

Chương 5

- Bài toán đường đi ngắn nhất
- Thuật toán tìm bao đóng bắt cầu

TRẦN QUỐC VIỆT

Tài liệu tham khảo

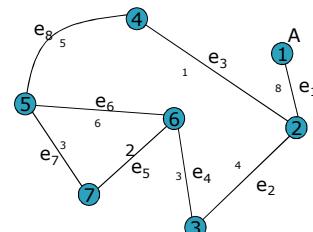
- ▶ Nguyễn Cam - Chu Đức Khánh, *Lý thuyết đồ thị* – NXB Trẻ Tp. HCM, 1998.
- ▶ Kenneth H. Rosen: *Discrete Mathematics and its Applications*, 7 Edition, McGraw Hill, 2010.

2

1. Giới thiệu

- Đồ thị có trọng số: Là đơn đồ thị, trong đó mỗi cạnh được gán một giá trị số, gọi là trọng số của cạnh
- Kí hiệu: $w(e)$ là trọng số của cạnh e

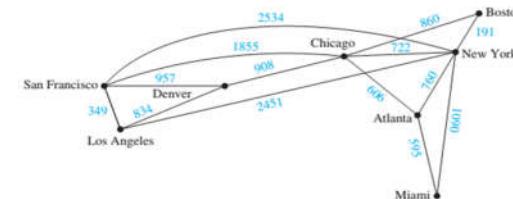
Ví dụ:



Giới thiệu

- Nhiều bài toán có thể được mô hình hóa bằng đồ thị có trọng số:
Ví dụ: Mô hình hóa một hệ thống đường hàng không nối giữa các thành phố

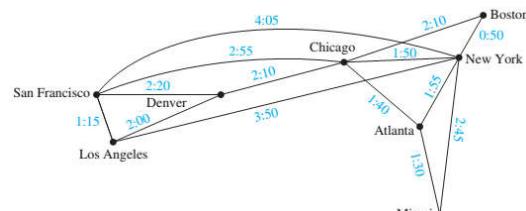
☞ Trọng số mỗi cạnh = Khoảng cách



Giới thiệu

Ví dụ: Mô hình hóa một hệ thống đường hàng không nối giữa các thành phố

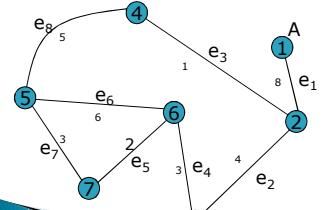
Trọng số mỗi cạnh = Thời gian bay



Giới thiệu

- Độ dài của một đường đi trong đồ thị có trọng số là tổng trọng số của tất cả các cạnh có trong đường đi đó.
 - Tìm đường đi ngắn nhất giữa 2 đỉnh trong đồ thị là một trong nhiều vấn đề liên quan đến đồ thị có trọng số.

Ví dụ

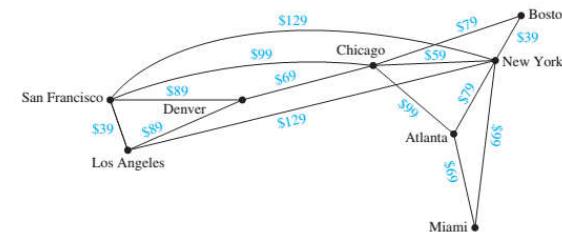


- Các đường đi từ 4 đến 6:
 4_85_6 . Độ dài: $5+6=11$
 $4_85_77_5$. Độ dài: $5+3+2=10$
 - Đường đi ngắn nhất giữa 4 và 6 là
 $4_32_23_4$. Độ dài: $1+4+3=8$

Giới thiệu

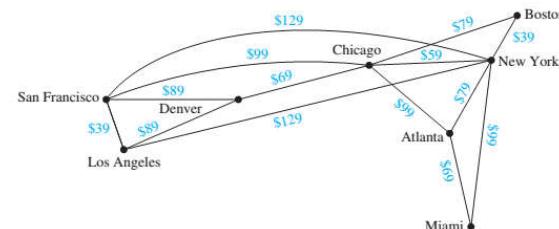
Ví dụ: Mô hình hóa một hệ thống đường hàng không nối giữa các thành phố

Trọng số mỗi cạnh = Giá vé



Giới thiệu

Ví dụ: Tìm một đường đi từ San Francisco đến Miami sao cho tổng tiền vé là ít nhất.



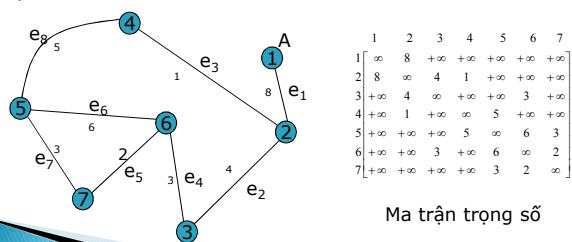
2. Ma trận trọng số

Cho đồ thị có trọng số $G = \langle V, E \rangle$, $|V| = n$

Ma trận trọng số của G được định nghĩa:

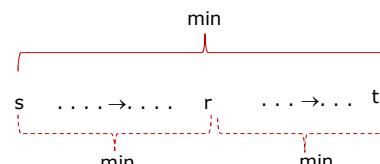
$$W = (w_{ij})_{n \times n}, \text{ với } w_{ij} = \begin{cases} w(\{v_i, v_j\}) & \text{nếu } \{v_i, v_j\} \in E \\ 0 \text{ (với } 0=0, -\infty \text{ hoặc } +\infty) & \text{nếu } \{v_i, v_j\} \notin E \end{cases}$$

Ví dụ



3. Định lý

Nếu đường đi $s \dots r \dots t$ là đường đi ngắn nhất từ s đến t thì $s.r$ và $r.t$ cũng là các đường đi ngắn nhất.

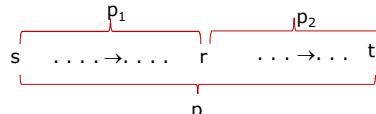


3. Định lý

- ▶ Chứng minh:
- ▶ Gọi p : là đường đi có độ dài nhỏ nhất từ s đến t

p_1 : là đoạn đường từ s đến r trên p

p_2 : là đoạn đường từ r đến t trên p



- Giả sử tồn tại đường đi $p'_1 \neq p_1$ từ s đến r nhỏ hơn p_1 . Khi đó:
 $I(p') = I(p'_1) + I(p_2) < I(p_1) + I(p_2) = I(p)$

Vô lý, vì p là đường đi ngắn nhất từ s đến $t \Rightarrow p_1$ là ngắn nhất
C/m tương tự: p_2 cũng là đường đi ngắn nhất

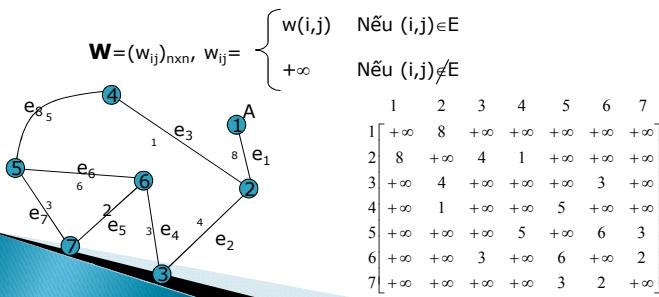
Thuật toán Dijkstra

4. Thuật toán Dijkstra

(tìm đường đi ngắn nhất từ đỉnh s đến tất cả các đỉnh khác)

Bài toán: Cho $G=(V,E)$ đơn đồ thị vô hướng (hoặc có hướng), $w(e_i) \geq 0$ là trọng số của cạnh (cung) e_i . Tìm đường đi có độ dài ngắn nhất từ đỉnh s cho trước đến các đỉnh khác trong G ?

- Ma trận khoảng cách/ma trận trọng số được định nghĩa:



4. Thuật toán Dijkstra

(tìm đường đi ngắn nhất từ source đến tất cả các đỉnh khác)

- Một số kí hiệu sử dụng:
- ✓ Gán nhãn cho đỉnh v ($L(v)$, $P(v)$): Đường đi từ s đến v có độ dài là $L(v)$, đỉnh trước kè với v trên đường đi là $P(v)$.
- ✓ $S =$ Tập các đỉnh đã xét, $R = V - S$

Procedure Dijkstra(G : Có trọng số và liên thông, s : Đỉnh nguồn)
Begin $R := V$;

For each v in R do

Begin $L[v] := \infty$;
 $P[v] := -$;

end

$L[s] = 0$;

While ($R \neq \emptyset$)

Begin $v =$ Đỉnh trong R có $L[v]$ nhỏ nhất;
 $R = R - \{v\}$

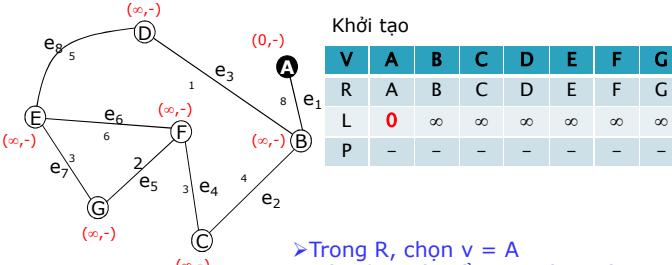
For each i in ($R \cap$ Tập đỉnh kè với v) do

If $(L[i] > L[v] + w[v][i])$ then
 $L[i] := L[v] + w[v][i]$; $P[i] = v$;

end

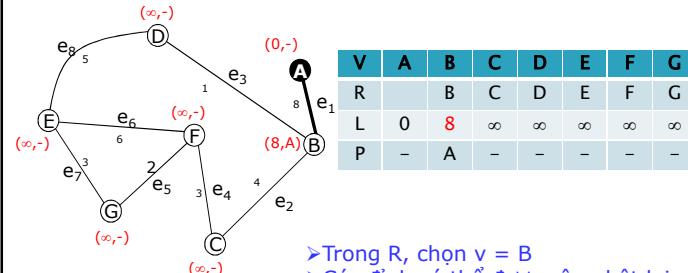
End

Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A

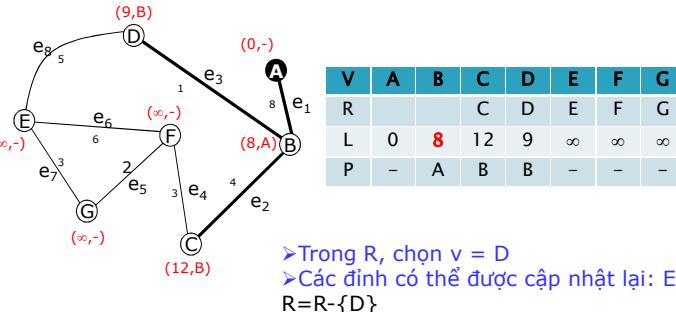


➤ Trong R , chọn $v = A$
➤ Các đỉnh có thể được cập nhật lại: B
➤ $R = R - \{A\}$

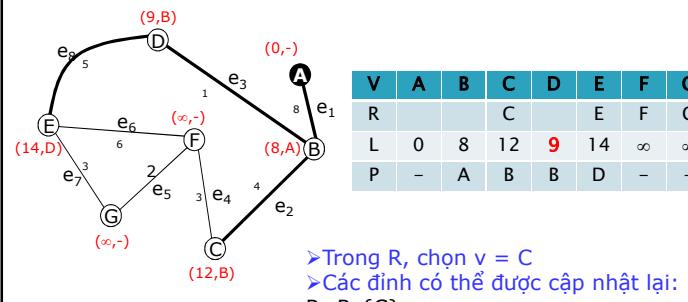
Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A



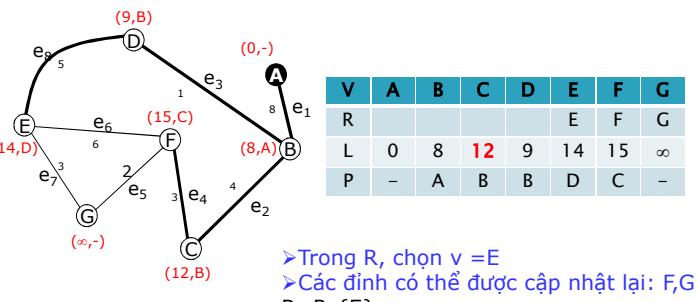
Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A



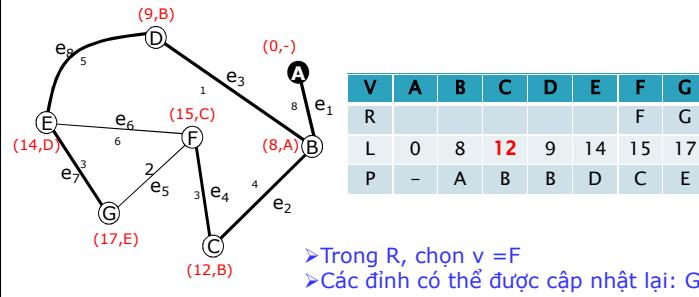
Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A



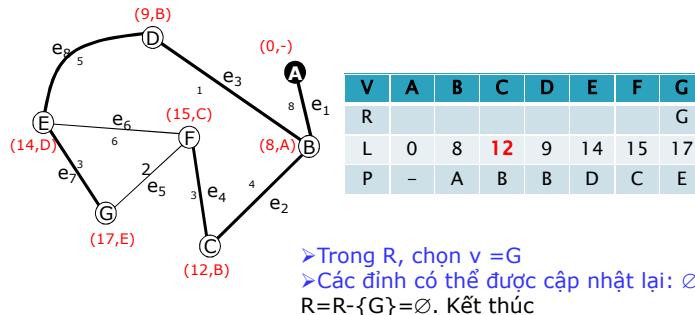
Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A



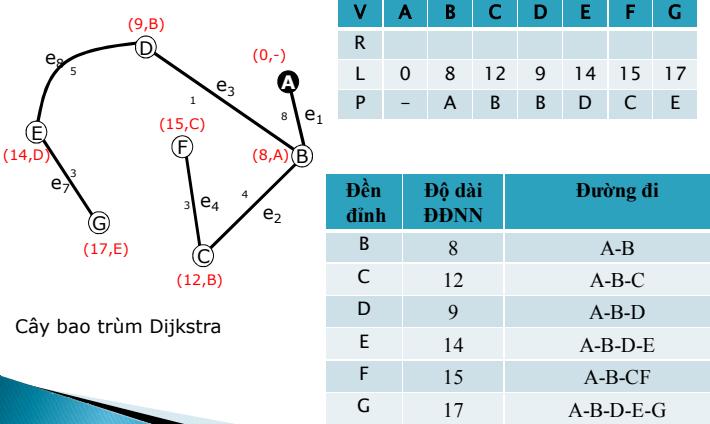
Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A



Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A



Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A



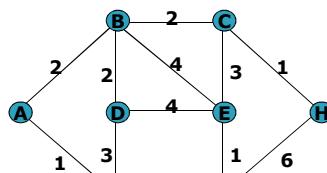
Nhận xét

- ❑ Thuật toán Dijkstra dùng được cho cả đồ thị vô hướng và có hướng
 - ❑ Độ phức tạp của thuật toán Dijkstra là $O(n^2)$
 - ❑ Thuật toán Dijkstra chỉ sử dụng với G không có cạnh có trọng số âm
-
- Tìm đường đi ngắn nhất từ đỉnh 3 đến đỉnh 5?
- ❑ Kết quả Khi thực hiện thuật toán Dijkstra, ta thu được một cây bao trùm của G gọi là cây bao trùm Dijkstra của G gốc s với khoảng cách ngắn nhất từ s đến từng đỉnh khác

Một số ví dụ khác

1) Cho đồ thị, chạy thuật toán Dijkstra, tìm đường đi ngắn nhất đến các đỉnh

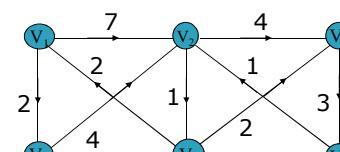
- Bắt đầu từ đỉnh A
- Bắt đầu từ đỉnh G



	A	B	C	D	E	F
A	∞	2	∞	∞	∞	1
B	2	∞	2	2	4	∞
C	∞	2	∞	∞	3	∞
D	∞	2	∞	∞	4	3
E	∞	4	3	4	∞	∞
F	1	∞	∞	3	∞	∞

Một số ví dụ khác

2. Chạy thuật toán Dijkstra, tìm đường đi ngắn nhất đến các đỉnh, bắt đầu từ đỉnh v_5



	V_1	V_2	V_3	V_4	V_5	V_6
V_1	0	7	∞	2	∞	∞
V_2	∞	0	4	∞	1	∞
V_3	∞	∞	0	∞	∞	3
V_4	∞	4	∞	0	∞	∞
V_5	2	∞	2	∞	0	∞
V_6	∞	1	∞	∞	∞	0

Thuật toán Floyd

Thuật toán Floyd

- ❑ Xét đơn đồ thị đồ thị có hướng có trọng số $G= \langle V, E \rangle$:
 - Tập đỉnh: $V = \{v_1, v_2, \dots, v_n\}$
 - Ma trận khoảng cách: $W = (w_{ij})$

$$W = (w_{ij})_{n \times n}, w_{ij} = \begin{cases} w(i,j) & \text{Nếu } (i,j) \in E \\ +\infty & \text{Nếu } (i,j) \notin E \end{cases}$$

- ❑ Thuật toán Floyd giúp xác định tất cả các đường đi ngắn nhất giữa tất cả các cặp đỉnh.

Thuật toán Floyd (tt)

Thuật toán Floyd xây dựng dãy các ma trận $n \times n$ W_k ($0 \leq k \leq n$) như sau:

Procedure Floyd(G: liên thông có ma trận trọng số W)

Begin

$W_0 := W$

For $k=1$ **to** n **do**

For $i=1$ **to** n **do**

For $j=1$ **to** n **do**

 if $(W_{k-1}[i,j] > W_{k-1}[i,k] + W_{k-1}[k,j])$ then

$W_k[i,j] = W_{k-1}[i,k] + W_{k-1}[k,j]$

 else

$W_k[i,j] = W_{k-1}[i,j];$

End

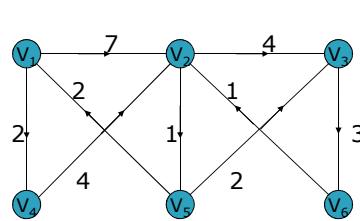
Thuật toán Floyd (tt)

Định lý

Thuật toán Floyd cho ta ma trận $W^* = W_n$ là ma trận khoảng cách nhỏ nhất của đồ thị G.

Chứng minh: Bài tập

Ví dụ



W

∞	7	∞	2	∞	∞
∞	∞	4	∞	1	∞
∞	∞	∞	∞	∞	3
∞	4	∞	∞	∞	∞
2	∞	2	∞	∞	∞
∞	1	∞	∞	∞	∞

W = W₀

Tính W_1 ($k=1$)

$W_0[1,2]$					
∞	7	∞	2	∞	∞
∞	∞	4	∞	1	∞
∞	∞	∞	∞	∞	3
∞	4	∞	∞	∞	∞
2	∞	2	∞	∞	∞
∞	1	∞	∞	∞	∞

$W_0[5,1]$

$W_0[5,2]$

∞	7	∞	2	∞	∞
∞	∞	4	∞	1	∞
∞	∞	∞	∞	∞	3
∞	4	∞	∞	∞	∞
2	9	∞	∞	∞	∞
∞	1	∞	∞	∞	∞

$i=5 \quad j=2$

$W_0[5,2] > w_0[5,1] + w_0[1,2]$

Nên $W_1[5,2] = w_0[5,1] + w_0[1,2]$

$W = W_0$

Tính $W_1 (k=1)$

$W_0[1,3]$	
$W_0[5,1]$	

$W_0[5,3] \quad k=1 \quad i=5 \quad j=3: \quad W_0[5,3] < w_0[5,1] + w_0[1,3]$
Nên $W_1[5,3] = W_0[5,3]$

Với giải thuật trên, ta tính được:
 W_1

∞	7	∞	2	∞	∞
∞	∞	4	∞	1	∞
∞	∞	∞	∞	∞	3
∞	4	∞	∞	∞	∞
2	9	2	∞	∞	∞
∞	1	∞	∞	∞	∞

W_2

∞	7	11	2	8	∞
∞	∞	4	∞	1	∞
∞	∞	∞	∞	∞	3
∞	4	8	∞	5	∞
2	9	2	4	10	∞
∞	1	5	∞	2	∞

W_3

∞	7	11	2	8	14
∞	∞	4	∞	1	7
∞	∞	∞	∞	∞	3
∞	4	8	∞	5	11
2	9	2	4	10	5
∞	1	5	∞	2	8

W_4

∞	6	10	2	7	13
∞	∞	4	∞	1	7
∞	∞	∞	∞	∞	3
∞	4	8	∞	5	11
2	8	2	4	9	5
∞	1	5	∞	2	8

 W_5

9	6	9	2	7	12
3	9	3	5	1	6
∞	∞	∞	∞	∞	3
7	4	7	9	5	10
2	8	2	4	9	5
4	1	4	6	2	7

 $W^* = W_6$

9	6	9	2	7	12
3	7	3	5	1	6
7	4	7	9	5	3
7	4	7	9	5	10
2	6	2	4	7	5
4	1	4	6	2	7

- Độ dài đường đi ngắn nhất từ đỉnh 4 đến đỉnh 6 là
 $W^*[4,6] = 10$
- Độ dài đường đi ngắn nhất từ đỉnh 5 đến đỉnh 4 là
 $W^*[5,4] = 4$
-

Thuật toán Floyd mở rộng

(Có xác định đường đi trong thuật toán Floyd)

- Đặt $P_0[i, j] = j$ nếu có cung $v_i v_j$.
- $P_0[i, j] = \infty$: không xác định.
- $P_k[i, j]$: đỉnh liền sau i trên đường đi ngắn nhất từ i đến j.
- Đường đi ngắn nhất từ đỉnh i đến đỉnh j được xác định bởi dây:
- $i, P^*[i,j], P^*[P^*[i,j],j], P^*[P^*[P^*[i,j],j],j], \dots, j$

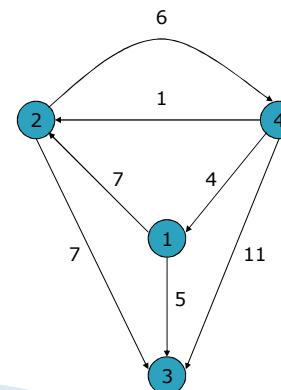
Thuật toán Floyd mở rộng (tt)

```

W0=W;
For k=1 to n do
  For i=1 to n do
    For j=1 to n do
      if (Wk-1[i,j] > Wk-1[i,k] + Wk-1[k,j]) then
        begin
          Wk[i,j] = Wk-1[i,k] + Wk-1[k,j];
          Pk[i,j] = Pk-1[i,k];
        end
      else
        begin
          Wk[i,j] = Wk-1[i,j];
          Pk[i,j] = Pk-1[i,j];
        end
  end
end

```

Thí dụ



Thí dụ

∞	7	5	∞
∞	∞	7	6
∞	∞	∞	∞
4	1	11	∞

$$P_0 =$$

∞	2	3	∞
∞	∞	3	4
∞	∞	∞	∞
1	2	3	∞

Thí dụ

∞	7	5	∞
∞	∞	7	6
∞	∞	∞	∞
4	1	9	∞

$$P_1 =$$

∞	2	3	∞
∞	∞	3	4
∞	∞	∞	∞
1	2	1	∞

Thí dụ

∞	7	5	13
∞	∞	7	6
$\bar{\infty}$	∞	∞	∞
4	1	8	7

 W_2

∞	2	3	2
∞	∞	3	4
∞	∞	∞	∞
1	2	2	2

 $P_2 =$

Thí dụ

∞	7	5	13
∞	∞	7	6
$\bar{\infty}$	∞	∞	∞
4	1	8	7

 W_3

∞	2	3	2
∞	∞	3	4
∞	∞	∞	∞
1	2	2	2

 $P_3 =$

Thí dụ

$W^* = W_4 =$	17	7	5	13
	10	7	7	6
	∞	∞	∞	∞
	4	1	8	7

$P^* = P_4 =$	2	2	3	2
	4	4	3	4
	∞	∞	∞	∞
	1	2	2	2

Nhận xét

- ▶ Thuật toán Floyd có thể áp dụng cho đồ thị G vô hướng (thay mỗi cạnh (u,v) bởi cặp cung có hướng (\vec{u},\vec{v}) và (\vec{v},\vec{u})).
- ▶ $\xrightarrow{\text{Đồ thị G có hướng là liên thông mạnh}} \forall u,v \in V, u \neq v \quad W^*[u,v] < \infty$.
- ▶ $\xrightarrow{\text{Đồ thị G có hướng có chu trình}} \exists u \in V, W^*[u,u] < \infty$
- ▶ Độ phức tạp của thuật toán Floyd: $O(|V|^3)$

Thuật toán Bellman–Ford

- Cho đơn đồ thị có trọng số G, không có chu trình âm, với ma trận trọng số W
- Thuật toán Bellman–Ford cho phép tìm đường đi ngắn nhất từ một đỉnh s đến tất cả các đỉnh còn lại
- Ma trận trọng số

$$W = (w_{ij})_{n \times n}, w_{ij} = \begin{cases} w(i,j) & \text{Nếu } (i,j) \in E \\ +\infty & \text{Nếu } (i,j) \notin E \end{cases}$$

Thuật toán Bellman–Ford

Input: W là ma trận trọng số của đơn đồ thị G
s: Đỉnh nguồn
Output: L: L[v] Khoảng cách ngắn nhất từ s đến các đỉnh v
P: P[v] là đỉnh trước đỉnh v

Thuật toán Bellman–Ford

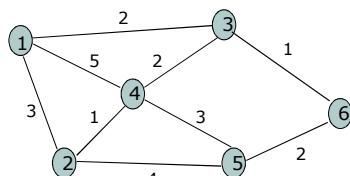
```
// Khởi tạo
// Mỗi đỉnh i, gán nhãn bởi cặp (pre[i], l[i])
For i=1 to n do
begin
  P[i]=-;
  if (i=s) then
    L[i]=0
  else
    L[i]=∞;
end
```

Thuật toán Bellman–Ford

```
stop=false;k=0;
while (not stop) do
begin stop=true;k=k+1;
  For each (i,j) in E do
    if L[j]>L[i]+w[i][j] then
      begin L[j]=L[i]+w[i][j]; P[j]=i;
      stop=false;
    end
  if (k>n) then
    begin if (stop=false) print "đồ thị có chu trình âm"
      stop=true;
    end
end
```

Thuật toán Bellman–Ford (tt)

Ví dụ: $s=6$



Định	1	2	3	4	5	6
Khởi tạo	($-\infty$)					
Lặp lần 1	($-\infty$)	($-\infty$)	(6,1)	($-\infty$)	(6,2)	($-\infty$)
Lặp lần 2	(3,3)	(5,6)	(6,1)	(3,3)	(6,2)	($-\infty$)
Lặp lần 3	(3,3)	(4,4)	(6,1)	(3,3)	(6,2)	($-\infty$)
Lặp lần 4	(3,3)	(4,4)	(6,1)	(3,3)	(6,2)	($-\infty$)
..

Nhận xét

- Thuật toán Bellman–Ford tìm đường đi ngắn nhất từ một đỉnh (đỉnh nguồn) đến tất cả các đỉnh còn lại (giống như Dijkstra)
- Thuật toán Bellman–Ford dùng được cho cả đồ thị vô hướng và có hướng, cho phép có cạnh âm, miễn là không có chu trình âm
- Độ phức tạp của thuật toán là $O(n^3)$

Thuật toán tính bao đóng bắt đầu – WARSHALL

- Cho đơn đồ thị G có tập đỉnh $V=\{v_1, v_2, \dots, v_n\}$, đỉnh v_j gọi là khả liên (reachable) từ đỉnh v_i nếu có một đường đi từ v_i đến v_j
- Ma trận kè $A=(a_{ij})$ của G là một ma trận Boolean
- Kí hiệu: $A^{(k)} = A^{(k-1)} \times A$ với phép cộng/nhân các phần tử tương ứng với phép toán or/and trên bit:

+/or	0	1
0	0	1
1	1	1

x/and	0	1
0	0	0
1	0	1

Thuật toán tính bao đóng bắt đầu – WARSHALL (tt)

- Dinh Iý:** Gọi $A = (a_{ij})$ là ma trận kè của đơn đồ thị G , $a_{ij}^{(k)} = 1 \Leftrightarrow$ có một đường đi độ dài k từ đỉnh v_i đến đỉnh v_j
- C/m: Sử dụng quy nạp
 - Với $k=1$, $a_{ij}^{(1)} = 1 \Leftrightarrow$ có đường đi trực tiếp (độ dài 1) từ v_i đến v_j
 - Giả sử $a_{ij}^{(k)} = 1 \Leftrightarrow$ có đường đi độ dài k từ v_i đến v_j
 - Ta có

$$a_{ij}^{(k+1)} = \sum_{t=1}^n a_{it}^{(k)} a_{tj}$$

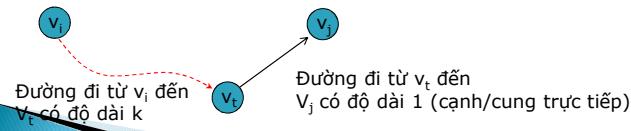
Thuật toán tính bao đóng bắt cầu – WARSHALL (tt)

$$a_{ij}^{(k+1)} = \sum_{t=1}^n a_{it}^{(k)} a_{tj}$$

$$a^{(k+1)}_{ij} = 1 \Leftrightarrow \exists t, a^{(k)}_{it} = 1 \wedge a_{tj} = 1$$

- $a^{(k)}_{it} = 1 \Leftrightarrow$ có đường đi độ dài k từ v_i đến v_t
- $a_{tj} = 1 \Leftrightarrow$ có đường đi độ dài 1 từ v_t đến v_j

Vậy có đường đi có độ dài $k+1$ từ v_i đến v_j



Thuật toán tính bao đóng bắt cầu – WARSHALL (tt)

Ma trận khả liên $R^{(k)} = (r_{ij})$:

$$R^{(k)} = A^{(1)} + A^{(2)} + \dots + A^{(k)}$$

Nhận xét: $r_{ij} = 1 \Leftrightarrow$ có một đường đi từ đỉnh v_i đến đỉnh v_j

Thuật toán tính bao đóng bắt cầu – WARSHALL

- ▶ Input: A: Ma trận kè của đơn đồ thị G
- ▶ Output: R: ma trận khả liên của G

```

R=A
For k=1 to n do
  For i=1 to n do
    For j=1 to n do
      R[i][j]=R[i][j] or (R[i][k] and R[k][j]);
Return R
  
```

Thực hành

1. Cài đặt thuật toán Dijkstra
 - a) Tìm đường đi ngắn nhất đến tất cả các đỉnh khác trong đồ thị từ một đỉnh cho trước. In ra tất cả các đường đi cùng với khoảng cách tìm được)
 - b) Tìm đường đi ngắn nhất từ đỉnh s đến đỉnh t cho trước
2. Cài đặt thuật toán Floyd
 - a) Tìm ma trận k/c ngắn nhất
 - b) In ra đường đi cùng với k/c ngắn nhất tìm được giữa 2 đỉnh s, t cho trước

Thực hành

3. Cài đặt thuật toán :

- a) Kiểm tra tính liên thông mạnh của đồ thị có hướng
- b) Kiểm tra một đồ thị có chu trình hay không
- c) Kiểm tra 2 đỉnh u, v có khả liên hay không (có đường đi từ u đến v hay không)

4. Cài đặt thuật toán WARSHALL

5. Cài đặt thuật toán

