МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Інститут комп’ютерних наук та інформаційних технологій
Кафедра програмного забезпечення

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторної роботи № 6 «Розв’язування задач оптимізації на мережах за допомогою алгоритмів Дейкстри та Флойда»

з дисципліни «Дослідження операцій»

для здобувачів першого (бакалаврського) рівня вищої освіти

спеціальності 121 «Інженерія програмного забезпечення»

Укладачі:
д.т.н., проф., проф. кафедри
Любов ЖУРАВЧАК
к.ф.-м..н., старший викладач кафедри
Наталія ІВАСЬКО

Львів 2024

Тема роботи: Розв'язування задач оптимізації на мережах за допомогою алгоритмів Дейкстри та Флойда

Мета роботи: Ознайомитись на практиці із основними алгоритмами розв'язування потокових задач, навчитись знаходити оптимальні маршрути між вершинами мережі за допомогою модифікованих алгоритмів Дейкстри та Флойда.

6.1. Поняття мережі

У межах теорії дослідження операцій розглядається велика кількість практичних задач, які можна сформулювати як мережеві моделі та розв'язати їх за спеціальними методами лінійного програмування. Приведемо декілька конкретних прикладів.

1. Проектування газопроводу, що сполучає бурові свердловини морського базування з приймальною станцією, що знаходиться на березі. Цільова функція відповідної моделі повинна мінімізувати вартість будівництва газопроводу.

2. Пошук найкоротшого маршруту між двома містами в наявній мережі доріг.

3. Визначення максимальної пропускної спроможності трубопроводу для транспортування вугільної пульпи від вугільних шахт до електростанцій.

4. Визначення схеми транспортування нафти від пунктів нафтовидобування до нафтопереробних заводів із мінімальною вартістю транспортування.

5. Складання тимчасового графіка будівельних робіт (визначення дат початку і завершення окремих етапів робіт).

Розв'язування описаних задач (як і багатьох аналогічних) вимагає застосування різних мережевих оптимізаційних алгоритмів. Задачі такого вигляду можна сформулювати і вирішувати як задачі лінійного програмування, але їхня специфічна структура дає можливість розробити спеціальні мережеві алгоритми, більш ефективні, ніж стандартний симплекс-метод.

Мережа складається з множини вузлів, зв'язаних дугами (або ребрами), тобто зображується графом. Отже, мережа описується парою множин (N,A), де N – множина вузлів (вершин), а A – множина дуг. Наприклад, на рис. 5.1 зображено деяку мережу, у якої $N=\{1,2,3,4,5\}$, а множина $A=\{(1,2);(1,3);(2,3);(2,5);(3,4);(3,5);(4,2);(4,5)\}$.

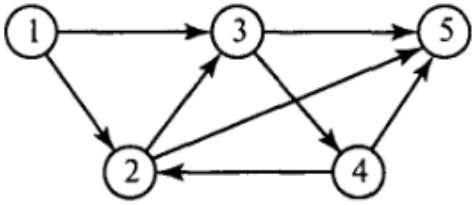


Рис. 6.1. Приклад мережі

Зв'язна мережа – це мережа, у якої будь-які два вузли зв'язані принаймні одним шляхом (дугою). На рис. 6.1 показаний саме такий тип мережі. Якщо дуги є спрямованими, то мережу називають *орієнтованою*. Якщо кожній дузі приписано деяке дійсне число (вагу), то мережу називають *зваженою*. Під *довжиною шляху* у зваженій мережі розуміють суму ваг дуг, що утворюють цей шлях, у незваженій – кількість дуг.

Зазвичай розглядають транспортні мережі. Найбільш поширеними задачами є визначення найкоротшої відстані від будь-якого пункту (вершини) до інших у заданій транспортній мережі.

Постановка задачі. Нехай у заданій зв'язній транспортній мережі кожній дузі, що виходить із точки p_i та входить у точку p_j, відповідає деяке дійсне **невід'ємне** число d_{ij} – її довжина. Треба визначити найкоротші шляхи в мережі від довільної вершини до всіх інших і вказати, через які вершини вони проходять.

Для розв'язування цієї задачі розглянемо модифіковані алгоритми Дейкстри та Флойда.

6.2. Модифікований алгоритм Дейкстри

Алгоритм Дейкстри винайдений нідерландським вченим Е. Дейкстрою в 1959 р. Його мета полягає в знаходженні найкоротшої відстані від однієї (початкової) вершини мережі до всіх інших.

Згідно з алгоритмом, кожній вершині відповідає мітка – мінімальна відома відстань від цієї вершини до початкової. Алгоритм працює покроково – на кожному кроці він "відвідує" одну вершину і намагається зменшувати мітки. Робота алгоритму завершується, коли всі вершини відвідані.

Ініціалізація. Мітка початкової вершини $d(a)=0$, мітки інших вершин – нескінченності ($d(x_i)= \infty$). Це зображає те, що відстані від початкової до інших вершин невідомі. Мітка вершини a стає **постійною** (x^*), мітки решти вершин – **тимчасовими**.

Крок 1. Якщо всі вершини мають постійну мітку, алгоритм завершений. Інакше, для всіх тимчасових міток, які суміжні з постійною вершиною (x^*), обчислюємо $d(x_i) = \min\{d(x_i), d(x^*) + c(x^*, x_i)\}$, де $c(x^*, x_i)$ – відстань від вершини x^* до вершини x_i.

Крок 2. Серед усіх тимчасових міток визначаємо постійну за правилом $x^* = \min(d(x_i))$. Переходимо до кроку 1.

Алгоритм Дейкстри дає змогу обчислити довжину найкоротшого шляху від початкової вершини *a* до заданої *t*. Для знаходження самого шляху потрібно ще накопичувати вектор вершин, із яких найкоротший шлях безпосередньо потрапляє у дану вершину. Для цього з кожною вершиною x_i, крім вершини x^*, пов'язують ще одну мітку $\theta(x_i)$. Крок 1 модифікують так.

Для кожної непереглянутої вершини x_i виконати: якщо $d(x_i) > d(x^*) + c(x^*, x_i)$, то $\theta(x_i) = x^*$. Інакше $\theta(x_i)$ не змінювати. Коли мітка $d^{(l)}(x)$ стане постійною, найкоротший шлях з *a* потраплятиме у вершину x_i безпосередньо з вершини x^*. З постійних міток $d(x_i)$ та $\theta(x_i)$ утворюємо вектори *d* та θ.

6.3 Приклад пошуку найкоротшого маршруту в мережі за модифікованим алгоритмом Дейкстри

Приклад 6.1. Знайти найкоротші маршрути з вершини *a* до решти вершин в мережі, зображеної на рис. 6.2, та вказати вершини, через які вони проходять, за допомогою модифікованого алгоритму Дейкстри.

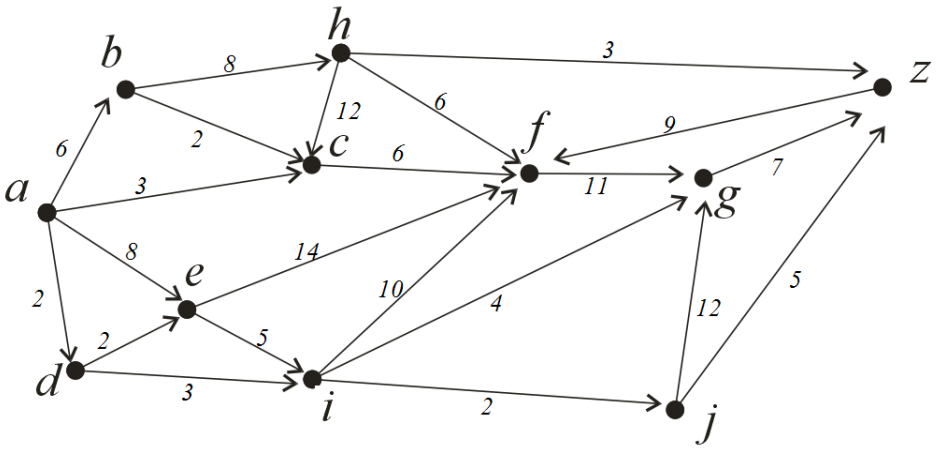


Рис. 6.2. Приклад мережі

Крок 1. Розглянемо першу вершину *a* (з якої необхідно знайти маршрут до решти вершин) – тимчасова мітка. Випишемо таблицю (табл. 6.1), у якій елементи першого рядка $d(x)$ міститимуть відстані від початкової вершини до решти, які ми переглянули, а елементи другого рядка $\theta(x)$ – вершини, через які проходить маршрут із мінімальною довжиною. На першому кроці відстань від вершини *a* до самої себе = 0, а до решти вершин = ∞, ці вершини ще не переглянуті. Другий рядок таблиці заповнюємо "-" (оскільки ще не обчислені мінімальні відстані до вершин, то і немає номерів чи назв самих вершин).

Таблиця 6.1

Вершини	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*z*
d(*x*)	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
x	-	-	-	-	-	-	-	-	-	-	-

Переглянемо усі суміжні вершини з вершиною *a*. Виконаємо обчислення:

$d(b) = \min\{d(b), d(a) + c(a,b)\} = \min\{\infty, 0 + 6\} = 6$,

$d(c) = \min\{d(c), d(a) + c(a,c)\} = \min\{\infty, 0 + 3\} = 3$,

$d(d) = \min\{d(d), d(a) + c(a,d)\} = \min\{\infty, 0 + 2\} = 2$,

$d(e) = \min\{d(e), d(a) + c(a, e)\} = \min\{\infty, 0 + 8\} = 8$.

Вершина *a* переглянута (*постійна* мітка).

Крок 2. У нову таблицю (табл. 6.2) записуємо обчислені значення елементів у перший рядок. У відповідних елементах другого рядка таблиці записуємо вершину *a.*

Таблиця 6.2

Вершини	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*z*
d(*x*)	0	6	3	2	8	∞	∞	∞	∞	∞	∞
x	-	*a*	*a*	*a*	*a*	-	-	-	-	-	-

Обираємо серед *непереглянутих* вершин у табл. 6.2 ту, яка має $\min(d(x))$. Це вершина *d* (*тимчасова* мітка).

Переглянемо усі суміжні вершини з вершиною *d*. Виконаємо обчислення:

$d(c) = \min\{d(c), d(d) + c(d,c)\} = \min\{3, 2 + \infty\} = 3$,

$d(i) = \min\{d(i), d(d) + c(d,i)\} = \min\{\infty, 2 + 3\} = 5$,

$d(e) = \min\{d(e), d(d) + c(d, e)\} = \min\{8, 2 + 2\} = 4$,

$d(b) = \min\{d(b), d(d) + c(d, b)\} = \min\{6, 2 + \infty\} = 6$.

Вершина *d* переглянута (*постійна* мітка).

Крок 3. Записуємо нову таблицю 6.3.

Таблиця 6.3

Вершини	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*z*
d(*x*)	0	6	3	2	4	∞	∞	∞	5	∞	∞
x	-	*a*	*a*	*a*	*d*	-	-	-	*d*	-	-

Обираємо серед непереглянутих вершин у табл. 6.3 ту, яка має $\min(d(x))$. Це вершина *c* (*тимчасова* мітка).

Переглянемо усі суміжні вершини з вершиною *c*. Виконаємо обчислення:

$d(f) = \min\{d(f), d(c) + c(c,f)\} = \min\{\infty, 3 + 6\} = 9$,

$d(i) = \min\{d(i), d(c) + c(c,i)\} = \min\{5, 3 + \infty\} = 5$,

$d(e) = \min\{d(e), d(c) + c(c, e)\} = \min\{4, 3 + \infty\} = 4$,

$d(b) = \min\{d(b), d(c) + c(c, b)\} = \min\{6, 3 + \infty\} = 6$.

Вершина *c* переглянута (*постійна* мітка).

Крок 4. Записуємо нову таблицю 6.4.

Таблиця 6.4

Вершини	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*z*
d(*x*)	0	6	3	2	4	9	∞	∞	5	∞	∞
x	-	*a*	*a*	*a*	*d*	*c*	-	-	*d*	-	-

Серед непереглянутих вершин у табл. 6.4, яка має $\min(d(x))$-це вершина *e* (*тимчасова* мітка).

Переглянемо усі суміжні вершини з вершиною *e*. Виконаємо обчислення:

$d(b) = \min\{d(b), d(e) + c(e, b)\} = \min\{6, 4 + \infty\} = 6,$

$d(f) = \min\{d(f), d(e) + c(e,f)\} = \min\{9, 4 + 14\} = 9,$

$d(i) = \min\{d(i), d(e) + c(e,i)\} = \min\{5, 4 + 5\} = 5.$

Вершина *e* переглянута (постійна мітка).

Так продовжуємо далі, допоки всі вершини таблиці не будуть переглянуті (табл. 6.5-6.11).

Таблиця 6.5

Вершини	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*Z*
d(*x*)	0	6	3	2	4	9	∞	∞	5	∞	∞
x	-	*a*	*a*	*a*	*d*	*c*	-	-	*d*	-	-

Таблиця 6.6

Вершини	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*z*
d(*x*)	0	6	3	2	4	9	9	∞	5	7	∞
x	-	*a*	*a*	*a*	*d*	*c*	*i*	-	*d*	*i*	-

Таблиця 6.7

Вершини	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*z*
d(*x*)	0	6	3	2	4	9	9	14	5	7	∞
x	-	*a*	*a*	*a*	*d*	*c*	*i*	*b*	*d*	*i*	-

Таблиця 6.8

Вершини	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*z*
d(*x*)	0	6	3	2	4	9	9	14	5	7	12
x	-	*a*	*a*	*a*	*d*	*c*	*i*	*b*	*d*	*i*	*j*

Таблиця 6.9

Вершини	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*z*
d(*x*)	0	6	3	2	4	9	9	14	5	7	12
x	-	*a*	*a*	*a*	*d*	*c*	*i*	*b*	*d*	*i*	*j*

Таблиця 6.10

Вершини	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*z*
d(*x*)	0	6	3	2	4	9	9	14	5	7	12
x	-	*a*	*a*	*a*	*d*	*c*	*i*	*b*	*d*	*i*	*j*

Таблиця 6.11

Вершини	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*z*
d(*x*)	0	6	3	2	4	9	9	14	5	7	12
x	-	*a*	*a*	*a*	*d*	*c*	*i*	*b*	*d*	*i*	*j*

Отже, всі вершини позначено і отримано остаточний результат (табл. 6.12).

Таблиця 6.12

Маршрут	Шлях	Довжина
a-a	-	0
a-b	$a \to b$	6
a-c	$a \to c$	3
a-d	$a \to d$	2
a-e	$a \to d \to e$	4
a-f	$a \to c \to f$	9
a-g	$a \to d \to i \to g$	9
a-h	$a \to b \to h$	14
a-i	$a \to d \to i$	5
a-j	$a \to d \to i \to j$	7
a-z	$a \to d \to i \to j \to z$	12

6.4. Модифікований алгоритм Флойда

Алгоритм Флойда – алгоритм динамічного програмування для знаходження найкоротших відстаней між усіма вершинами зваженого орієнтованого графа. Розроблений цей алгоритм у 1962 році Робертом Флойдом і Стівеном Воршеллом.

Суть алгоритму Флойда полягає у перевірці того, чи не виявиться шлях із вершини *i* у вершину *j* коротшим, якщо він буде проходити через деяку проміжну вершину. Алгоритм Флойда реалізується з використанням двох матриць: матриці найкоротших довжин *D* та матриці шляхів *S*. Зауважимо, що в цьому алгоритмі довжини дуг можуть бути *від'ємними*, однак довжина кожного циклу має бути *невід'ємною*.

Спочатку потрібно ініціалізувати початкову матрицю відстаней D_1 (матрицю, кожен елемент якої d_{ij} дорівнює відстані від вершини *i* до вершини *j*, якщо існує ребро (*i*,*j*), і дорівнює нескінченності в іншому випадку) і матрицю марш-

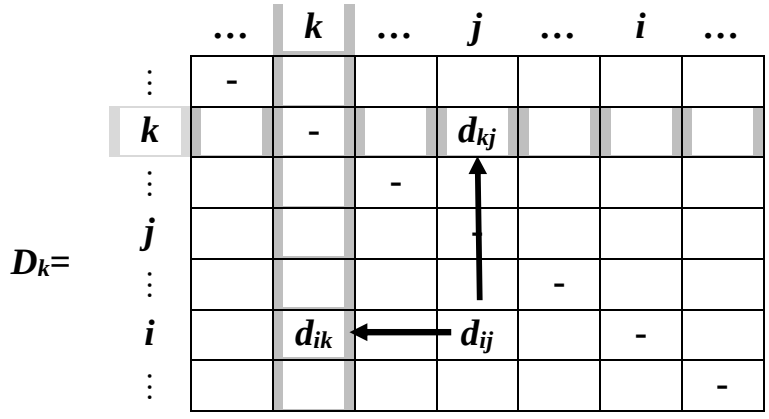
рутів (послідовностей вершин) S_1 (кожен елемент матриці дорівнює номеру відповідного стовпця). Діагональні елементи обох матриць позначають знаком "-", оскільки їх в обчисленнях не враховують.

$$D_1 = \begin{bmatrix} - & d_{12} & d_{13} \ldots & d_{1n} \\ d_{21} & - & d_{23} \ldots & d_{2n} \\ \ldots & \ldots & \ldots \ldots & \ldots \\ d_{n1} & d_{n2} & d_{n3} \ldots & d_{nn} \end{bmatrix} \qquad S_1 = \begin{bmatrix} - & 2 & 3 & \ldots & n \\ 1 & - & 3 & \ldots & n \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & 2 & 3 & \ldots & n \end{bmatrix}$$

Далі на кожному кроці потрібно виділити **базовий вузол** (провідні рядок та стовпець). На першому кроці (k=1) це буде перший рядок та стовпець, на другому кроці – другий рядок та стовпець тощо. Викреслюємо базові стовпець та рядок матриці D_k, а також ті її рядки та стовпці, які мають значення ∞, що знаходиться в базових рядку чи стовпці.

Для утворення нової матриці D_{k+1} необхідно переписати викреслені рядки і стовпці без змін, а інші елементи перерахувати за таким правилом: якщо $d_{ik} + d_{kj} < d_{ij}$, то у матриці D_{k+1} змінюємо елемент $d_{ij} = d_{ik} + d_{kj}$, в іншому випадку значення елементу залишаємо попереднім.

Розрахунки на цьому кроці доцільно виконувати з урахуванням *викреслених* рядків та стовпчиків. Викреслення полегшує розрахунки за вищенаведеними формулами та усуває непотрібні перерахунки значень. Порівнюємо кожен *невикреслений* елемент d_{ij} та переписуємо перераховані значення в матриці наступної ітерації. Викреслені елементи переписуємо без змін.



Після зміни на кроці k елемента d_{ij} матриці відстаней D_{k+1} у матриці шляхів S_{k+1} на місці відповідного елемента s_{ij} записуємо номер кроку k, якщо $s_{ik} = k$, а в протилежному випадку присвоюємо йому значення s_{ik}.

Алгоритм Флойда завершується через n кроків (n – кількість вершин мережі).

6.5. Приклад знаходження найкоротшого маршруту в мережі за допомогою алгоритму Флойда

Приклад 6.2. Знайти мінімальні ланцюги між вершинами мережі, зображеної на рис. 6.3, за допомогою алгоритму Флойда.

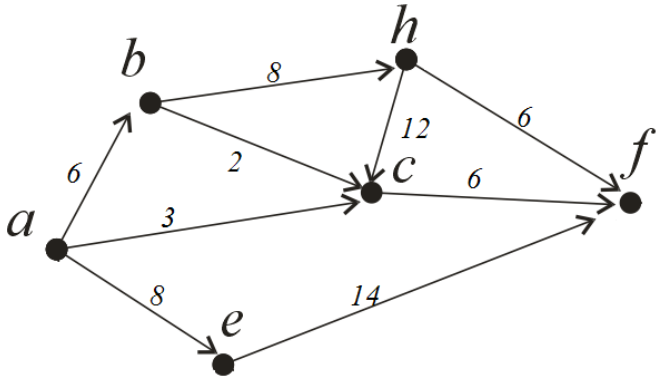


Рис. 6.3. Приклад мережі

Крок 1. Визначимо початкові матриці.

D_1=

	a	b	c	e	h	f
a	0	6	3	8	∞	∞
b	∞	0	2	∞	8	∞
c	∞	∞	0	∞	∞	6
e	∞	∞	∞	0	∞	14
h	∞	∞	12	∞	0	6
f	∞	∞	∞	∞	∞	0

S_1=

	a	b	c	e	h	f
a	*0*	b	c	e	h	f
b	a	*0*	c	e	h	f
c	a	b	*0*	e	h	f
e	a	b	c	*0*	h	f
h	a	b	c	e	*0*	f
f	a	b	c	e	h	*0*

На першому кроці виділяємо *перші* рядок та стовпець матриці D_1. Для зменшення кількості обчислень користуємось **правилом**: якщо у виділеному рядку (стовпці) є елементи рівні ∞, то викреслюємо і всі стовпці (рядки), які їм відповідають, тобто їх **не перераховуємо**. Оскільки у матриці D_1 у першому стовпчику всі елементи рівні ∞, то жоден її елемент не змінюємо (так само і в матриці S_1).

Крок 2. Виділяємо у матриці D_2 другі рядок та стовпець.

D_2=

	a	b	c	e	h	f
a	0	6	3	8	∞	∞
b	∞	0	2	∞	8	∞
c	∞	∞	0	∞	∞	6
e	∞	∞	∞	0	∞	14
h	∞	∞	12	∞	0	6
f	∞	∞	∞	∞	∞	0

S_2=

	a	b	c	e	h	f
a	*0*	b	c	e	h	f
b	a	*0*	c	e	h	f
c	a	b	*0*	e	h	f
e	a	b	c	*0*	h	f
h	a	b	c	e	*0*	f
f	a	b	c	e	h	*0*

У матриці D_2 необхідно перерахувати лише d_{13}^3 та d_{15}^3.

$$d_{13}^3 = \min\{d_{13}^2; d_{12}^2 + d_{23}^2\} = \min\{3; 6+2\} = 3;$$

$$d_{15}^3 = \min\{d_{15}^2; d_{12}^2 + d_{25}^2\} = \min\{\infty; 6+8\} = 14.$$

Отже, змінився елемент d_{15}^3, тому значення відповідного елемента s_{15}^3 буде дорівнювати *b* (номер кроку = 2).

Крок 3. Виділяємо у матриці D_3 третій рядок та стовпець.

$D_3=$

	a	*b*	*c*	*e*	*h*	*f*
a	0	6	3	8	14	∞
b	∞	0	2	∞	8	∞
c	∞	∞	0	∞	∞	6
e	∞	∞	∞	0	∞	14
h	∞	∞	12	∞	0	6
f	∞	∞	∞	∞	∞	0

$S_3=$

	a	*b*	*c*	*e*	*h*	*f*
a	*0*	*b*	*c*	*e*	*b*	*f*
b	*a*	*0*	*c*	*e*	*h*	*f*
c	*a*	*b*	*0*	*e*	*h*	*f*
e	*a*	*b*	*c*	*0*	*h*	*f*
h	*a*	*b*	*c*	*e*	*0*	*f*
f	*a*	*b*	*c*	*e*	*h*	*0*

Обчислимо нові елементи: $d_{16}^4 = \min\{d_{16}^3; d_{13}^3 + d_{36}^3\} = \min\{\infty; 6+3\} = 9;$

$d_{26}^4 = \min\{d_{26}^3; d_{23}^3 + d_{36}^3\} = \min\{\infty; 6+2\} = 8;$

$d_{56}^4 = \min\{d_{56}^3; d_{53}^3 + d_{36}^3\} = \min\{6; 6+12\} = 6.$

Відповідні елементи s_{16}^4 і s_{26}^4 матриці S_4 замінюємо на *c*.

Крок 4. Виділяємо у матриці D_4 четвертий рядок та стовпець.

$D_4=$

	a	*b*	*c*	*e*	*h*	*f*
a	0	6	3	8	14	9
b	∞	0	2	∞	8	8
c	∞	∞	0	∞	∞	6
e	∞	∞	∞	0	∞	14
h	∞	∞	12	∞	0	6
f	∞	∞	∞	∞	∞	0

$S_4=$

	a	*b*	*c*	*e*	*h*	*f*
a	*0*	*b*	*c*	*e*	*b*	*c*
b	*a*	*0*	*c*	*e*	*h*	*c*
c	*a*	*b*	*0*	*e*	*h*	*f*
e	*a*	*b*	*c*	*0*	*h*	*f*
h	*a*	*b*	*c*	*e*	*0*	*f*
f	*a*	*b*	*c*	*e*	*h*	*0*

$d_{16}^5 = \min\{d_{16}^4; d_{14}^4 + d_{46}^4\} = \min\{9; 8+14\} = 9.$

Крок 5. Виділяємо у матриці D_5 п’ятий рядок та стовпець.

$D_5=$

	a	*b*	*c*	*e*	*h*	*f*
a	0	6	3	8	14	9
b	∞	0	2	∞	8	8
c	∞	∞	0	∞	∞	6
e	∞	∞	∞	0	∞	14
h	∞	∞	12	∞	0	6
f	∞	∞	∞	∞	∞	0

$S_5=$

	a	*b*	*c*	*e*	*h*	*f*
a	*0*	*b*	*c*	*e*	*b*	*c*
b	*a*	*0*	*c*	*e*	*h*	*c*
c	*a*	*b*	*0*	*e*	*h*	*f*
e	*a*	*b*	*c*	*0*	*h*	*f*
h	*a*	*b*	*c*	*e*	*0*	*f*
f	*a*	*b*	*c*	*e*	*h*	*0*

$d_{13}^6 = \min\{d_{13}^5; d_{15}^5 + d_{53}^5\} = \min\{3; 12+14\} = 3;$

$d_{23}^6 = \min\{d_{23}^5; d_{25}^5 + d_{53}^5\} = \min\{2; 12+8\} = 2;$

$d_{16}^6 = \min\{d_{16}^5; d_{15}^5 + d_{56}^5\} = \min\{9; 14+6\} = 9;$

$d_{26}^6 = \min\{d_{26}^5; d_{25}^5 + d_{56}^5\} = \min\{8; 8+6\} = 8$. *Крок 6*. Виділяємо у матриці D_6 шостий рядок та стовпець.

D_6=

	a	*b*	*c*	*e*	*h*	*f*
a	0	6	3	8	14	9
b	∞	0	2	∞	8	8
c	∞	∞	0	∞	∞	6
e	∞	∞	∞	0	∞	14
h	∞	∞	12	∞	0	6
f	∞	∞	∞	∞	∞	0

S_6=

	a	*b*	*c*	*e*	*h*	*f*
a	*0*	*b*	*c*	*e*	*b*	*c*
b	∞	*0*	*c*	∞	*h*	*c*
c	∞	∞	*0*	∞	∞	*f*
e	∞	∞	∞	*0*	∞	*f*
h	∞	∞	*c*	∞	*0*	*f*
f	∞	∞	∞	∞	∞	*0*

У матриці D_6 немає елементів, які необхідно обчислити. Всі елементи, які в матриці D_6 дорівнюють ∞, в матриці S_6 теж дорівнюють ∞, тобто у вершину *a* не можна попасти зі жодної вершини.

Отже, всі кроки алгоритму Флойда виконані, отримані кінцеві матриці D_6 та S_6.

На основі цих двох матриць можна обчислити всі можливі маршрути між вершинами мережі. Наприклад: найкоротша відстань між вершинами *a* та *f* $d_{16}^6 = 9$. Для того, щоб знайти вершини мережі, через які проходить даний маршрут, необхідно проаналізувати матрицю S_6. Елемент $s_{16}^6 = c$, а це означає, що проміжною точкою даного маршруту є вершина *c*. Далі потрібно переглянути елемент $s_{36}^6 = f$, тобто маршрут від вершини *a* до вершини *f* проходить через такі вершини: $a \to c \to f$.

Контрольні запитання до лабораторної роботи № 6

1. Що таке мережа?
2. Що таке зв'язана мережа?
3. Що таке орієнтована і зважена мережа?
4. Що розуміють під довжиною шляху у зваженій (незваженій) мережі?
5. Наведіть приклади практичного застосування мережевих задач.
6. На якій ідеї ґрунтується алгоритм Дейкстри та які задачі можна розв'язувати за його допомогою?
7. Поясніть процедуру знаходження найкоротшого маршруту в мережі.
8. Опишіть послідовність кроків алгоритму Дейкстри.
9. Вкажіть, за допомогою якої модифікації можна, окрім довжини найкоротшого шляху, знайти і сам шлях у алгоритмі Дейкстри.
10. Яка умова завершення алгоритму Дейкстри?
11. Опишіть послідовність кроків алгоритму Флойда.
12. У чому полягає суть алгоритму Флойда?
13. Яка умова завершення алгоритму Флойда?

14. Як за допомогою алгоритму Флойда визначити через які вершини проходять найкоротші шляхи в мережі від довільної вершини до всіх інших?
15. Для чого на кожному кроці алгоритму Флойда потрібно виділяти базовий вузол?
16. Назвіть переваги алгоритму Флойда над алгоритмом Дейкстри.

Завдання до лабораторної роботи № 6

1. Отримати індивідуальний варіант завдання.
2. Написати програму розв'язування потокових задач за модифікованими методами Дейкстри та Флойда з Додатка до лабораторної роботи № 6.
3. Оформити звіт про виконану роботу.
4. Продемонструвати викладачеві результати, відповісти на запитання стосовно виконання роботи.

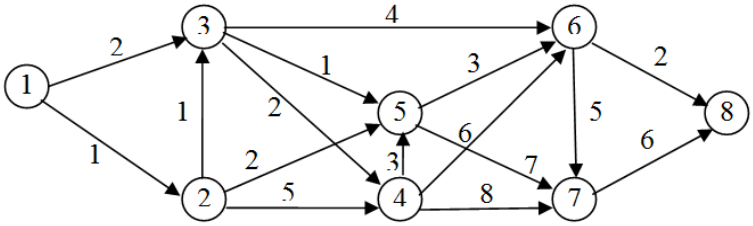
Вимоги до програми

Програма має передбачати такі можливості:

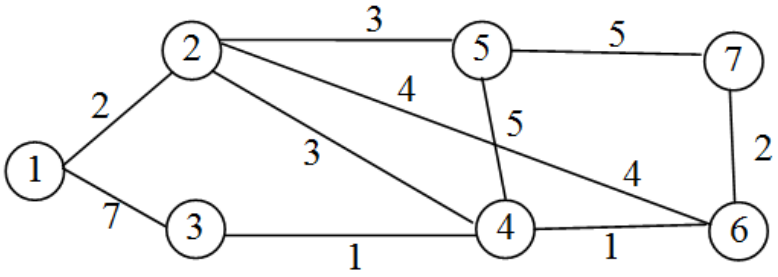
1. Автоматичне знаходження найкоротшого маршруту в мережі за модифікованим алгоритмом Дейкстри.
2. Автоматичне знаходження найкоротшого маршруту в мережі за модифікованим алгоритмом Флойда.
3. Введення вхідних даних вручну (вагові матриці).
4. Передбачити можливість некоректного введення даних.
5. Передбачити можливість покрокового відображення проміжних таблиць.
6. Підписання усіх таблиць.
7. Виведення відповідного повідомлення у випадку відсутності розв'язку.

Додаток до лабораторної роботи № 6

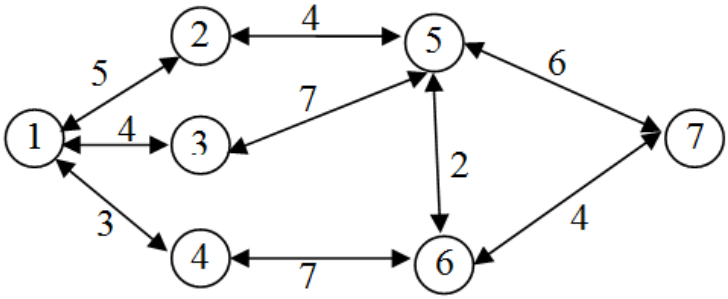
1



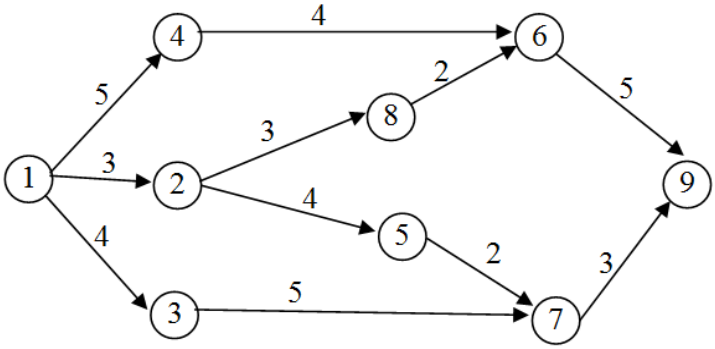
2



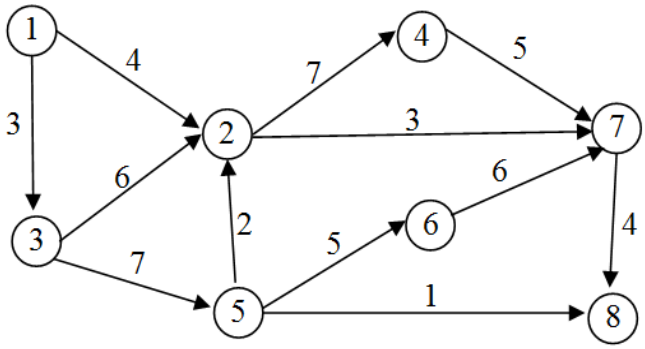
3



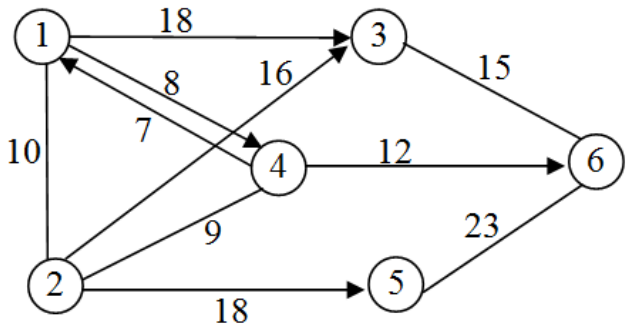
4



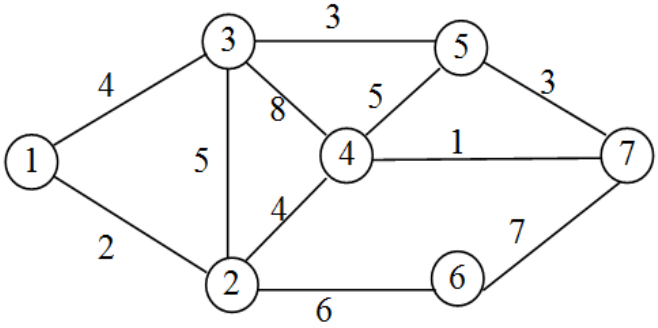
5



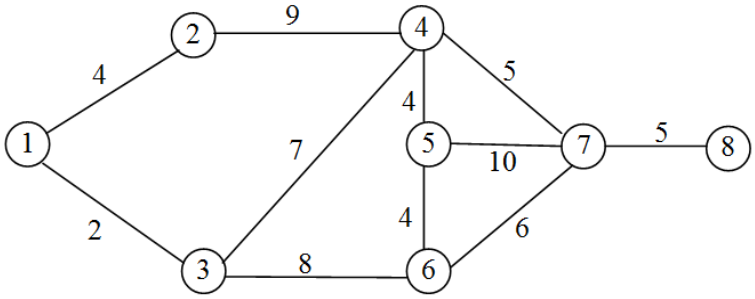
6


1
3
2
4
5
6
18
15
8
16
7
10
12
9
23
18

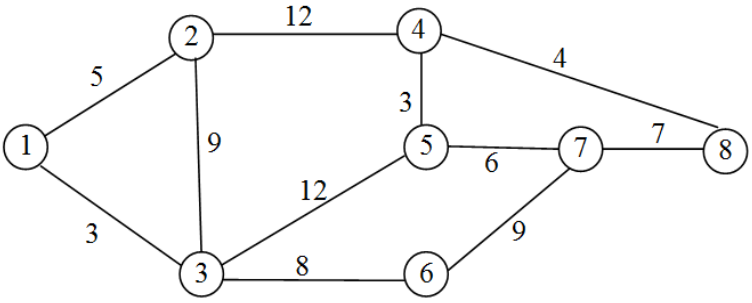
7


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

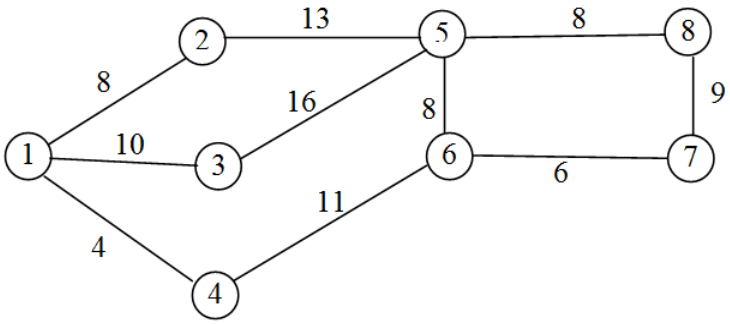
8


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

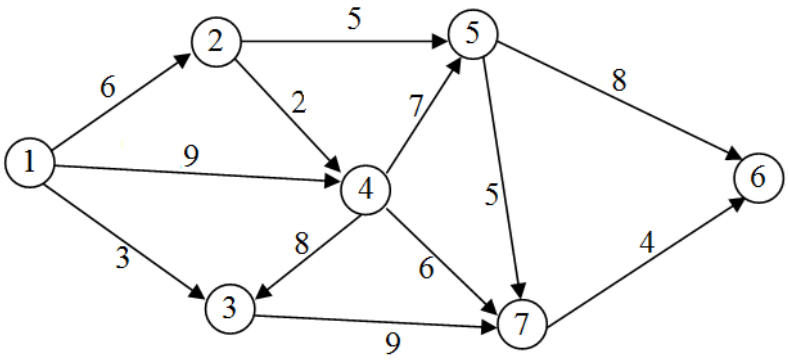
9


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

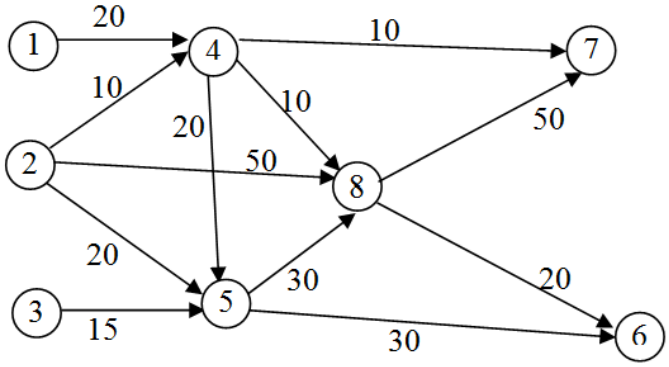
10


2
13
5
8
8
8
9
16
8
1
10
3
6
7
6
11
4
4

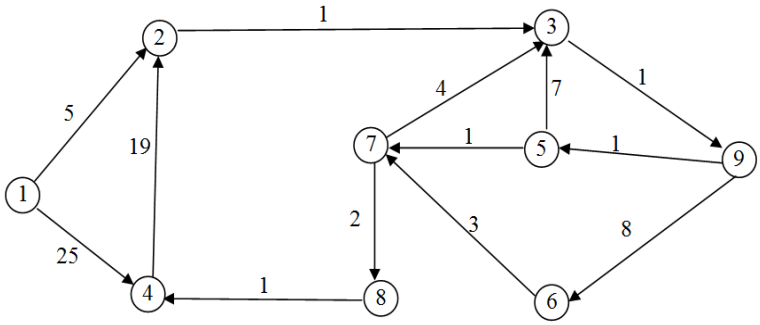
11


1
2
3
4
5
6
7
6
5
2
9
7
8
3
8
5
6
4
9

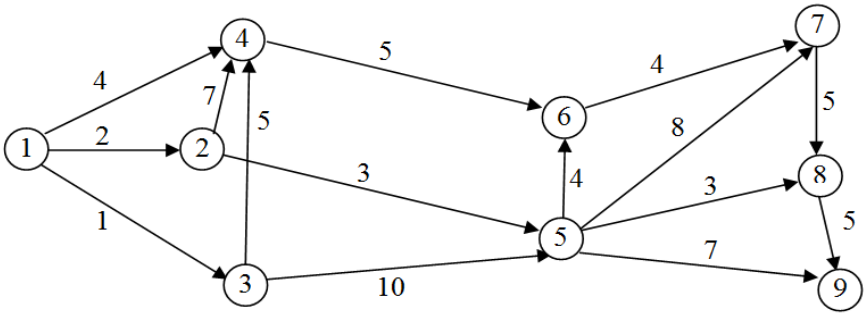
12


1
2
3
4
5
6
7
8
20
10
10
20
50
10
50
20
30
20
15
30

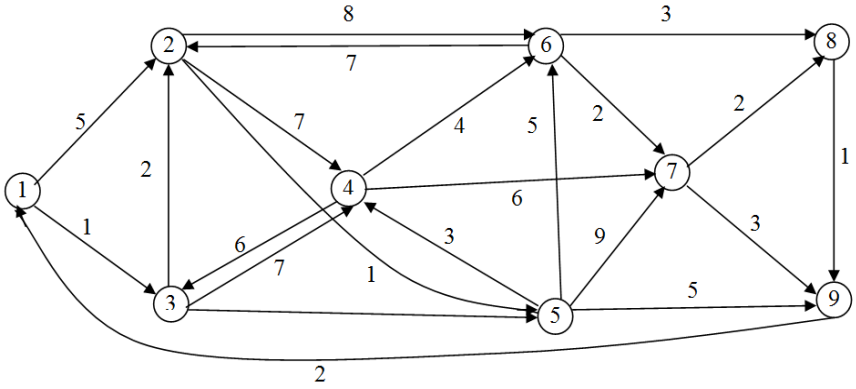
13


1
2
3
4
5
6
7
8
9
1
5
19
4
7
1
1
1
2
3
8
25
1

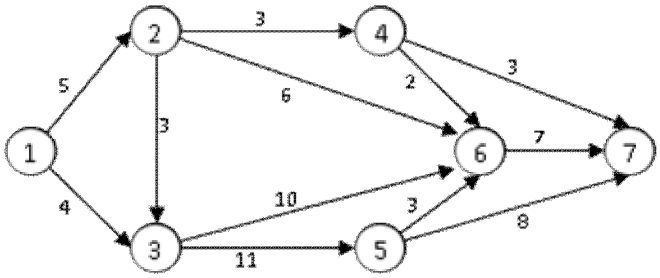
14


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

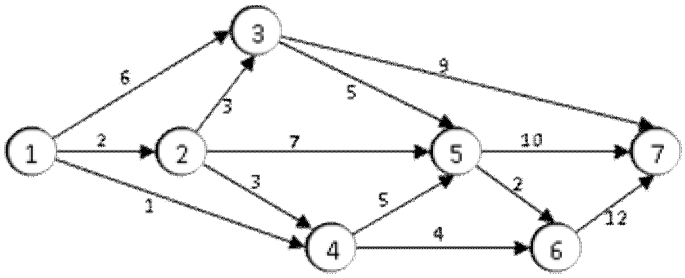
15


1
2
3
4
5
6
7
8
9
8
7
3
5
2
7
4
5
2
2
1
6
1
6
3
9
7
1
3
5
2

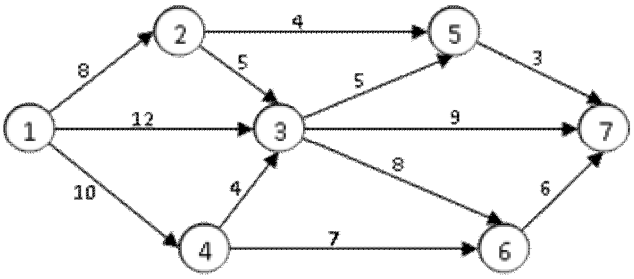
16


1
2
3
4
5
6
7
5
4
3
3
6
2
3
10
11
3
7
8

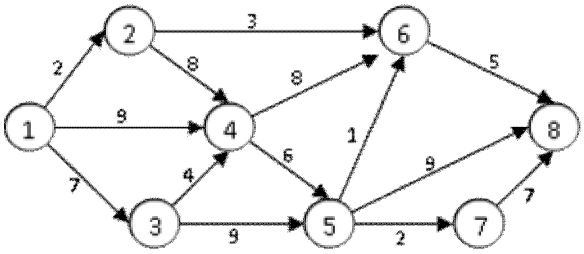
17


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

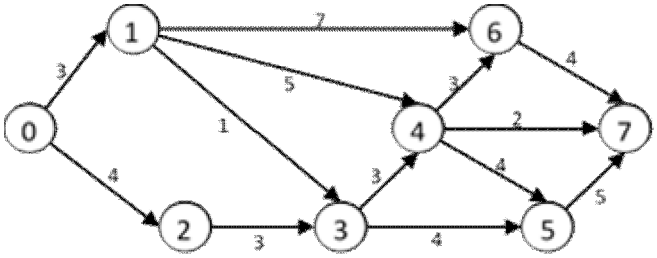
18


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

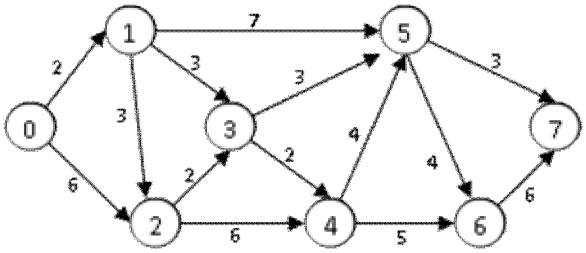
19


1
2
3
4
5
6
7
8
2
7
9
8
4
3
8
6
1
5
9
7
9
2
7

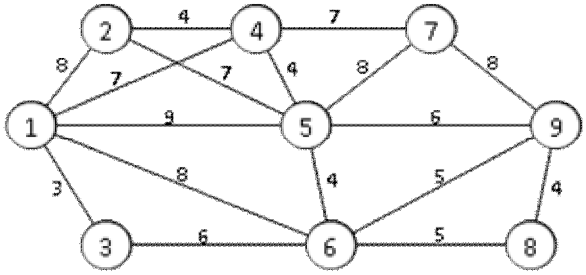
20


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

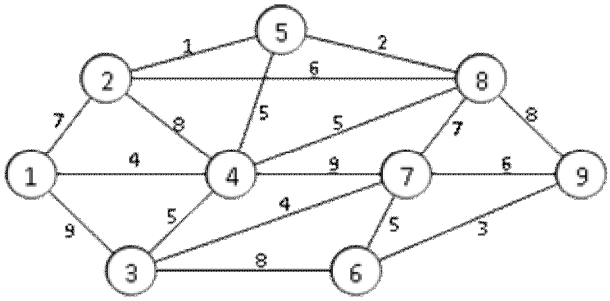
21



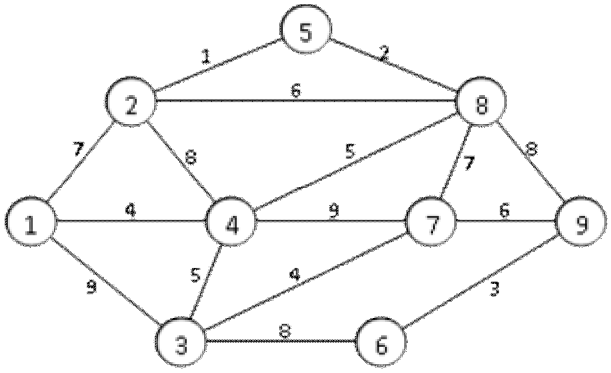
22



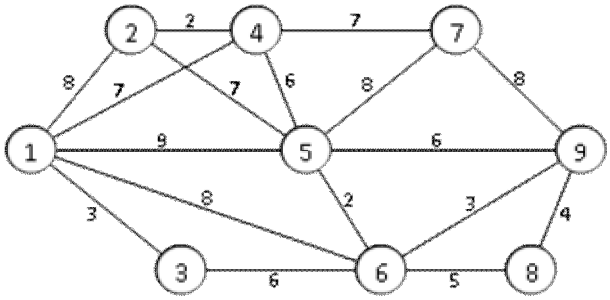
23



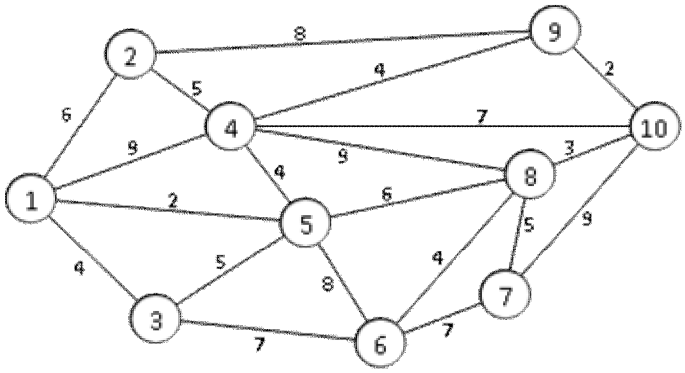
24



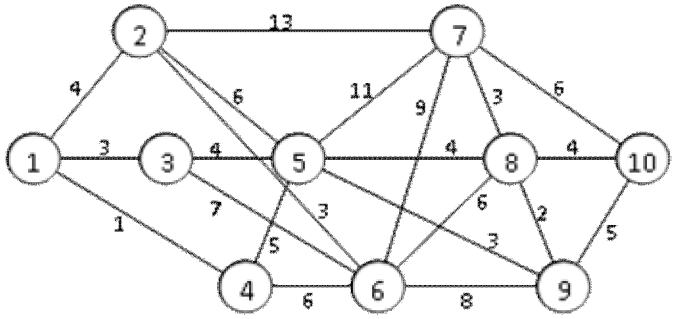
25



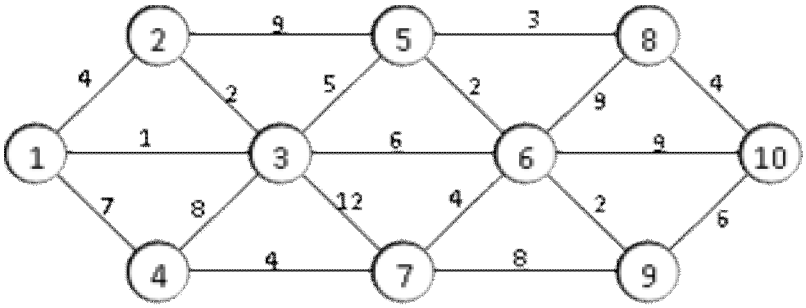
26


1
2
3
4
5
6
7
8
9
10

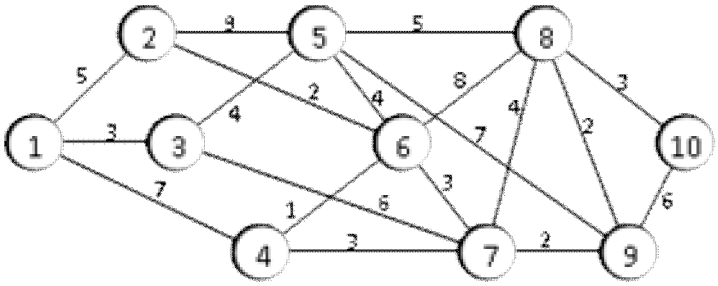
27


1
2
3
4
5
6
7
8
9
10

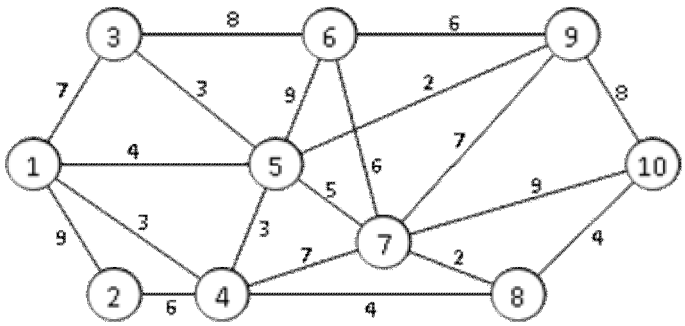
28


1
2
3
4
5
6
7
8
9
10

29


1
2
3
4
5
6
7
8
9
10

30


1
2
3
4
5
6
7
8
9
10