Chương 2. Đường đi ngắn nhất

2.1 Biểu diễn đồ thị bằng ma trận :

2.1.1 Biểu diễn bằng ma trận kề :

2.1.1.1 Định nghĩa: Cho G=(V, E), tập các đỉnh V={0, …, n-1} . **Ma trận kề A của G** là ma trận vuông, cấp n có a_{ij} được định nghĩa:

a_{ij} số cạnh kề giữa đỉnh i và đỉnh j.

Nếu G có hướng thì

a_{ij} là số cạnh hướng từ đỉnh i đến đỉnh j .

Chú ý : *A cho ta biết cặp đỉnh có cạnh song song, đỉnh có vòng*.

2.1.2 Biểu diễn bằng ma trận liên kết :

2.1.2.1 Định nghĩa : Cho G=(V, E) , có V={ 0, 1,...,n-1} và E ={e_i : i=1,… , m} . **Ma trận liên kết A** là ma trận có a_{ij} được định nghĩa:

a_{ij} = 1 nếu e_j *liên kết* (kề) với đỉnh i và a_{ij}=0 nếu ngược lại.

Nếu G có hướng thì

a_{ij} = 1 , nếu e_j hướng ra từ đỉnh i và a_{ij}= -1 nếu e_j hướng vào đỉnh i.

→ Vòng ở đỉnh i được xem là *hướng ra từ* i.

→ a_{ij} = 0 , nếu e_j không liên kết với v_i.

2.1.3 Biểu diễn bằng ma trận trọng số :

2.1.3.1 Định nghĩa :

- Đồ thị G=(V, E) có hướng, V = {0, 1, 2, . ., n-1}, có trọng số, không cạnh song song cùng hướng , không vòng. Ma trận trong số W được định nghĩa :

$$- W = (w_{ij}) = \begin{cases} 0 & \text{nếu } i=j, \\ \text{trọng số của cạnh có hướng } (i, j) \in E & \\ \infty & \text{nếu } i \neq j \text{ và } (i, j) \notin E. \end{cases}$$

Nếu G vô hướng thì . . .

2.2 Đường đi ngắn nhất :

2.2.1 Định nghĩa:

Cho G=(V, E), có trọng số , có hướng.

- Trọng số của một đường đi từ đỉnh v đến w là tổng trọng số các cạnh của đường đi đó.

- Đường đi ngắn nhất giữa 2 đỉnh v, w là đường đi trọng số bé nhất.

2.3 Thuật toán Dijkstra:

2.3.1 Yêu cầu :

- Đồ thị G=(V, E) có hướng, có trọng số, không cạnh song song cùng hướng , không vòng.
- $W = (w_{ij})$, $w_{ij} \geq 0$ hoặc $w_{ij} = \infty$.
- Thuật toán tìm đường đi ngắn nhất từ đỉnh s đến tất cả các đỉnh .
- Giả sử có đường đi từ s đến tất cả các đỉnh.

2.3.2 Thuật toán :

Bước 1 :

- **i = 0**
- Với mọi v , d[v]=∞;
- d[s]=0;
- p[s]= - 1;
- T=V;

Bước 2 :

i = 1 // Bước lặp thứ i.

While T $\neq\varnothing$

{

1. Chọn u $\in$T với d[u] bé nhất.

2. T=T-{u};

3. Với mỗi v $\in$ T ***kề với u*** ((u,v) $\in$ E) thực hiện (if . . .):

if (d[v] > d[u]+w_{uv})

{

31. d[v] = d[u]+w_{uv}

32. p[v] = u

}

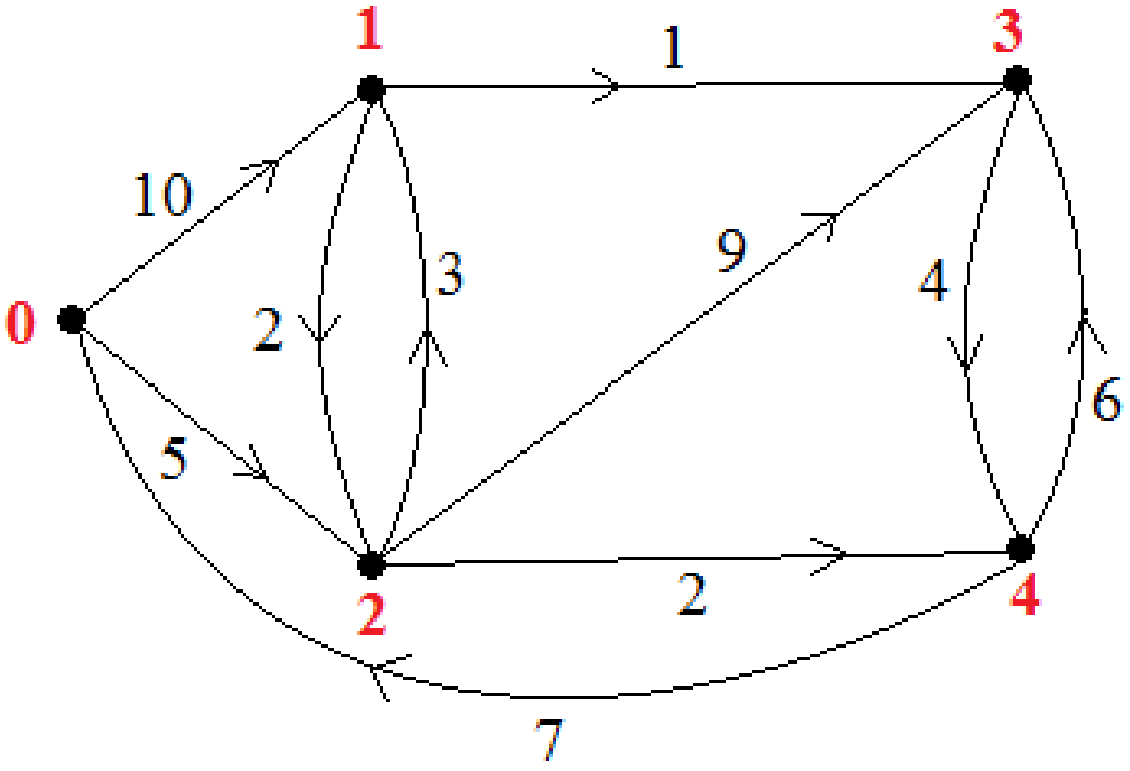
4. i = i + 1

} (Kết thúc While)

Bước 3 :

Viết d, p.

Đỉnh s = 0



Lời giải :

i	d[0]	d[1]	d[2]	d[3]	d[4]
0	**0**	∞	∞	∞	∞
1					
2					
3					
. . .					

i	p[0]	p[1]	p[2]	p[3]	p[4]
0	**-1**				
1					
2					
3					
. . .					

Qui tắc tính với ∞ :

1) Với mọi $a \in R$, $a < \infty \Rightarrow a + \infty = \infty$

2) $\infty + \infty = \infty$

3) $\infty = \infty$ (phép so sánh)

4) Với mọi $a \in R$ không phải là ∞ thì $a < \infty$ (phép so sánh)

➢ Trong cài đặt chương trình, ∞ có thể được thay thế bởi bất ký hiệu nào miễn là ký hiệu đó thỏa 4 tính chất trên.

2.4 Thuật toán Bellman-Ford :

Thuật toán tìm đường đi ngắn nhất từ s đến các đỉnh.

2.4.1 Yêu cầu:

- Đồ thị G=(V, E) có hướng, có trọng số (ma trận W).
- Không có cạnh song song cùng hướng , không có vòng.

2.4.2 Thuật toán :

Bước 1 :

- **i = 0**
- Với mỗi $v \in V$ thực hiện :

$$d[v] = \infty$$

- d[s] = 0
- p[s] = -1

Bước 2 :

for (i = 1 ; i ≤ |V|-1; i ++)

{

Với mỗi cạnh (u,v) $\in$ E thực hiện :

if (d[v] > d[u] + w_{uv})

{

d[v] = d[u] + w_{uv}

p[v] = u

}

}

Bước 3:

Với mỗi cạnh (u,v) $\in$ E thực hiện :

if (d[v] > d[u] + w_{uv})

{

Thông báo **có chu trình âm**

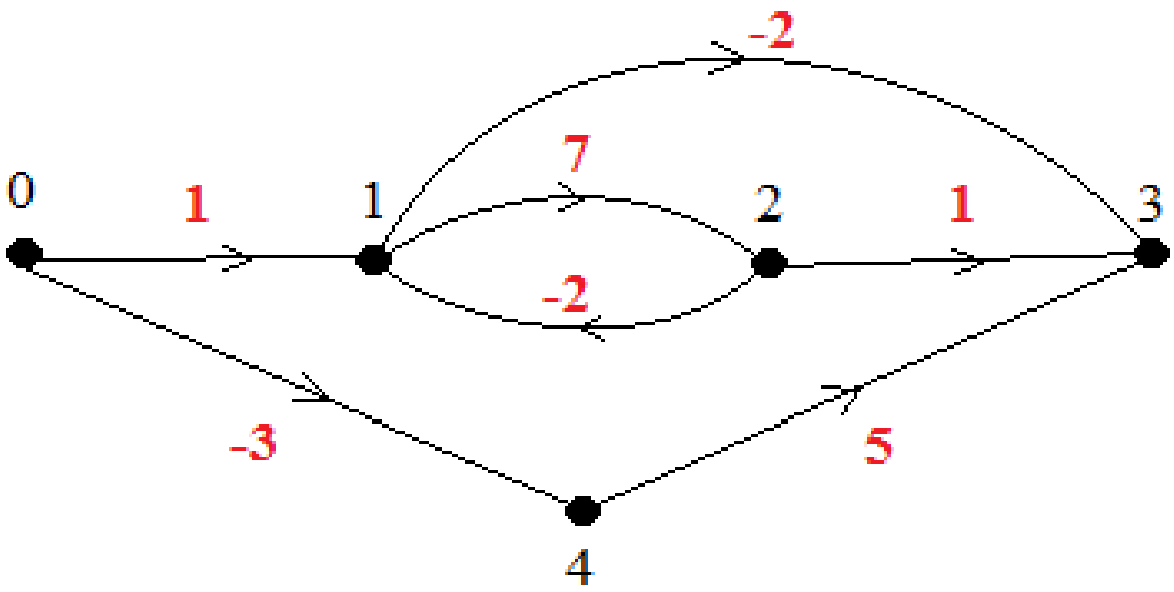
Kết thúc thuật toán

}

Bước 4 :

Viết d, p

Ví dụ :



Lời giải :

i	d[0]	d[1]	d[2]	d[3]	d[4]
0	0	∞	∞	∞	∞
1					
2					
3					
. . .					

i	p[0]	p[1]	p[2]	p[3]	p[4]
0	-1				
1					
2					
3					
. . .					

2.5 Thuật toán Floyd-Warshall :

2.5.1 Yêu cầu:

- Đồ thị G=(V, E) có hướng, có trọng số. Không cạnh song song cùng hướng , không vòng, **không chu trình âm**.

- n : số đỉnh.

- $D^{(k)} = (d_{ij}^{(k)})$. $D^{(n)} = (d_{ij}^{(n)})$ là kết quả của đường đi ngắn nhất từ đỉnh i đến đỉnh j.

- $P = (p_{ij})$, $p_{ij} = \begin{cases} -1 \text{ , nếu } i=j \text{ hay } w_{ij} = \infty \\ \\ i \text{ , nếu } i \neq j \text{ và } w_{ij} < \infty \end{cases}$

2.5.2 Thuật toán :

Bước 1 :

$D^{(0)} = W$, $P^{(0)}=P$

Bước 2 :

for (k = 1 ;k <= n ; k++) {

for (i=1; i<=n; i++) {

for (j=1; j<=n; j++){

$If\ (d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$

{ $d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$

$p_{ij}^{(k)} = p_{kj}^{(k-1)}$;

}

else { $d_{ij}^{(k)} = d_{ij}^{(k-1)}$; $p_{ij}^{(k)} = p_{ij}^{(k-1)}$ }

} *End For j*

} *End For i*

} *End For k*

Thuật toán in đường đi từ i đến j.

Print-Path(i, j)

{

If (i==j) printf(i);

Else If (P_{ij} == -1) printf("Khong co duong di i, j.")

Else {

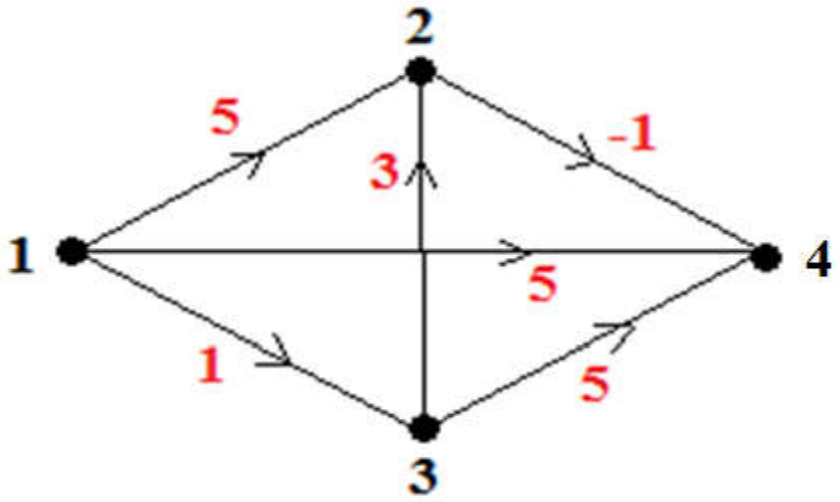
Print-Path(i, P_{ij});

printf(j);

}

}

Ví dụ :



k=1, 2, 3, 4.

$D^{(0)}$:

	1	2	3	4
1	0	5	1	5
2	vc	0	vc	-1
3	vc	3	0	5
4	vc	vc	vc	0

$P^{(0)}$:

	1	2	3	4
1	-1	1	1	1
2	-1	-1	-1	2
3	-1	3	-1	3
4	-1	-1	-1	-1

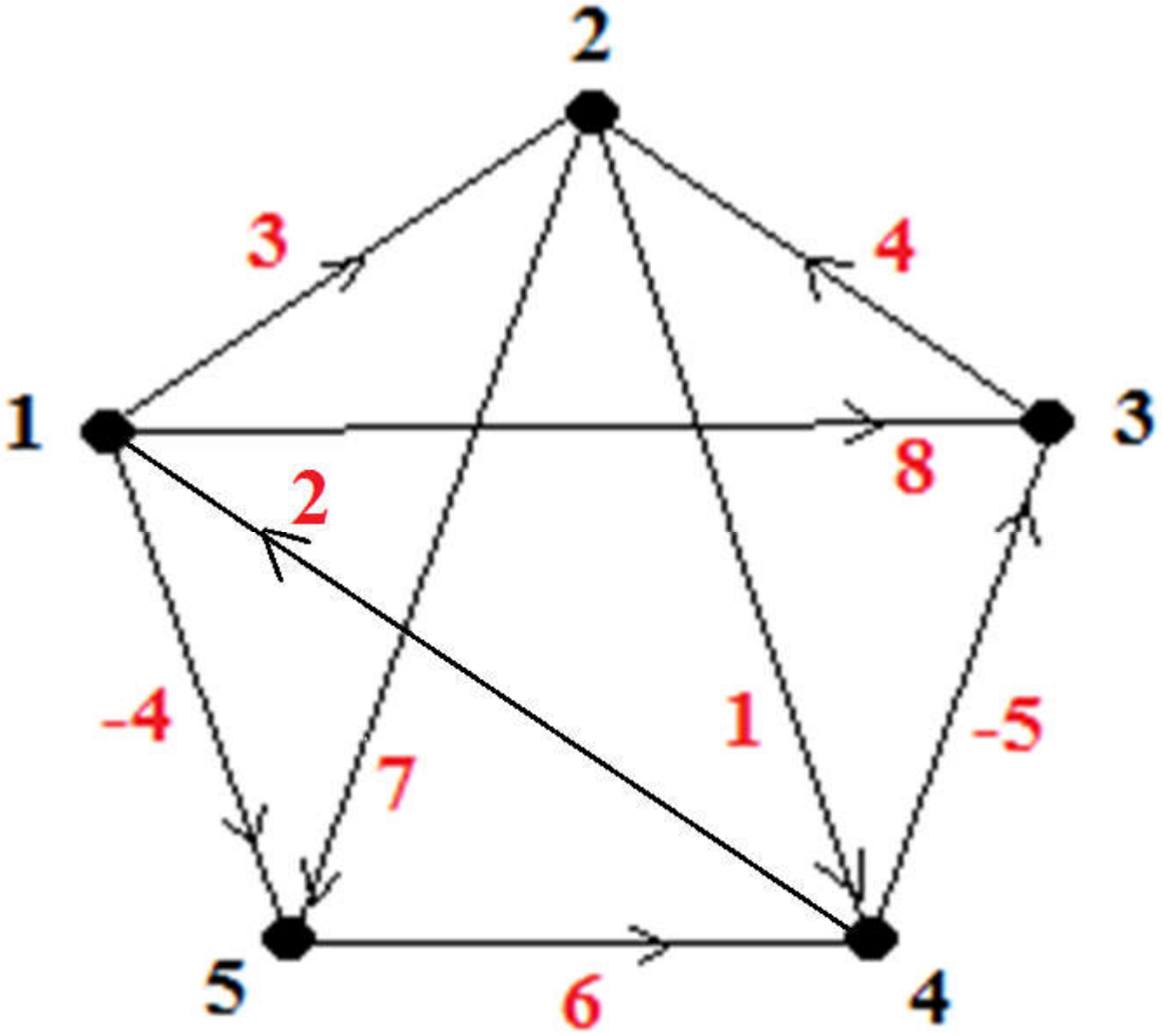
$D^{(1)}$:

	1	2	3	4
1				
2				
3				
4				

$P^{(1)}$:

	1	2	3	4
1				
2				
3				
4				

Bài tập :



Bài tập :

$D^{(0)}$ = ?, $P^{(0)}$ = ? , $D^{(5)}$ = ? , $P^{(5)}$ = ?, **Print-Path**(3, 4) = ?.

$D^{(4)}$ =

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

$P^{(4)}$=

-1	1	4	2	1
4	-1	4	2	1
4	3	-1	2	1
4	3	4	-1	1
4	3	4	5	-1

Bài giải:

Print-Path(3, 4) = ?.

$D^{(5)}=$

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

$P^{(5)}=$

-1	3	4	5	1
4	-1	4	2	1
4	3	-1	2	1
4	3	4	-1	1
4	3	4	5	-1

(Hoặc chỉ tính các phần tử dòng i).

Tài liệu tham khảo:

1. Discrete Mathematics , Richard Johnsonbaugh

2. Algorithms, Thomas h. Cormen

3. Toán Rời Rạc Nâng Cao, Trần Ngọc Danh, ĐHQG TP HCM

4. Lý Thuyết Đồ Thị, Đặng Trường Sơn, Lê văn Vinh, ĐHSP Kỹ Thuật TP HCM