Лекція 13. Пошук найкоротшого шляху в графі

Текст слайда

§1 Постановка задачі

Задача про найкоротший шлях полягає у знаходженні найкоротшого шляху від заданої початкової вершини *a* до заданої вершини z.

Наступні дві задачі є безпосередніми узагальненнями сформульованої задачі про найкоротший шлях.

1. Для заданої початкової вершини знайти найкоротші шляхи від *a* до всіх інших вершин.
2. Знайти найкоротші шляхи між всіма парами вершин.

Розглянемо два алгоритми. Перший алгоритм розв'язує задачу 1, другий - спеціально призначений для розв'язування задачі 2.

§2 Алгоритм Дейкстри

Найефективнішим алгоритмом знаходження довжини найкоротшого шляху від фіксованої вершини до будь-якої іншої є алгоритм, запропонований 1959 р. нідерландським математиком Е. Дейкстрою (E. Dijkstra).

Цей алгоритм застосовується лише у випадку, коли **вага кожної дуги додатня**.

Нехай G=(V,E) – орієнтований граф, $w(v_i, v_j)$ – вага дуги (v_i, v_j) .

Пошук мінімального шляху здійснюється за допомогою присвоювання вершинам міток. Мітки є двох типів - тимчасові й постійні. Вершини з постійними мітками групують у множину M, яку називають *множиною позначених вершин*. Решта вершин має тимчасові мітки, і множину таких вершин позначають через T (T=V\M).

Величина постійної мітки вершини l(v) дорівнює довжині найкоротшого шляху від вершини *a* до вершини *v*. Якщо ж мітка тимчасова, то вона дорівнює довжині найкоротшого шляху, який проходить лише через вершини з постійними мітками.

Формальний опис алгоритму Дейкстри:

Крок 1. *Присвоювання початкових значень*. Виконати $l(a)=0$ і вважати цю мітку постійною. Виконати $l(v)=\infty$ для всіх $v \neq a$ і вважати ці мітки тимчасовими. Виконати $x=a$, $M=\{a\}$.

Крок 2. *Оновлення міток*. Для кожної вершини $v \in V \setminus M$ замінити мітки:

$$l(v)=\min\{ l(v), l(x)+w(x,v) \},$$

тобто оновлювати тимчасові мітки вершин, у які з вершини x іде дуга.

Крок 3. *Перетворення мітки у постійну.* Серед усіх вершин з тимчасовими мітками знайти вершину з мінімальною міткою, тобто знайти вершину v^* з умови

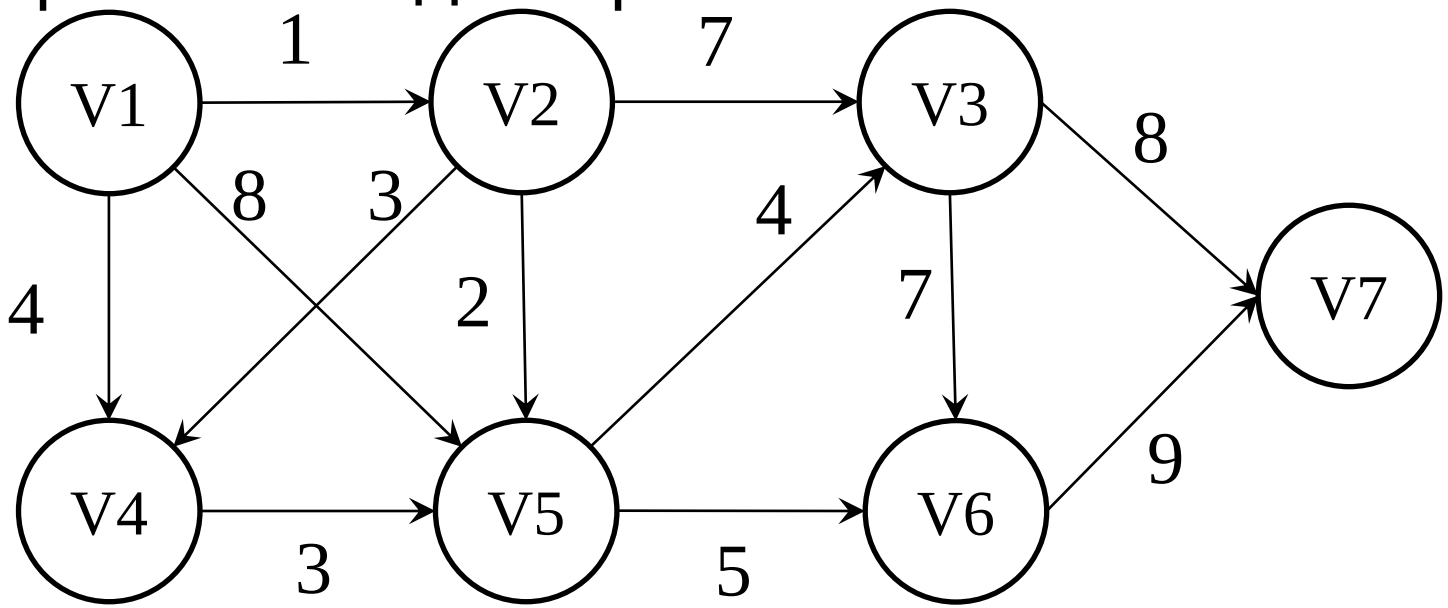
$$l(v^*)=\min l(v),\ v\in T,\quad T=V\setminus M.$$

Крок 4. Вважати мітку вершини v^* постійною і покласти $M=M\cup v^*$, $x=v^*$.

Крок 5. (*а*) (Якщо потрібно знайти шлях від *a* до *z*.) Якщо , $x=z$, то $l(z)$ – довжина найкоротшого шляху від *a* до *z*; зупинитись. Якщо $x\neq z$, то перейти до кроку 2.

(*б*) (Якщо потрібно знайти шлях від *a* до всіх інших вершин.) Якщо всі вершини отримали постійні мітки (включені у множину M), то ці мітки дають довжини найкоротших шляхів; зупинитись. Якщо деякі вершини мають тимчасові мітки, то перейти до кроку 2.

Приклад. Знайти найкоротший шлях від вершини V1 до вершини V7.



Початкові значення:

M={V1}, T=∅, $l(V_1)=0$, $l(V_2)=l(V_3)=\dots l(V_7)=\infty$

Крок 1. T1= {V2(1, V1),V4(4,V1),V5(8,V1)} – min V2

M1= {V1, V2(1, V1)}

Крок 2. T2= {V4(4,V1),V5(8,V1),V4(4,V2),V5(3,V2), V3(8,V2)}= {V4(4,V1),V5(3,V2),V3(8,V2)}– min V5

M2= {V1, V2(1, V1),V5(3,V2)}

Крок 3. T_3= {V4(4,V_1),V3(8,V_2),V3(7,V_5),V6(8,V_5)}=
= {V4(4,V_1),V3(7,V_5),V6(8,V_5)} – min V4

M_3= {V_1, V2(1, V_1),V5(3,V_2),V4(4,V_1)}

Крок 4. T_4= {V3(7,V_5),V6(8,V_5)} – min V3

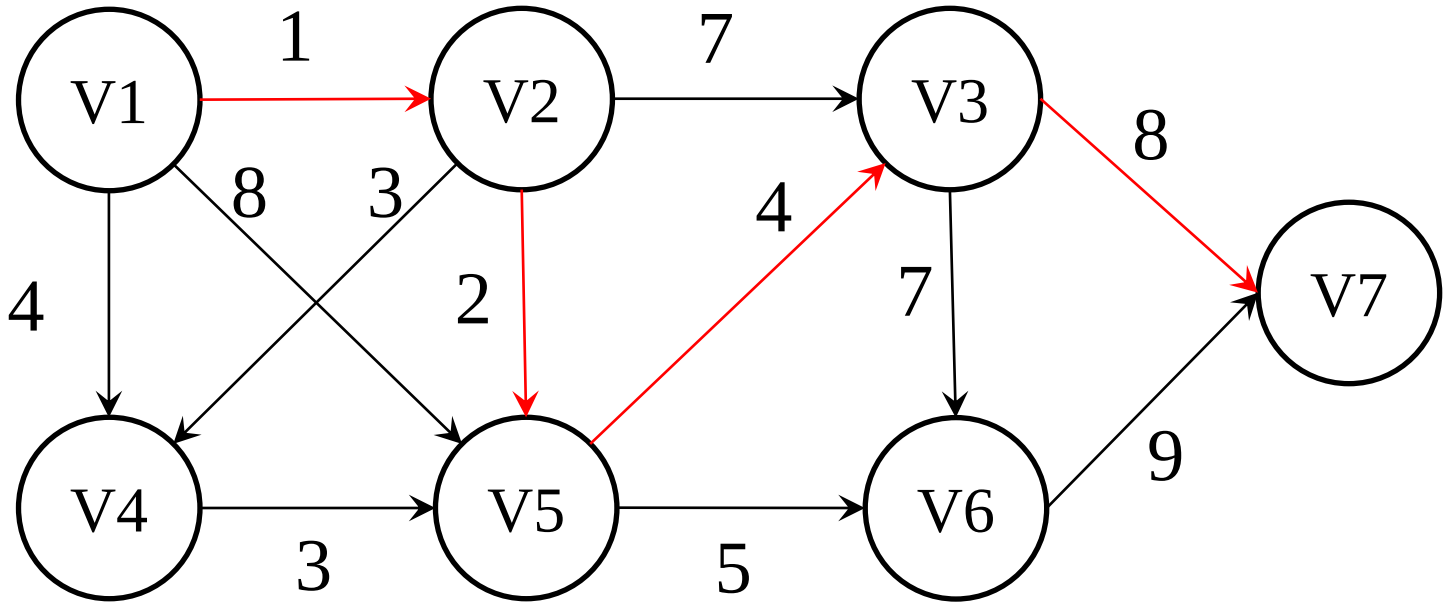
M_4= {V_1, V2(1, V_1),V5(3,V_2),V4(4,V_1),V3(7,V_5)}

Крок 5. T_5= {V6(8,V_5),V6(14,V_3),V7(15,V_3)}=
= {V6(8,V_5),V7(15,V_3)} – min V6

M_5= {V_1, V2(1, V_1),V5(3,V_2),V4(4,V_1),V3(7,V_5),
V6(8,V_5)}

Крок 6. T6= {V7(15,V_3),V7(17,V6)}= {V7(15,V_3)}

M6= {V_1, V2(1, V_1),V5(3,V_2),V4(4,V_1),V3(7,V_5),
V6(8,V_5),V7(15,V_3)}


V1
V2
V3
V4
V5
V6
V7
1
7
8
8
3
4
4
2
7
9
3
5

§3 Алгоритм Флойда

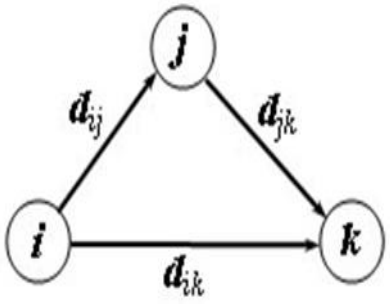
Алгоритм Флойда – алгоритм пошуку в графі найкоротших шляхів між кожною парою вершин.

В алгоритмі Флойда для довжин дуг дозволені від'ємні значення, проте не дозволена наявність циклів від'ємної довжини.

Використовує матрицю суміжності ваг та матрицю маршрутів.

- Цей алгоритм більш загальний на відміну від алгоритма Дейкстри, так як він знаходить найкоротші шляхи між будь-якии двома вузлами мережі. цьому алгоритмі мережа представлена у вигляді квадратної матриці з **n** рядками і **n** стовпчиками. Елемент **(i, j)** дорівнює відстані $\mathbf{d_{ij}}$ від вузла **i** до вузла **j**, яке має кінцеве значення, якщо існує дуга **(i, j)**, і дорівнює нескінченості в протилежному випадку.

- Покажемо спочатку ідею методу Флойда. Нехай задані три вузли **i, j** і **k** і задані відстані між ними (рис. 1). Якщо виконується нерівність $\mathbf{d_{ij} + d_{jk} < d_{ik}}$, то доцільно замінити шлях **i -> k** на шлях **i -> j -> k**. Така заміна (далі її будемо умовно називати ***трикутним оператором***) виконується систематично в процесі виконання алгоритму Флойда.



Ідея алгоритму Флойда:

Припустимо, що нам відомі:

1) найкоротший шлях з вершини *i* у вершину k , в якому як внутрішні допускають використання лише перших (k-1) вершин;

2) найкоротший шлях з вершини k у вершину *j* , у якому як внутрішні допускають використання лише перших (k-1) вершин;

3) найкоротший шлях з вершини *i* у вершину *j*, у якому як внутрішні допускають використання лише перших (k-1) вершин.

Оскільки за припущенням граф G не містить циклів від'ємної довжини, то один з двох шляхів – шлях 3) або об'єднання шляхів 1) та 2) – є найкоротшим шляхом з вершини i у вершину j, у якому як внутрішні допускають використання лише перших k вершин.

$$w_{ij}^{(k)} = \min\left\{w_{ik}^{(k-1)} + w_{kj}^{(k-1)}, w_{ij}^{(k-1)}\right\}$$

Основний алгоритм методу Флойда

- ***Крок 0.*** Визначаємо вхідну матрицю відстаней $\mathbf{D_0}$ і матрицю послідовності вузлів $\mathbf{S_0}$. Діагональні елементи обох матриць помічаються знаком "-", який показує, що ці елементи в не приймають участі в обчисленнях. Покладаємо $\mathbf{k = 1}$:

$D_0 =$

	1	2	...	j	...	n
1	—	d_{12}	...	d_{1j}	...	d_{1n}
2	d_{21}	—	...	d_{2j}	...	d_{2n}
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
i	d_{i1}	d_{i2}	...	d_{ij}	...	d_{in}
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
n	d_{n1}	d_{n2}	...	d_{nj}	...	—

$S_0 =$

	1	2	...	j	...	n
1	—	2	...	j	...	n
2	1	—	...	j	...	n
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
i	1	2	...	j	...	n
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
n	1	2	...	j	...	—

- ***Основний крок k.*** Задаємо рядок **k** і стовпчик **k** як ***головний рядок*** и ***головний стовпчик***. Розглядаємо можливість застосування трикутного оператора до всії елементів d_{ij} матриці D_{k-1}. Якщо виконується нерівність $d_{ik} + d_{kj} < d_{ij}$, **(i <> k, j <> k, i <> j)**, тоді виконуємо наступні дії:

- створюємо матрицю D_k шляхом заміни в матриці D_{k-1} елемента d_{ij} на суму $d_{ik} + d_{kj}$,

- створюємо матрицю S_k шляхом заміни в матриці S_{k-1} елемента s_{ij} на **k**. Покладаємо **k = k + 1** і повторюємо крок **k**.

- Пояснимо дії, що виконуються на **k**-му кроці алгоритму, представивши матрицю D_{k-1} так, як вона показана на рисунку 3. На цьому рисунку рядок **k** і стовпчи **k** є головними. Рядок **i** – будь-який рядок з номером від 1 до **k - 1**, а рядок **p** – довільний рядок з номером від **k + 1** до **n**. Аналогічно стовпчик **j** представляє будь-який стовпчик з номером від 1 до **k - 1**, стовпчик **q** - довільний стовпчик з номером від **k + 1** до **n**. Трикутний оператор виконується наступним чином. Якщо сума елементів головних рядка і стовпчика (що показані в квадратах) менша за суму елементів, що знаходяться на перетині стовпчика і рядка (показані в колах), що відповідають головним елементам, що розглядаюься, то відстань (елемент в колі) замінюється на суму відстаней, що представлені головними елементами:

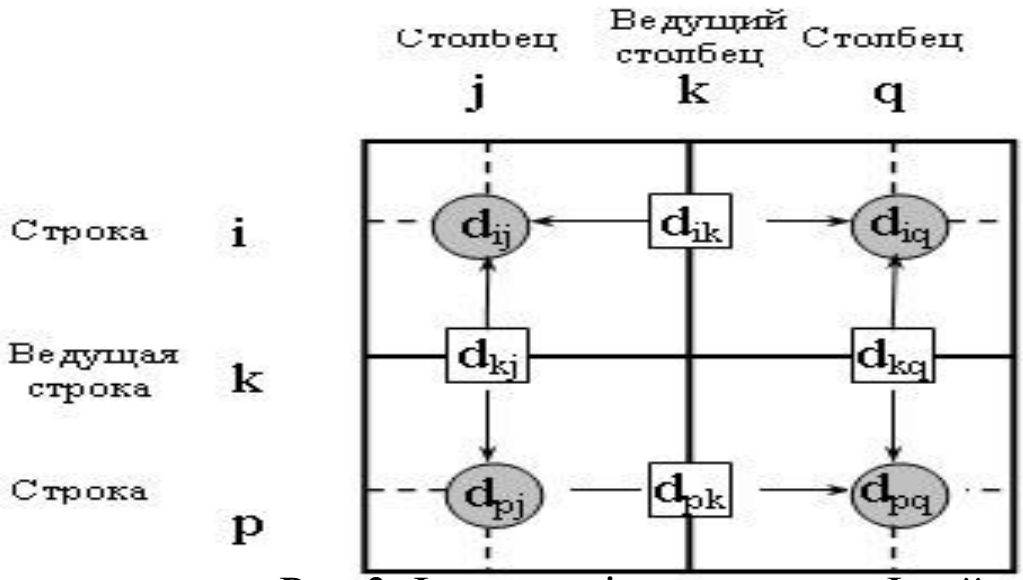
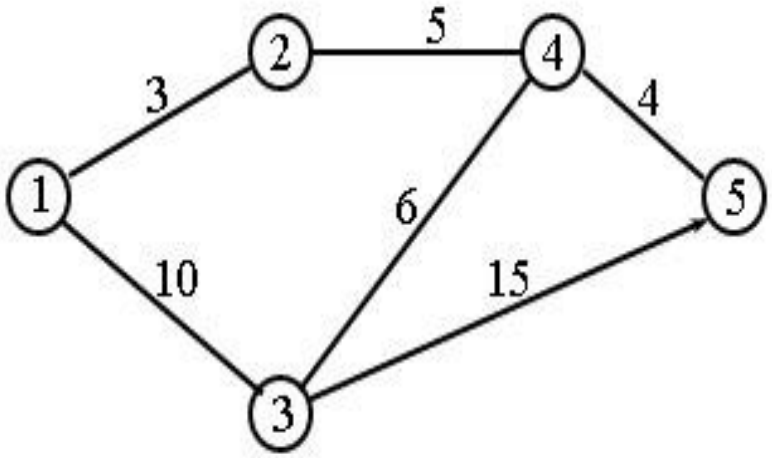


Рис.3. Ілюстрація алгоритму Флойда

- Після реалізації **n** кроків алгоритму визначення за матрицями $\mathbf{D_n}$ і $\mathbf{S_n}$ найкоротшого шляху між вузлами **i** і **j** виконується за наступними правилами.

1. Відстань між вузлами **i** і **j** дорівнює елементу $\mathbf{d_{ij}}$ в матриці $\mathbf{D_n}$.
2. Проміжні вузли шляху від вузла **i** до вузла **j** визначаємо за матрицею $\mathbf{S_n}$. Нехай $\mathbf{s_{ij} = k}$, тоді маємо шлях **i -> k -> j**. Якщо далі $\mathbf{s_{ik} = k}$ і $\mathbf{s_{kj} = j}$, тоді вважвємо, що весь шлях визначений, так як знайдені всі проміжні вузли. В іншому випадку повторюємо описану процедуру для шляхів від вузла **i** до вузла **k** і від вузла **k** до вузла **j**.

- Приклад. Знайдемо для мережі, що показана на рисунку 4, найкоротші шляхи між будь-якими двома вузлами. Відстань між вузлами цієї мережі проставлені на рисунку біля відповідних ребер. Ребро (3, 5) орієнтоване, тому не допускається рух від вузла 5 до вузла 3. Всі інші ребра допускають рух в обох напрямках:



- ***Крок 0.*** Початкові матриці $\mathbf{D_0}$ і $\mathbf{S_0}$ будуються безпосередньо за заданою схемою мережі. Матриця $\mathbf{D_0}$ симетрична, за виключенням пари елементів $\mathbf{d_{35}}$ і $\mathbf{d_{53}}$, де $\mathbf{d_{53}}$ дорівнює нескінечнності, оскільки неможливий перехід від вузла 5 до вузла 3:

D_0

	1	2	3	4	5
1	—	3	10	∞	∞
2	3	—	∞	5	∞
3	10	∞	—	6	15
4	∞	5	6	—	4
5	∞	∞	∞	4	—

S_0

	1	2	3	4	5
1	—	2	3	4	5
2	1	—	3	4	5
3	1	2	—	4	5
4	1	2	3	—	5
5	1	2	3	4	—

- ***Крок 1*.** В матриці **Ди$_0$** виділені головні рядок і стовпчик **(k = 1)**.

  Подвійною рамкою представлені елементи **d_{23}** і **d_{32}**, єдині серед елементів матриці **D_0**, значення яких можна покращати за допомогою трикутного оператора. Таким чином, щоб на основі матриць **D_0** і **S_0** отримати матриці **D_1** і **S_1**, виконуємо наступні дії.

1. Замінюємо **d_{23}** на **$d_{21} + d_{13} = 3 + 10 = 13$** і встановлюємо **$s_{23} = 1$**.
2. Замінюємо **d_{32}** на **$d_{31} + d_{12} = 10 + 3 = 13$** і встановлюємо **$s_{32} = 1$**.

- Матриці **D_1** і **S_1** мають наступний вигляд:

D_1

	1	2	3	4	5
1	—	3	10	∞	∞
2	3	—	**13**	5	∞
3	10	**13**	—	6	15
4	∞	5	6	—	4
5	∞	∞	∞	4	—

S_1

	1	2	3	4	5
1	—	2	3	4	5
2	1	—	**1**	4	5
3	1	**1**	—	4	5
4	1	2	3	—	5
5	1	2	3	4	—

- ***Крок 2*.** Покладаємо **k = 2**; в матриці $\mathbf{D_1}$ виділені наступні рядок і стовпчик. Трикутний оператор застосовується до елементів матриць $\mathbf{D_1}$ і $\mathbf{S_1}$, що виділені подвійною рамкою. В результаті отримуємо матриці $\mathbf{D_2}$ і $\mathbf{S_2}$:

D_2

	1	2	3	4	5
1	—	3	10	**8**	∞
2	3	—	13	5	∞
3	10	13	—	6	15
4	**8**	5	6	—	4
5	∞	∞	∞	4	—

S_2

	1	2	3	4	5
1	—	2	3	**2**	5
2	1	—	1	4	5
3	1	1	—	4	5
4	**2**	2	3	—	5
5	1	2	3	4	—

- ***Крок 3*.** Покладаємо **k = 3**; в матриці $\mathbf{D_2}$ виділені головні рядок і стовпчик. Трикутний оператор застосовується до елементів матриць $\mathbf{D_2}$ і $\mathbf{S_2}$, що виділені подвійною рамкою. В результаті отримуємо матриці $\mathbf{D_3}$ і $\mathbf{S_3}$:

D_3

	1	2	3	4	5
1	—	3	10	8	**25**
2	3	—	13	5	**28**
3	10	13	—	6	15
4	8	5	6	—	4
5	∞	∞	∞	4	—

S_3

	1	2	3	4	5
1	—	2	3	2	**3**
2	1	—	1	4	**3**
3	1	1	—	4	5
4	2	2	3	—	5
5	1	2	3	4	—

- ***Крок 4.*** Покладаємо **k = 4**, головні рядок і стовпчик в матриці $\mathbf{D_3}$ виділені.

  Отримуємо нові матриці $\mathbf{D_4}$ і $\mathbf{S_4}$:

D_4

	1	2	3	4	5
1	—	3	10	8	**12**
2	3	—	**11**	5	**9**
3	10	**11**	—	6	**10**
4	8	5	6	—	4
5	**12**	**9**	**10**	4	—

S_4

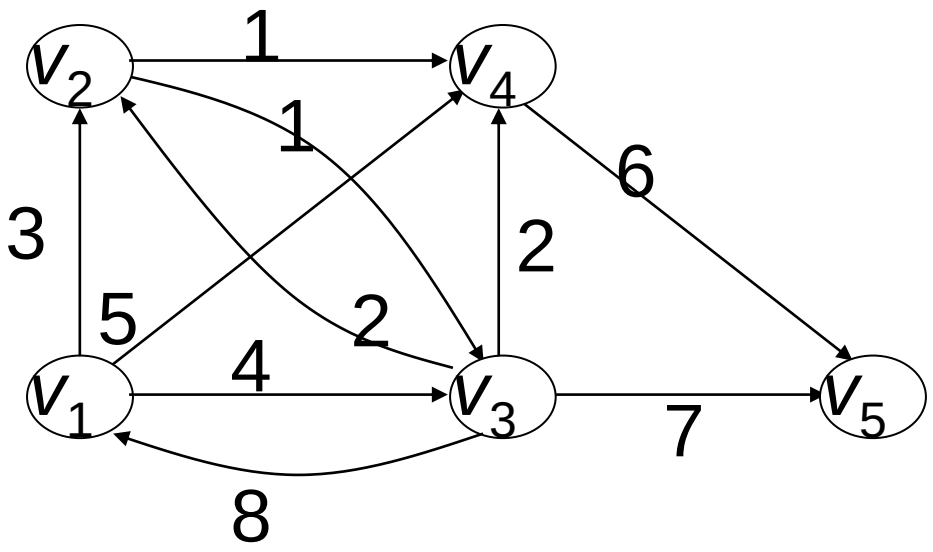
	1	2	3	4	5
1	—	2	3	2	**4**
2	1	—	**4**	4	**4**
3	1	**4**	—	4	**4**
4	**2**	2	3	—	5
5	**4**	**4**	**4**	4	—

- ***Крок 5*.** Покладаємо **k = 5**, головні рядок і стовпчик в матриці $\mathbf{D_4}$ виділені. Ніяких дій на цьому кроці не виконуємо; обчислення закінчені.

- Кінцеві матриці $\mathbf{D_4}$ і $\mathbf{S_4}$ мають всю інформацію, необхідну для визначення найкоротших шляхів між будь-якими двома вузлами мережі. Наприклад, найкоротша відстань між вузлами 1 і 5 дорівнює $\mathbf{d_{15} = 12}$.

- Для знаходження відповідних маршрутів нагадаємо, що сегмент маршруту **(i, j)** складається з ребра **(i, j)** тільки в тому випадку, коли $\mathbf{s_{ij} = j}$. В протилежному випадку вузли **i** і **j** зв`язані, меншою мірою, через один проміжний вузол. Наприклад, оскільки $\mathbf{s_{15} = 4}$ і $\mathbf{s_{45} = 5}$, спочатку найкоротший маршрут між вузлами 1 і 5 буде мати вигляд **1->4->5**. Але так як $\mathbf{s_{14}}$ не дорівнює 4, вузли 1 і 4 у визначеному шляху не зв`язані одним ребром (але у вхідній мережі вони можуть бути зв`язані безпосередньо). Далі слід визначити проміжний вузол (вузли) між першим і четвертим вузлами. Маємо $\mathbf{s_{14} = 2}$ і $\mathbf{s_{24} = 4}$, тому маршрут **1->4** замінюємо **1->2->4**. Оскільки $\mathbf{s_{12} = 2}$ і $\mathbf{s_{24} = 4}$, інших проміжних вузлів немає. Комбінуючи визначені сегменти маршруту, нарешті отримаємо наступний найкоротший шлях від вузла 1 до вузла 5: **1->2->4->5**. Довжина цього шляху дорівнює 12 кілометрам.

Приклад. Знайти найкоротший шлях між всіма парами вершин.



Матриця суміжності ваг

	v_1	v_2	v_3	v_4	v_5
v_1	0	3	4	5	∞
v_2	∞	0	1	1	∞
v_3	8	2	0	2	7
v_4	∞	∞	∞	0	6
v_5	∞	∞	∞	∞	0

$k = 1$

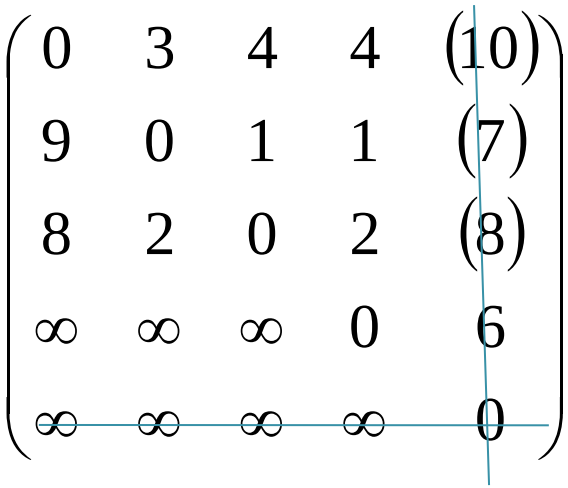
Матриця маршруів

	v_1	v_2	v_3	v_4	v_5
v_1	1	1	1	1	1
v_2	2	2	2	2	2
v_3	3	3	3	3	3
v_4	4	4	4	4	4
v_5	5	5	5	5	5

$$\begin{pmatrix} 0 & 3 & 4 & 5 & \infty \\ \infty & 0 & 1 & 1 & \infty \\ 8 & 2 & 0 & 2 & 7 \\ \infty & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix} \quad k = 2 \qquad \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 3 & 4 & (4) & \infty \\ \infty & 0 & 1 & 1 & \infty \\ 8 & 2 & 0 & 2 & \infty \\ \infty & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix} \quad k = 3 \qquad \begin{pmatrix} 1 & 1 & 1 & (2) & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 3 & 4 & 4 & \infty \\ (9) & 0 & 1 & 1 & \infty \\ 8 & 2 & 0 & 2 & \infty \\ \infty & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix} \quad k = 4 \qquad \begin{pmatrix} 1 & 1 & 1 & 2 & 1 \\ (3) & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix}$$



$$\begin{pmatrix} 0 & 3 & 4 & 4 & (10) \\ 9 & 0 & 1 & 1 & (7) \\ 8 & 2 & 0 & 2 & (8) \\ \infty & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

$k = 5$

$$\begin{pmatrix} 1 & 1 & 1 & 2 & (4) \\ 3 & 2 & 2 & 2 & (4) \\ 3 & 3 & 3 & 3 & (4) \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix}$$

Відповідь:

$$\begin{pmatrix} 0 & 3 & 4 & 4 & 10 \\ 9 & 0 & 1 & 1 & 7 \\ 8 & 2 & 0 & 2 & 8 \\ \infty & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 & 2 & 4 \\ 3 & 2 & 2 & 2 & 4 \\ 3 & 3 & 3 & 3 & 4 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix}$$