Chương 5

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

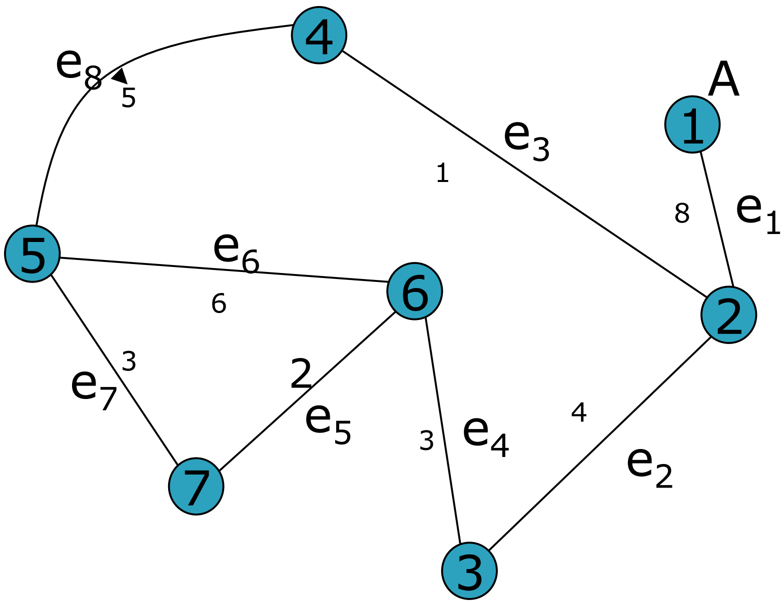
Tài liệu tham khảo:

- Silde bài giảng ThS. Trần Quốc Việt
- Silde bài giảng ThS. Lê Phi Hùng
- Nguyễn Cam, Chu Đức Khánh, [redacted]
- Kenneth H. Rosen, Discrete Mathematics and Its Applications

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

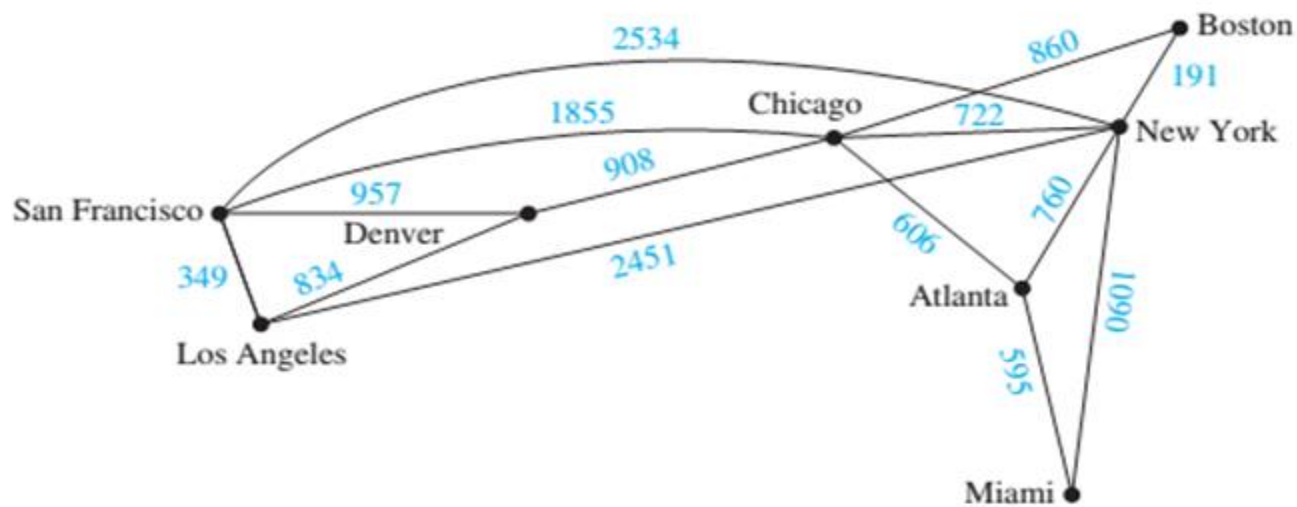
<u>*Ví dụ:*</u>



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

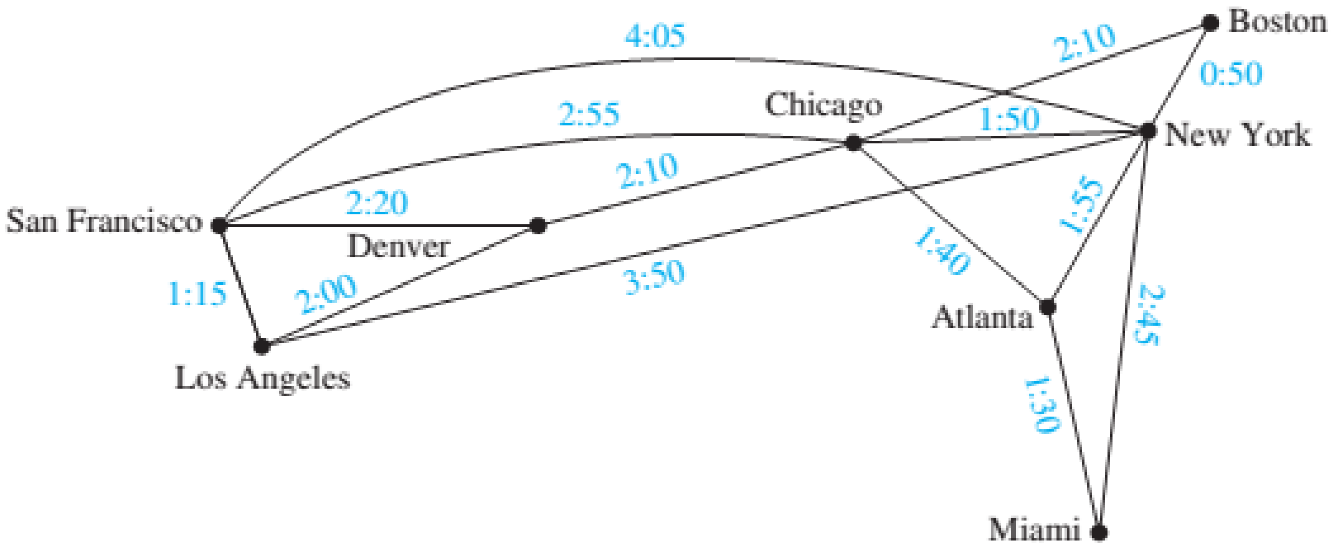
  ☞ ***Trọng số mỗi cạnh= Khoảng cách***



Giới thiệu

<u>Ví dụ</u>:*Mô hình hóa một hệ thống đường hàng không nối giữa các thành phố*

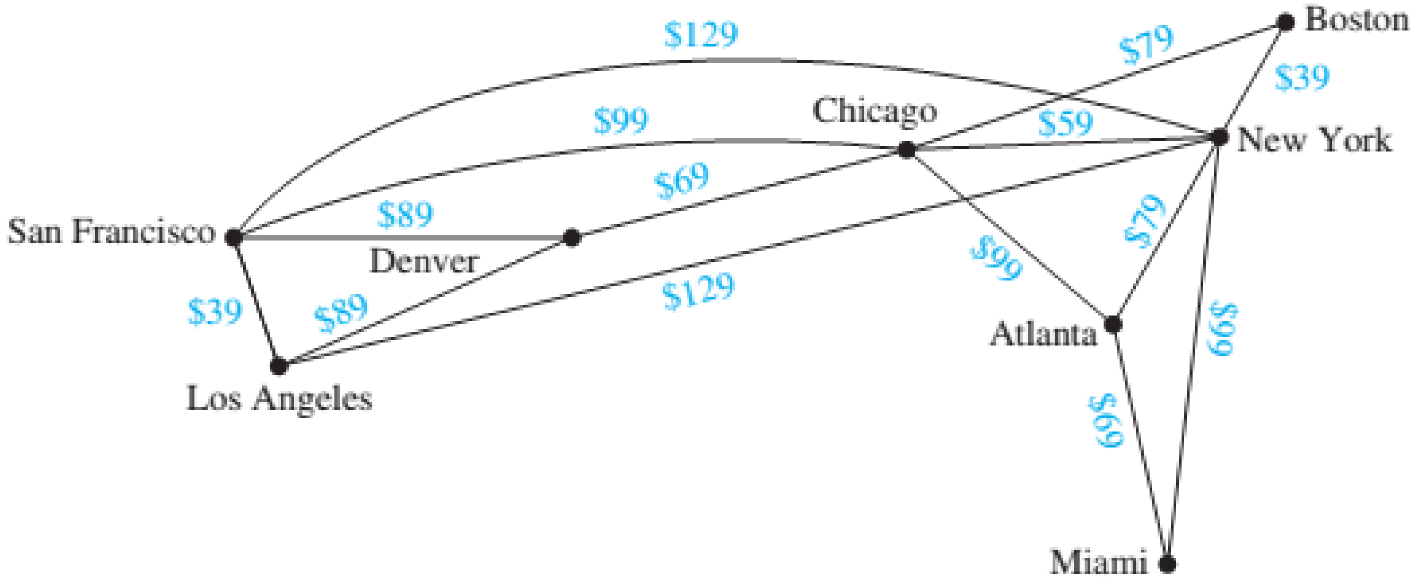
Trọng số mỗi cạnh= Thời gian bay



Giới thiệu

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

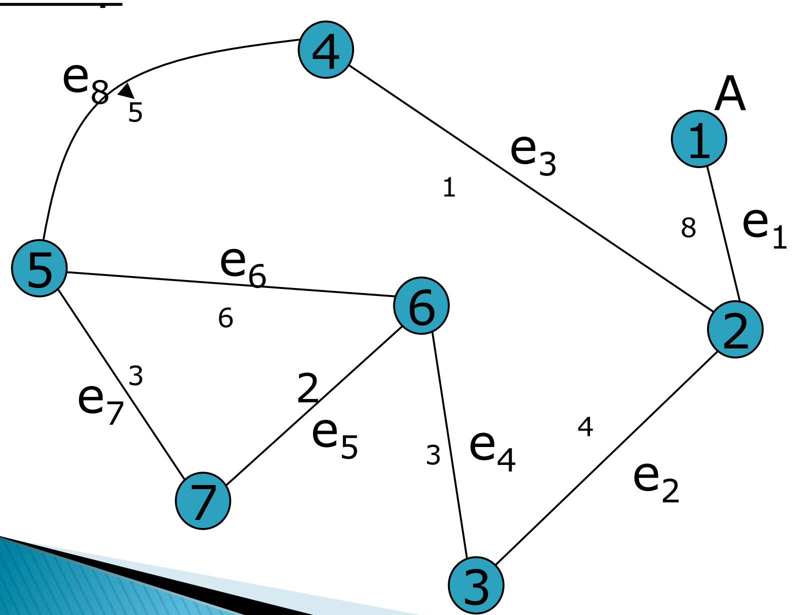
Trọng số mỗi cạnh= Giá vé



Giới thiệu

- 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.

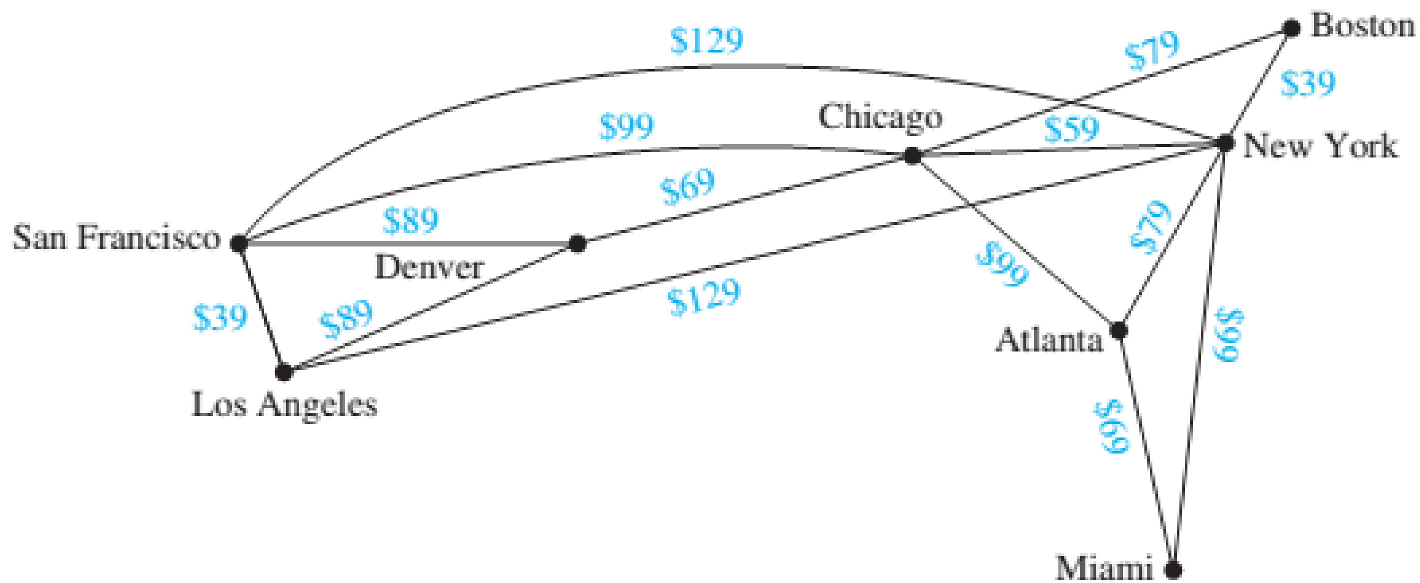
Ví dụ



- Các đường đi từ 4 đến 6:
  $4e_8 5e_6 6$. Độ dài: 5+6=12
  $4e_8 5e_7 7e_5 6$. Độ dài: 5+3+2=10
  $4e_3 2e_2 3e_4 6$. Độ dài: 1+4+3=8
- Đường đi ngắn nhất giữa 4 và 6 là $4e_3 2e_2 3e_4 6$ với độ dài 8.

Giới thiệu

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

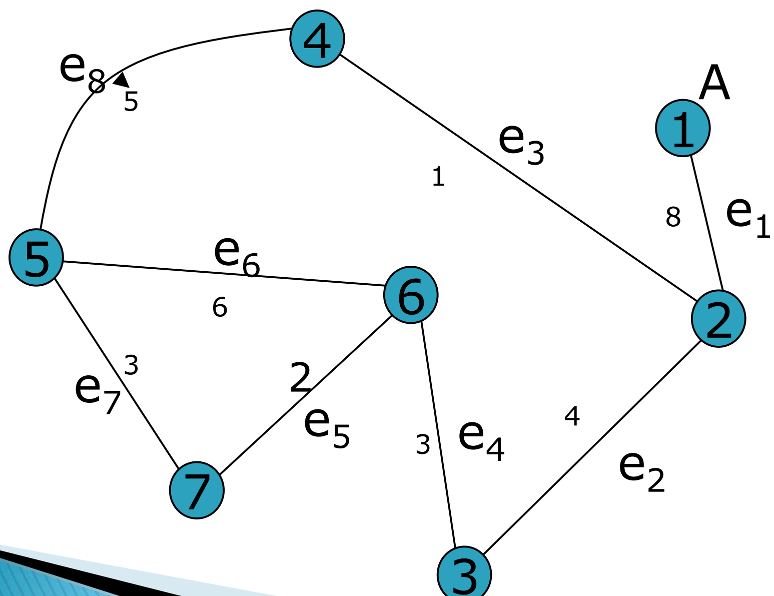


2. Ma trận trọng số

- Cho đồ thị có trọng số G=<V,E>, |V|=n
- Ma trận trọng số của G được định nghĩa:

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

Ví dụ



$$\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & \infty & 8 & +\infty & +\infty & +\infty & +\infty & +\infty \\ 2 & 8 & \infty & 4 & 1 & +\infty & +\infty & +\infty \\ 3 & +\infty & 4 & \infty & +\infty & +\infty & 3 & +\infty \\ 4 & +\infty & 1 & +\infty & \infty & 5 & +\infty & +\infty \\ 5 & +\infty & +\infty & +\infty & 5 & \infty & 6 & 3 \\ 6 & +\infty & +\infty & 3 & +\infty & 6 & \infty & 2 \\ 7 & +\infty & +\infty & +\infty & +\infty & 3 & 2 & \infty \end{array}$$

Ma trận trọng số

3. Định lý

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

$$\overbrace{\underbrace{s \quad .\,.\,.\,.\,.\!\to\!.\,.\,.\,. \quad r}_{\text{min}} \quad \underbrace{.\,.\,.\!\to\!.\,.\,. \quad t}_{\text{min}}}^{\text{min}}$$

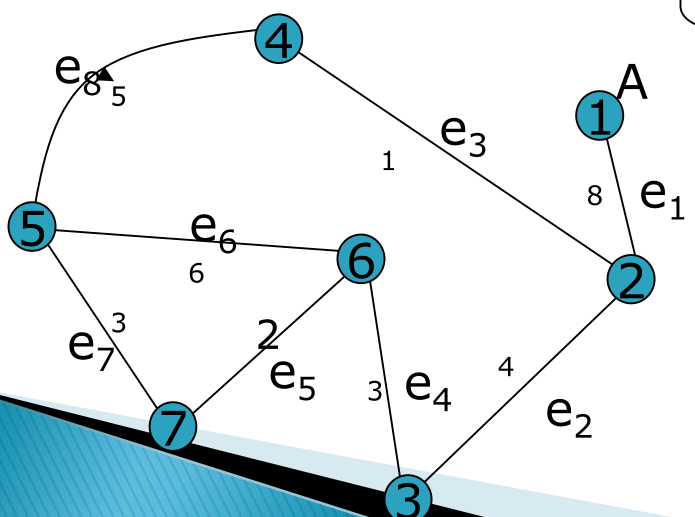
Thuật toán Dijkstra

4. Thuật toán Dijsktra
(tìm đường đi ngắn nhất từ đỉnh s đến tất cả các đỉnh khác)

<u>*Bài toán*</u>: 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 đỉnh ***s*** cho trước đến các đỉnh khác trong G?

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

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



	1	2	3	4	5	6	7
1	$+\infty$	8	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
2	8	$+\infty$	4	1	$+\infty$	$+\infty$	$+\infty$
3	$+\infty$	4	$+\infty$	$+\infty$	$+\infty$	3	$+\infty$
4	$+\infty$	1	$+\infty$	$+\infty$	5	$+\infty$	$+\infty$
5	$+\infty$	$+\infty$	$+\infty$	5	$+\infty$	6	3
6	$+\infty$	$+\infty$	3	$+\infty$	6	$+\infty$	2
7	$+\infty$	$+\infty$	$+\infty$	$+\infty$	3	2	$+\infty$

4. Thuật toán Dijsktra
(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 kề với v trên đường đi là P(v).
- ✓ **S**=Tập các đỉnh đã xét, **R** = **V**-**S**

```
Procedure Dijsktra(G: Có trọng số và liên thông,s: Đỉnh nguồn)
Begin  R:=V;
        For each v in R do
        Begin           L[v]: = ∞;
                        P[v]: = - ;
        end
       L[s]=0;
       While (R≠∅)
        Begin      v = Đỉnh trong R có L[v] nhỏ nhất;
                   R=R-{v}
                   For each i in (R ∩ Tập đỉnh kề với v) do
                        If  (L[i]> L[v]+w[v][i]) then
                              L[i]:=L[v]+w[v][i]; P[i]=v;
       end
End
```

Thuật toán Dijsktra

Bước1. i:=0, S:=V\{u_0}, L(u_0):=0, L(v):= ∞ với mọi v ∈S và đánh dấu đỉnh v bởi (∞,-). Nếu n=1 thì xuất d(u_0,u_0)=0=L(u_0)

Bước 2. Với mọi v ∈S và kề với u_i (nếu đồ thị có hướng thì v là đỉnh sau của u_i), đặt L(v):=min{L(v),L(u_i)+w(u_i v)}.

Xác định k =minL(v) ,v∈S.

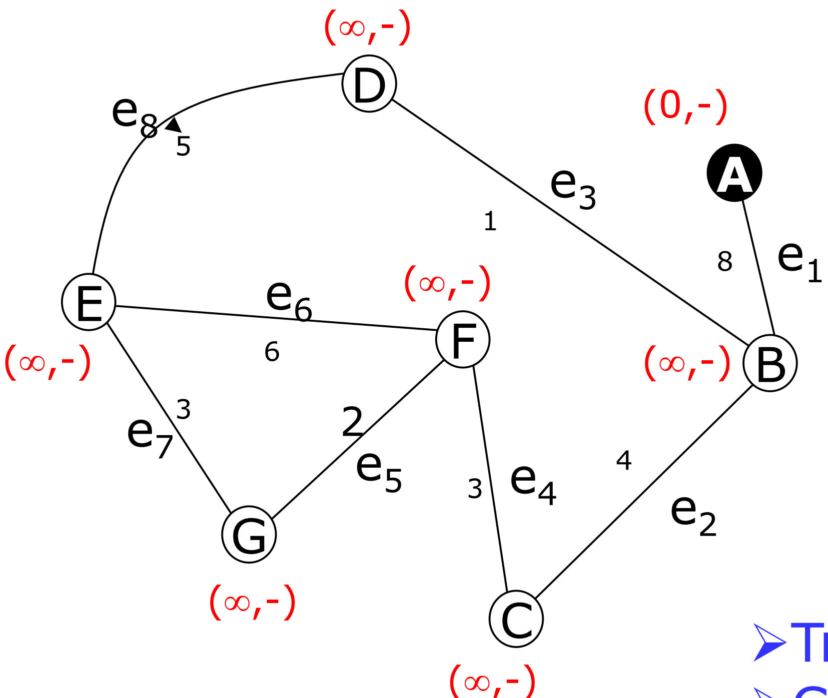
Nếu k=L(v_j) thì xuất d(u_0,v_j)=k và đánh dấu v_j bởi (L(v_j);u_i). u_{i+1}:=v_j
S:=S\{u_{i+1}}

Bước3. i:=i+1

Nếu i = n-1 thì kết thúc

Nếu không thì quay lại Bước 2

Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A

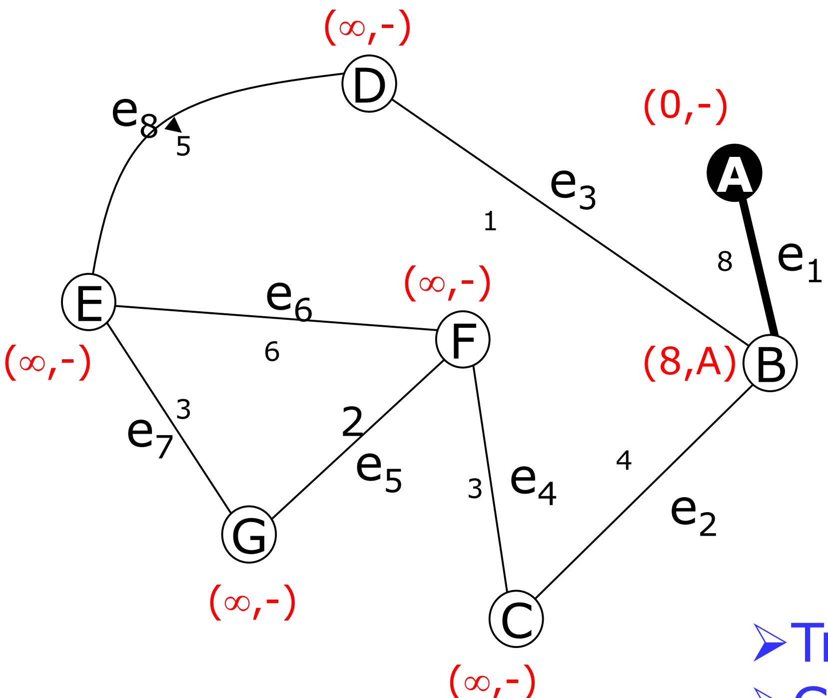


Khởi tạo

V	A	B	C	D	E	F	G
R	A	B	C	D	E	F	G
L	0	∞	∞	∞	∞	∞	∞
P	–	–	–	–	–	–	–

- 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



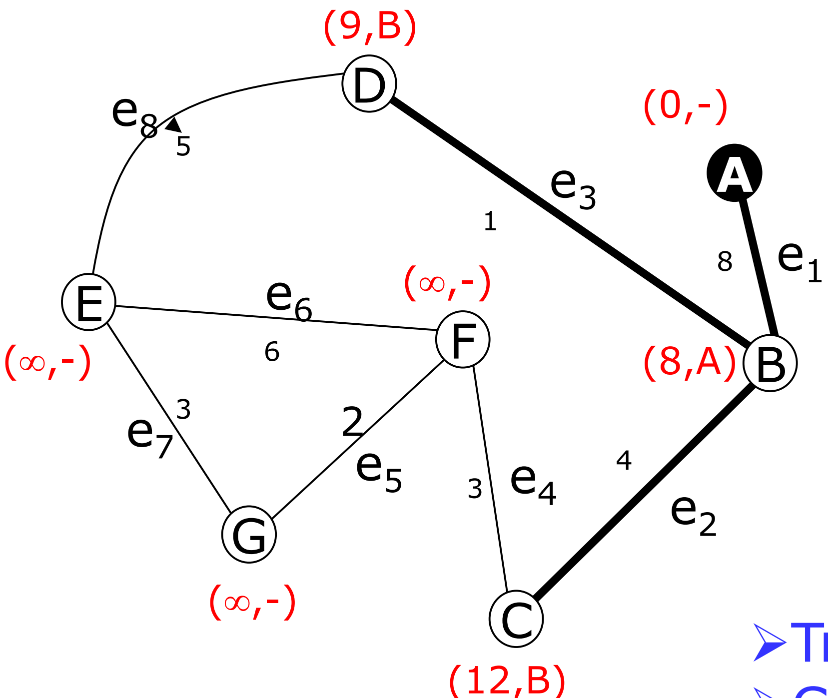
V	A	B	C	D	E	F	G
R		B	C	D	E	F	G
L	0	8	∞	∞	∞	∞	∞
P	–	A	–	–	–	–	–

- Trong R, chọn v = B
- Các đỉnh có thể được cập nhật lại:

Các đỉnh D và C

R=R-{B}

Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A

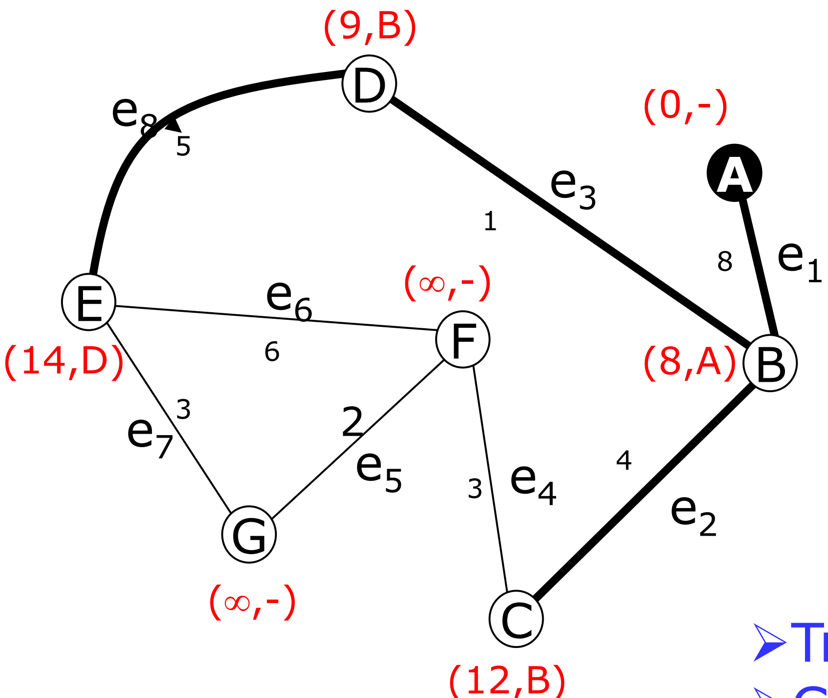


V	A	B	C	D	E	F	G
R			C	D	E	F	G
L	0	**8**	12	9	∞	∞	∞
P	–	A	B	B	–	–	–

- Trong R, chọn v = D
- Các đỉnh có thể được cập nhật lại: E

R=R-{D}

Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A

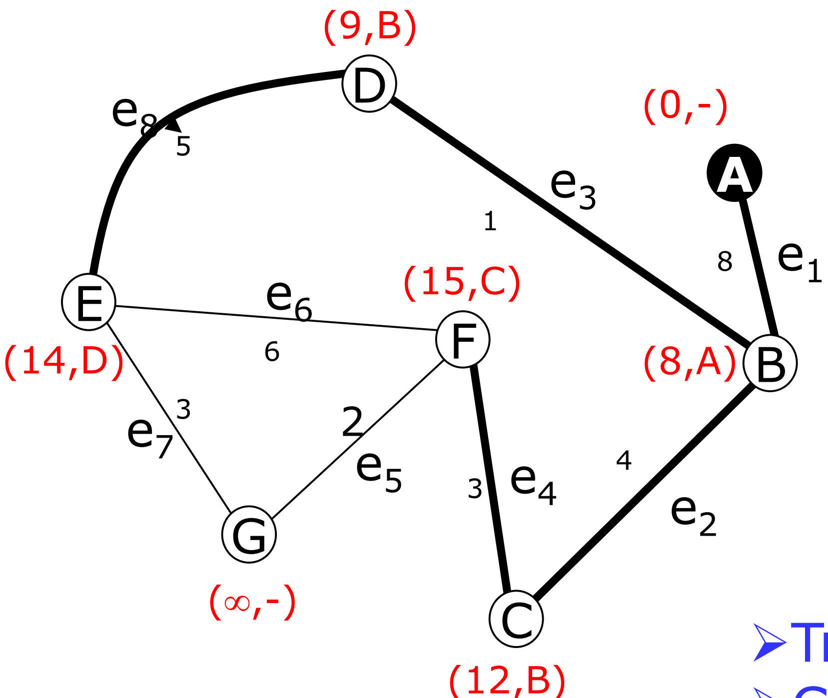


V	A	B	C	D	E	F	G
R			C		E	F	G
L	0	8	12	9	14	∞	∞
P	–	A	B	B	D	–	–

- Trong R, chọn v = C
- Các đỉnh có thể được cập nhật lại: F

R=R-{C}

Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A

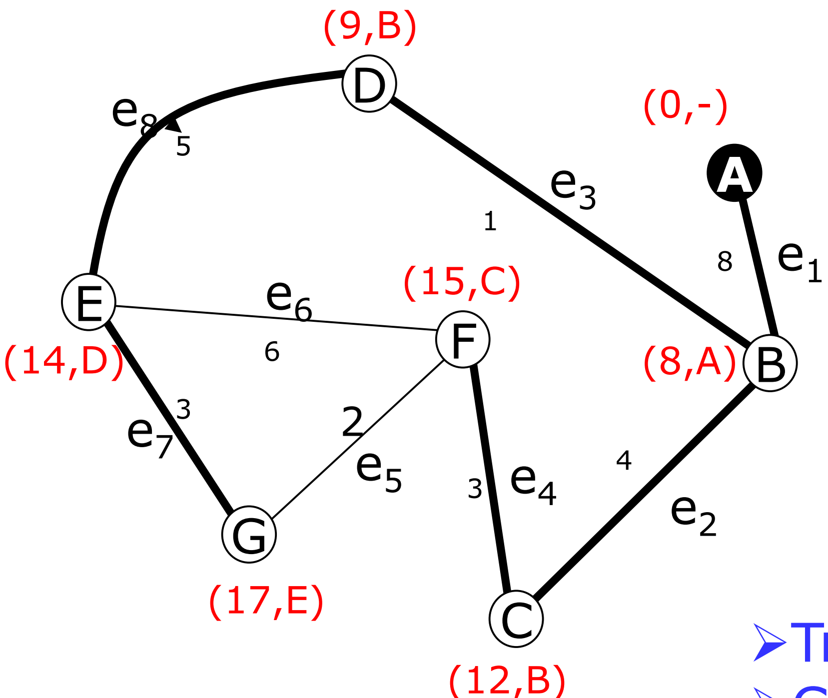


V	A	B	C	D	E	F	G
R					E	F	G
L	0	8	12	9	14	15	∞
P	–	A	B	B	D	C	–

- Trong R, chọn v =E
- Các đỉnh có thể được cập nhật lại: F,G

R=R-{E}

Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A

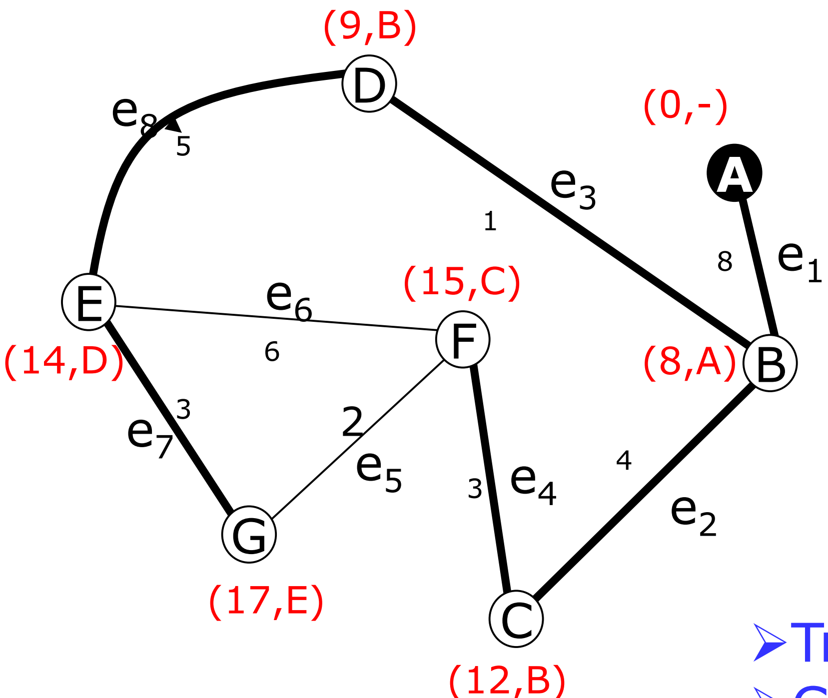


V	A	B	C	D	E	F	G
R						F	G
L	0	8	12	9	14	15	17
P	–	A	B	B	D	C	E

- Trong R, chọn v =F
- Các đỉnh có thể được cập nhật lại: G

R=R-{F}

Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A

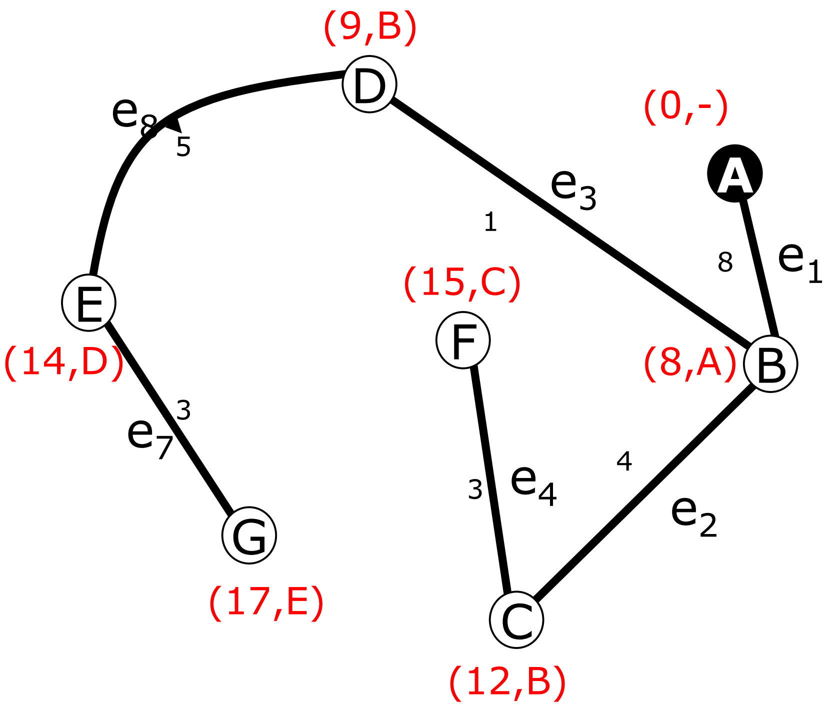


V	A	B	C	D	E	F	G
R							G
L	0	8	12	9	14	15	17
P	–	A	B	B	D	C	E

- Trong R, chọn v =G
- Các đỉnh có thể được cập nhật lại: ∅

R=R-{G}=∅. Kết thúc

Ví dụ: Tìm đường đi ngắn nhất từ đỉnh A



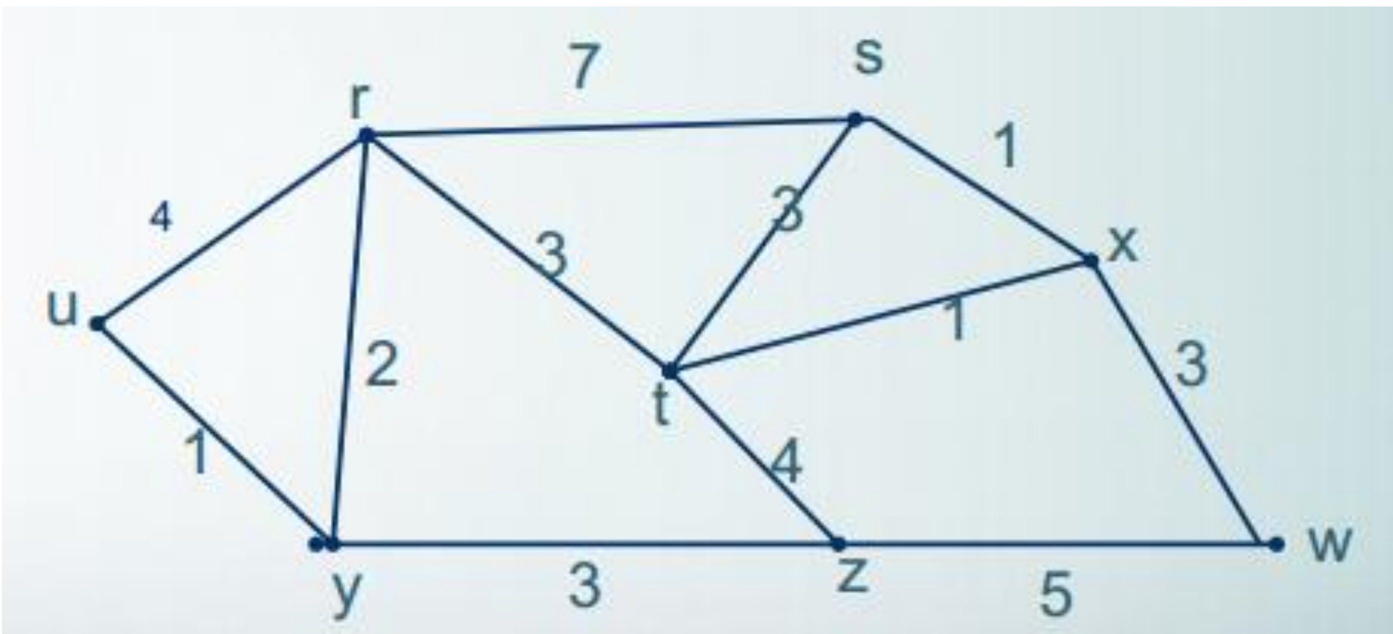
Cây bao trùm Dijkstra

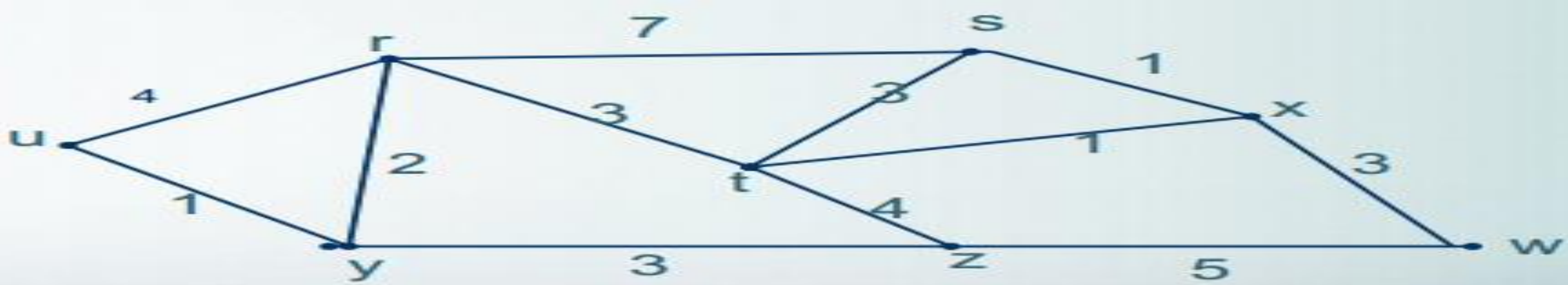
V	A	B	C	D	E	F	G
R							
L	0	8	12	9	14	15	17
P	–	A	B	B	D	C	E

Đền đỉnh	Độ dài ĐĐNN	Đường đi
B	8	A-B
C	12	A-B-C
D	9	A-B-D
E	14	A-B-D-E
F	15	A-B-CF
G	17	A-B-D-E-G

VÍ DỤ 2: Dijkstra

- Tìmđườngđi ngắn nhất từ u đến các đỉnh còn lại

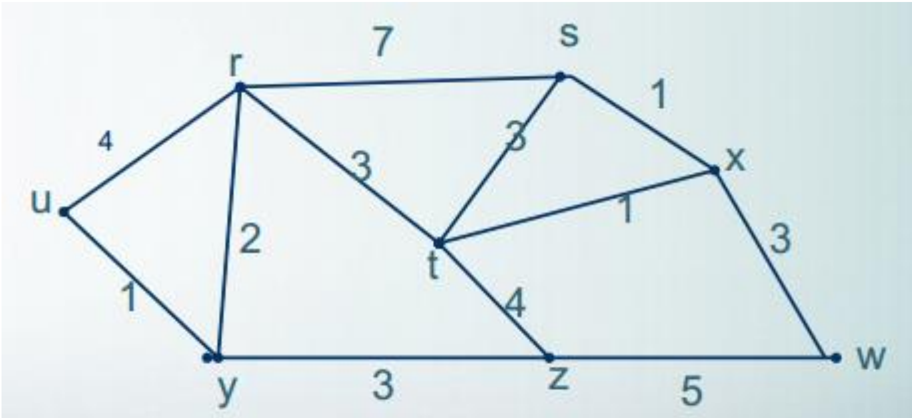


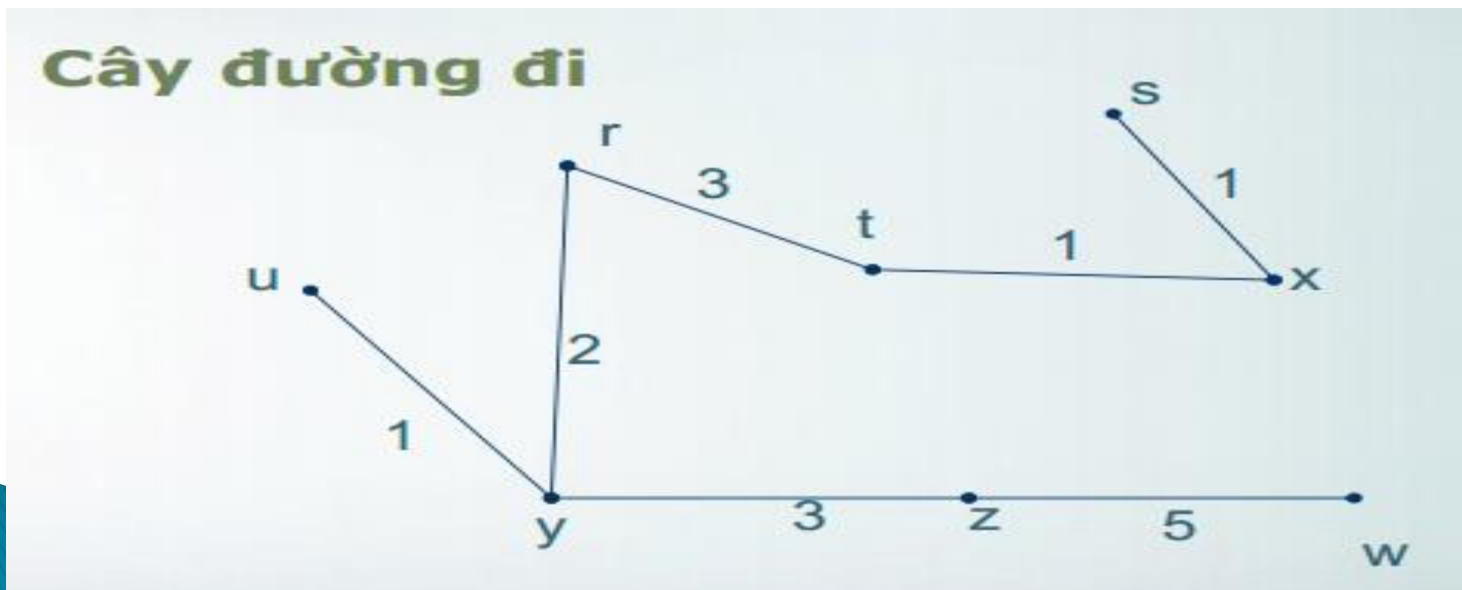


- Lập bảng thực hiện từng bước đi

BƯỚC	U	R	S	T	X	Y	Z	W
K. TẠO	0*	(∞,–)	(∞,–)	(∞,–)	(∞,–)	(∞,–)	(∞,–)	(∞,–)
1	–	(4,U)	(∞,–)	(∞,–)	(∞,–)	(1,U)*	(∞,–)	(∞,–)
2	–	(3,Y)*	(∞,–)	(∞,–)	(∞,–)	–	(4,Y)	(∞,–)
3	–	–	(10,R)	(6,R)	(∞,–)	–	(4,Y)*	(∞,–)
4	–	–	(10,R)	(6,R)*	(∞,–)	–	–	(9,Z)
5	–	–	(9,T)	–	(7,T)*	–	–	(9,Z)
6	–	–	(8,X)*	–	–	–	–	(9,Z)
7	–	–	–	–	–	–	–	(9,Z)*

VÍ DỤ 2: Dijkstra





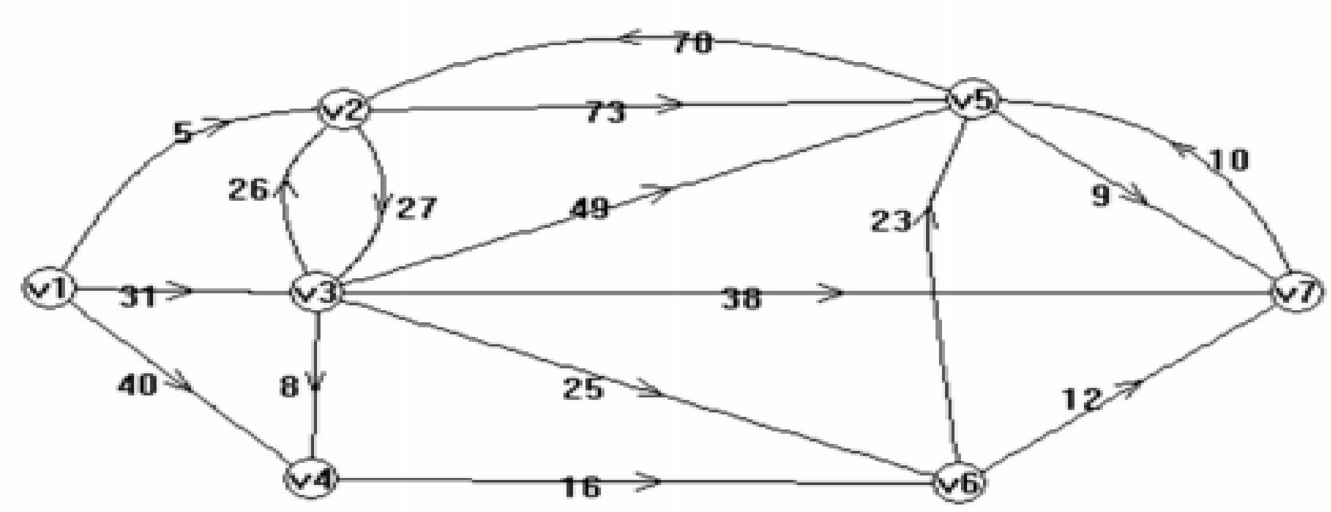
VÍ DỤ 3: Dijkstra ĐỒ THỊ CÓ HƯỚNG

- Cho đồ thị có trọng số G = (V, E) , V = { v1 , v2 , v3 , v4 , v5 , v6 , v7 } xác định bởi ma trận trọng số D. Dùng thuật toán Dijkstra tìm đường đi ngắn nhất từ v1 đến các đỉnh v2 ,v3 ,v4, v5 , v6 ,v7

$D =$

V1	V2	V3	V4	V5	V6	V7	
0	5	31	40	∞	∞	∞	V1
∞	0	27	∞	73	∞	∞	V2
∞	26	0	8	49	25	38	V3
∞	∞	∞	0	∞	16	∞	V4
∞	70	∞	∞	0	∞	9	V5
∞	∞	∞	∞	23	0	12	V6
∞	∞	∞	∞	10	∞	0	V7

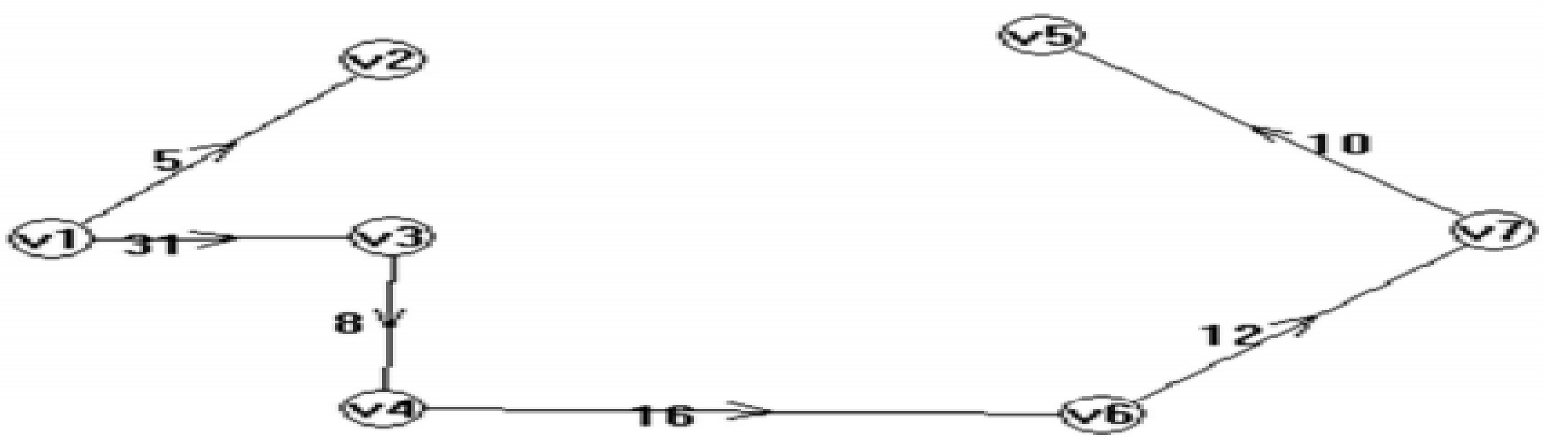
VÍ DỤ 3: Dijkstra ĐỒ THỊ CÓ HƯỚNG



BƯỚC	V1	V2	V3	V4	V5	V6	V7
K. TẠO	0*	(∞,–)	(∞,–)	(∞,–)	(∞,–)	(∞,–)	(∞,–)
1	–	(5,V1)*	(31,V1)	(40,V1)	(∞,–)	(∞,–)	(∞,–)
2	–	–	(31,V1)*	(40,V1)	(78,V2)	(∞,–)	(∞,–)
3	–	–	–	(39,V3)*	(78,V2)	(56,V3)	(69,V3)
4	–	–	–	–	(78,V2)	(55,V4)*	(69,V3)
5	–	–	–	–	(78,V2)	–	(67,V6)*
6	–	–	–	–	(77,V7)*	–	–

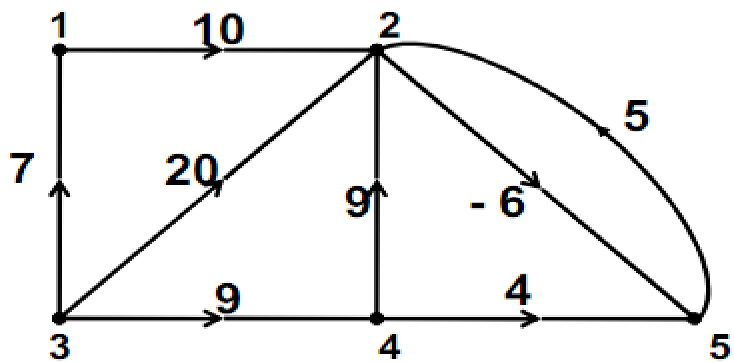
VÍ DỤ 3: Dijkstra ĐỒ THỊ CÓ HƯỚNG

BƯỚC	V1	V2	V3	V4	V5	V6	V7
K. TẠO	0*	(∞,–)	(∞,–)	(∞,–)	(∞,–)	(∞,–)	(∞,–)
1	–	(5,V1)*	(31,V1)	(40,V1)	(∞,–)	(∞,–)	(∞,–)
2	–	–	(31,V1)*	(40,V1)	(78,V2)	(∞,–)	(∞,–)
3	–	–	–	(39,V3)*	(78,V2)	(56,V3)	(69,V3)
4	–	–	–	–	(78,V2)	(55,V4)*	(69,V3)
5	–	–	–	–	(78,V2)	–	(67,V6)*
6	–	–	–	–	(77,V7)*	–	–



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



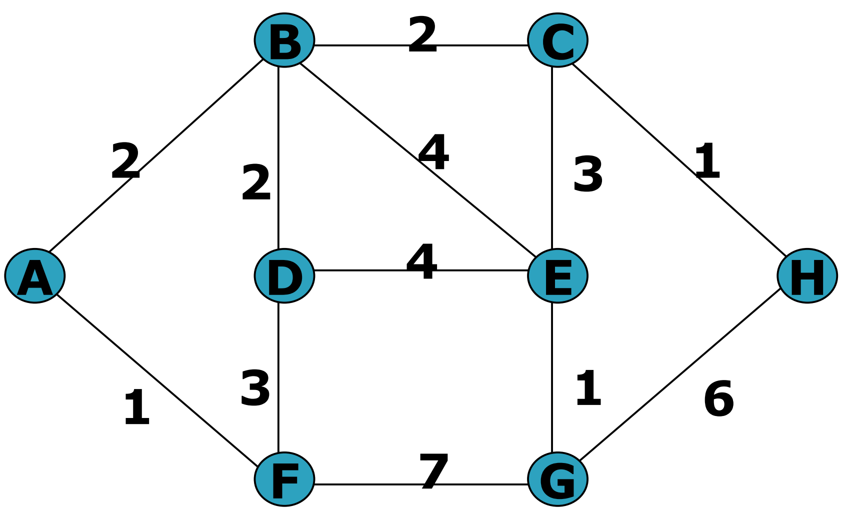
Tìm đường đi ngắn nhất từ đỉnh 3 đến đỉnh 5?

- 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

BÀI TẬP 1

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

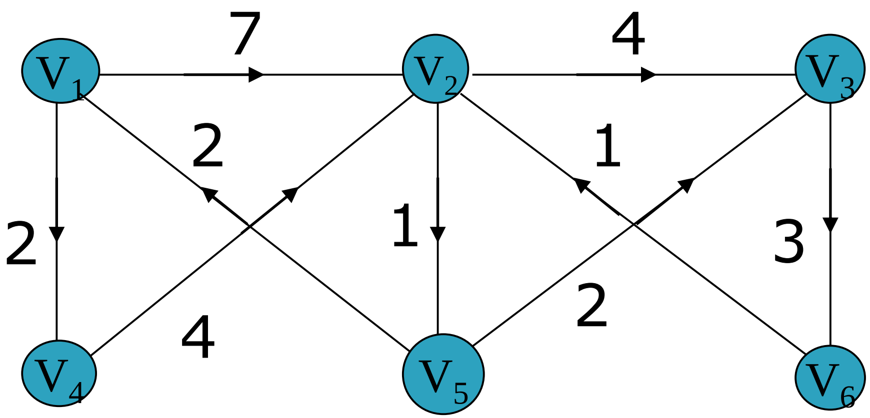
- ❑ Bắt đầu từ đỉnh A
- ❑ Bắt đầu từ đỉnh G



	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	∞	∞

BÀI TẬP 2

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=<V,E>:
  - Tập đỉnh: $V=\{v_1, v_2, \ldots, v_n\}$
  - Ma trận khoảng cách: $W = (w_{ij})$

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

- 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×n W_k ($0 \le k \le n$) 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

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

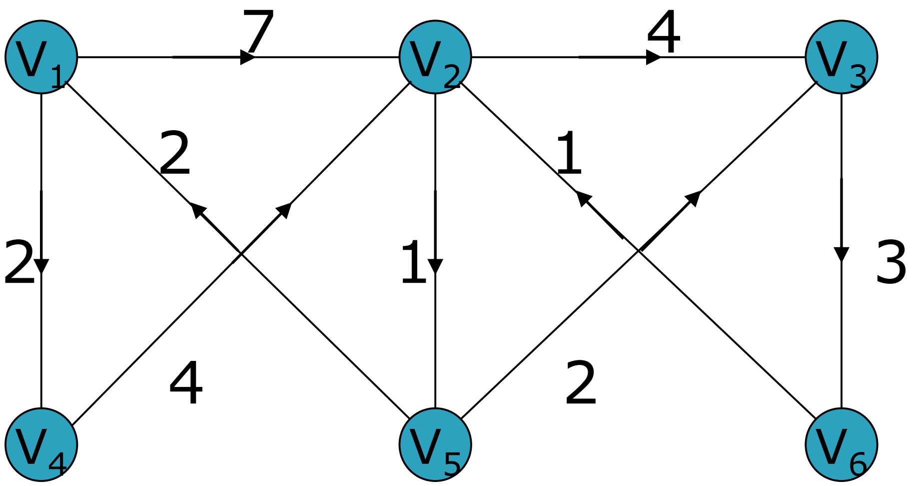
End

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.

Ví dụ



W

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

$W = W_0$

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 J=2

$W_0[5,2] > w_0[5,1] + w_0[1,2]$ Nên $\mathbf{W_1[5,2]} =$ $w_0[5,1] + w_0[1,2]$

$W = W_0$

Tính W_1 (k=1)

$W_0[1,3]$

$W_0[5,1]$

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

$W_0[5,3]$

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

k=1 i=5 J=3:

$W_0[5,3] < w_0[5,1] + w_0[1,3]$

Nên $\mathbf{W_1[5,3]} = W_0[5,3]$

Với giải thuật trên, ta tính được: W_1

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

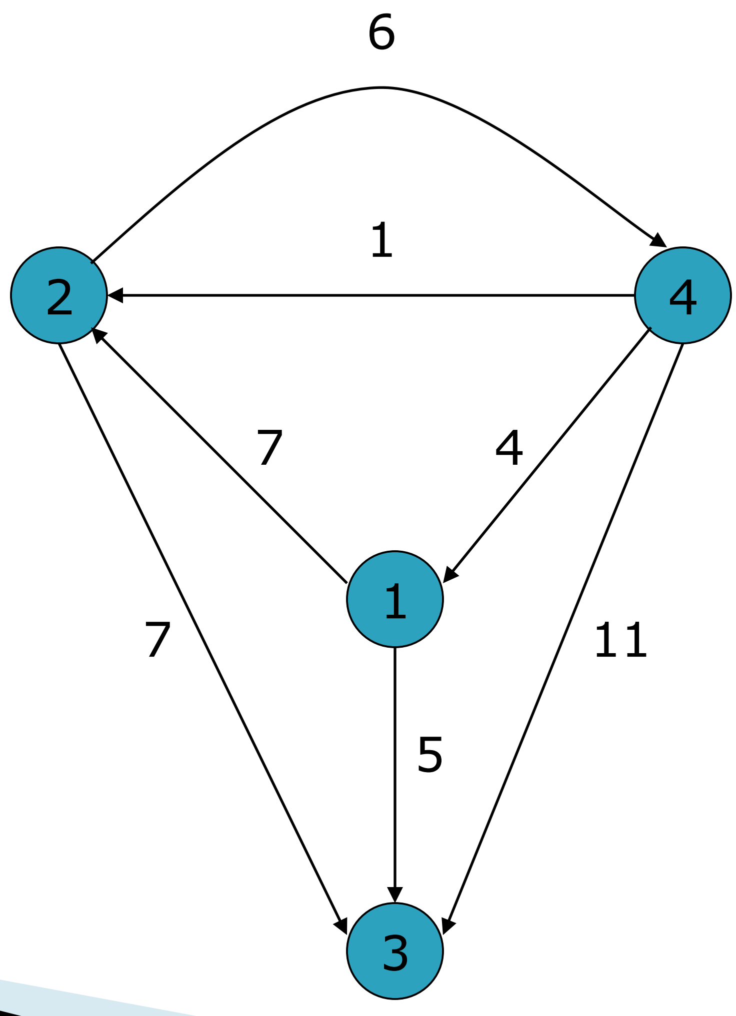
(Có xác định đường đi trong thuật toán Floyd)

- Đặt $P_0[i, j] = j$ nếu có cung $\overrightarrow{v_i v_j}$.
- $P_0[i, j] = \infty$: 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 dãy:
- i, P*[i,j],P*[P*[i,j],j], P*[P*[P*[i,j],j],j],j],…,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
```

Thí dụ



Thí dụ

$W_0 =$

∞	7	5	∞
∞	∞	7	6
∞	∞	∞	∞
4	1	11	∞

$P_0 =$

∞	2	3	∞
∞	∞	3	4
∞	∞	∞	∞
1	2	3	∞

Thí dụ

$W_1 =$

∞	7	5	∞
∞	∞	7	6
∞	∞	∞	∞
4	1	9	∞

$P_1 =$

∞	2	3	∞
∞	∞	3	4
∞	∞	∞	∞
1	2	1	∞

Thí dụ

$W_2 =$

∞	7	5	13
∞	∞	7	6
∞	∞	∞	∞
4	1	8	7

$P_2 =$

∞	2	3	2
∞	∞	3	4
∞	∞	∞	∞
1	2	2	2

Thí dụ

$W_3 =$

∞	7	5	13
∞	∞	7	6
∞	∞	∞	∞
4	1	8	7

$P_3 =$

∞	2	3	2
∞	∞	3	4
∞	∞	∞	∞
1	2	2	2

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 $(\overrightarrow{u,v})$ và $(\overrightarrow{v,u})$).
- Đồ thị G có hướng là liên thông mạnh $\Leftrightarrow \forall u,v \in V$, $u \neq v$ $W^*[u,v] < \infty$.
- Đồ thị G có hướng có chu trình $\Leftrightarrow \exists\ u \in V$, $W^*[u,u] < \infty$
- 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

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

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 Ford – Bellman

Tìm đường đi ngắn nhất từ u_0 đến các đỉnh hoặc chỉ ra đồ thị có mạch âm.

Bước 1. $L_0(u_0) = 0$ và $L_0(v) = \infty \ \forall v \neq u_0$. Đánh dấu đỉnh v bằng ($\infty$,-) ; k=1.

Bước 2. $L_k(u_0) = 0$ và

$L_k(v) = \min\{L_{k-1}(u) + w(uv) / u$ là đỉnh trước của v$\}$

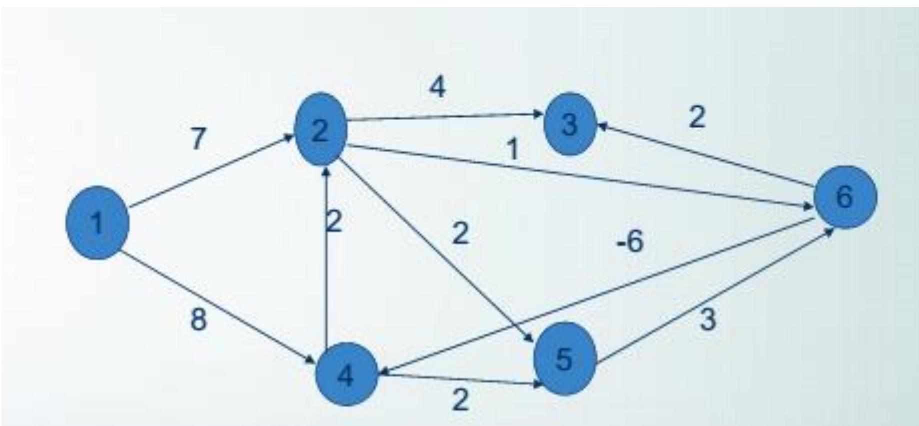
Nếu $L_k(v) = L_{k-1}(y) + w(yv)$ thì đánh dấu đỉnh v bởi $(L_k(v), y)$

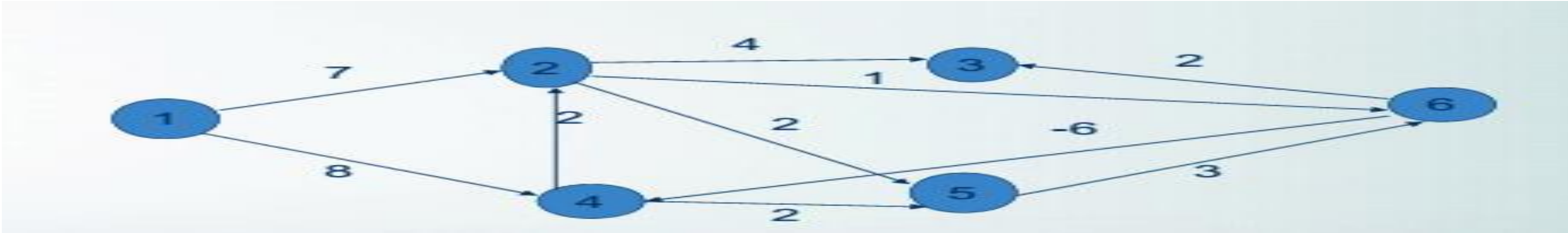
Bước 3. Nếu $L_k(v) = L_{k-1}(v)$ với mọi v, tức $L_k(v)$ ổn định thì dừng. Ngược lại đến bước 4.

Bước 4. Nếu k = n thì dừng. G có mạch âm. Nếu $k \leq n-1$ thì trở về bước 2 với k:=k+1

Ví dụ 1

- Cho đồ thị G, dùng thuật toán Ford – Bellman, tìm đường đi ngắn nhất từ đỉnh 1 đến các đỉnh của đồ thị hoặc chỉ ra 1 chu trình âm



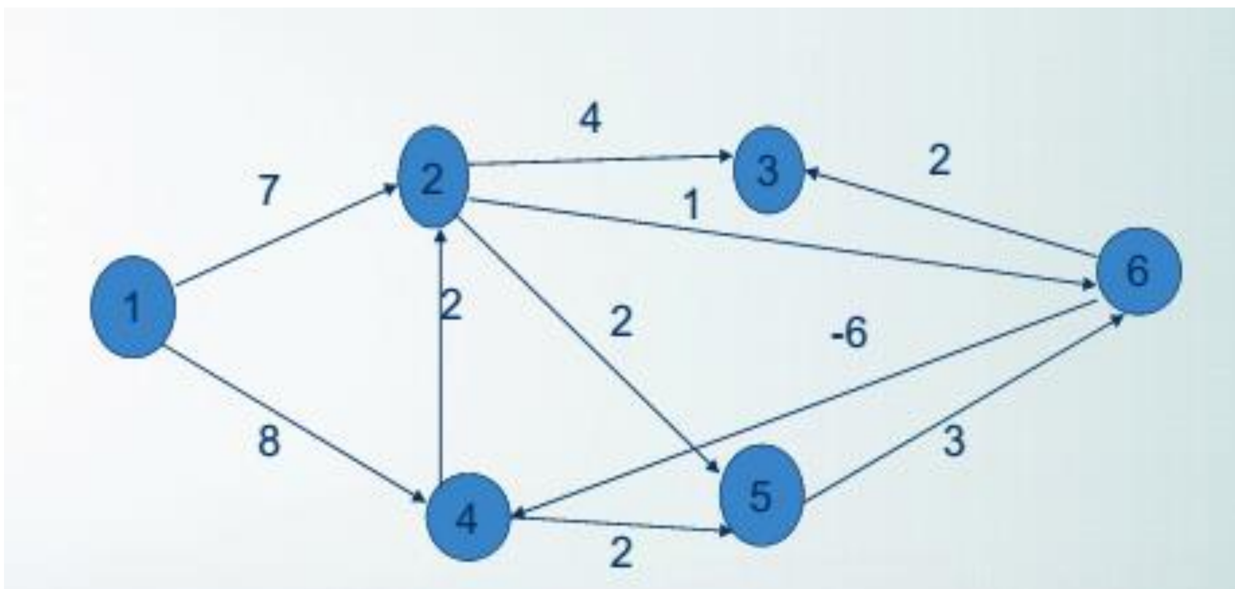


- Lập bảng chạy từng bước

k	1	2	3	4	5	6
KT	(0,-)	(∞,–)	(∞,–)	(∞,–)	(∞,–)	(∞,–)
1	(0,-)	(7,1)	(∞,–)	(8,1)	(∞,–)	(∞,–)
2	(0,-)	(7,1)	(11,2)	**(8,1)**	(9,2)	**(8,2)**
3	(0,-)	(7,1)	(10,6)	(2;6)	(9,2)	**(8,2)**
4	(0,-)	**(4,4)**	(10,6)	(2;6)	**(4;4)**	**(8,2)**
5	(0,-)	**(4,4)**	**(8,2)**	(2;6)	**(4;4)**	**(5,2)**
6	(0,-)	**(4,4)**	(7,6)	(-1,6)	**(4;4)**	**(5,2)**

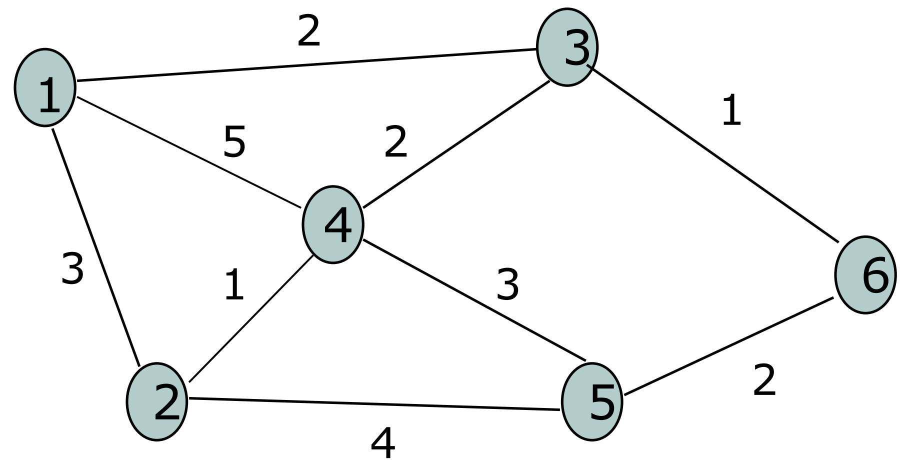
Ví dụ 1

- k = n = 6 . $L_k(i)$ chưa ổn định nênđồ thị có chu trình âm. Chẳng hạn:
  4→ 2→ 6→ 4 cóđộ dài -3



Ví dụ 2

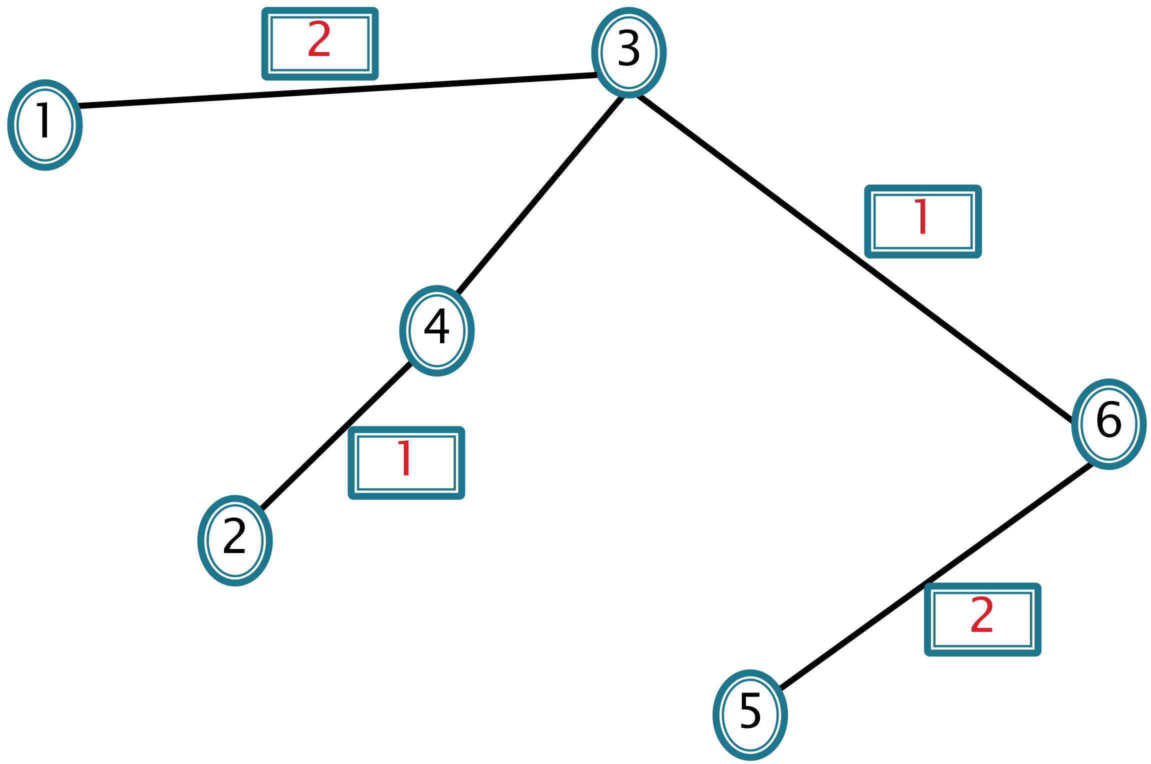
Ví dụ: s=6



k	1	2	3	4	5	6
KT	(∞,-)	(∞,–)	(∞,–)	(∞,–)	(∞,–)	(0,–)
1			(1,6)		(2,6)	(0,–)
2	(3,3)	(6,5)	(1,6)	(3,3)	(2,6)	(0,–)
3	(3,3)	(4,4)	(1,6)	(3,3)	(2,6)	(0,–)
4	(3,3)	(4,4)	(1,6)	(3,3)	(2,6)	(0,–)
5	(3,3)	(4,4)	(1,6)	(3,3)	(2,6)	(0,–)
6	(3,3)	(4,4)	(1,6)	(3,3)	(2,6)	(0,–)

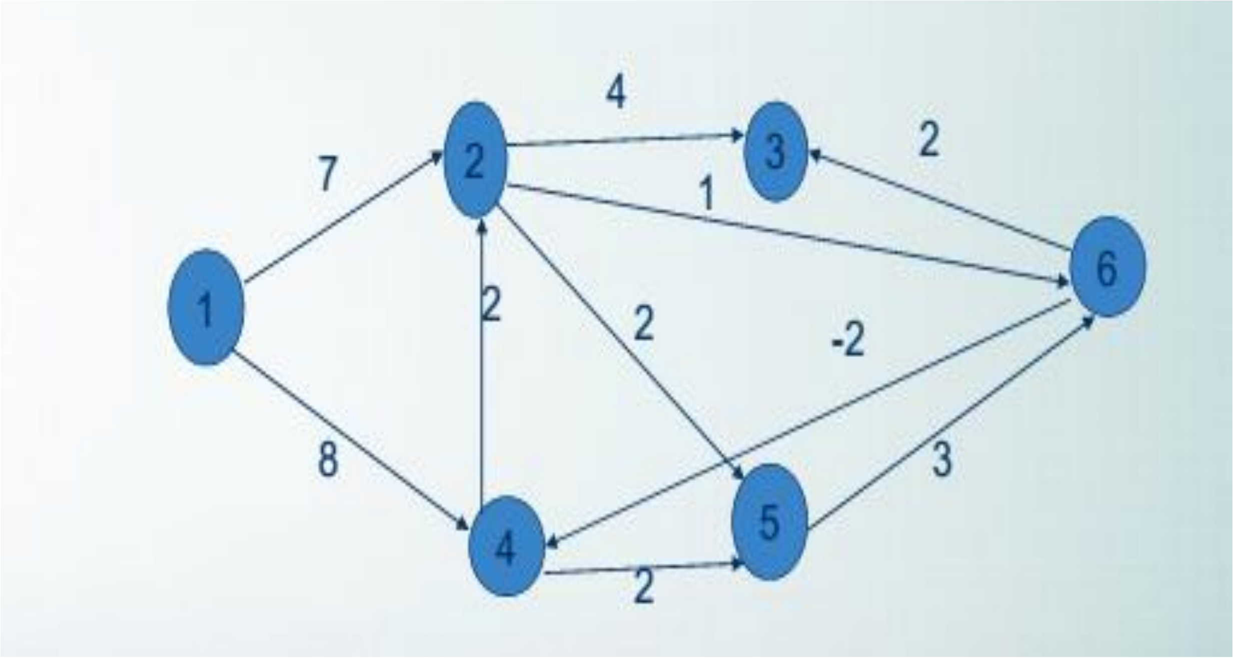
Ví dụ 2

- Kết quả



BÀI TẬP

- Cho đồ thị G, dùng thuật toán Floy –Bellman tìm đường đi từ đỉnh 1 đến các đỉnh còn lại.



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 cầu - WARSHALL

- Cho đơn đồ thị G có tập đỉnh $V=\{v_1,v_2,\ldots,v_n\}$, 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 kề $A=(a_{ij})$ của G là một ma trận Boole
- 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 cầu – WARSHALL (tt)

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

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

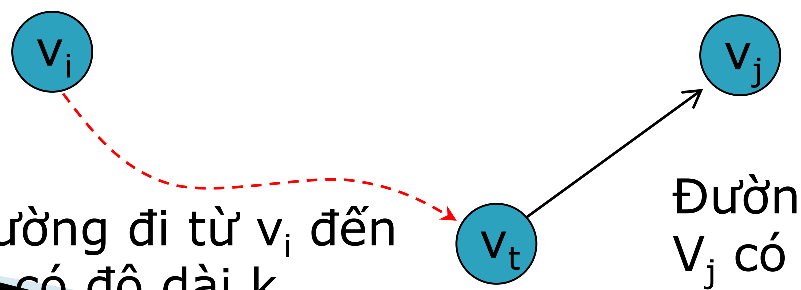
Thuật toán tính bao đóng bắt cầu – WARSHALL (tt)

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

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

- $a^{(k)}{}_{it} = 1 \Leftrightarrow$ 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 đến vj

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



Đường đi từ v_i đến V_t có độ dài k

Đường đi từ v_t đến V_j có độ dài 1 (cạnh/cung trực tiếp)

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

Ma trận khả liên $R^{(k)}=(r_{ij})$:

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

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 kề 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]);
         ⇔R[i][j]=R[i][j]||(R[i][k]&&R[j][v]);
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