & 2 & 3 \\ \mathbf{4} & \mathbf{1} & \text{NIL} & 3 \\ 4 & 1 & 2 & \text{NIL} \end{pmatrix}$$

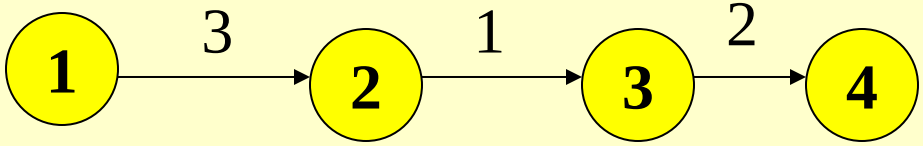
Ví dụ (tiếp)

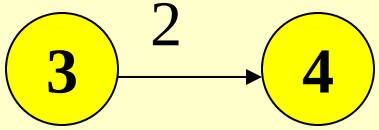


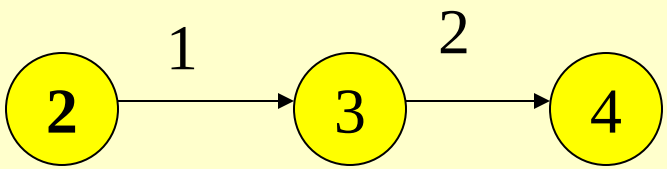


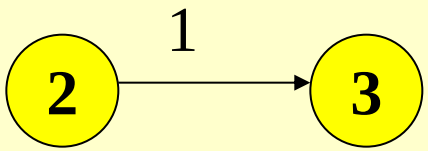






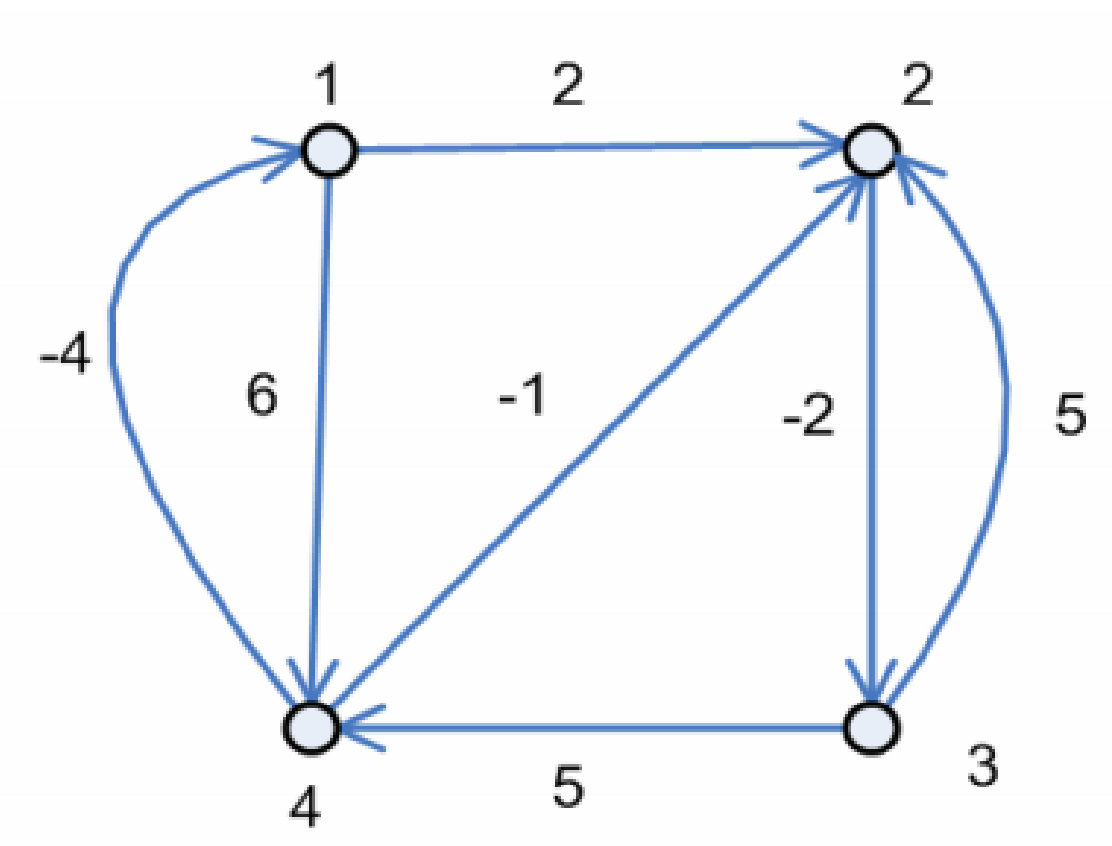






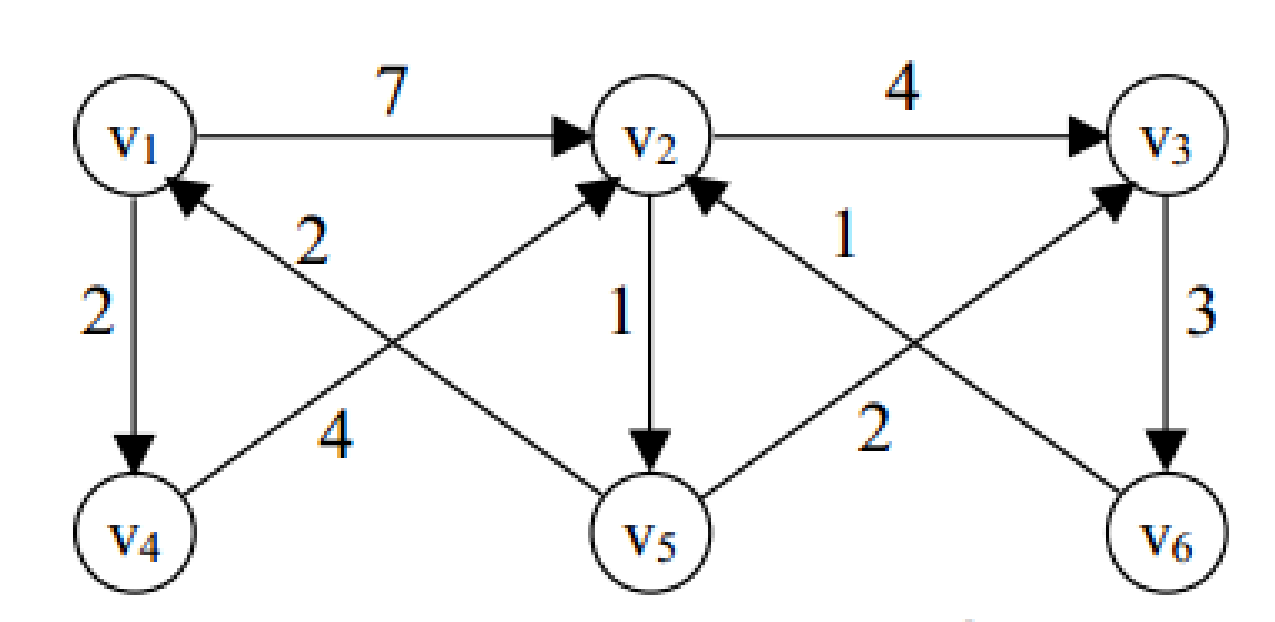
Bài tập

Áp dụng thuật toán Floyd-Warshall tìm đường đi ngắn nhất giữa các cặp đỉnh?



Bài tập

Áp dụng thuật toán Floyd-Warshall tìm đường đi ngắn nhất giữa các cặp đỉnh?



Chú ý

- Thuật toán Floyd có thể áp dụng cho đồ thị vô hướng cũng như đồ thị có hướng.
- Ta chỉ cần thay mỗi cạnh vô hướng (u,v) bằng một cặp cạnh có hướng (u,v) và (v,u) với m(u,v)=m(v,u). Tuy nhiên, trong trường hợp này, các phần tử trên đường chéo của ma trận D cần đặt bằng 0.