fyfМІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ „ЛЬВІВСЬКА ПОЛІТЕХНІКА”

ЗАДАЧІ ПРО ПОТОКИ В МЕРЕЖАХ

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт №5 та №6
з дисципліни „Дослідження операцій”
для студентів спеціальності
6. 050103 „Програмна інженерія”

Затверджено
на засіданні кафедри
програмного забезпечення
Протокол № __ від _____ р.

Львів – 2016

Задачі про потоки в мережах. : Методичні вказівки до виконання лабораторних робіт №5 та №6 із дисципліни „Дослідження операцій” для студентів спеціальності „Програмна інженерія” / Укл.: Л.М. Журавчак, О.О. Нитребич – Львів: Видавництво Національного університету „Львівська політехніка”, 2016. – 29 с.

Укладачі Л.М. Журавчак, д-р тех. наук, проф.
О.О. Нитребич, канд. тех. наук, асист. кафедри ПЗ

Відповідальний за випуск Федасюк Д.В., д-р тех. наук, проф.

Зміст

Мета роботи: Ознайомитись на практиці із основними алгоритмами розв'язування потокових задач, навчитись знаходити оптимальні маршрути між вершинами мережі за допомогою алгоритмів Дейкстри та Флойда, а також розв'язувати задачі про максимальний потік з використанням алгоритму Гоморі-Ху.

Вступ

В рамках теорії дослідження операцій розглядається велика кількість практичних задач, які можна сформулювати як мережеві моделі і розв'язати їх 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 існуючій мережі доріг.

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

4. Визначення схеми транспортування нафти від пунктів нафтовидобутку до нафтопереробних заводів з мінімальною вартістю транспортування.

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

Розв'язок 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.

Лабораторна робота №5

Розв'язування задач оптимізації на мережах: пошук найкоротшого маршруту в мережі модифікованим алгоритмом Дейкстри та багатополюсного найкоротшого ланцюга за допомогою алгоритму Флойда

1. Поняття мережі

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.1.1 зображено

деяку мережу, у якої $N=\{1,2,3,4,5\}$, а множина $A=\{(1,2);(1,3);(2,3);(2,5);(3,4);(3,5);(4,2);(4,5)\}$.

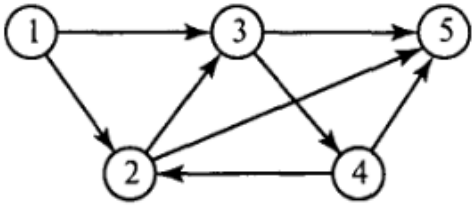


Рис. 1.1. Приклад мережі

Зв'язна мережа – це мережа, у якої будь-які два вузли зв'язані принаймні одним шляхом (дугою). На рис. 1.1 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 задана зв'язна транспортна мережа, кожній дузі якої, що виходить із точки p_i та входить у точку p_j, поставлено у відповідність деяке дійсне **невід'ємне** число d_{ij} – її довжину. Треба визначити найкоротші шляхи в мережі від довільної вершини до всіх інших і вказати, через які вершини вони проходять.

Для розв'язування даної задачі розглянемо алгоритми Дейкстри та Флойда.

2. 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 алгоритму кожній вершині ставиться у відповідність 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 2. 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 3. Серед усіх тимчасових міток визначаємо постійну за правилом $x^* = \min(d(x_i))$. Переходимо на крок 2.

3. Приклад пошуку найкоротшого маршруту в мережі модифікованим алгоритмом Дейкстри

Приклад 3.1. Знайти найкоротші маршрути з вершини *a* до решти вершин в мережі, зображеної на рис. 3.1, та вказатати вершини, через які вони проходять, за допомогою модифікованого алгоритму Дейкстри.

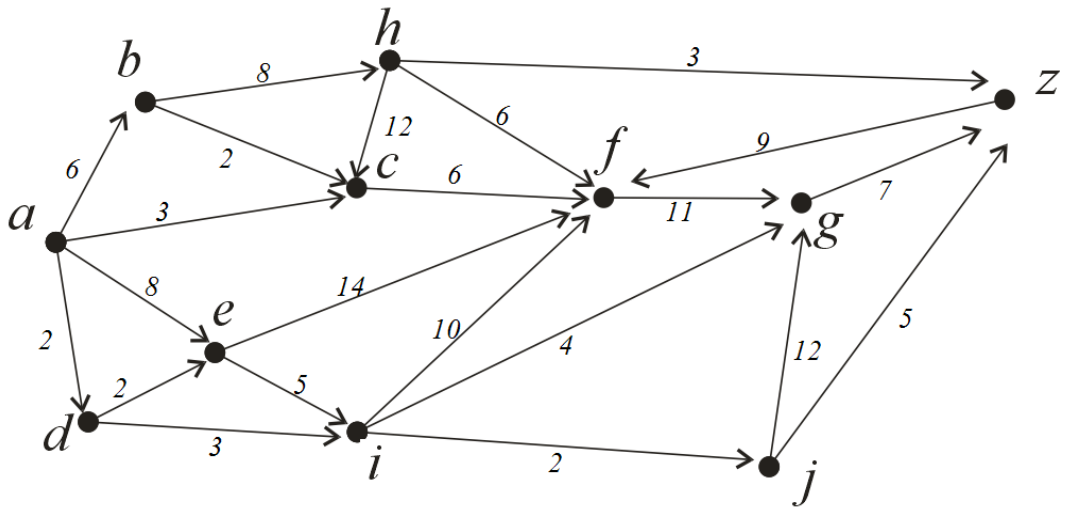


Рис.3.1. Приклад мережі

Крок 1. Розглянемо першу вершину *a* (з якої необхідно знайти маршрут до решти вершин) – тимчасова мітка. Випишемо таблицю (табл.3.1), у якій елементи першого рядка ($d(x)$) будуть містити відстані від початкової вершини до решти, які ми переглянули, а елементи другого рядка (x) – вершини, через які проходить маршрут з 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 3.1

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.3.2) записуємо обчислені значення елементів у перший рядок. У відповідних елементах другого рядка таблиці записуємо вершину *a.*

Таблиця 3.2

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.3.2 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 3.3.

Таблиця 3.3

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.3.3 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 3.4.

Таблиця 3.4

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*	-	-

Обираємо серед непереглянутих вершин у табл.3.4 ту, 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. 3.5-3.11).

Таблиця 3.5

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 3.6

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 3.7

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 3.8

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 3.9

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 3.10

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 3.11

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, отримано остаточний результат:

Таблиця 3.12

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

4. Алгоритм Флойда

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

Суть алгоритму Флойда полягає у перевірці того, чи не виявиться шлях з 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*[*i*,*j*] дорівнює відстані від вершини *i* до вершини *j*, якщо існує ребро (*i*,*j*), і дорівнює нескінченності в іншому випадку) і матрицю маршрутів (послідовностей вершин) S_1 (кожен елемент матриці дорівнює номеру відповідного стовпця). Діагональні елементи обох матриць позначають знаком "-", оскільки їх в обчисленнях не враховують.

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

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 і т.д.

Для утворення нової матриці D_{k+1} необхідно переписати провідні рядки і стовпці без змін, а решта елементів 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.

D_k=

	…	**k**	…	**j**	…	**i**	…
⋮	-						
k		-		d_{kj}			
⋮			-				
j							
⋮					-		
i		d_{ik}	←	d_{ij}		-	
⋮							-

Якщо змінився деякий елемент d_{ij} матриці відстаней на кроці k, то в матриці S_{k+1} на місці відповідного елементу записуємо номер кроку k.

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

5. Приклад знаходження найкоротшого ланцюга за допомогою алгоритму Флойда

Приклад 5.1. Знайти мінімальні ланцюги між вершинами мережі, зображеної на рис.5.1, за допомогою алгоритму Флойда.

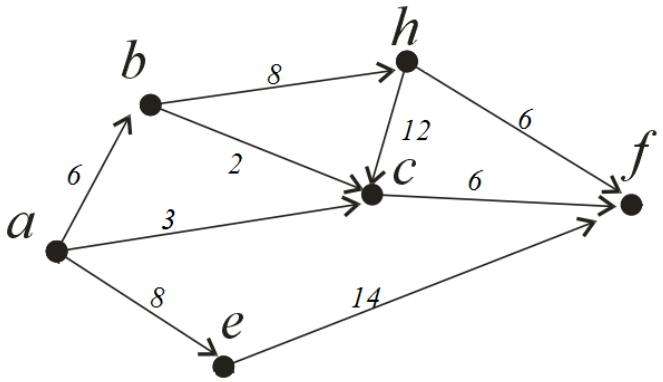


Рис. 5.1. Приклад мережі

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

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

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

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

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

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

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

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

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

$d_{26}^6 = \min\{d_{26}^5; d_{25}^5 + d_{56}^5\} = \min\{8; 8+6\} = 8.$

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

У матриці D_6 немає елементів, які необхідно обчислити.

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 чи є проміжні точки в маршруті $a \to c$, елемент $s_{13}^6 = c$, тобто проміжних вершин нема. Тобто маршрут від вершини *a* до вершини *f* проходить через такі вершини: $a \to c \to f$.

Контрольні запитання до лабораторної роботи №5

1. Що таке мережа?
2. Що таке зв'язана мережа?
3. Поясніть процедуру знаходження найкоротшого маршруту в мережі.
4. Наведіть приклади практичного застосування мережевих задач.
5. На якій ідеї ґрунтується алгоритм Дейкстри та які задачі можна розв'язувати за його допомогою?
6. Вкажіть, за допомогою якої модифікації можна, окрім довжини найкоротшого шляху, знайти і сам шлях.
7. Опишіть послідовність кроків алгоритму Флойда.
8. Назвіть переваги алгоритму Флойда стосовно алгоритму Дейкстри.

Завдання до лабораторної роботи № 5

1. Отримати індивідуальний варіант завдання.
2. Написати програму розв'язування потокових задач методами Дейкстри та Флойда з Додатку 1.
3. Оформити звіт про виконану роботу.
4. Продемонструвати викладачеві результати, відповісти на запитання стосовно виконання роботи.

Вимоги до звіту

1. Титульний аркуш.
2. Тема звіту.

3. Мета звіту.
4. Теоретичні відомості.
   i. Дати відповідь на контрольне запитання у відповідності із номером журналу.
5. Текст програми з коментарями (алгоритм Дейкстри, Флойда).
6. Вигляд реалізованої програми.
7. Висновки.

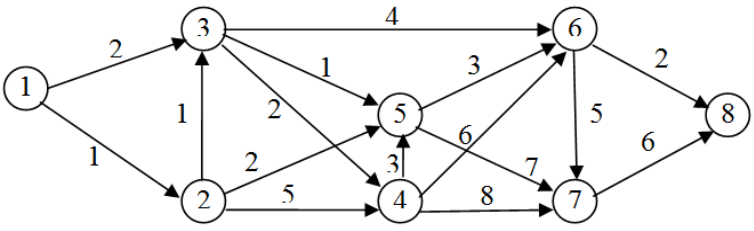
Вимоги до програми

Програма має передбачати наступні можливості:

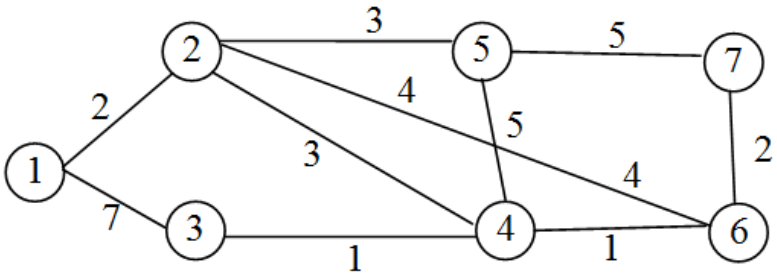
1. Автоматичне знаходження найкоротшого маршруту в мережі модифікованим алгоритмом Дейкстри.
2. Автоматичне знаходження найкоротшого маршруту в мережі алгоритмом Флойда.
3. Ввід вхідних даних вручну (матриці суміжності).
4. Передбачити можливість некоректного введення даних.
5. Передбачити можливість покрокового відображення проміжних таблиць.
6. Підпис усіх таблиць.
7. Вивід необхідного повідомлення у випадку неіснування розв’язку.

Додаток 1

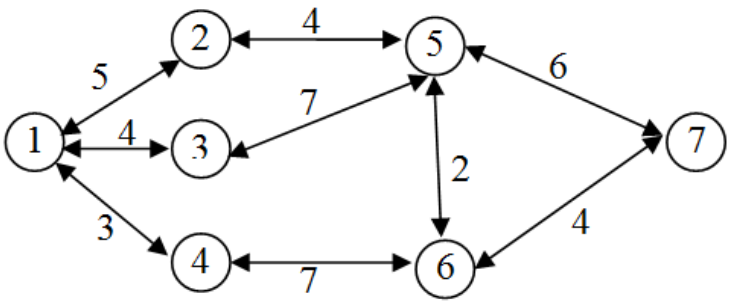
1



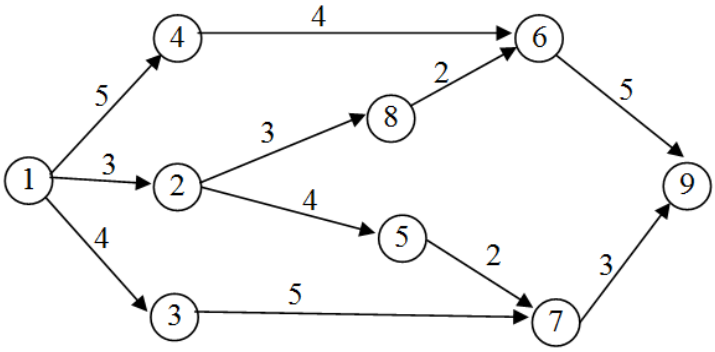
2



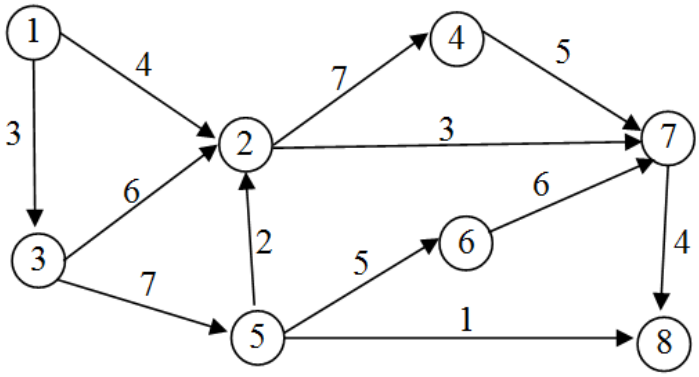
3


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

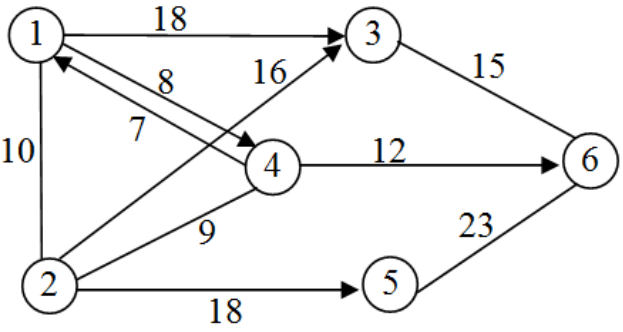
4


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

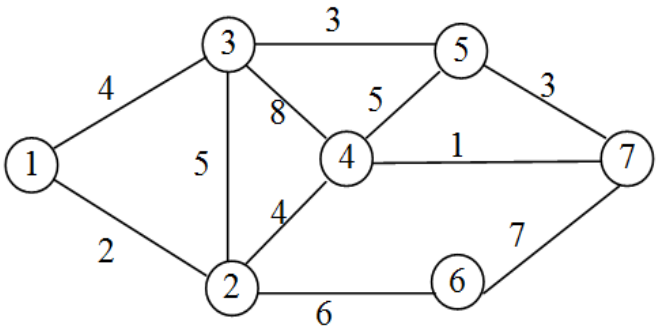
5


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

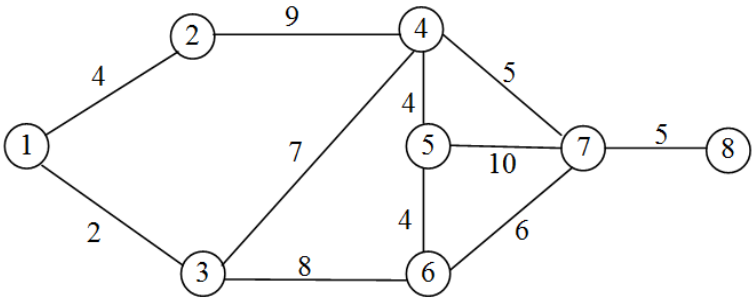
6


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

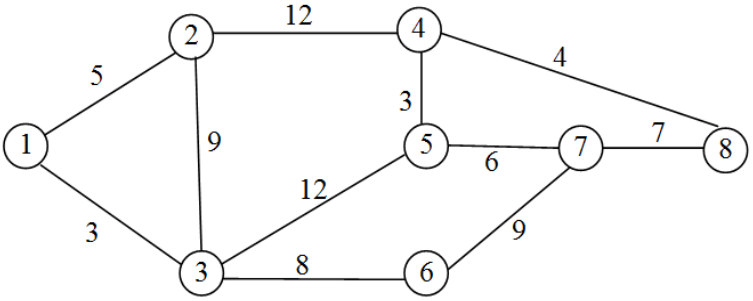
7


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

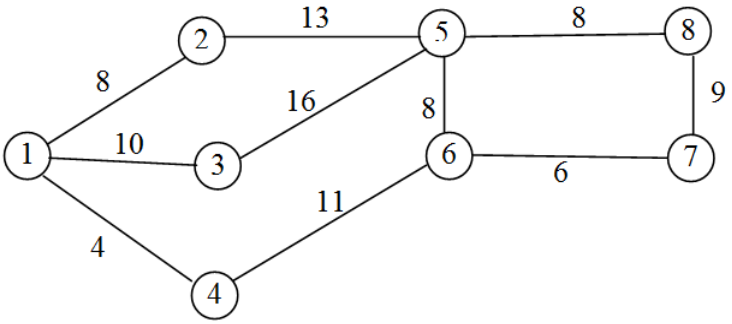
8


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

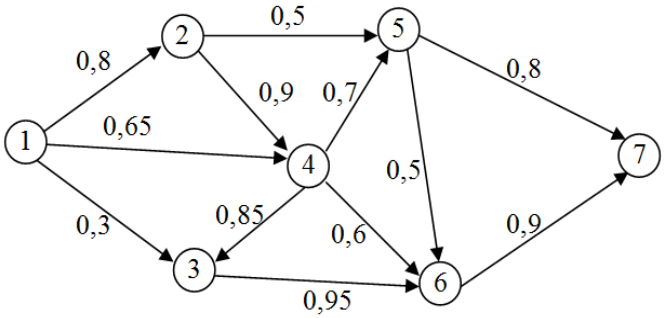
9


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

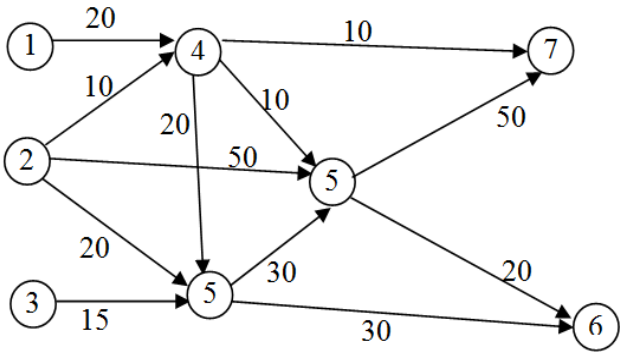
10


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

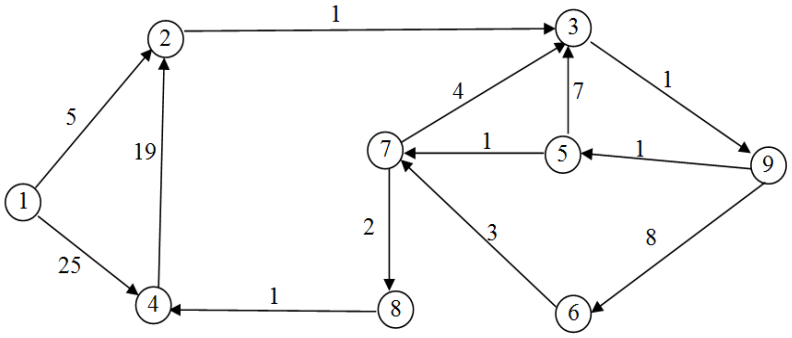
11


1
2
3
4
5
6
7
0,8
0,65
0,3
0,5
0,9
0,7
0,85
0,6
0,95
0,5
0,8
0,9

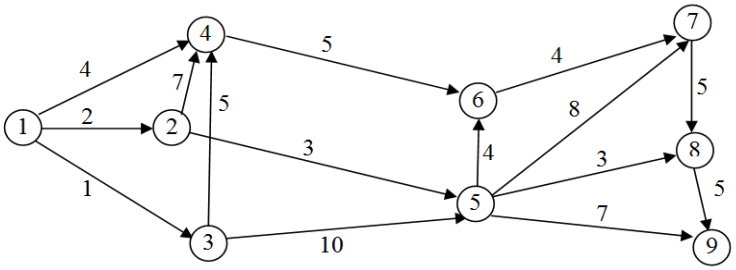
12


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

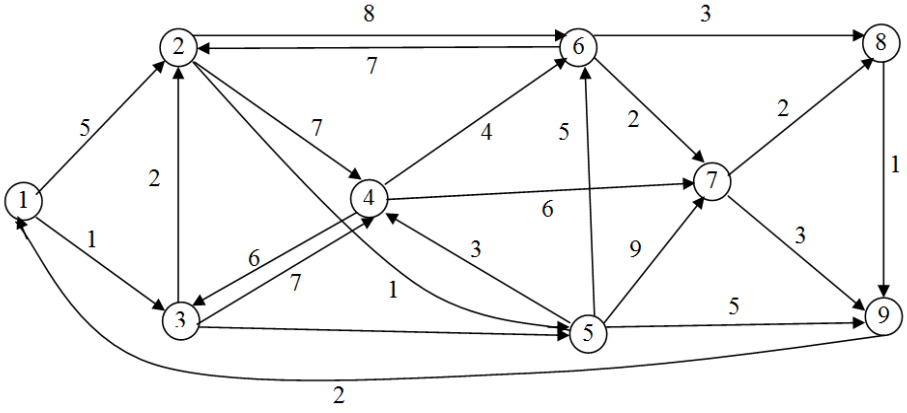
13


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
4
7
5
5
3
4
8
4
3
5
7
10

15


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

Лабораторна робота №6
Розв'язування задачі про багатополюсний максимальний потік за допомогою алгоритму Гоморі-Ху

6. Задача про максимальний потік

Під час опису багатьох реальних ситуацій, які можна моделювати за допомогою мережі, наприклад, рух транспорту вулицями міста, використовують поняття «потік». Існує багато технічних і економічних задач, в яких системи можуть бути наближено описані у вигляді потокових моделей. Прикладами таких систем є транспортні мережі, де автостради зображують дугами з пропускними спроможності, що відповідають максимально допустимій інтенсивності руху; телефонні мережі, де телефонні лінії описують дугами, а пропускні здатності відповідають максимальному числу викликів, які можуть обслуговуватися в кожен момент часу і т.д. У всіх цих задачах передбачається існування декількох джерел деякого продукту. Передбачається також, що величина продукту, який може транспортуватися до декількох витоків, обмежена тільки пропускними здатностями дуг.

Задача про максимальний потік. Нехай $G = (N, A)$ – орієнтована мережа з одним джерелом $s \in N$ і одним витоком $t \in N$, і нехай дуги $(i, j) \in A$ мають обмежену пропускну здатність. Задача про максимальний потік полягає в пошуку таких потоків по дугах, що належать множині A, щоб результуючий потік, який витікає із s в t, був максимальним.

Для знаходження розв'язку даної задачі використовують алгоритм, розроблений американськими математиками Гоморі і Ху.

7. Алгоритм Гоморі-Ху

Ідея алгоритму Гоморі-Ху полягає в ітеративній побудові дерева розрізів.

Крок 1. Множина гілок дерева розрізів пуста. Всі вузли об'єднані в одну групу. Обираємо довільну пару вузлів s та t. Покладемо $l = 1$.

Крок 2. Вибираємо один або два конденсовані вузли, в один з яких входить s, в інший t.

Крок 3. Знаходимо мінімальний розріз, що відділяє s від t. Зображуємо цей розріз гілкою в дереві розрізів, вага якої рівна пропускній здатності цього розрізу. Ця гілка повинна сполучати вузли чи групи вузлів, які розташовані по різні боки від знайденого мінімального розрізу.

Крок 4. Якщо $l = n-1$, то кінець: дерево розрізів побудоване; інакше переходимо до наступного кроку.

Крок 5. Обираємо будь-яку пару вузлів *i* та *j*, які ще не відділені один від одного в дереві розрізів. Покладаємо *s=i*, *t=j*.

Крок 6. Сконденсовуємо в один вузол кожну зв'язну підмережу, що з'єднана з групою, в якій знаходяться вузли *i* та *j*. Покладаємо *l=l*+1. Переходимо до кроку 3.

8. Приклад розв'язування потокової задачі методом Гоморі-Ху

Приклад 8.1. Для кожної пари вузлів мережі, зображеної на рис. 8.1, визначити величину максимального потоку між ними.

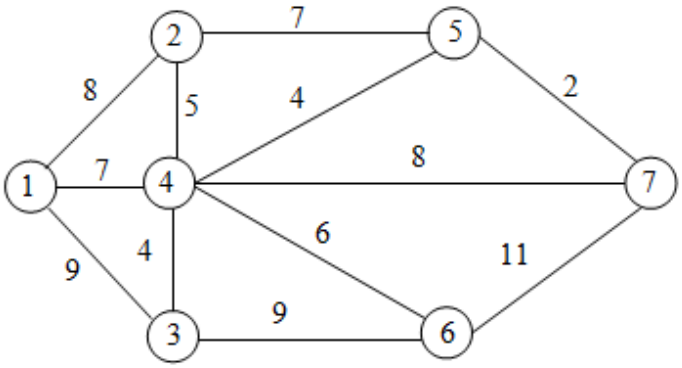


Рис. 8.1. Приклад мережі в задачі про багатополюсний максимальний потік

Дана задача розв'язується за *n*-1 = 7-1 = 6 ітерацій алгоритму Гоморі-Ху.

Крок 1. Об'єднуємо, наприклад, всі вузли, окрім 7-го (до нього йде ребро з найбільшою пропускною здатністю, тому його краще зразу поставити у кінець ланцюжка), у конденсований вузол, тобто покладаємо *t*=7. Величина максимального потоку між цими вузлами дорівнює 21 (бо такою є сума ваг всіх ребер, що входять у 7: 2+8+11=21). Побудова дерева розрізів починається з гілки, яка з'єднує вузол 7 і конденсований вузол 1,2,3,4,5,6 (рис. 8.2). Вага даного ребра дорівнює 21.

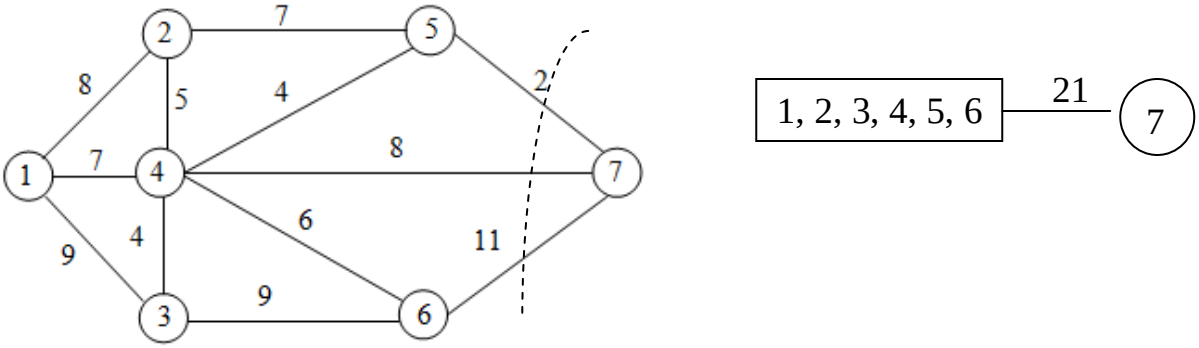


Рис. 8.2. Задача про максимальний потік (крок 1)

Крок 2. З конденсованого 1,2,3,4,5,6 вузла виділяємо вузол 6 (до нього з вершини 7 йде ребро з найбільшою пропускною здатністю), тобто покладаємо t=6. Розглядаємо два розрізи: в першому відтинаємо лише вузол 6 (пропускна здатність такого розрізу 9+6+11=26), в другому – 6 та 7 (його пропускна здатність 9+6+8+2=25). За теоремою Форда-Фалкерсона (теоремою про максимальний потік в мережі) величина максимального потоку дорівнює пропускній здатності мінімального розрізу, тому вибираємо другий. Вузли 6 і 7 лежать по одну сторону розрізу, а решта – по іншу (рис. 8.3).

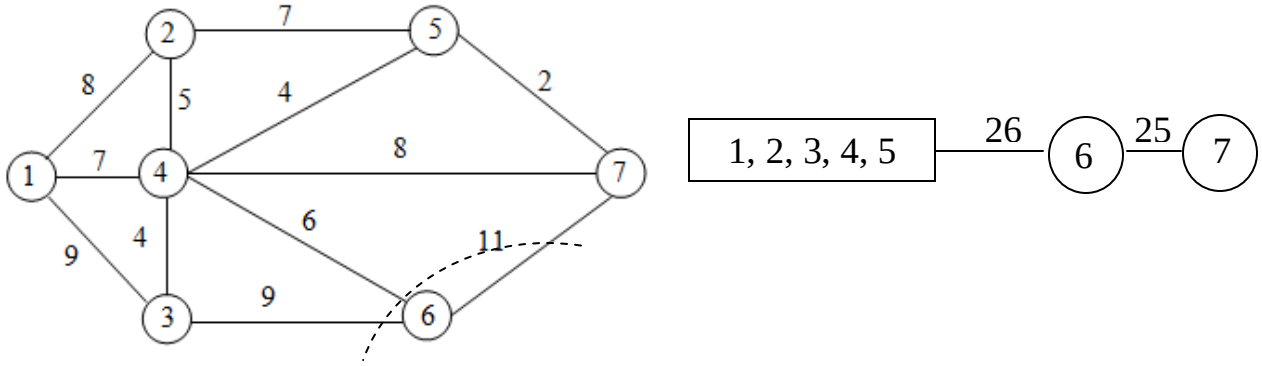


Рис. 8.3. Задача про максимальний потік (крок 2)

Крок 3. З конденсованого вузла 1,2,3,4,5 виділяємо вузол 3 (до нього з вершини 6 йде ребро з найбільшою пропускною здатністю), тобто покладаємо t=3. Пропускна здатність мінімального розрізу (і відповідно величина максимального потоку) 9+4+9=22. Вузол 3 лежить по один бік від конденсованого вузла 1,2,4,5, а вузли 6,7 – по інший (рис. 8.4).

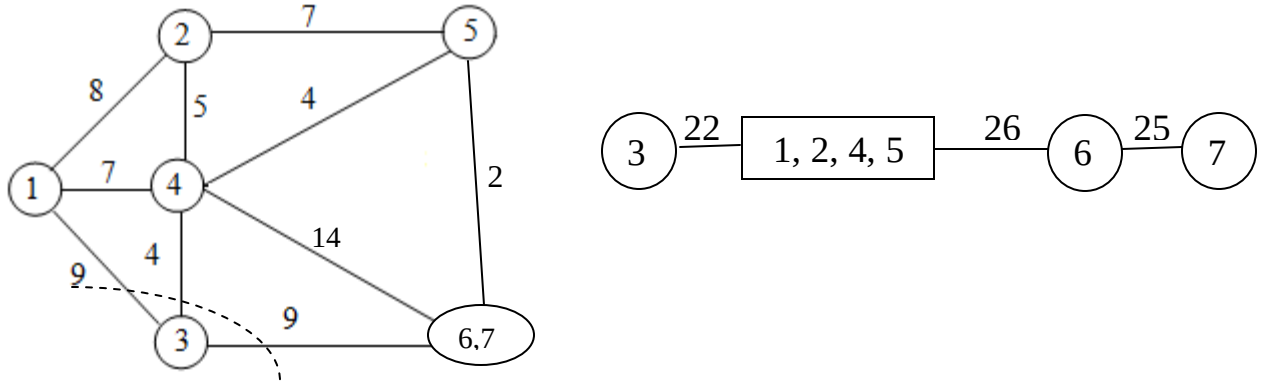


Рис. 8.4. Задача про максимальний потік (крок 3)

Крок 4. З конденсованого вузла 1,2,4,5 виділяємо вузол 1, тобто покладаємо t=1. Величина максимального потоку дорівнює 8+7+9=24. Він розташований зліва від конденсованого вузла 2,4,5, вузол 3 – внизу, а вузли 6 і 7 – справа (рис. 8.5).

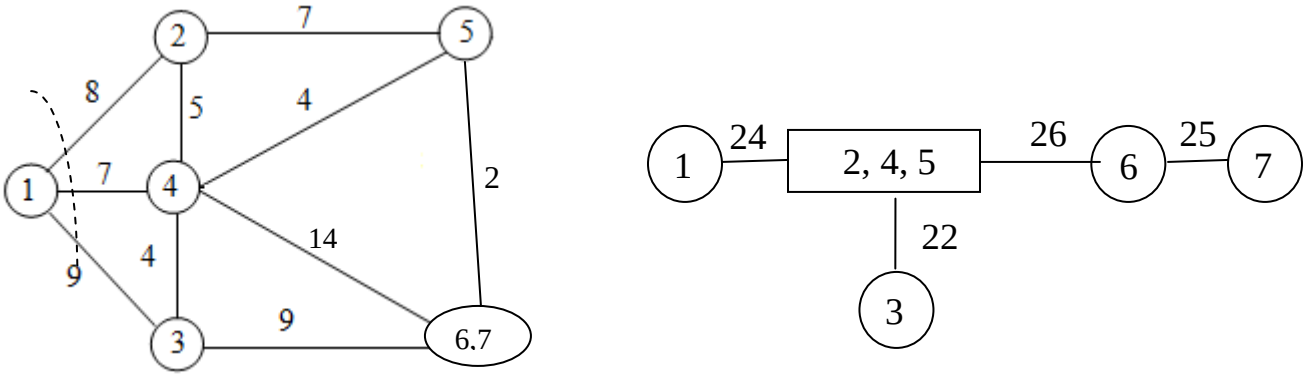


Рис. 8.5. Задача про максимальний потік (крок 4)

Крок 5. З конденсованого вузла 2,4,5 виділяємо вузол 4. Величина максимального потоку дорівнює 8+9+2=19. Вгору від нього розташовуємо конденсований вузол 2,5 (рис. 8.6).

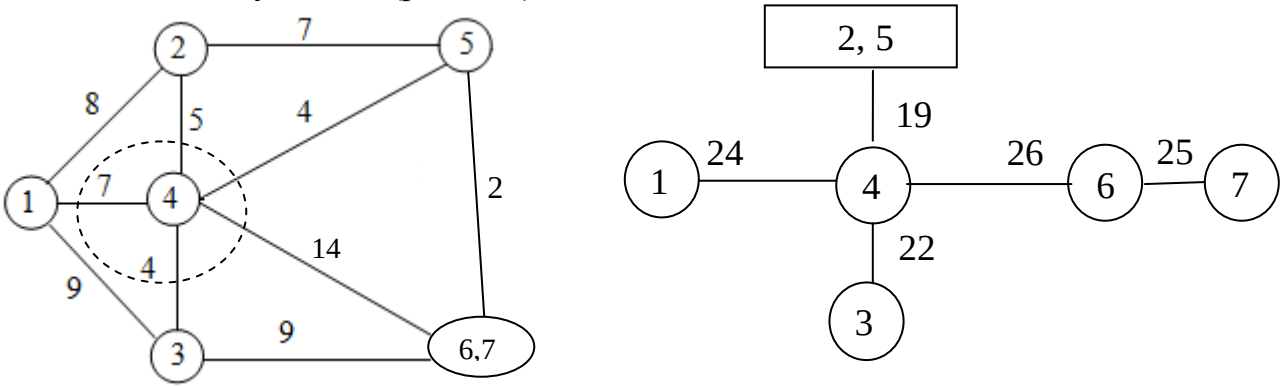


Рис. 8.6. Задача про максимальний потік (крок 5)

Крок 6. З конденсованого вузла 2,5 виділяємо вузол 5. Величина максимального потоку дорівнює 7+4+2=13. Його розташовуємо вгору від вузла 2 (рис. 8.7).

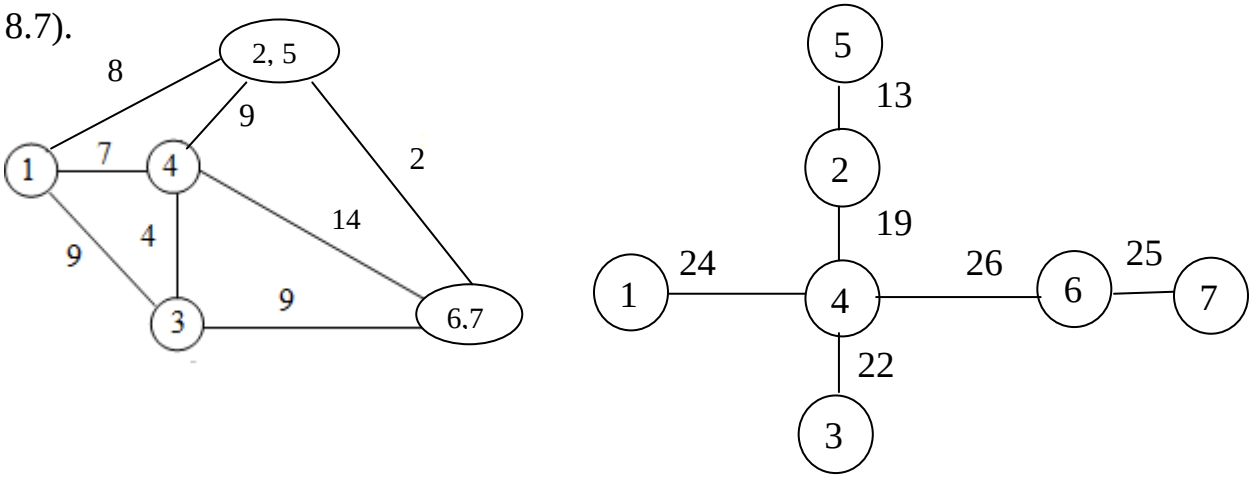


Рис. 8.7. Задача про максимальний потік (крок 6)

У результаті одержано повне дерево розрізів. Величини максимальних потоків записуються у вигляді такої матриці (елемент матриці v_{ij} дорівнює *найменшій* вазі ребра серед тих, що сполучають вузли *i* та *j*):

$$V = \begin{bmatrix} - & 19 & 22 & 24 & 13 & 24 & 21 \\ 19 & - & 19 & 19 & 13 & 19 & 19 \\ 22 & 19 & - & 22 & 13 & 22 & 21 \\ 24 & 19 & 22 & - & 13 & 25 & 21 \\ 13 & 13 & 13 & 13 & - & 13 & 13 \\ 24 & 19 & 22 & 25 & 13 & - & 21 \\ 21 & 19 & 21 & 21 & 13 & 21 & - \end{bmatrix}$$

Контрольні запитання до лабораторної роботи №6

1. Що таке потік?
2. Сформуйте задачу про максимальний потік.
3. Наведіть приклади задач про максимальний потік.
4. Що таке дерево розрізів?
5. Яка основна ідея алгоритму Гоморі-Ху?
6. Які основні кроки алгоритму Гоморі-Ху?

Завдання до лабораторної роботи № 6

1. Отримати індивідуальний варіант завдання.
2. Написати програму розв’язування задачі про максимальний потік методом Гоморі-Ху.
3. Оформити звіт про виконану роботу.
4. Продемонструвати викладачеві результати, відповісти на запитання стосовно виконання роботи.

Вимоги до звіту

1. Титульний аркуш.
2. Тема звіту.
3. Мета звіту.
4. Теоретичні відомості.
    i. Дати відповідь на контрольне запитання у відповідності із номером журналу.
5. Текст програми з коментарями (алгоритм Гоморі-Ху).
6. Вигляд реалізованої програми.
7. Висновки.

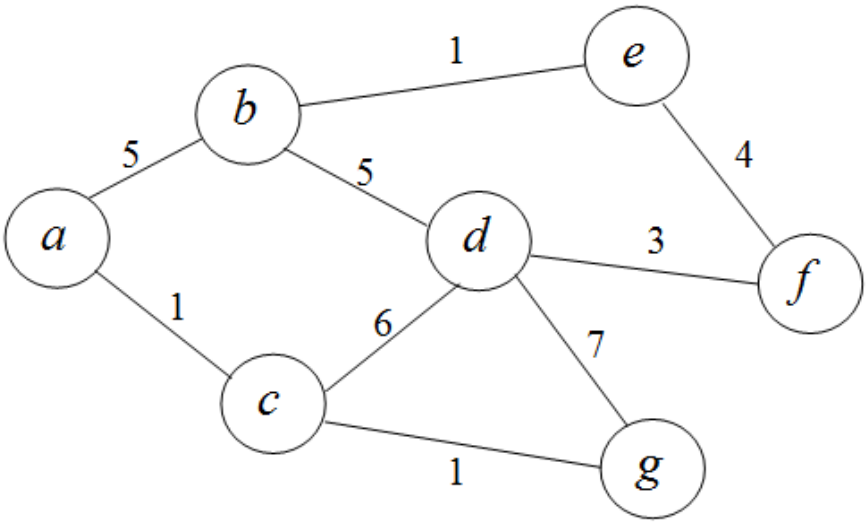
Вимоги до програми

Програма має передбачати наступні можливості:

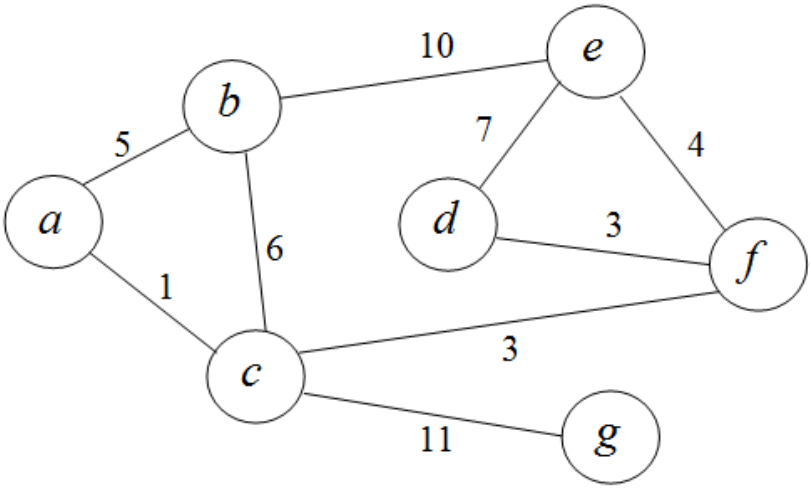
1. Автоматичне знаходження максимального потоку з використанням алгоритму Гоморі-Ху.
2. Ввід вхідних даних вручну (матриці суміжності).
3. Передбачити можливість некоректного введення даних.
4. Передбачити можливість покрокового відображення проміжних дерев розрізів.
5. Підпис усіх кроків.

Додаток 2

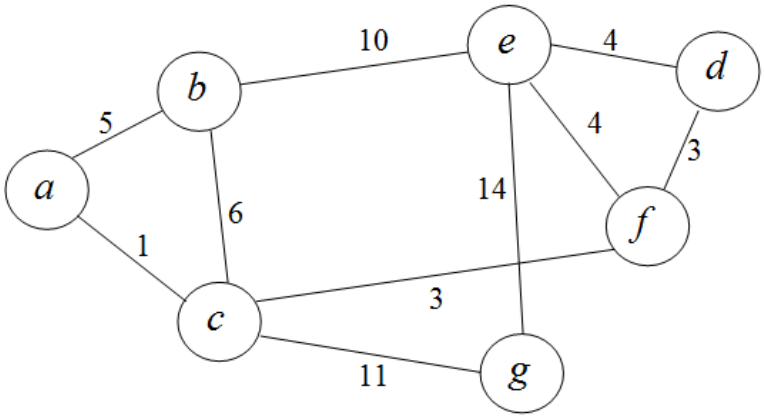
1



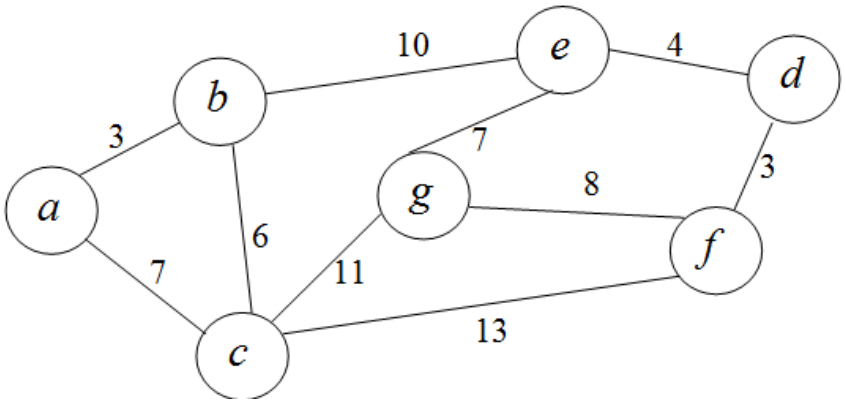
2



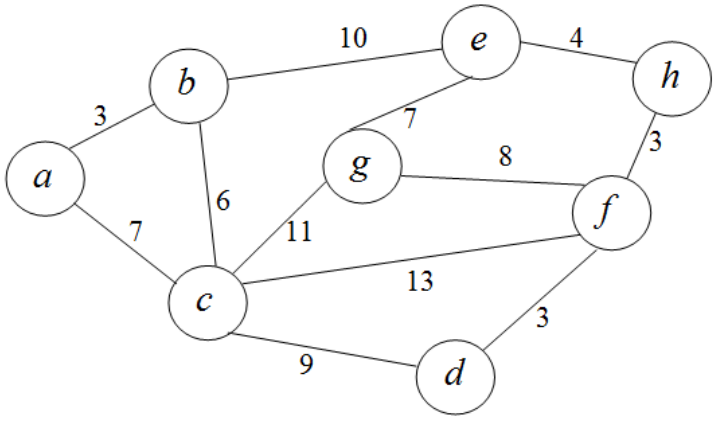
3



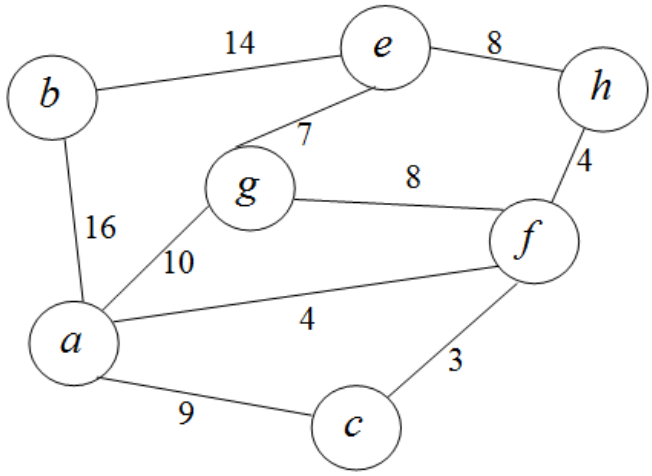
4


a
b
c
d
e
f
g
10
4
3
7
8
3
6
7
11
13

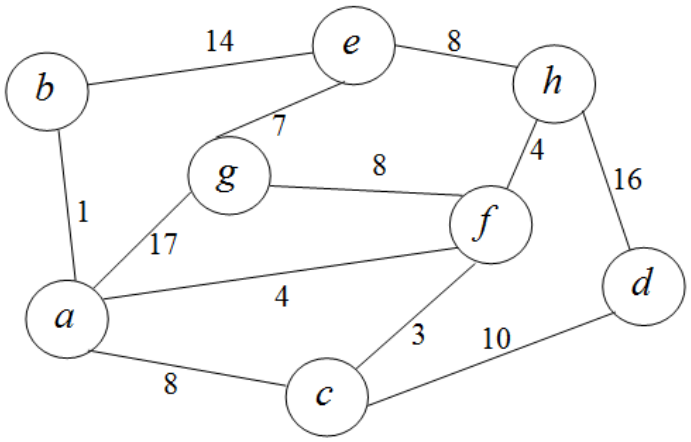
5


a
b
c
d
e
f
g
h
10
4
3
7
8
3
6
7
11
13
3
9

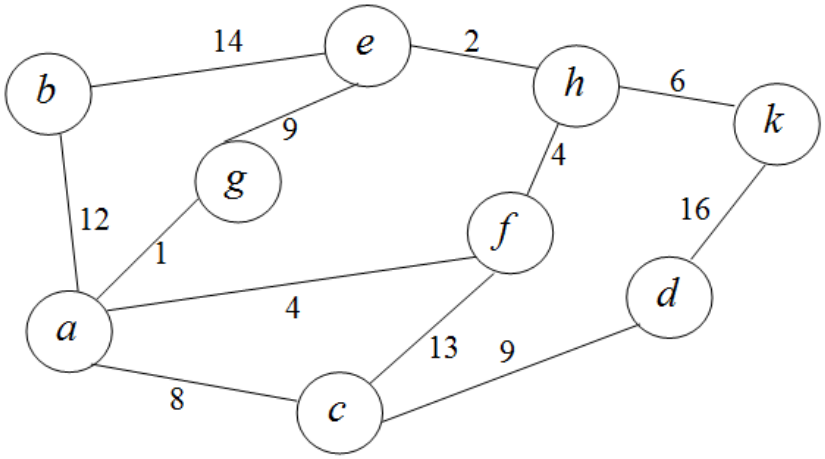
6


a
b
c
e
f
g
h
14
8
7
4
8
16
10
4
3
9

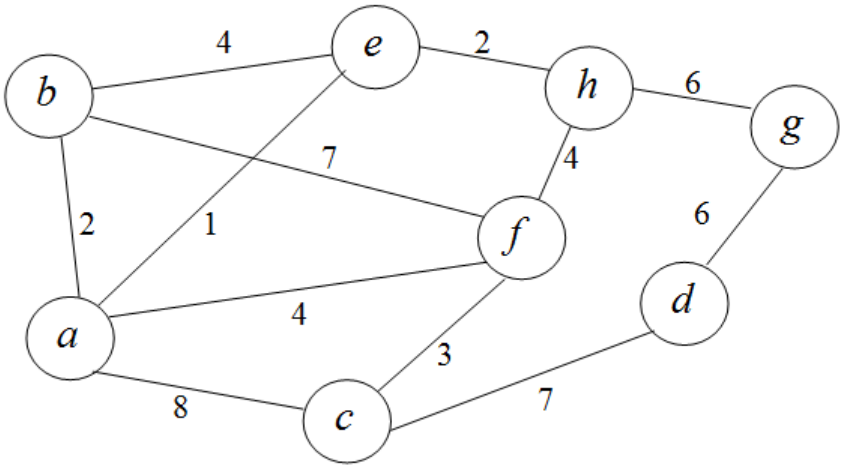
7


b
e
h
g
f
d
a
c
14
8
7
4
16
1
17
8
4
3
10
8

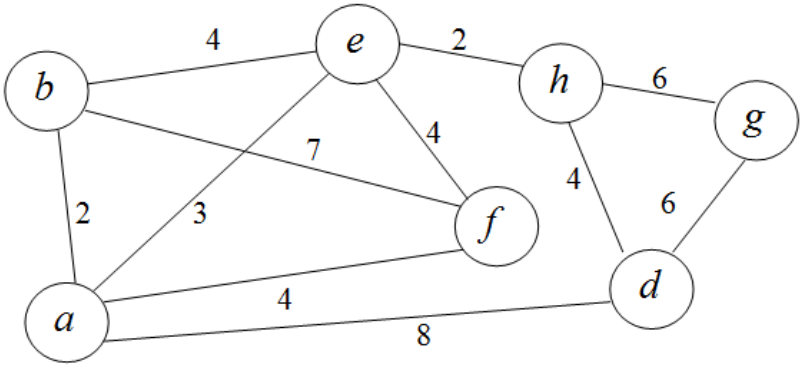
8


b
e
h
k
g
f
d
a
c
14
2
6
9
4
12
16
1
4
13
9
8

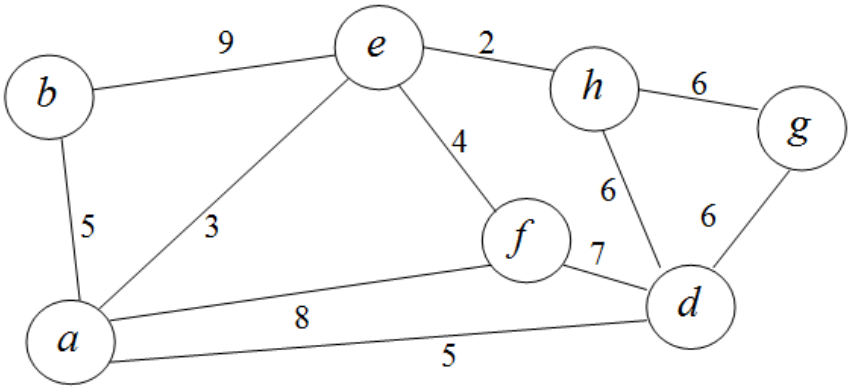
9


b
e
h
g
f
d
a
c
4
2
6
7
4
2
1
6
4
3
8
7

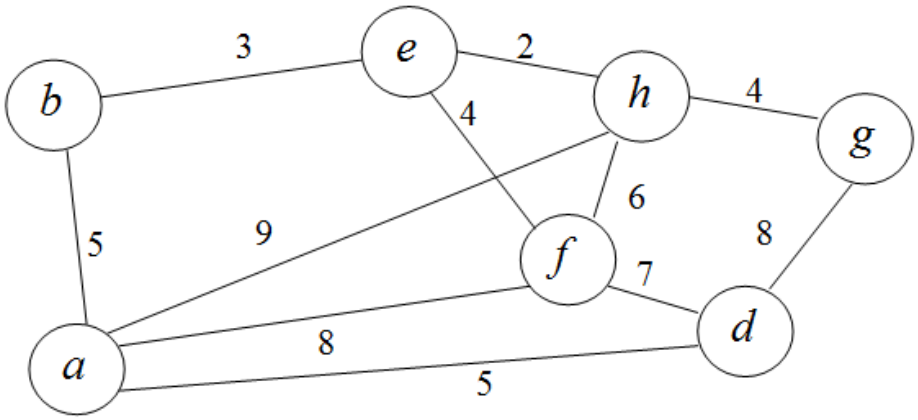
10


b
e
h
g
f
d
a
4
2
6
7
4
2
3
4
6
4
8

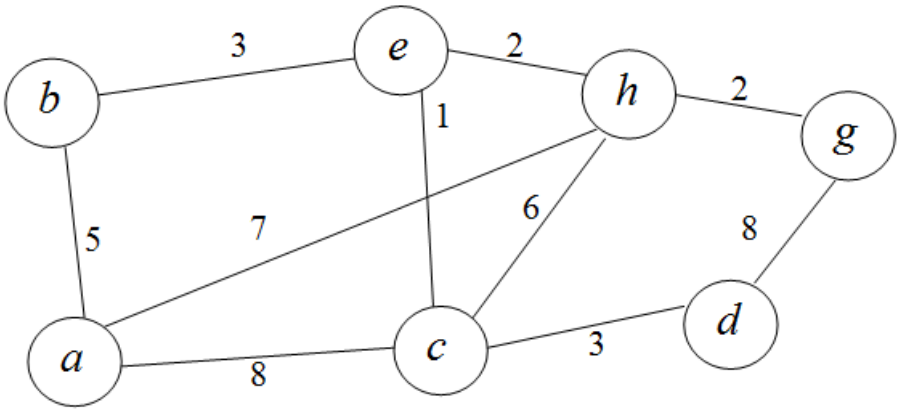
11


b
e
h
g
f
d
a
9
2
6
4
6
5
3
6
7
8
5

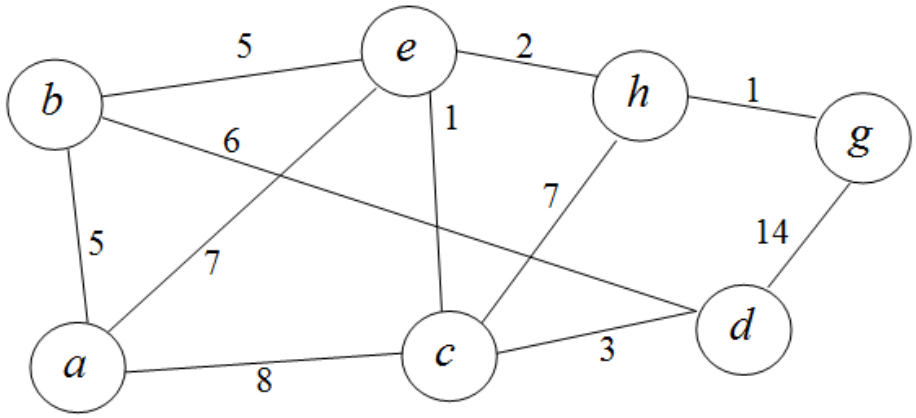
12


b
e
h
g
f
d
a
3
2
4
4
6
5
9
8
7
8
5

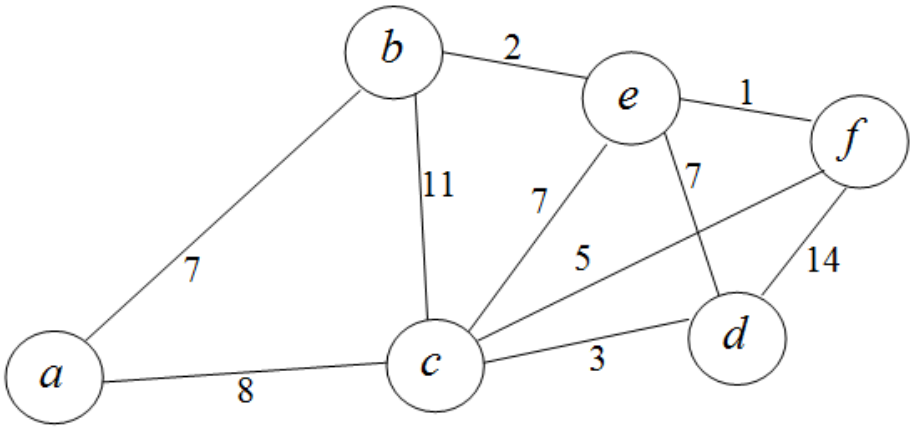
13


b
e
h
g
d
a
c
3
2
2
1
5
7
6
8
8
3

14



15



СПИСОК ЛІТЕРАТУРИ

1. Зайченко Ю.П. Дослідження операцій. – Київ: ЗАТ – ВІПОЛ, 2000. – 688 с.
2. Давыдов Э.Г. Исследование операций. – М.: Высшая школа, 1990. – 383с.
3. Вентцель Е.С. Исследование операций. Задачи, принципы, методология. – М.: Наука, 1988. – 208 с.
4. Грешилов А.А. Математические методы принятия решений: учебное пособие для вузов / Грешилов А.А. – М. : Изд-во МГТУ им. Н.Э. Баумана, 2014. – 647 с.

НАВЧАЛЬНЕ ВИДАННЯ

ЗАДАЧІ ПРО ПОТОКИ В МЕРЕЖАХ

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт №5 та №6
з дисципліни „ Дослідження операцій”
для студентів спеціальності
„ Програмна інженерія ”

Укладачі Журавчак Любов Михайлівна
Нитребич Оксана Олександрівна

Редактор

Комп’ютерне верстання