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 liên thuộc với đỉnh i và đỉnh j.

i chỉ số hàng, j chỉ số cột.

Ví dụ : Ma trận kề A được lập như bảng sau :

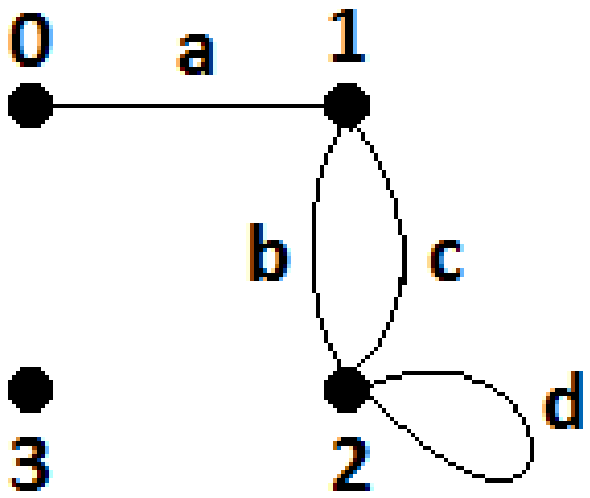
A =

	0	**1**	**2**	**3**
0	0	1	0	0
1	1	0	**2**	0
2	0	2	1	0
3	0	0	0	0

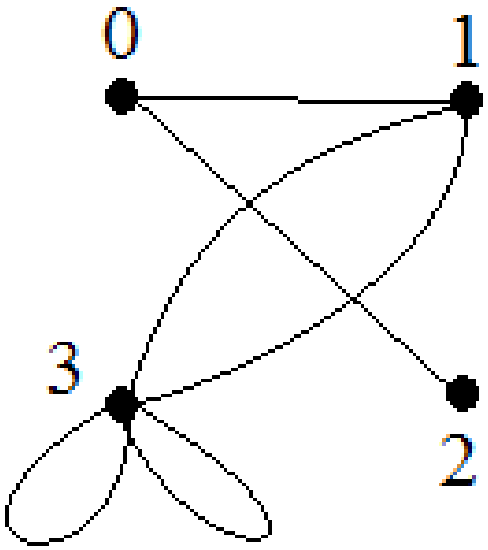
- a_{ij} số cạnh liên thuộc với đỉnh i và đỉnh j.
- i chỉ số hàng, j chỉ số cột.

$a_{12} = 2$
(đỉnh 1 và 2 có 2 cạnh liên thuộc với b và c)

$a_{22} = 1$
(đỉnh 2 có 1 cạnh liên thuộc với d)



a_{ij} số cạnh liên thuộc với đỉnh i và đỉnh j

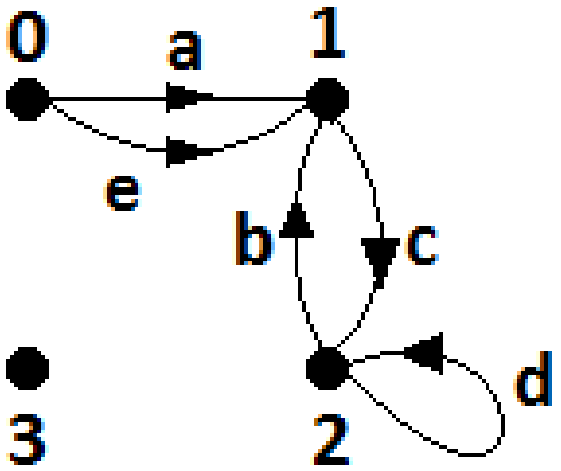


	0	1	2	3
0				
1				
2				
3				

- Ma trận kề của G có hướng :

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

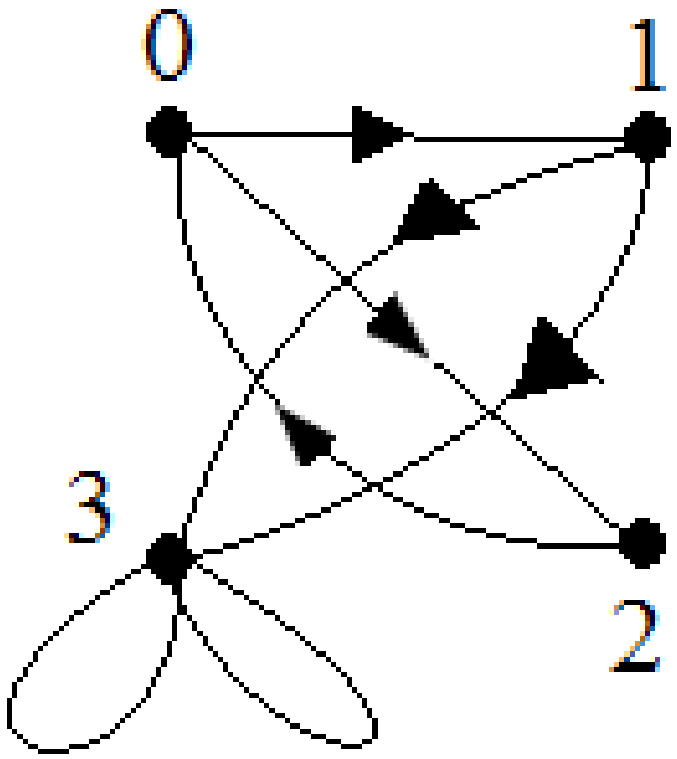
	0	1	2	3
0	0	**2**	0	0
1	0	0	1	0
2	0	1	**1**	0
3	0	0	0	0



$a_{01} = 2$
(có 2 cạnh kề a và e hướng từ đỉnh 0 đến đỉnh 1)

$a_{22} = 1$
(đỉnh 2 có 1 cạnh d hướng đến chính nó)

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



	0	1	2	3
0				
1				
2				
3				

2.1.2 Biểu diễn bằng ma trận liên thuộc :

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

a_{ij} = 1 nếu e_j *liên thuộc* 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 đỉnh* i.

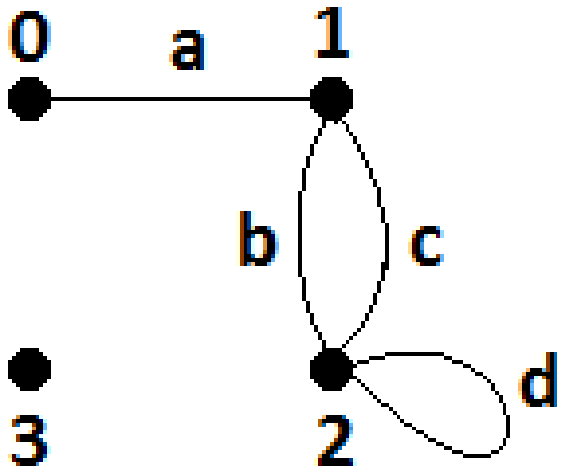
→ a_{ij} = 0 , nếu e_j không liên kết với đỉnh i.

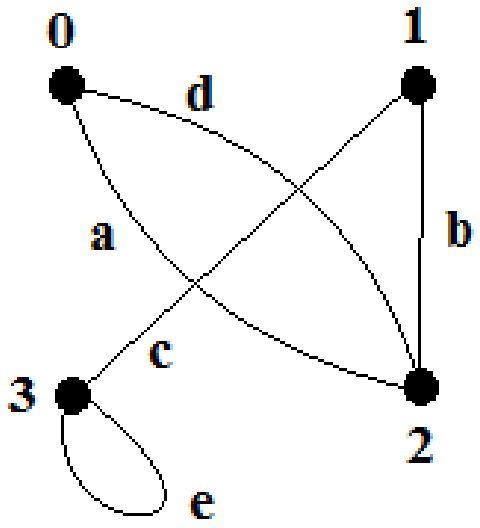
Ví dụ : Ma trận liên kết A được lập như bảng sau :

A =

	e_0	e_1	**e_2**	e_3
	a	b	**c**	d
0	**1**	0	0	0
1	1	1	1	0
2	0	1	1	1
3	0	0	0	0

$a_{1c} = 1$
(cạnh c liên thuộc với 1)





	e_0	e_1	e_2	e_3	e_4
	a	b	c	d	e
0					
1					
2					
3					

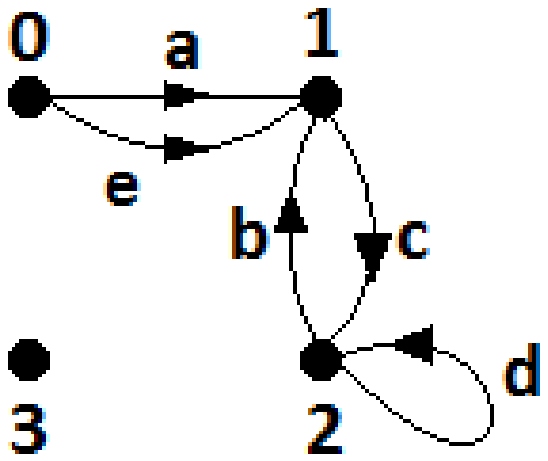
$a_{ij} = 1$ nếu e_j *liên thuộc* với đỉnh i và $a_{ij}=0$ nếu ngược lại

- Ma trận liên thuộc của G có hướng :

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

	a	b	c	d	e
0	1	0	0	0	1
1	-1	-1	1	0	-1
2	0	1	-1	1	0
3	0	0	0	0	0

$a_{1e} = -1$
(cạnh hướng đỉnh 1)



2.1.3 Biểu diễn bằng ma trận trọng số :

2.1.3.1 Định nghĩa :

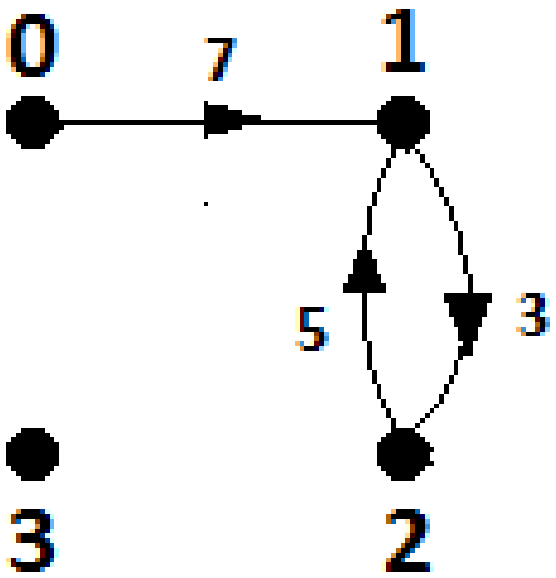
- Đồ thị G=(V, E) có hướng, V = {0, 1, 2, . ., n-1}, có trọng số, không cạnh song song cùng hướng , không vòng. Ma trận trọng số W được định nghĩa :

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

- Ma trận trọng số của G có hướng :

	0	**1**	**2**	**3**
0	0	7	∞	∞
1	∞	0	3	∞
2	∞	5	0	∞
3	∞	∞	∞	0

$a_{13} = \infty$
(không có cạnh hướng từ đỉnh 1 đến 3)



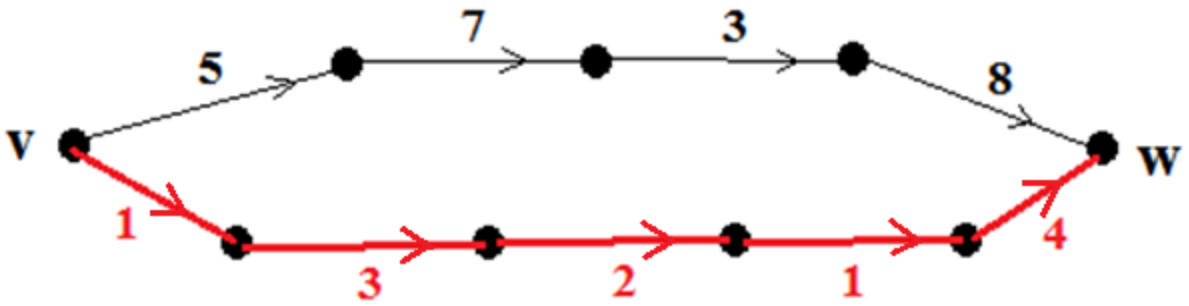
2.2 Đường đi ngắn nhất :

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.



Qui tắc tính với ∞ :

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

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

3) $\infty = \infty$ (phép so sánh)

4) Với mọi $a \in \mathbb{R}$ 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.

2.3 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.

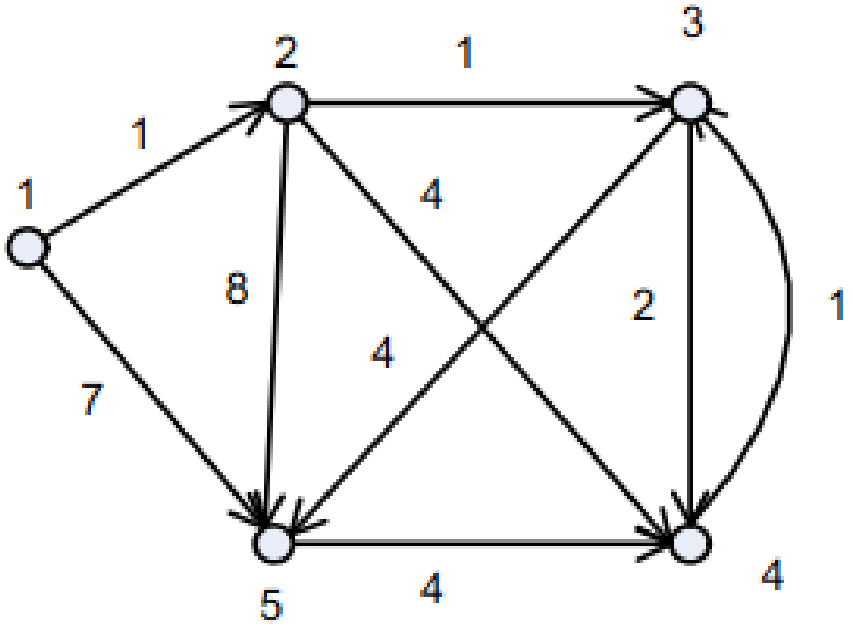
2.3.2 Thuật toán :

```
void Dijkstra
{
  for (v ∈ V)
  {
     d[v] = w[s,v];
     Truoc[v] = s;
  }

  d[s] = 0;
  T = V \ {s};
```

```
while (T != Ø)
{
  Tìm u ∈ T sao cho d[u] = min { d[z]: z ∈ T }
  T = T \ {u};
  for (v ∈ T) do
    if (d[v] > d[u] + w[u,v]) then
    {
      d[v] = d[u] + w[u,v];  Truoc[v] = u;
    }
} /* Kết thúc while (T != Ø) */
} /* Kết thúc void Dijkstra */
```

s = 1



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

T	d[2], Truoc[2]	d[3], Truoc[3]	d[4], Truoc[4]	d[5], Truoc[5]
2, 3, 4, 5	1, 1	∞, 1	∞, 1	7, 1
3, 4, 5		2, 2	5, 2	7, 1
4, 5			4, 3	6, 3
5				6, 3
∅	1, 1	2, 2	4, 3	6, 3

2.4 Thuật toán Bellman-Ford :

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

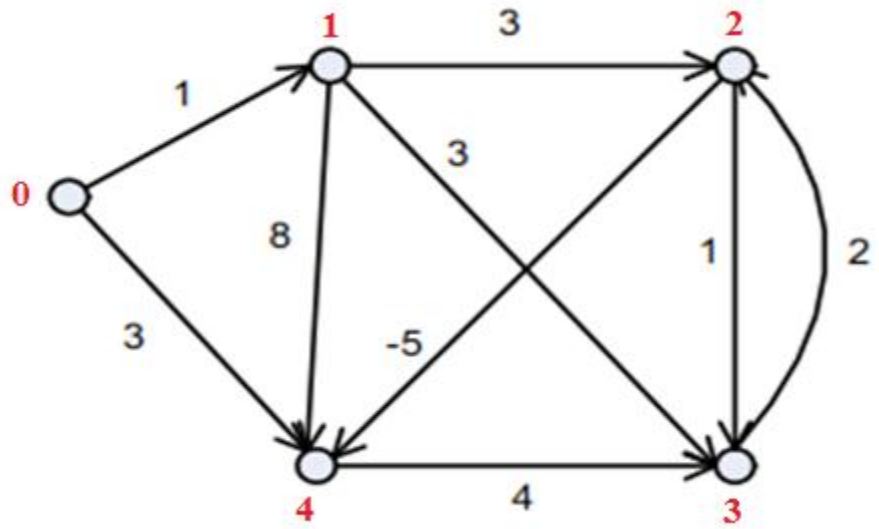
2.4.1 Yêu cầu:

- 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 :

```
void Ford_Bellman()
{
   for (v ∈ V)
   {
        d[v] = w[s,v];
        Truoc[v] = s;
   }
   d[s] = 0;
```

```
for (k = 1; k < n-1; k++) // n : số đỉnh
    for (v ∈ V \ {s})
        for (u ∈ V)
            if (d[v] > d[u] + w[u,v])
            {
                d[v] = d[u] + w[u,v];
                Truoc[v] = u;
             }
} /* Kết thúc Ford_Bellman()
```



S = 0, v = 4, 3, 2, 1, u=0, 1, 2, 3, 4

k	d[1], Truoc[1]	d[2], Truoc[2]	d[3], Truoc[3]	d[4], Truoc[4]
	1, 0	∞, 0	∞, 0	3, 0
1	1, 0	4, 1	4, 1	3, 0
2	1, 0	4, 1	3, 4	-1, 2
3	1, 0	4, 1	3, 4	-1, 2
	1, 0	4, 1	3, 4	-1, 2

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.

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

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 {

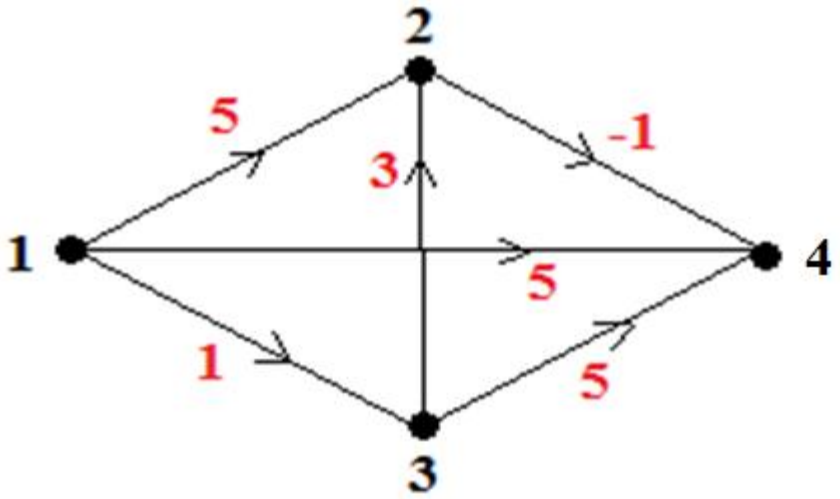
Print-Path(i, P_{ij});

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

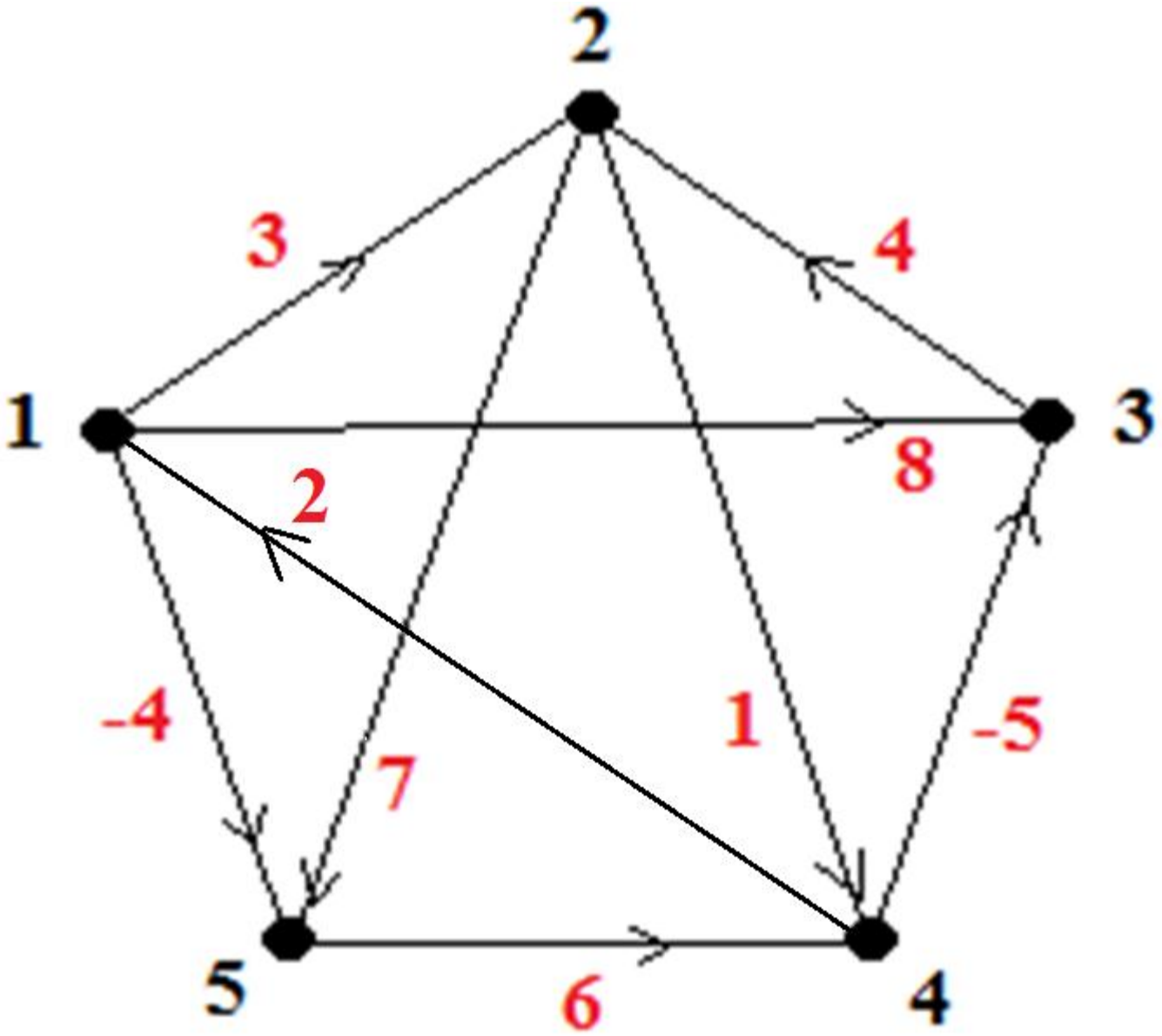
$D^{(1)}$:

	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

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