Міністерство освіти і науки України
Львівський національний університет імені Івана Франка

**М. Я. Бартіш, І. М. Дудзяний**

# Дослідження операцій

## Частина 2. Алгоритми оптимізації на графах

Рекомендовано
Міністерством освіти і науки України
як підручник
для студентів вищих навчальних закладів

Львів
Видавничий центр ЛНУ імені Івана Франка
2007

УДК 519.8
ББК 22.183
Б 26

**Рецензенти**:

д-р фіз.-мат. наук, проф. *В.А. Кривень*
(Тернопільський державний технічний університет імені Івана Пулюя);
д-р фіз.-мат. наук, проф. *І.В. Огірко*
(Українська академія друкарства, м. Львів);
д-р фіз.-мат. наук, проф. *Р.В. Слоньовський*
(Національний університет "Львівська політехніка")

*Рекомендовано до друку Міністерством освіти і науки України*
*як підручник для студентів вищих навчальних закладів*
(*лист № 14/18–Г–609 від 01.08.06 р.*)

**Бартіш М. Я., Дудзяний І. М.**

Б 26 Дослідження операцій. Частина 2. Алгоритми оптимізації на графах: Підручник. - Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. - 120 с.

ISBN 966-613-496-9
ISBN 978-966-613-533-2. Ч. 2

Висвітлено фундаментальні алгоритми оптимізації на графах: побудова мінімальних каркасів; відшукання найкоротших/найдовших шляхів в орграфах у різноманітних постановках (алгоритм Дейкстри, алгоритм Беллмана-Форда, алгоритм Флойда-Уоршолла, оптимальні шляхи в ациклічному орграфі); відшукання максимального потоку та потоку мінімальної вартості. Детально розглянуто задачі, які зводяться до задач оптимізації на графах (динамічне програмування в орграфі, узагальнена транспортна задача, календарне планування у мережах тощо). Зміст підручника відповідає програмі обов'язкового курсу "Дослідження операцій" для базового напряму "Прикладна математика".

Для бакалаврів, спеціалістів і магістрів вищих закладів освіти, де викладають предмети "Дослідження операцій", "Математичне програмування" тощо.

УДК 519.8
ББК 22.183

ISBN 966-613-496-9
ISBN 978-966-613-533-2. Ч. 2

# ЗМІСТ

## ПЕРЕДМОВА

У рамках дисципліни *дослідження операцій* (ДО) вивчається велика кількість практичних задач, які зручно інтерпретувати як задачі оптимізації на графах. Прикладами таких задач є відшукання найкоротшого маршруту між двома населеними пунктами, визначення максимальної пропускної здатності нафтопроводу, укладення календарного плану виконання робіт проекту тощо.

Задачі, які випливають із перелічених прикладів, можна сформулювати та розв'язати як задачі лінійного програмування. Однак особлива структура цих задач дає змогу розробити спеціальні алгоритми оптимізації на графах, які значно ефективніші за стандартний алгоритм симплекс-методу.

У *першому розділі* конспективно викладено базові поняття та терміни теорії графів. Розглянуто неорієнтовані графи, орієнтовані графи (або орграфи), навантажені графи. Важливе значення для розуміння наступних розділів має поняття дерева, теорема про упорядковану нумерацію вершин ациклічного орграфа, структури даних для збереження графів у пам'яті комп'ютера.

У *другому розділі* детально розглянуто питання виокремлення мінімального каркасу в неорієнтованих графах та схематично – питання виокремлення максимального орієнтованого лісу в орграфах. Математично обґрунтовано алгоритм Пріма виокремлення мінімального каркасу та наведено його реалізацію на псевдокоді для випадку, коли граф задано списком суміжних вершин.

*Третій розділ* присвячено різним варіантам задачі про найкоротші шляхи в орграфі. Математично обґрунтовані алгоритм Дейкстри та алгоритм Беллмана-Форда відшукання найкоротшого шляху від деякої вершини орграфа до всіх інших вершин, а також алгоритм Флойда-Уоршолла відшукання найкоротших шляхів між усіма парами вершин орграфа. Для ациклічного орграфа наведено алгоритм відшукання найкоротшого шляху від деякої вершини орграфа до всіх інших вершин, який базується на упорядкованій нумерації вершин цього орграфа. Усі алгоритми реалізовані на псевдокоді (орграф задано списком суміжних вершин). Наведено модифікацію деяких алгоритмів для пошуку найдовших шляхів.

У *четвертому розділі* розглянуто метод *дискретного динамічного програмування* (ДДП) в орграфі на прикладі розв'язування таких класичних задач ДО, як задача заміни устаткування, задача оптимального розподілу коштів та задача оптимального розподілу обмеженого ресурсу. Метод ДДП є одним з найефективніших методів реалізації багатокрокової процедури прийняття рішення в умовах повної визначеності. Проілюстровано, що ця процедура відповідає задачі вибору найкоротшого шляху в ациклічному орграфі.

*П'ятий розділ* присвячено розв'язуванню класичної задачі про максимальний потік, яка має важливе практичне значення. Теоретичним фундаментом цієї задачі слугує теорема Форда-Фалкерсона, яка дає змогу побудувати ефективний алгоритм відшукання максимального потоку та зв'язаного з ним мінімального перерізу. У розділі наведено механізм зведення деяких важливих задач дослідження операцій до задачі про максимальний потік.

Важливим ускладненим випадком задачі про максимальний потік є задача про потік мінімальної вартості, якій присвячено *шостий розділ* підручника. Інтерпретація цієї задачі у термінах потоку в орграфі дає змогу застосувати спеціальний алгоритм відшукання оптимального розв'язку (алгоритм викреслювання циклів), простіший за симплексний метод. У розділі наведено механізм зведення узагальненої транспортної задачі до задачі про потік мінімальної вартості.

*Сьомий розділ* висвітлює питання календарного планування робіт з виконання деякого проекту за допомогою мережевого графіка (спеціального ациклічного орграфа).

Кожний розділ підручника супроводжується розглядом прикладів, а також наводиться список запитань для самоперевірки і низка задач для самостійної роботи студентів.

Зміст підручника відповідає програмі обов'язкового курсу "Дослідження операцій" для базового напряму "Прикладна математика". Розрахований на бакалаврів, спеціалістів і магістрів. Ним можуть скористатися аспіранти та викладачі вищих закладів освіти, в яких викладають предмети "Дослідження операцій", "Математичне програмування", "Математичні методи в економіці" тощо.

# 1. ДЕЯКІ ОЗНАЧЕННЯ І ТЕОРЕМИ ТЕОРІЇ ГРАФІВ

**План викладу матеріалу**:

1. Неорієнтовані графи.
2. Орієнтовані графи.
3. Навантажені графи.

**Ключові терміни розділу**

- *Граф*: *вершини*, *ребра*
- *Орієнтований граф* (*орграф*)
- *Зображення графа*
- *Шлях і цикл в орграфі*
- *Лема про вузли ациклічного орграфа*
- *Зв'язаний граф*; *дерево*, *ліс*
- *Навантажений* (*зважений*) *граф*
- *Неорієнтовані графи*
- *Вузли і дуги орграфа*
- *Шлях і цикл у графі*
- *Ациклічний орграф*
- *Теорема про нумерацію вершин ациклічного орграфа*
- *Каркас* (*покривне дерево*)
- *Зберігання навантаженого графа*

У багатьох задачах математики, інформатики та технічних дисциплін виникає необхідність відображення відношень між деякими об'єктами. Графи – це природна модель для таких відношень. Найкращою ілюстрацією може слугувати карта автошляхів. Міста – це вершини графа, а дороги – його ребра. Іноді графи є орієнтованими (подібно до вулиць з одностороннім рухом) або навантаженими (кожен шлях має відстань і відповідну вартість проїзду).

Під час викладу матеріалу цього параграфа опиратимемося на підручник [14].

## 1.1. Неорієнтовані графи

Задано непорожню скінченну множину $V$ і множину $E$ невпорядкованих пар *різних* елементів з множини $V$. *Простим графом* $G$ називають пару множин $V$ і $E$, тобто $G = (V, E)$.

Елементи множини $V$ називають *вершинами* графа $G$, а невпорядковані пари різних вершин – *ребрами*. Кількість вершин позначають як $|V|$, а кількість ребер – як $|G|$. Якщо пара вершин $a$ і $b$ є ребром, то її позначають $[a, b]$. Вершини $a$ і $b$ є *кінцями* ребра. Зауважимо, що $[a, b]$ і $[b, a]$ позначають одне і те ж ребро.

На рисунках ребра зображають прямою чи кривою лінією без стрілок, а вершини позначають літерами або числами. Оскільки $E$ – *множина*, то у простому графі довільну пару вершин може з'єднувати не більше одного ребра.

У деяких випадках розглядають графи, у яких дві вершини може з'єднувати декілька ребер. Виникає поняття *мультиграфа* – пари $(V, E)$, де $V$ – непорожня скінченна множина, а $E$ – *сім'я* невпорядкованих пар *різних* елементів з множини $V$.

Термін "сім'я" означає, що елементи $E$ (ребра) можуть повторюватися. Ребра, які з'єднують одну й ту ж пару вершин, називають *кратними* (або *паралельними*) ребрами.

Наступне узагальнення полягає у тому, що окрім кратних ребер допускають наявність *петель* – ребер, які з'єднують вершину саму з собою. *Псевдографом* називають пару $(V, E)$, де $V$ – непорожня скінченна множина, а $E$ – *сім'я* невпорядкованих пар елементів з множини $V$ (не обов'язково різних).

Три типи графів, які введені вище, називають *неорієнтованими графами*, причому:

- *псевдограф* – може мати петлі і кратні ребра;
- *мультиграф* – може мати кратні ребра (без петель);
- *простий граф* – без петель і кратних ребер.

Вершини $a$ і $b$ в неорієнтованому графі називають *суміжними*, якщо $[a, b]$ є ребром. Два ребра називають *суміжними*, якщо вони мають спільний кінець. Вершина $v$ і ребро $e$ називають *інцидентними*, якщо вершина $v$ є кінцем ребра $e$. Отже, суміжність відображає зв'язок між однорідними елементами графа, а інцидентність – між неоднорідними елементами.

*Степінь вершини* $v$ (позначають $deg(v)$) у неорієнтованому графі – це кількість ребер, інцидентних вершині $v$, причому петлю враховують двічі. Якщо $deg(v) = 0$, то вершину $v$ називають *ізольованою*; якщо $deg(v) = 1$, то вершину $v$ називають *висячою*.

Послідовність ребер $[v_0, v_1], [v_1, v_2], [v_2, v_3], \ldots, [v_{n-1}, v_n]$ неорієнтованого графа називають *шляхом* довжини $n$, що з'єднує вершини $v_0$ та $v_n$ (ребро враховують стільки разів, скільки воно входить у шлях). Якщо при цьому $v_0 = v_n$, то шлях називають *циклом*.

Якщо усі ребра шляху/циклу *різні*, то його називають *простим* шляхом/циклом. Шлях з крайніми вершинами $v_0$ та $v_n$ позначають $<v_0, v_n>$ і називають $<v_0, v_n>$-шляхом.

Граф $G' = (V', E')$ називають підграфом графа $G = (V, E)$, якщо $V' \subseteq V$ і $E' \subseteq E$.

Підграф неорієнтованого графа називають *зв'язним*, якщо у ньому для кожної пари вершин знайдеться хоча б один шлях, що їх з'єднує. Відношення "з'єднані шляхом" є *відношенням еквівалентності* на множині вершин. Класи еквівалентності називають *компонентами зв'язності* графа (слово "*зв'язності*" у терміні іноді опускають). На рис. 1.1 зображено граф, що складається з двох компонент зв'язності: $\{a, b, f\}$ і $\{d, c, e, g\}$.

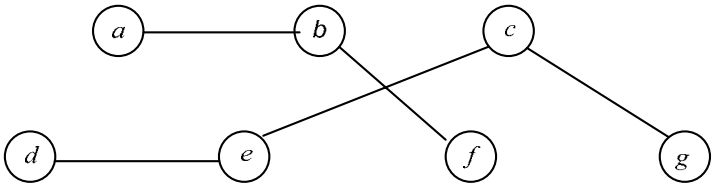



Рис. 1.1. Приклад двокомпонентного графа

Неорієнтований граф *зв'язний*, якщо він складається з єдиної компоненти зв'язності.

Граф називають *деревом*, якщо він зв'язний і не містить простих циклів. Граф, який не містить простих циклів і складається з $k$ компонент, називають *лісом* з $k$ дерев.

**Увага**! З означень випливає, що дерева і ліси є простими графами.

На рис. 1.1 зображено ліс, що складається з двох дерев. Дерева лісу можна зобразити наочніше, як на рис. 1.2.

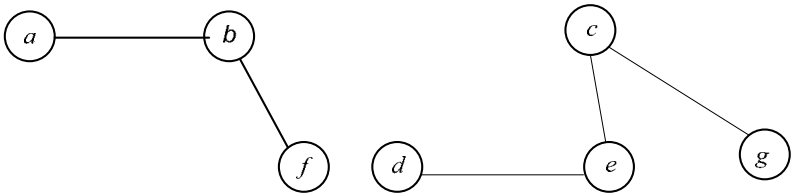



Рис. 1.2. Ліс з двох дерев

**Теорема 1.1**. Для графа $G$, що має $n$ вершин і $m$ ребер (тобто $|V| = n$; $|G| = m$ ), такі твердження *еквівалентні*:

1. $G$ – дерево.
2. $G$ – зв'язний граф і $m = n - 1$.
3. $G$ – не містить простих циклів і $m = n - 1$.
4. $G$ – зв'язний граф, проте вилучення довільного ребра робить його незв'язним.
5. Будь-які дві вершини $G$ з'єднані єдиним простим шляхом.
6. $G$ – не містить простих циклів і при з'єднанні ребром двох несуміжних вершин у новому графі буде єдиний простий цикл.

➢ Доведення див. [14] ≺

**Наслідок 1.1** (з теореми 1.1, 2). У лісі з $k$ дерев: $m = n - k$.

*Каркасом* (*з'єднувальним деревом*) простого зв'язного графа $G$ називають його підграф, який є деревом і містить усі вершини $G$.

## 1.2. Орієнтовані графи

*Орієнтованим графом* (або *орграфом*) називають пару множин $V$ і $E$, де $V$ – непорожня скінченна множина, а $E$ – множина *упорядкованих* пар елементів з множини $V$.

Елементи множини $V$ називають *вершинами* (або *вузлами*), а множини $E$ – *дугами* (або *орієнтованими ребрами*). Якщо пара вершин $a$ і $b$ є дугою, то її позначають $(a, b)$. Вершину $a$ називають *початковою*, а вершину $b$ – *кінцевою*. Дугу $(a, a)$ називають *петлею*. Зауважимо, що орграф може мати петлі (проаналізуйте означення).

На рисунках дугу позначають прямою чи кривою лінією зі стрілкою, яка вказує на кінцеву вершину. Дуга – це впорядкована пара вершин, причому дуги $(a, b)$ і $(b, a)$ є *різними* дугами.

*Орієнтованим мультиграфом* називають пару множин $V$ і $E$, де $V$ – непорожня скінченна множина, а $E$ – *сім'я* упорядкованих пар елементів з множини $V$.

Елементи $E$ (дуги) в орієнтованому мультиграфі можуть повторюватися, тоді їх називають *кратними* дугами. Кратні дуги з'єднують одну й ту ж пару вершин та *однаково напрямлені*.

Граф, в якого є і ребра, і дуги, називають *змішаним*. Його зводять до орграфа заміною кожного ребра [*a*, *b*] дугами (*a*, *b*) і (*b*, *a*).

**Приклад 1.1**. Граф, зображений на рис. 1.3, описати як орграф.

➢ Граф $G = (V, E)$, зображений на рис. 1.3, є *змішаним*: $V=\{1, 2, 3, 4\}$; $E = \{[1, 2], [1, 4], (2, 4), (3, 4)\}$. Якщо описати його орграфом, то $E =\{(1, 2), (2, 1), (1, 4), (4, 1), (2, 4), (3, 4)\}$. ≺

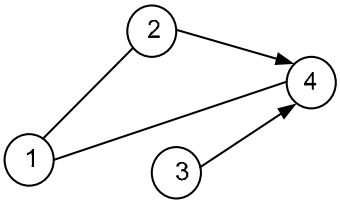


Рис. 1.3. Граф прикладу 1.1

В орграфі *напівстепенем входу* вершини $v$ називають кількість дуг з *кінцевою* вершиною $v$ (позначають $deg^-(v)$). *Напівстепенем виходу* вершини $v$ називають кількість дуг, для яких вершина $v$ є *початковою* (позначають $deg^+(v)$). Степінь вершини $v$ визначають так: $deg(v) = deg^-(v) + deg^-(v)$.

Послідовність дуг $(v_0, v_1), (v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)$ орієнтованого графа називають *орієнтованим шляхом* (або просто *шляхом*), що з'єднує вузли $v_0$ та $v_n$ (кінець довільної дуги шляху, окрім останньої, є початком наступної дуги). Довжиною шляху називають кількість дуг, з яких він складається.

В орграфі шлях однозначно визначається упорядкованою послідовністю вершин $p = <v_0, v_1, \ldots, v_{n-1}, v_n>$. У цьому випадку кажуть, що для вершин $v_0$ і $v_n$ існує шлях $p$ від $v_0$ до $v_n$ (або вершина $v_n$ є *досяжною* з вершини $v_0$) і позначають $v_0 \xrightarrow{p} v_n$.

Якщо в орієнтованому шляху $v_0 = v_n$, то цей шлях називають *орієнтованим циклом* (або просто *циклом*). Зауважимо, що петля є спеціальним типом циклу. Орграф, що не має циклів, називають *ациклічним* орграфом.

Якщо усі дуги орієнтованого шляху/циклу в орграфі – *різні*, то його називають *простим* орієнтованим шляхом/циклом.

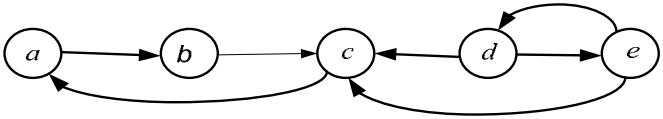


Рис. 1.4. Приклад орієнтованого графа

На рис. 1.4 можна вказати простий шлях <*a*, *b*, *c*>. Інші можливі прості шляхи: <*e*, *d*, *c*, *a*, *b*>, <*e*, *c*, *a*, *b*> і т. д. Шлях <*a*, *b*, *c*, *a*> є циклом. Знайдіть інші цикли (самостійно).

**Лема 1.1**. В ациклічному орграфі є хоча б одна вершина, в яку не входить жодна дуга.

➢ Доведення від *супротивного*: у будь-яку вершину входить дуга. Розглянемо довільну вершину $i_1$. Оскільки у цю вершину входить дуга, то перейдемо по ній до початку дуги – вершини $i_2$. У вершину $i_2$ теж входить дуга, отож перейдемо по ній до початку дуги – вершини $i_3$ і т.д. Однак орграф має скінченну кількість вершин, і в деякий момент ми знову потрапимо у вершину, в якій були раніше, тобто отримаємо цикл (*протиріччя*). ◅

**Теорема 1.2**. Вершини ациклічного орграфа можна занумерувати так, що кінець кожної дуги матиме більший номер, ніж номер початку дуги.

➢ За лемою 1.1 в ациклічному орграфі є хоча б одна вершина, в яку не входить жодна дуга. Присвоїмо їй номер 1. Якщо є декілька таких вершин, то вибираємо довільну з них.

Видаляємо з орграфа цю вершину разом з дугами, що з неї виходять. В отриманому орграфі також відсутні цикли, а тому є хоча б одна вершина, в яку не входить жодна дуга. Присвоїмо їй номер 2. Якщо є декілька таких вершин, то вибираємо довільну з них. Цей процес виконуватиметься тоди, доки не занумеруємо всі вершини орграфа. Теорему доведено. ◅

Для орграфа поняття зв'язності вводять по-різному, залежно від того, чи враховують напрям дуг. Орієнтований підграф називають *сильнозв'язним*, якщо в ньому для кожної пари вершин *u* і *v* знайдеться шлях від *u* до *v* та шлях від *v* до *u*.

Будь-який орграф можна розбити на *компоненти сильної зв'язності*, які визначають як класи еквівалентності відношення "*v* є досяжною з *u* і *u* є досяжною з *v*". Наприклад, орграф на рис. 1.4 має дві компоненти сильної зв'язності: {*a*, *b*, *c*} і {*e*, *d*}.

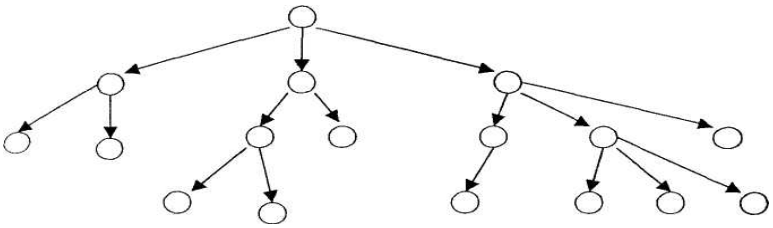
Орієнтований граф *сильнозв'язний*, якщо він складається з єдиної компоненти сильної зв'язності.

Орієнтований граф називають *слабкозв'язним*, якщо існує шлях між будь-якими двома різними вершинами в його неорієнтованому варіанті (тобто *без урахування* напряму дуг). Очевидно, що сильнозв'язний орграф є водночас і слабкозв'язним.

Орієнтований граф називають *орієнтованим* (або *кореневим*) *деревом*, яке росте з вершини $v_0$ (*кореня* дерева), якщо:

- орграф утворює дерево в його неорієнтованому варіанті;
- $v_0$ – єдина вершина орграфа, в яку не заходить жодна дуга, а у всі інші вершини заходить тільки одна дуга.

На рис. 1.5 наведено приклад орієнтованого дерева.



1.5. Орієнтоване дерево

Якщо $(a, b)$ є дугою орієнтованого дерева, то вершину $a$ називають *предком* (або *батьком*), а вершину $b$ – *нащадком* (або *сином*). Вершини орієнтованого дерева, які не мають синів, називають *листками*. Вершини (окрім кореня), які мають синів, називають *внутрішніми*.

*Орієнтований ліс* – це ліс, що складається з орієнтованих дерев. *З'єднувальним орієнтованим деревом* (або *орієнтованим каркасом*) орграфа називають орієнтоване дерево, що містить усі вершини цього орграфа. *З'єднувальним орієнтованим лісом* орграфа називають орієнтований ліс, що містить усі вершини цього графа.

### 1.3. Навантажені графи

Часто у реальних задачах ребро $[u, v]$ неорієнтованого графа чи дугу $(u, v)$ орграфа *навантажують* дійсним числом – *вагою* $w(u, v)$, яка відображає кількісну характеристику ребра/дуги (наприклад: відстань, швидкість, ціна, пропускна здатність, дохід, прибу-

ток тощо). Ваги, зазвичай, є невід'ємними числами. У деяких випадках ваги мають від'ємні значення (витрати, штрафи тощо).

Граф, у якому кожне ребро/дуга має вагу, називають *навантаженим* (у деяких джерелах – *зваженим*) графом. Приклад орієнтованого навантаженого графа наведено на рис. 1.6 (поруч з кожною дугою проставлено її вагу).

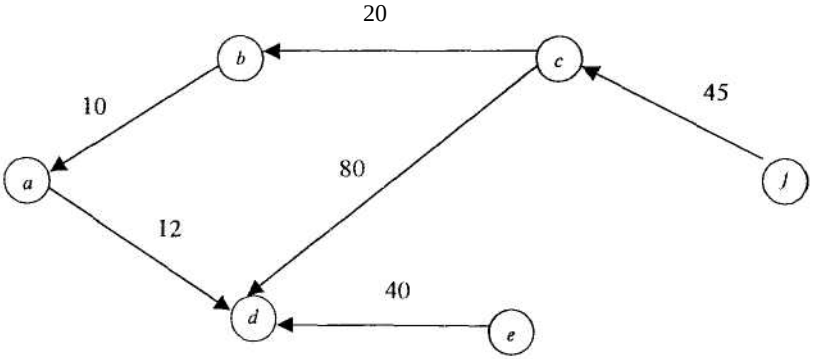



Рис. 1.6. Приклад орієнтованого навантаженого графа

Найчастіше для зберігання навантаженого графа $G = (V, E)$ у пам'яті комп'ютера використовують *матрицю ваг*. Для цього нумерують вершини графа $G$ числами 1, 2, …, $n$, де $n = |V|$, і розглядають матрицю $(w_{ij})_{n\times n}$, у якої $w_{ij} = w(i, j)$, якщо ребро $[i, j]$ чи дуга $(i, j) \in G$. Якщо ж ребро $[i, j]$ чи дуга $(i, j) \notin G$, то $w_{ij} = 0$ чи $w_{ij} = \infty$ (залежно від змісту задачі).

Якщо вершинам навантаженого орграфа на рис. 1.6 приписати номери літер в алфавітному порядку ($a - 1$, $b - 2$, $c - 3$, $d - 4$, $e - 5$, $f - 6$), то отримаємо таку матрицю ваг (справа):

$$\begin{pmatrix} 0 & 0 & 0 & 12 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 20 & 0 & 80 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 40 & 0 & 0 \\ 0 & 0 & 45 & 0 & 0 & 0 \end{pmatrix}$$

Для простого навантаженого графа матриця ваг є симетричною відносно головної діагоналі.

Матриці ваг неефективно використовують пам'ять під час збереження розріджених графів, у яких $|E| << |V|^2$. Для зберігання

розрідженого навантаженого графа $G = (V, E)$ використовують *списки суміжних вершин* (*adjacencylist representation*), базою яких є вектор вказівників **`Adj[n]`** на $n$ списків – по одному на кожну вершину графа. Список будь-якої вершини графа містить у довільному порядку номери суміжних з нею вершин та відповідні ваги.

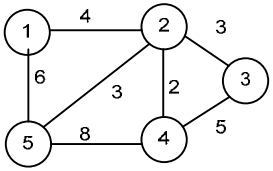



Рис. 1.7. Простий зважений граф

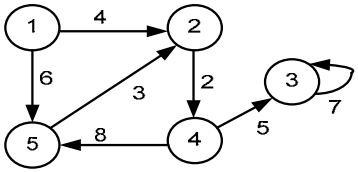
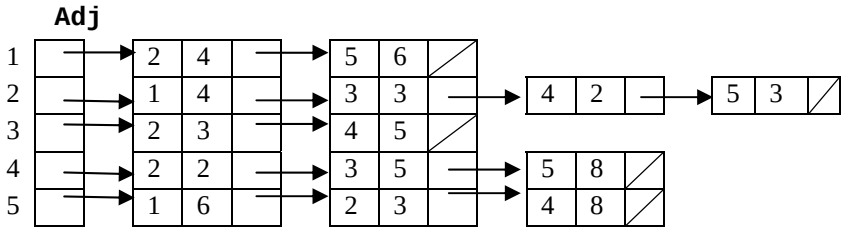


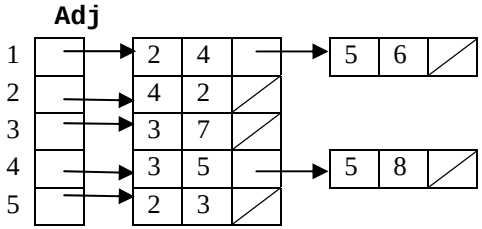

Рис. 1.8. Орієнтований зважений граф

Зображення простого навантаженого графа (рис. 1.7) списками суміжних вершин наведено у табл. 1.1, а зображення орієнтованого навантаженого графа (рис. 1.8) – у табл. 1.2. Коса риска зображає порожній вказівник (**`nil`**).

*Таблиця 1.1*. Зображення простого навантаженого графа (рис. 1.7)




*Таблиця 1.2*. Зображення орієнтованого навантаженого графа (рис. 1.8)




Для графів з великою кількістю вершин, передусім, постає задача про існування зв'язків між *усіма* його вершинами (задача встановлення факту *зв'язності графа*). Цю задачу розв'язують шляхом *виокремлення* із графа деякого каркасу.

Інша задача полягає у виокремленні таких ребер/дуг, які *мінімізують/максимізують* певну адитивну кількісну характеристику графа. Цю задачу розв'язують *виокремленням* із навантаженого графа деякого каркасу мінімальної/максимальної *сумарної ваги* (суми ваг усіх ребер/дуг дерева). Такий каркас просто називають *мінімальним/максимальним каркасом*.

Алгоритми виокремлення каркасів істотно залежать від орієнтованості графа. Приклад мінімального каркасу простого графа наведено на рис. 1.9 (жирною лінією наведено ребра каркасу сумарної ваги 37). Для графа на рис. 1.9 можна виокремити й інший каркас сумарної ваги 37, замінивши у попередньому каркасі ребро [2, 3] на ребро [1, 8].

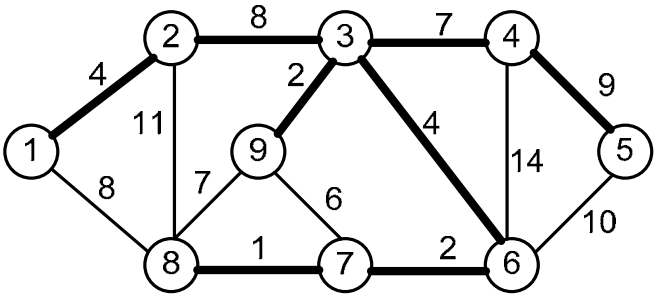



Рис. 1.9. Приклад мінімального каркасу мережі

### ? Запитання для самоперевірки

1. Сформулюйте означення простого графа.
2. Сформулюйте означення мультиграфа.
3. Сформулюйте означення дерева.
4. Скільки ребер має дерево, яке має 9 вершин?
5. Скільки вершин має дерево, яке має 6 ребер?
6. Сформулюйте означення орієнтованого графа.
7. Сформулюйте означення простого шляху в орграфі.
8. Сформулюйте і доведіть лему щодо вершини ациклічного орграфа.
9. Сформулюйте і доведіть теорему щодо вершини ациклічного орграфа.
10. Сформулюйте означення орієнтованого дерева.
11. Сформулюйте означення навантаженого графа.
12. Сформулюйте означення каркасу.

# 2. АЛГОРИТМИ ВИОКРЕМЛЕННЯ КАРКАСІВ

**План викладу матеріалу**:

1. Типові задачі виокремлення каркасів у простих графах.
2. Виокремлення мінімального каркасу в простих графах.
3. Алгоритм Пріма.
4. Каркаси в орграфах.

**Ключові терміни розділу**

- *Каркас у простому графі*
- *Переріз у простому графі*
- *Граф компонент*
- *Безпечне ребро*
- *Легке ребро*
- *Каркас в орграфі*

## 2.1. Типові задачі виокремлення каркасів у простих графах

Розглянемо типові задачі, які розв'язують у рамках теорії виокремлення каркасів у простих графах[1]. Першою сформулюємо задачу, з якої історично розпочався розвиток цієї теорії.

**Приклад 2.1**. *Поширення чуток*. Розглядають населений пункт, у якому кожен мешканець зустрічає інших мешканців і спілкується з ними. Чи може у цьому населеному пункті поширитися чутка?

➢ Нехай кожному мешканцю відповідає вершина графа. З'єднують дві вершини ребром у тому випадку, якщо відповідні мешканці спілкуються між собою. За умови зв'язності отриманого графа на поставлене запитання можна відповісти позитивно. Граф буде зв'язним, якщо для нього існує каркас. Отже, необхідно перевірити, чи у графа можна виокремити *каркас*. ≺

**Приклад 2.2**. *Аналіз мережі комунікацій*. Мережа доріг зв'язує кожен населений пункт із деякими іншими. Визначити, чи можна, користуючись цими дорогами, потрапити з кожного населеного пункту в будь-який інший.

➢ Нехай кожному населеному пункту і точці перетинання наявних доріг відповідає вершина графа. Дві вершини з'єднаємо ребром у тому випадку, якщо між ними існує сполучення. Якщо граф міс-

[1] Задачі обрано з посібника [12].

тить *каркас*, то між будь-якими пунктами можна проїхати по існуючій мережі доріг. ◅

Задача прикладу 2.2 має багато практичних узагальнень. Подібні задачі можна ставити для будь-яких комунікацій (водопроводів, газо- і нафтопроводів, електромереж, транспортних мереж, інформаційних мереж, телефонних систем тощо).

**Приклад 2.3**. *Проект реконструкції* (*будівництва*) *мережі комунікацій мінімальної вартості*. Заплановано реконструкцію мережі доріг, які сполучатимуть задану кількість населених пунктів. Вартість будівництва або реконструкції доріг між будь-якими двома пунктами відома. Необхідно реконструювати (чи побудувати) мережу доріг так, щоб вартість будівництва була *мінімальною* і цією мережею можна було б проїхати між будь-якими двома пунктами.

➢ Розв'язання задачі розпочнемо з побудови простого графа, вершини якого відповідають населеним пунктам і точкам перетинання доріг, а ребра – дорогам. Кожному ребру необхідно приписати вагу, яка дорівнює вартості будівництва (чи реконструкції) цієї ділянки дороги. Проект зводиться до виокремлення в отриманому графі *мінімального каркасу*. ◅

Задача прикладу 2.3 на побудову *мінімального каркасу* має багато практичних узагальнень, наприклад:

- як покласти телевізійний кабель між мікрорайонами міста для організації кабельного телебачення;
- як побудувати мережу доріг з твердим покриттям у сільській місцевості, яка сполучатиме усі села;
- як покласти труби для забезпечення газом населених пунктів району і т.п.

У цих проектах необхідно обирати шлях найкоротшої довжини з тим, щоб мінімізувати витрати кабелю, асфальту чи труб.

**Приклад 2.4**. *Проект реконструкції мережі комунікацій мінімальної вартості з фіксованими зв'язками*. За умовою попередньої задачі потрібно реконструювати мережу доріг так, щоб деякі ділянки доріг увійшли до створюваної мережі незалежно від вартості їхньої реконструкції.

➢ Проект зводиться до виокремлення в отриманому графі мінімального *каркасу* із заздалегідь зафіксованими зв'язками, які фіксують безпосередньо перед розв'язанням задачі. ➢

Типові задачі на побудову *максимального каркасу* найчастіше належать до сфери проектування мереж за умови максимізації прибутку від експлуатації мережі.

Зазначимо, що за умови здійснення формалізації задачі у вигляді простого графа і сформульованих вимог його оптимізації зміст початкової задачі не має значення для самого процесу розв'язування. Зміст початкової задачі знову буде необхідний для інтерпретації отриманого розв'язку.

## 2.2. Виокремлення мінімального каркасу в простих графах

Нехай задано зв'язний простий граф $G = (V, E)$ і вагову функцію $w: E \to R$. Опишемо загальний метод виокремлення мінімального каркасу для цього графа, опираючись на підручник [8].

Нехай підмножина $A \subseteq E$ є одночасно і *підмножиною* деякого *мінімального каркасу*. Шуканий каркас будують поступово:

> до початково *порожньої* множини $A$ на кожному кроці методу додається деяке ребро $[u, v]$ так, щоб множина $A \cup \{[u, v]\}$ теж була *підмножиною мінімального каркасу* (ребро $[u, v]$ називають *безпечним* ребром для $A$).

Запишемо цей метод на псевдокоді:

```
A := ∅;
Доки (A не каркас)
  { Знайти безпечне ребро [u, v] для A;
    A := A ∪ {[u, v]}; // {[u, v]} - множина
  }
Повернути A.
```

Безпечні ребра на будь-якій ітерації циклу забезпечують виконання *вимоги*, щоб $A$ залишалася підмножиною деякого мінімального каркасу (для порожньої множини вимога справджується).

Далі опишемо правило знаходження безпечних ребер (теорема 2.1). Почнемо з визначень. Нехай $S \subseteq V$ – підмножина вершин

графа *G*, які уже з'єднані ребрами підмножини $A \subseteq E$, а $V \setminus S$ – містить решту вершин *V* (очевидно, що $(V \setminus S) \cap S = \varnothing$).

*Перерізом* $(S, V \setminus S)$ неорієнтованого графа $G = (V, E)$ називають підмножину таких ребер з *E*, що один з його кінців лежить у *S*, а інший – у $V \setminus S$ (див. рис. 2.1).

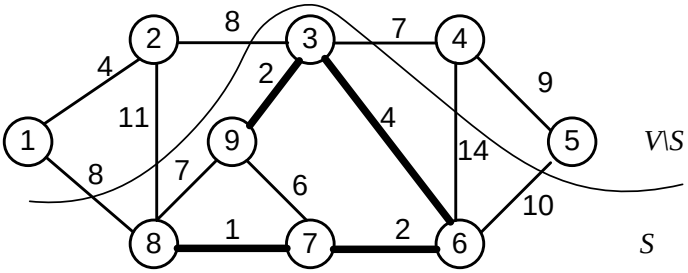



Рис. 2.1. Переріз $(S, V \setminus S)$ неорієнтованого графа *G*

Переріз *узгоджений* з множиною ребер *A*, якщо жодне ребро з *A* не перетинає цей переріз. У множині ребер, що належать перерізу, виокремлюють ребра *найменшої* ваги (назвемо їх *легкими* ребрами). На рис. 2.1 переріз узгоджений з *A*, а ребро [3, 4] – *легке* ребро.

**Теорема 2.1**. Нехай $G = (V, E)$ – зв'язний простий граф з ваговою функцією $w: E \to R$, *A* – підмножина ребер деякого мінімального каркасу $T \subseteq G$, $(S, V \setminus S)$ – переріз *G*, узгоджений з *A*. Якщо [*u*, *v*] – легке ребро цього перерізу, то [*u*, *v*] є безпечним для *A*.

➢ Нехай *T* не містить ребра [*u*, *v*], оскільки у цьому випадку твердження теореми є очевидним. Покажемо, що існує інший мінімальний каркас *T**, що містить $A \cup \{[u, v]\}$ (тоді [*u*, *v*] є безпечним).

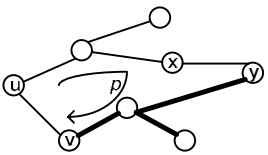



Рис. 2.2. Замикання *p*

Каркас *T* зв'язний, отож містить єдиний шлях *p* з *u* до *v* (рис.2.2); ребро [*u*, *v*] замикає *p* у цикл. Оскільки *u* і *v* належать різним підмножинам перерізу $(S, V \setminus S)$, узгодженого з *A*, то шлях *p* має принаймні одне ребро [*x*, *y*], що перетинає переріз і не лежить в *A*.

Додавши до дерева $T$ ребро $[u, v]$ і видаливши з нього ребро $[x, y]$, одержимо новий каркас $T^* = (T \cup \{[u, v]\}) - \{[x, y]\}$. Залишилося довести, що $T^*$ – мінімальний каркас.

Оскільки $[u, v]$ – легке ребро перерізу $(S, V \setminus S)$, то вилучене з $T$ ребро $[x, y]$ має не меншу вагу, ніж вага ребра $[u, v]$. Отже, вага каркасу $T^*$ могла тільки *зменшитися*. Проте каркас $T$ – *мінімальний*. Відповідно, вага нового каркасу $T^*$ дорівнює вазі $T$. Отож ребро $[u, v]$, що міститься у $T^*$, є *безпечним*. ≺

Для реалізації методу виокремлення мінімального каркасу в простих графах найчастіше використовують алгоритми *Краскала* (J. B. Kruskal) і *Пріма* (R. C. Prim). Обидва алгоритми однаково використовують загальну схему методу, проте по-різному обирають безпечне ребро. Алгоритм Краскала описано у [14], отож у наступному пункті розглянемо тільки алгоритм Пріма.

## 2.3. Алгоритм Пріма

Множина ребер $A$ зображається одним деревом, формування якого починається з довільної кореневої вершини. На кожній ітерації циклу до дерева додається ребро мінімальної ваги серед усіх ребер, які з'єднують вершини $A$ з іншими вершинами графа. Якщо таких ребер є декілька, то обирають будь-яке з них. Згідно з теоремою 2.1 ці ребра є безпечними для $A$.

Опишемо алгоритм Пріма виокремлення мінімального каркасу для простих зважених графів, які зображають *списками суміжних вершин*. На вході алгоритм отримує зв'язний простий граф $G$ з ваговою функцією $w: E \to R$ і корінь $r$ мінімального каркасу. На виході алгоритм повертає масиви $\pi$ і *key*, які виокремлюють мінімальний каркас.

Під час роботи алгоритму вершини $V \setminus S$ зберігаються у *черзі з пріоритетами* (позначимо $Q$). Пріоритет вершини $u \in Q$ (позначимо *key*$[u]$) дорівнює мінімальній вазі серед ваг усіх ребер, що з'єднують $u$ з вершинами, які належать $S = V / Q$. Якщо таких ребер ще немає, то *key*$[u]=\infty$. Поле $\pi[u]$ для вершини $u \in Q$ указує на вершину дерева $A$, до якої прямує ребро ваги *key*$[u]$.

Поле $\pi[v]$ для вершини $v \in S$ указує на вершину-предка у цьому дереві (для кореня $\pi[v] = 0$). Множину ребер дерева $A$ під час роботи алгоритму визначають як $A = \{[\pi[v], v]: v \in S = V \setminus Q\}$. Після завершення роботи алгоритму черга $Q$ – порожня, і множина $A = \{[\pi[v], v]: v \in V\}$ є множиною ребер мінімального каркасу.

Запишемо алгоритм Пріма на псевдокоді:

```
Alg_Prim(G, w, r, π, key);
 Q := V; Для (кожної вершини u ∈ Q) key[u] := ∞;
 key[r] := 0; π[r] := 0;
 Доки (Q не ∅)
  { v := Extract_Min(Q);
    Для (кожної вершини u ∈ Adj[v])
      Якщо ((u ∈ Q) І (w(u, v) < key[u]))
         { π[u] := v; key[u] := w(u, v) }
  }.
```

Процедура `Extract_Min(Q)` *обирає* з черги $Q$ вершину найменшого пріоритету (водночас вершина *вилучається* з черги).

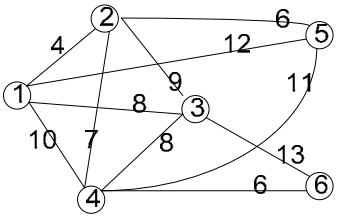



Рис. 2.3. Структура мережі

**Приклад 2.5**. Телевізійна компанія планує під'єднати до кабельної мережі п'ять районів. На рис. 2.3 проілюстровано структуру мережі і відстань (у *км*) між районами. Побудувати кабельну мережу мінімальної сумарної відстані, якщо вершина 1 відповідає телевізійній компанії.

➢ Вершина 1 на рис. 2.3 відповідає телевізійній компанії. Цю вершину доцільно обрати коренем дерева. Для кожної вершини $v$, обраної процедурою `Extract_Min`, фактично *фіксуються* значення $\pi[v]$, *key*$[v]$ (змінюватися можуть тільки елементи черги $Q$). Послідовність виконання алгоритму Пріма оформимо таблицею (елемент, який обрано з черги, позначатимемо косою лінією):

| *Черга Q* (для кожного *u*: π[*u*], *key*[*u*]) | | | | | | *v* | *Результат* |
|---|---|---|---|---|---|---|---|
| *u* = 1 | *u* = 2 | *u* = 3 | *u* = 4 | *u* = 5 | *u* = 6 | | π[*v*], *key*[*v*] |
| 0, 0 | ?, ∞ | ?, ∞ | ?, ∞ | ?, ∞ | ?, ∞ | 1 | 0, 0 |
| | 1, 4 | 1, 8 | 1, 10 | 1, 12 | ?, ∞ | 2 | 1, 4 |
| | | 1, 8 | 2, 7 | 2, 6 | ?, ∞ | 5 | 2, 6 |
| | | 1, 8 | 2, 7 | | ?, ∞ | 4 | 2, 7 |
| | | 1, 8 | | | 4, 6 | 6 | 4, 6 |
| | | 1, 8 | | | | 3 | 1, 8 |

Мінімальний каркас кабельної мережі сумарної відстані 31 *км* зображено на рис. 2.4.

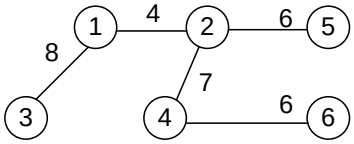



Рис. 2.4. Мінімальний каркас ⩤

Внаслідок того, що ребра з однаковими вагами обираються у довільному порядку, можна отримати декілька варіантів розв'язування. Рішення про те, який з варіантів найприйнятніший, має обумовлюватися змістом задачі і прийматися керівництвом.

Алгоритм Пріма можна застосувати для визначення *максимального* каркасу. У цьому випадку безпечним ребром буде ребро перерізу з *найбільшою* вагою.

Цей алгоритм можна також застосувати для визначення простого каркасу (без вимог оптимальності; див. прикл. 2.1 і 2.2). Оскільки у цьому випадку кожне ребро має умовну вагу 1, то безпечним ребром буде будь-яке ребро перерізу.

Припустимо тепер, що на початковій мережі деякі зв'язки зафіксовані. Така умова може бути закладена з різних причин.

Наприклад, у задачі прикладу 2.4 щодо проекту реконструкції мережі районних автошляхів з мінімальними витратами додатково ставиться природна вимога, щоб ділянки шляху, зв'язані з районним центром, належали майбутній мережі у будь-якому випадку, незалежно від вартості їхньої реконструкції.

Мінімальний каркас для цієї мережі будують відповідно до алгоритму Пріма, проте з умовою, що фіксовані зв'язки (ребра) обов'язково належатимуть каркасу. Виконання алгоритму розпочинають з ребра мінімальної ваги, яке не зафіксоване.

## 2.4. Каркаси в орграфах

Під час аналізу систем на зв'язність істотною є відмінність орієнтованих графів від неорієнтованих. Для будь-якого зв'язного неорієнтованого графа каркас завжди існує, хоча для зв'язного орграфа орієнтований каркас існує не завжди. Отож для орієнтованих графів передусім важливим залишається поняття *сильної зв'язності* та виокремлення *з'єднувального орієнтованого лісу*.

Розглянемо типові задачі виокремлення оптимального *з'єднувального* орієнтованого лісу в навантажених орграфах[1].

**Приклад 2.6**. *Оптимальне розміщення представників*. Фірма планує розмістити генеральних представників на всій території країни. З цією метою здійснено маркетингові дослідження, враховано зв'язки між ринками областей і максимальну кількість контрактів, які укладатимуть на кожному ринку. Необхідно визначити оптимальне розташування генеральних представників фірми.

➢ Побудувати орієнтований граф, вершинами якого слугуватимуть ринки країни, дугами – зв'язки між ринками, вагами вважати максимальну кількість передбачуваних контрактів фірми. Розв'язування задачі зводиться до виокремлення максимального з'єднувального орієнтованого лісу отриманого графа. Корені виявлених дерев відповідатимуть можливим представникам фірми. ≺

**Приклад 2.7**. *Розміщення складів*. Підприємство має мережу магазинів. Успішна діяльність підприємства дає змогу розширити обсяг продаж. З цією метою вирішено на території деяких магазинів розмістити склади. Магазини обслуговує кілька транспортних одиниць, які підвозять до них товар. Сформовані зв'язки між магазинами і спосіб розвезення товарів між ними відомі. У яких місцях варто розташувати склади, щоб загальний час підвезення товарів був мінімальним?

➢ Побудувати орієнтований граф. Вершинами мережі взяти точки розташування магазинів. Дугами відобразити сформовані транспортні зв'язки. Вагами дуг буде час переїзду між магазинами. Розв'язування задачі зводиться до виокремлення мінімального

[1] Задачі обрано з посібника [12].

з'єднувального орієнтованого лісу для отриманого графа. Корені виявлених дерев відповідатимуть можливим складам. ⋖

**Приклад 2.8**. *Вибір лідерів*. Нехай колектив людей націлений на виконання певного завдання. Необхідно на основі системи переваг членів колективу обрати серед них лідерів.

➢ Побудувати орієнтований граф з вершинами, які відповідають членам колективу. Дугами відобразити бажання кожного члена колективу бачити своїм лідером того чи іншого члена. За ваги дуг узяти кількісну оцінку цього бажання (бальну чи будь-яку іншу). Розв'язування задачі зводиться до виокремлення максимального з'єднувального орієнтованого лісу для графа. Корені виявлених дерев відповідатимуть ймовірним лідерам. ⋖

Чимало алгоритмів виокремлення каркасів, які працюють з орграфами, розпочинається зі *знаходження компонент сильної зв'язності*: після цього задачу розв'язують окремо для кожної компоненти, а потім розв'язки комбінують відповідно до змісту задачі та зв'язків між компонентами, які зображають "*графом компонент*".

Ідея побудови "*графа компонент*" полягає у стягуванні дуг і вершин виявлених *компонент сильної зв'язності* (циклів) у *фіктивні* вершини, як це проілюстровано на рис. 2.5.

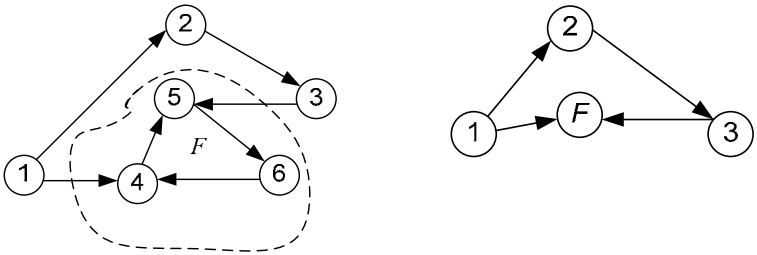



Рис. 2.5. Стягування циклу у фіктивну вершину

Розглянемо побудову максимального з'єднувального орієнтованого лісу на прикладі. Нехай мережу ринків для задачі прикладу 2.6 зображено на рис. 2.6. Вершинами орієнтованого графа, у яких бажано розмістити представників фірми, взято обласні центри (позначення латинськими літерами), а вагами – передбачувана

кількість контрактів. Побудуємо максимальний з'єднувальний орієнтований ліс для отриманого графа.

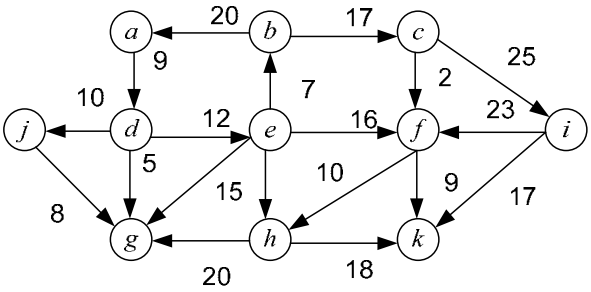



Рис. 2.6. Мережа ринків прикладу 2.6

Послідовно переглядаючи вершини, можна візуально визначити цикл, утворений вершинами *a*, *d*, *e*, *b*. Замінимо цикл фіктивною вершиною *F* (рис. 2.7). Решта вершин мережі циклів не утворює. Виходячи з вимоги максимальності лісу, вершину *j* логічно зв'язати з фіктивною вершиною *F*, а для решти вершин виокремити максимальний каркас. На рис. 2.8 зображено два дерева, які у підсумку отримаємо.

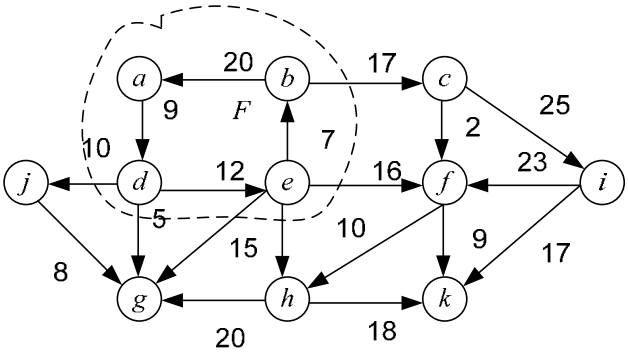



Рис. 2.7. Виокремлення фіктивної вершини

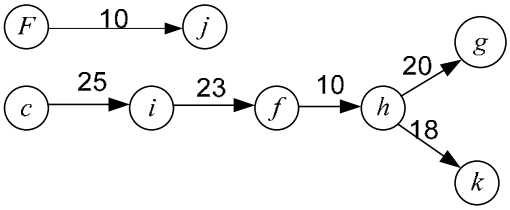



Рис. 2.8. Дерева максимального з'єднувального лісу

Розгорнемо фіктивну вершину F. У циклі, що її утворює, видалимо дугу $(e, b)$, що має найменшу вагу. Отримаємо дерево, зображене на рис. 2.9.

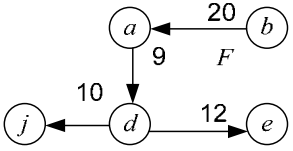



Рис. 2.9. Розгортання вершини $F$

Алгоритм виявлення компонент сильної зв'язності детально описано у підручнику [8]. В основу алгоритму покладено метод *пошуку углиб* (або DFS-метод) у зв'язному орграфі [8, 14].

## ? Запитання для самоперевірки

1. Сформулюйте означення мінімального каркасу.
2. Нехай $[u, v]$ – ребро мінімальної ваги у графі $G$. Доведіть, що $[u, v]$ належить деякому мінімальному каркасу цього дерева.
3. Доведіть, що якщо ребро $[u, v]$ міститься у мінімальному каркасі графа $G$, то існує деякий переріз графа $G$, для якого $[u, v]$ є легким ребром, що перетинає цей переріз.
4. Нехай $T$ – мінімальний каркас графа $G$. Складемо упорядкований список ваг усіх ребер каркасу $T$. Доведіть, що для довільного іншого каркасу отримаємо такий же список.
5. Доведіть, що якщо для довільного перерізу графа $G$ існує єдине легке ребро, яке перетинає цей переріз, то у графі є тільки один мінімальний каркас.
6. Як може змінитися кількість сильнозв'язаних компонент орграфа за умови додавання до нього однієї дуги?
7. Сформулюйте і доведіть теорему щодо легких ребер графа.
8. Сформулюйте алгоритм Пріма, якщо граф задано матрицею ваг.
9. Сформулюйте означення з'єднувального орієнтованого лісу.

## 🕮 Завдання для самостійної роботи

**Завдання 2.1**. Нехай заплановано побудувати мережу автошляхів з твердим покриттям, щоб з'єднати усі села району. Вартість будівництва автошляхів між будь-якими двома селами відома (рис.2.10).

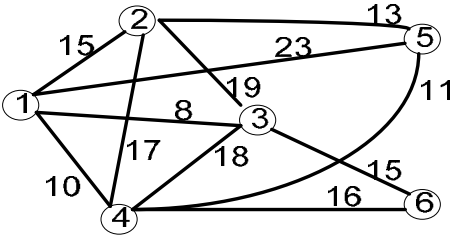



Рис. 2.10. Вартості будівництва автошляхів між селами

Необхідно побудувати мережу доріг так, щоб вартість будівництва була мінімальною і цією мережею можна було б проїхати між будь-якими двома селами.

**Завдання 2.2**. Нехай заплановано побудувати газопроводи, щоб з'єднати газовидобувні платформи у морі. Відстань (у *км*) між будь-якими двома платформами відома (рис.2.11). Платформа 1 знаходиться найближче до берега, отож вона має необхідне обладнання, щоб перекачувати газ від інших платформ до приймальної станції на березі моря.

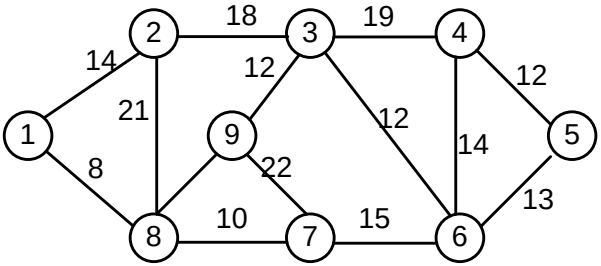



Рис. 2.11. Відстані між платформами

Необхідно спроектувати мережу газопроводів мінімальної сумарної довжини.

# 3. ОПТИМАЛЬНІ ШЛЯХИ В ОРГРАФАХ

**План викладу матеріалу**:

1. Постановка задач.
2. Найкоротші шляхи та релаксація.
3. Алгоритм Дейкстри.
4. Алгоритм Беллмана-Форда.
5. Оптимальні шляхи в ациклічному орграфі.
6. Алгоритм Флойда-Уоршолла.

**Ключові терміни розділу**

- *Довжина шляху*
- *Цикл від'ємної ваги*
- *Релаксація дуг*
- *Найдовший шлях*
- *Найкоротший шлях*
- *Попередник вершини*
- *Властивості релаксації дуг*
- *Шляхи в ациклічному орграфі*

## 3.1. Постановка задач

Під час викладу матеріалу цього параграфа опиратимемося на підручник [8].

Розглядаємо навантажений орієнтований граф, який може мати чи не мати циклів. Довжина довільної дуги ($a$, $b$) може не збігатися з довжиною дуги ($b$, $a$). Для неорієнтованих графів будь-яке ребро [$i$, $j$] довжиною $d_{ij}$ можна замінити парою дуг ($i$, $j$) та ($j$, $i$), кожна з яких матиме довжину $d_{ij}$.

Під оптимальними шляхами у графі, зазвичай, розуміють *найкоротші шляхи* між двома (або між усіма) вершинами графа. Проте у деяких задачах необхідно визначати і *найдовші шляхи*.

Уточнимо деякі визначення. У задачі про найкоротші шляхи дано орієнтований граф $G = (V, E)$ з ваговою функцією $w: E \to R$. *Довжиною* шляху $p = \langle v_0, v_1, \ldots, v_{n-1}, v_n \rangle$ називають суму довжин (ваг) дуг, які належать до цього шляху, тобто $w(p) = \sum_{i=1}^{n} w(v_{i-1}, v_i)$.

*Довжину найкоротшого шляху* з $u$ в $v$ визначають так:

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \xrightarrow{p} v\}, & \text{якщо } \exists u \xrightarrow{p} v, \\ \infty, & \text{інакше.} \end{cases}$$

Отже, якщо деякий шлях $p$ з $u$ до $v$ є *найкоротшим* шляхом з $u$ до $v$, то $w(p) = \delta(u, v)$. Визначимо такі задачі:

1. Від заданої *початкової* вершини графа $s$ виокремити найкоротші шляхи до *всіх* інших вершин графа.
2. Від *усіх* вершин графа виокремити найкоротші шляхи до деякої *кінцевої* вершини графа $t$.
3. Від *заданої* вершини графа $v_i$ виокремити найкоротший шлях до деякої *іншої* вершини графа $v_j$.
4. Виокремити найкоротші шляхи між *усіма парами* вершин графа.

Алгоритм, який розв'язує першу задачу, дає змогу розв'язати й інші три задачі. Розв'язування задачі 2 зводиться до розв'язування задачі 1, якщо умовно поміняти напрям усіх дуг. Якщо відшукаємо найкоротші шляхи від початкової вершини $v_i$ до *всіх* інших вершин графа, то одночасно відшукаємо і найкоротший шлях від вершини $v_i$ до вершини $v_j$ (задача 3).

Задачу 4 можна розв'язати або багатократним застосуванням алгоритму задачі 1, де на кожнім кроці за початкову вершину $s$ беруть інші вершини графа, або однократним застосуванням *спеціального* алгоритму.

У деяких застосуваннях ваги дуг можуть бути *від'ємними*. У цьому випадку важливо, чи є цикли від'ємної довжини. Якщо з вершини $s$ можна добратися до такого циклу, то потім його можна обходити як завгодно довго, і вага шляху усе зменшуватиметься, отож для вершин цього циклу найкоротших шляхів не існує (рис. 3.1).

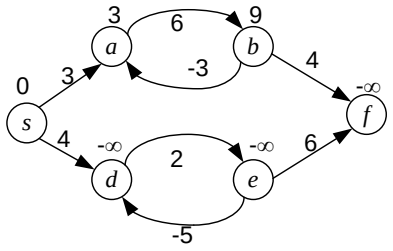


Рис. 3.1. Цикл від'ємної ваги

У цьому випадку вважатимемо, що вага найкоротшого шляху дорівнює $-\infty$. На рис. 3.1 поблизу кожної вершини проставлено вагу найкоротшого шляху від вершини $s$. Якщо ж циклів від'ємної ваги немає, то будь-який цикл можна викинути, не подовжуючи шляху.

### 3.2. Найкоротші шляхи та релаксація

Для зображення дуг найкоротших шляхів використовуватимемо такий же спосіб, як і в алгоритмі Пріма. Для кожної вершини $v \in V$ зберігатимемо її *попередника* (*predecessor*) $\pi[v]$. Попередник вершини – це або інша вершина, або 0 (для початкової вершини $s$).

Під час роботи алгоритмів виокремлення найкоротших шляхів утворюється орієнтований підграф попередників $G_\pi = (V_\pi, E_\pi)$, де $V_\pi = \{v \in V : \pi[v] \neq 0\} \cup \{s\}$; $E_\pi = \{(\pi[v], v) \in E : v \in V_\pi \setminus \{s\}\}$. Після закінчення роботи цих алгоритмів граф $G_\pi$ буде *деревом найкоротших шляхів* від кореня $s$ до будь-якої, *досяжної* з кореня, вершини $v \in V$. Найкоротших шляхів від $s$ до вершин, досяжних з $s$, а також дерев найкоротших шляхів у графі $G$, може бути декілька.

Усі алгоритми виокремлення найкоротшого (або найдовшого) шляху в графі є *багатокроковими процедурами ухвалення рішень* в умовах визначеності, оскільки при досягненні певної вершини графа обирається наступна дуга цього шляху. Під час реалізації цієї *процедури* користуються *принципом оптимальності* (або принципом Беллмана[1]), який для навантаженого орграфа формулюють такою лемою.

**Лема 3.1**. *Відрізки найкоротших шляхів є найкоротшими*. Нехай $G = (V, E)$ – орієнтований граф з ваговою функцією $w: E \to R$. Якщо $p = \langle v_0, v_1, \ldots, v_{n-1}, v_n \rangle$ – найкоротший шлях з $v_0$ до $v_n$ і $0 \le i < j \le n$, то $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_{j-1}, v_j \rangle$ є найкоротшим шляхом з $v_i$ до $v_j$.

➢ Доведення від супротивного. Нехай шлях $p_{ij}$ не найкоротший. Замінивши шлях з $v_i$ до $v_j$ на коротший, ми тим самим зменшуємо довжину шляху з $v_0$ до $v_n$ (*протиріччя*). ➢

**Наслідок з леми 3.1**. Нехай $G = (V, E)$ – орієнтований граф з ваговою функцією $w: E \to R$ і $p$ – найкоротший шлях з $s$ до $v$. Якщо $(u, v)$ – остання дуга цього шляху, то $\delta(s, v) = \delta(s, u) + w(u, v)$.

➢ Нехай $p'$ – шлях з $s$ до $u$. За лемою 3.1 $p'$ – найкоротший шлях, отож $\delta(s, v) = w(p') + w(u, v) = \delta(s, u) + w(u, v)$. ➢

[1] Вчений, який вперше сформулював принцип оптимальності.

**Лема 3.2**. Нехай $G = (V, E)$ – орієнтований граф з ваговою функцією $w: E \to R$ і $s \in V$. Тоді для будь-якої дуги $(u, v) \in E$ вірна нерівність $\delta(s, v) \le \delta(s, u) + w(u, v)$.

➢ Довжина найкоротшого шляху з $s$ до $v$ не перевищує довжину будь-якого шляху з $s$ до $v$, у тім числі й шляху через дугу $(u, v)$. ◅

Усі алгоритми виокремлення найкоротшого (або найдовшого) шляху, які описано у цьому параграфі, базуються на принципі *релаксації* (від *relaxation* – "полегшення"). Згідно з цим принципом кожній вершині $v \in V$ приписують число $d[v]$ – верхню оцінку довжини найкоротшого шляху з $s$ до $v$ (або просто, *оцінку найкоротшого шляху*).

Початкові значення оцінок найкоротшого шляху та елементів масиву $\pi$ встановлюють такою процедурою:

```
Init_Source(G, s, π, d);
 Для (усіх вершин v ∈ V[G]) {d[v] := ∞; π[v] := 0;}
 d[s] := 0.
```

*Релаксація* дуги $(u, v) \in E$ полягає у такому:

> значення $d[v]$ зменшується до величини $d[u] + w(u, v)$, якщо $d[u] + w(u, v) < d[v]$.

У цьому випадку $d[v]$ залишиться верхньою оцінкою (див. теорему 3.1). Водночас значення $\pi[v]$ має указувати на шлях, використаний при одержанні цієї верхньої оцінки, отож водночас змінюється значення $\pi[v]$:

```
Relax(u, v, w, π, d);
 Якщо (d[v] > d[u] + w(u, v))
  { d[v] := d[u] + w(u, v); π[v] := u;}.
```

На рис. 3.2 наведено два приклади релаксації: в одному випадку оцінка найкоротшого шляху зменшується, в іншому – ні. Над вершинами указані оцінки найкоротших шляхів.

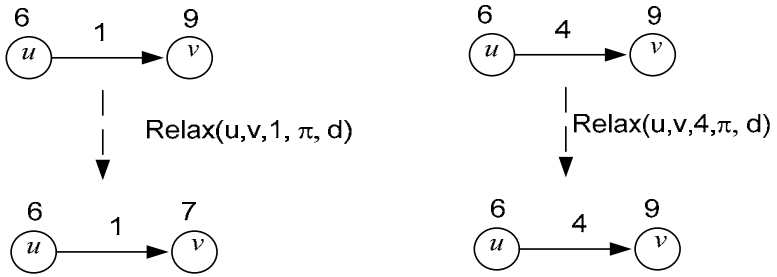



Рис. 3.2. Приклади релаксації дуг

**Теорема 3.1**. *Властивості релаксації дуг*. Нехай $G = (V, E)$ – орієнтований граф з ваговою функцією $w: E \to R$, $s \in V$ – початкова вершина і $(u, v) \in E$. Тоді після виконання процедури `Init_Source(G, s, π, d)` і довільної послідовності релаксації дуг, для кожної вершини $v \in V$ виконується таке:

$$\delta(s, v) \le d[v] \le d[u] + w(u, v). \qquad (3.1)$$

Якщо ж для деякої вершини $v$ маємо $\delta(s, v) = d[v]$, то ця рівність залишиться вірною і при наступних релаксаціях дуг.

➢ Права частина нерівності (3.1) є очевидною (відповідно до визначень оцінки найкоротшого шляху та релаксації дуги).

Доведемо ліву частину нерівності (3.1). Після ініціалізації значення $d[v]$ – нескінченні при $v \ne s$, отож слугують оцінкою зверху для чого завгодно, а $d[s] = 0$ (теж правильно). Якщо ж для дуги $(u, v)$ здійснено релаксацію, то

$$d[v] = d[u] + w(u, v) \ge \delta(s, u) + w(u, v) \ge \delta(s, v). \qquad (3.2)$$

Остання нерівність (3.2) виконується згідно з лемою (3.2).

У процесі релаксації значення $d$ можуть тільки зменшуватися, а після досягнення рівності $d[v] = \delta(s, v)$ далі зменшуватися нікуди. ≺

**Теорема 3.2**. Нехай $G = (V, E)$ – орієнтований граф з ваговою функцією $w: E \to R$, $s \in V$ – початкова вершина і $s \longrightarrow v$ – найкоротший шлях від $s$ до $v$ з останньою дугою $(u, v)$. Нехай виконано процедуру `Init_Source(G, s, π, d)` і довільну послідовність ре-

лаксації дуг, у тім числі дуги $(u, v)$. Якщо у деякий момент до релаксації дуги $(u, v)$ виконувалася рівність $d[u] = \delta(s,u)$, то після релаксації дуги $(u, v)$ виконуватиметься рівність $d[v] = \delta(s,v)$.

➢ Рівність $d[u] = \delta(s,u)$ за теоремою 3.1 збережеться і після релаксації дуги $(u, v)$. Тоді $d[v] \le d[u]+w(u, v) = \delta(s,u)+w(u, v) = \delta(s,v)$ (остання рівність за наслідком з леми 3.1).

Однак за теоремою 3.1: $\delta(s,v) \le d[v]$, отож $d[v] = \delta(s,v)$, що і треба було довести. ➢

**Лема 3.3**. Нехай $G = (V, E)$ – орієнтований граф з ваговою функцією $w: E \to R$ і $s \in V$ – початкова вершина, причому у графі $G$ немає циклів від'ємної ваги, досяжних з $s$. Тоді після виконання процедури `Init_Source(G, s, π, d)` і довільної послідовності релаксації дуг, підграф попередників $G_\pi$ буде деревом з коренем $s$.

➢ Доведення леми можна знайти у [8, ст. 486]. ➢

### 3.3. Алгоритм Дейкстри

Алгоритм Дейкстри розв'язує задачу про найкоротші[1] шляхи для навантаженого орієнтованого графа $G = (V, E)$ від початкової вершини $s \in V$ до всіх інших вершин, досяжних з $s$, у якому довжини усіх дуг *невід'ємні*. Вважають, що граф задано за допомогою *списків суміжних вершин*.

У процесі роботи алгоритму Дейкстри підтримується множина $S \subseteq V$, яка складається з вершин $v$, для яких $\delta(s,v)$ уже знайдено (тобто $d[v] = \delta(s,v)$). Вершини $u$, які не належать $S$ (тобто вершини $u$ з підмножини $V \setminus S$) зберігаються у черзі $Q$ із пріоритетами, обумовленими значеннями $d[u]$.

Алгоритм вибирає з черги $Q$ чергову вершину $u$, яка має найменший пріоритет $d[u]$, додає її до множини $S$ (тобто відбувається неявний перехід вершини типу $u \in V \setminus S$ до вершини типу $v \in S$) і здійснює релаксацію усіх ребер, що виходять з $u$, після чого цикл повторюється.

---

[1] У деяких випадках алгоритм видає правильні результати і під час пошуку найдовших шляхів (для ациклічних орграфів – *завжди* правильні).

```
DEJKSTRA(G, w, s, π, d);
 Init_Source(G, s, π, d); S := ∅;
 Q := V[G]+d; // формування черги з пріоритетами
 Доки (Q ≠ ∅)
  { u := EXTRACT_MIN(Q); S := S ∪ {u}; // множина
    Для (усіх y ∈ Adj[u]) RELAX(u, y, w, π, d);
  }.
```

У циклі **Доки** кожна вершина додається до множини $S$ лише один раз. Отже, кількість ітерацій циклу **Доки** дорівнює $|V|$.

Обґрунтування правильності алгоритму Дейкстри можна знайти у [8, 14].

**Приклад 3.1**. Визначити найкоротші шляхи від вершини 1 до інших вершин орієнтованого графа на рис. 3.3.

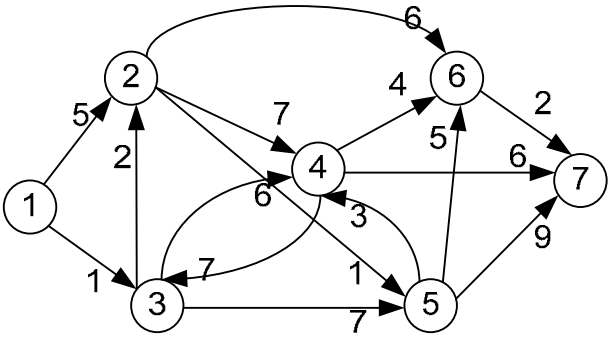



Рис. 3.3. Орієнтований граф

➢ Процес відшукання найкоротших шляхів від вершини 1 до інших вершин орієнтованого графа за допомогою алгоритму Дейкстри оформимо таблицею (табл. 3.1).

Для зручності обчислень, які зв'язані з релаксацією дуг, на початку кожного рядка вказано вершину $u$, яка є початком дуг, що піддаються релаксації. Дерево найкоротших шляхів для цього орграфа зображено на рис. 3.4 (над вершиною $u$ зазначено $\delta[s, u]$).

*Таблиця 3.1.* Відшукання найкоротших шляхів (алгоритм Дейкстри)

| *Черга Q* (для кожного *u*: π[*u*], *d*[*u*]) | | | | | | | *Результат* |
|---|---|---|---|---|---|---|---|
| *u* = 1 | *u* = 2 | *u* = 3 | *u* = 4 | *u* = 5 | *u* = 6 | *u* = 7 | π[*v*], *v*, *d*[*v*] |
| 0, 0 | 0, ∞ | 0, ∞ | 0, ∞ | 0, ∞ | 0, ∞ | 0, ∞ | 0, 1, 0 |
| *u*=1 | 1, 5 | 1, 1 | 0, ∞ | 0, ∞ | 0, ∞ | 0, ∞ | 1, 3, 1 |
| *u*=3 | 3, 3 | | 3, 7 | 3, 8 | 0, ∞ | 0, ∞ | 3, 2, 3 |
| *u*=2 | | | 3, 7 | 2, 4 | 2, 9 | 0, ∞ | 2, 5, 4 |
| *u*=5 | | | 3, 7 | | 2, 9 | 5, 13 | 3, 4, 7 |
| *u*=4 | | | | | 2, 9 | 5, 13 | 2, 6, 9 |
| *u*=6 | | | | | | 6, 11 | 6, 7, 11 |

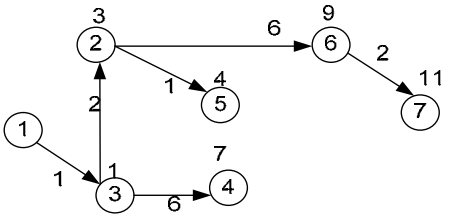



Рис. 3.4. Дерево найкоротших шляхів орграфа на рис. 3.3 ➢

**Приклад 3.2**. Визначити *найдовші* шляхи від вершини 1 до інших вершин орієнтованого графа на рис. 3.3.

➢ Модифікуємо процедуру `Initi_Source` так, щоб $d[s] = 1$, а для решти вершин $v$: $d[v] = 0$. У цьому випадку необхідно "навантажувати" (або дерелаксовувати) дуги. *Дерелаксація* дуги $(u, v)$ полягатиме у заміні $d[v]$ на $d[u] + w(u, v)$, якщо $d[u] + w(u, v) > d[v]$. Алгоритм Дейкстри у цьому випадку модифікують так:

```
…
Доки (Q ≠ ∅)
 { u := EXTRACT_MAX(Q);  Якщо (u = s) d[u] = 0;
   S := S ∪ {u}; // фігурні дужки задають множину
   для (усіх y ∈ Adj[u]) DERELAX(u, y, w, π, d); }.
```

Дерево найдовших шляхів від вершини 1 до інших вершин орієнтованого графа (рис. 3.3) зображено на рис. 3.5.

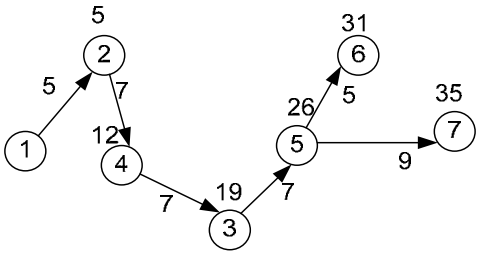



Рис. 3.5. Дерево найдовших шляхів графа (рис. 3.3) ➢

### 3.4. Алгоритм Беллмана-Форда

Алгоритм Беллмана-Форда (Bellman-Ford) розв'язує задачу про найкоротші шляхи з однієї початкової вершини до решти вершин орієнтованого графа для випадку, коли ваги дуг можуть бути від'ємними. Цей алгоритм повертає значення **TRUE**, якщо в графі немає циклу від'ємної ваги, досяжного з початкової вершини, і **FALSE**, якщо такий цикл є. У першому випадку алгоритм знаходить найкоротші шляхи; у другому випадку найкоротших шляхів (принаймні для деяких вершин) не існує.

Запишемо алгоритм Беллмана-Форда на псевдокоді:

```
BELLMAN-FORD(G, w, s, π, d, ozn);
 Init_Source(G, s, π, d);
 Для (i:= 1; i<= |V[G]| - 1; i++)
  Для (кожної дуги (u, v) ∈ E[G]) RELAX(u, v, w, π, d);
 Для (кожної дуги (u, v) ∈ E[G])
  Якщо (d[v] > d[u] + w(u, v))
       {ozn:= FALSE; Вихід_з_алгоритму; }
  ozn:= TRUE.
```

Після ініціалізації алгоритм |V| – 1 раз робить одне і те ж:

> перебирає по одному разові усі дуги графа і піддає кожне з них релаксації.

Наприкінці алгоритм перевіряє, чи немає циклу від'ємної ваги, досяжного з початкової вершини $s$.

Доведемо, що алгоритм Беллмана-Форда працює правильно.

**Лема 3.4**. Нехай $G = (V, E)$ – орієнтований граф з ваговою функцією $w: E \to R$ і $s \in V$ – початкова вершина, причому у графі $G$ немає циклів від'ємної ваги, досяжних з $s$. Тоді після закінчення роботи процедури `BELLMAN-FORD` рівність $d[v] = \delta(s, v)$ виконуватиметься для усіх вершин $v$, досяжних з $s$.

➢ Нехай $p = <s = v_0, v_1, ..., v_k = v>$ – найкоротший шлях з $s$ у $v$. Можна вважати, що цей шлях не містить циклів (вони тільки збільшують сумарну вагу шляху), отож $k \le |V| - 1$.

Доведемо за індукцією, що після $i$-ої ітерації циклу **`Для`** буде виконана рівність $d[v_i] = \delta(s, v_i)$. Для $i = 0$ це очевидно, оскільки $d[s] = \delta(s, s) = 0$ при вході у цикл.

Нехай після $(i - 1)$-ої ітерації було $d[v_{i-1}] = \delta(s, v_{i-1})$. Під час $i$-ої ітерації відбудеться релаксація ребра $(v_{i-1}, v_i)$, після чого за теоремою 1.5 установиться рівність $d[v_i] = \delta(s, v_i)$.

Оскільки всього виконується $|V| - 1$ ітерація, то за теоремою 3.1 рівність $d[v] = \delta(s, v)$ буде виконуватися для усіх вершин $v$, досяжних з $s$. ➢

**Теорема 3.3**. *Правильність алгоритму Беллмана-Форда*. Нехай $G = (V, E)$ – орієнтований граф з ваговою функцією $w: E \to R$, $s \in V$ – початкова вершина. Якщо у графі $G$ немає циклів від'ємної довжини, досяжних з $s$, то процедура `BELLMAN-FORD` повертає значення **`TRUE`**. Якщо ж у графі $G$ є цикли від'ємної ваги, досяжні з $s$, то процедура `BELLMAN-FORD` повертає значення **`FALSE`**.

➢ Якщо у графі $G$ немає циклів від'ємної ваги, досяжних з $s$, то за лемою 3.4 після виконання процедури `BELLMAN-FORD` рівність $d[v] = \delta(s, v)$ виконуватиметься для усіх вершин $v$, досяжних з $s$. Згідно з формулою (3.1) для довільної дуги $(u, v)$: $d[v] \le d[u] + w(u, v)$.

А це означає, що в процедурі `BELLMAN-FORD` жодна умова `d[v] > d[u] + w(u, v)` не виконуватиметься, і алгоритм поверне значення **`TRUE`**.

Нехай тепер у графі є цикл від'ємної ваги $<v_0, v_1, ..., v_k = v_0>$, досяжний з вершини $s$. Доведемо, що алгоритм поверне **FALSE**. Справді, якщо

$$d[v_i] \le d[v_{i-1}] + w(v_{i-1}, v_i),\ i = \overline{1,k}$$

то, додаючи ці $k$ нерівностей і скорочуючи $\sum d[v_i]$ в обох частинах, одержимо $0 \le \sum_{i=1}^{k} w(v_{i-1}, v_i)$, що суперечить вибору циклу.

Виходить, що для деякого $i$ маємо $d[v_i] > d[v_{i-1}] + w(v_{i-1}, v_i)$ і алгоритм поверне значення **FALSE**. ◅

**Приклад 3.3**. Визначити *найкоротші* шляхи від вершини $z$ до інших вершин орієнтованого графа (рис. 3.6).

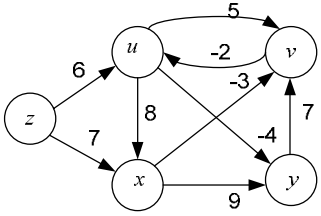



Рис. 3.6. Орієнтований граф з від'ємними вагами

➢ Дуги в алгоритмі можна піддавати релаксації у довільному порядку. Для зручності у даному прикладі при кожній ітерації циклу дуги піддаються релаксації в лексикографічному порядку: $(u, v)$, $(u, x)$, $(u, y)$, $(v, u)$, $(x, v)$, $(x, y)$, $(y, v)$, $(z, u)$, $(z, x)$.

Послідовні стани мережі після кожної ітерації циклу зображено на рис. 3.7 – 3.10. Дуги найкоротшого шляху – потовщені.

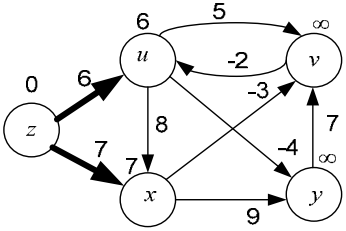



Рис. 3.7. Перша ітерація

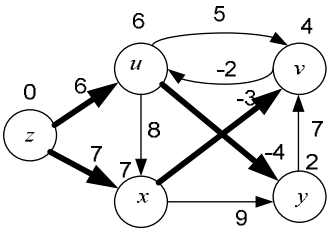



Рис. 3.8. Друга ітерація

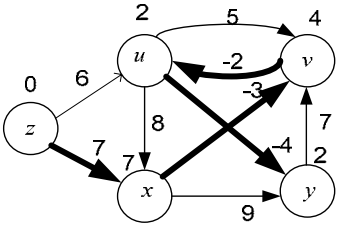



Рис. 3.9. Третя ітерація

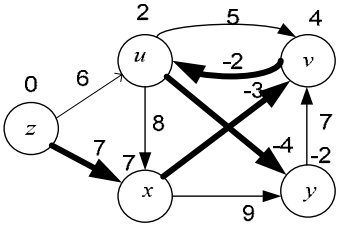



Рис. 3.10. Четверта ітерація

У цьому прикладі алгоритм поверне значення **TRUE**. ⋖

### 3.5. Оптимальні шляхи в ациклічному орграфі

Розглянемо задачу пошуку найкоротших шляхів від деякої початкової вершини $s$ до інших вершин в ациклічному орграфі. За теоремою 1.2 для такого орграфа можлива нумерація вершин, за якої для кожної дуги $(i, j)$ справедлива нерівність $i < j$.

Після такої нумерації усі дуги орграфа прямують в одному напрямі. В алгоритмі пошуку найкоротших шляхів в ациклічному орієнтованому графі вершини переглядають у заданому порядку від початкової вершини. Для кожної вершини піддають релаксації усі дуги, які виходять з неї. Запишемо алгоритм на псевдокоді:

```
Dag_Shortest_Paths(G, w, s, π, d);
 Відсортувати_вершини(V[G]);
 Init_Source(G, s, π, d);
 Для (усіх вершин і відповідно до упорядкування)
  Для (усіх вершин j ∈ Adj[i]) RELAX(i, j, w, π, d);
```

Доведемо, що алгоритм `Dag_Shortest_Paths` правильний.

**Теорема 3.4**. Нехай $G = (V, E)$ – орієнтований граф без циклів з ваговою функцією $w: E \to R$, $s \in V$ – початкова вершина. Тоді після завершення роботи процедури `Dag_Shortest_Paths` для усіх $v \in V$ виконуватимуться рівності $d[v] = \delta(s, v)$, а підграф попередників $G_\pi$ буде деревом найкоротших шляхів.

➢ Якщо вершина $v$ недосяжна з $s$, то $d[v] = \delta(s, v) = \infty$. Нехай вершина $v$ досяжна з $s$ і $p = <s = v_0, v_1, ..., v_k = v>$ – найкоротший шлях

з $s$ до $v$. Оскільки вершини упорядковані, то передусім опрацьовують дугу $(v_0, v_1)$, потім дугу $(v_1, v_2)$ і т.д. Індукція по $i$ (аналогічно доведенню леми 1.5) доводить, що $d[v_i] = \delta(s, v_i)$ для усіх $i$. ◅

**Приклад 3.4**. Знайти найкоротші шляхи від вершини 2 до інших вершин орграфа на рис. 3.11. Вершини мережі перенумеровані так, що для будь-якої дуги $(i, j)$ справедлива нерівність $i < j$

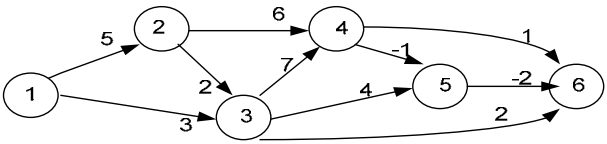



Рис. 3.11. Ациклічний упорядкований орграф

➢ Стан орграфа після ініціалізації зображено на рис. 3.12.

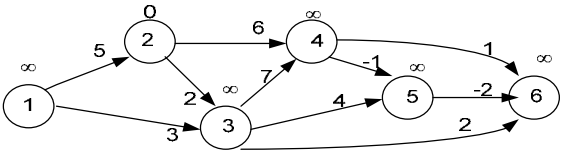



Рис. 3.12. Стан орграфа після ініціалізації

Стан орграфа після першої ітерації ($i = 1$) не змінюється (рис. 3.12).

Послідовні стани мережі після наступних ітерацій циклу по $i$ зображено на рис. 3.13 – 3.16. Дуги найкоротшого шляху – потовщені.

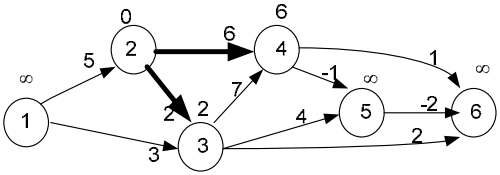



Рис. 3.13. Стан орграфа після другої ітерації ($i = 2$)

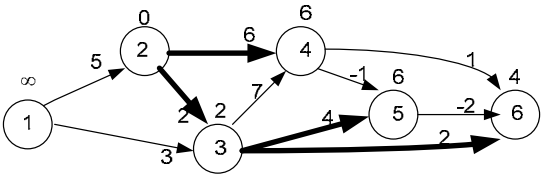



Рис. 3.14. Стан орграфа після третьої ітерації ($i = 3$)

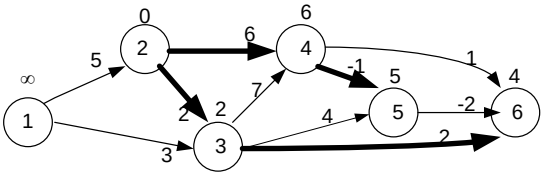



Рис. 3.15. Стан орграфа після четвертої ітерації ($i$ = 4)

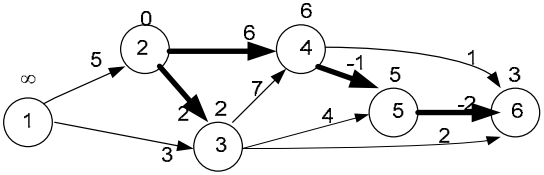



Рис. 3.16. Стан орграфа після п'ятої ітерації ($i$ = 5)

Стан мережі після шостої ітерації ($i$ = 6) не змінюється (рис. 3.16).◅

Для відшукання в ациклічному орграфі *найдовшого шляху* можна скористатися способом *дерелаксації* дуг (див. приклад 3.2). У цьому випадку треба модифікувати процедуру `Init_Source` так, щоб $d[s] = 0$ і $d[v] = -\infty$ для решти вершин $v$.

**Приклад 3.5**. Знайти найдовші шляхи від вершини 2 до інших вершин мережі на рис. 3.11.

➢ Стан мережі після ініціалізації зображено на рис. 3.17.

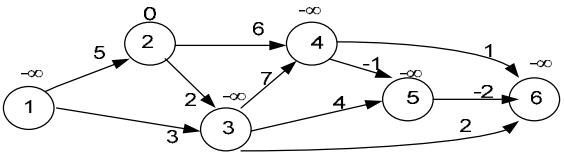



Рис. 3.17. Стан мережі після ініціалізації

Стан мережі після першої ітерації ($i$ = 1) не змінюється (рис. 3.17).

Послідовні стани мережі після наступних ітерацій циклу по $i$ зображено на рис. 3.18 – 3.20. Дуги найдовшого шляху – потовщені.

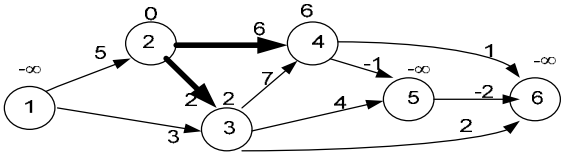



Рис. 3.18. Стан мережі після другої ітерації ($i = 2$)

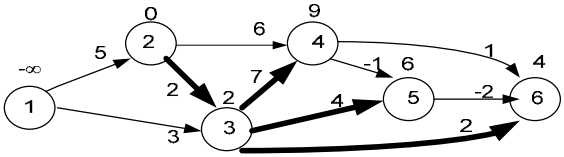



Рис. 3.19. Стан мережі після третьої ітерації ($i = 3$)

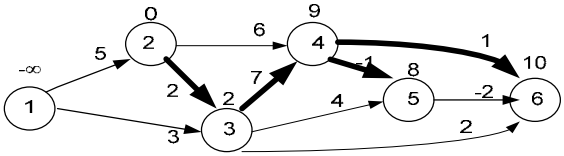



Рис. 3.20. Стан мережі після четвертої ітерації ($i = 4$)

Стан мережі далі не змінюється. Дерево *найдовших шляхів* від вершини 2 до решти вершин, досяжних з неї, зображено на рис. 3.20. ≺

## 3.6. Алгоритм Флойда-Уоршолла

*Алгоритм Флойда-Уоршолла* дає змогу знайти довжини найкоротших шляхів між усіма парами $n$ вершин орграфа. Допускаються дуги з від'ємними довжинами, проте без циклів з сумарною від'ємною довжиною.

Довжини дуг орграфа з $n$ вершинами зображають квадратною матрицею $D$ порядку $n$, у якій елемент $d_{ij}$ є довжиною дуги $(i, j)$. Якщо дуги $(i, j)$ не існує, то $d_{ij} = \infty$. Очевидно, що $d_{ii} = 0$.

Водночас з матрицею $D$ формується і матриця $S$ порядку $n$, у якій $s_{ij} = j$, $s_{ii} = 0$. Матриця $S$ – це своєрідна матриця попередників.

Викладемо базову ідею алгоритму Флойда-Уоршолла. Нехай маємо три вершини $i, j$ та $k$, відстані між якими $d_{ij}$, $d_{ik}$ та $d_{kj}$, відповідно. Якщо виконується нерівність

$$d_{ik} + d_{kj} < d_{ij}, \ (i = \overline{1,n};\ \ j = \overline{1,n};\ \ k = \overline{1,n}), \text{ то}$$

- шлях дугою $(i, j)$ заміняють на шлях двома послідовними дугами $(i, k)$, $(k, j)$;
- $s_{ij} := k$ – ознака наявності шляху $(i, k)$, $(k, j)$.

Запишемо алгоритм Флойда-Уоршолла на псевдокоді:

```
FLOYD_WARSHALL(D, S)
 n := кількість_рядків(D);
 Для (k := 1; k <= n; k++)
  Для (i := 1; i <= n; i++)
   Для (j := 1; j <= n; j++)
    Якщо (dik + dkj < dij)
      { dij := dik + dkj; sij := k; }.
```

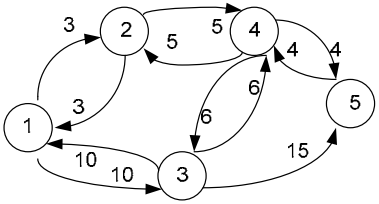



Рис. 3.21. Орграф прикладу 1.15

**Приклад 3.6**. Знайти найкоротші шляхи між усіма парами вершин орграфа (рис. 3.21).

➢ Усі обчислення оформимо як таблиці (елементи матриці $D$, які задовольняють нерівність, затоновано).

*D* *S*

$k = 1$:

| | | | | |
|---|---|---|---|---|
| 0 | 3 | 10 | ∞ | ∞ |
| 3 | 0 | ∞ | 5 | ∞ |
| 10 | ∞ | 0 | 6 | 15 |
| ∞ | 5 | 6 | 0 | 4 |
| ∞ | ∞ | ∞ | 4 | 0 |

| | | | | |
|---|---|---|---|---|
| 0 | 2 | 3 | 4 | 5 |
| 1 | 0 | 3 | 4 | 5 |
| 1 | 2 | 0 | 4 | 5 |
| 1 | 2 | 3 | 0 | 5 |
| 1 | 2 | 3 | 4 | 0 |

*k* = 2:

| | | | | |
|---|---|---|---|---|
| 0 | 3 | 10 | ∞ | ∞ |
| 3 | 0 | 13 | 5 | ∞ |
| 10 | 13 | 0 | 6 | 15 |
| ∞ | 5 | 6 | 0 | 4 |
| ∞ | ∞ | ∞ | 4 | 0 |

| | | | | |
|---|---|---|---|---|
| 0 | 2 | 3 | 4 | 5 |
| 1 | 0 | 1 | 4 | 5 |
| 1 | 1 | 0 | 4 | 5 |
| 1 | 2 | 3 | 0 | 5 |
| 1 | 2 | 3 | 4 | 0 |

*k* = 3:

| | | | | |
|---|---|---|---|---|
| 0 | 3 | 10 | 8 | ∞ |
| 3 | 0 | 13 | 5 | ∞ |
| 10 | 13 | 0 | 6 | 15 |
| 8 | 5 | 6 | 0 | 4 |
| ∞ | ∞ | ∞ | 4 | 0 |

| | | | | |
|---|---|---|---|---|
| 0 | 2 | 3 | 2 | 5 |
| 1 | 0 | 1 | 4 | 5 |
| 1 | 1 | 0 | 4 | 5 |
| 2 | 2 | 3 | 0 | 5 |
| 1 | 2 | 3 | 4 | 0 |

*k* = 4:

| | | | | |
|---|---|---|---|---|
| 0 | 3 | 10 | 8 | 25 |
| 3 | 0 | 13 | 5 | 28 |
| 10 | 13 | 0 | 6 | 15 |
| 8 | 5 | 6 | 0 | 4 |
| ∞ | ∞ | ∞ | 4 | 0 |

| | | | | |
|---|---|---|---|---|
| 0 | 2 | 3 | 2 | 3 |
| 1 | 0 | 1 | 4 | 3 |
| 1 | 1 | 0 | 4 | 5 |
| 2 | 2 | 3 | 0 | 5 |
| 1 | 2 | 3 | 4 | 0 |

*k* = 5:

| | | | | |
|---|---|---|---|---|
| 0 | 3 | 10 | 8 | 12 |
| 3 | 0 | 11 | 5 | 9 |
| 10 | 11 | 0 | 6 | 10 |
| 8 | 5 | 6 | 0 | 4 |
| 12 | 9 | 10 | 4 | 0 |

| | | | | |
|---|---|---|---|---|
| 0 | 2 | 3 | 2 | 4 |
| 1 | 0 | 4 | 4 | 4 |
| 1 | 4 | 0 | 4 | 4 |
| 2 | 2 | 3 | 0 | 5 |
| 4 | 4 | 4 | 4 | 0 |

➢

Найкоротші шляхи між довільною парою вершин орграфа визначають за допомогою матриць *D* і *S*, які отримують після реалізації *n* кроків алгоритму Флойда-Уоршолла. Доведення правиль-

ності цього алгоритму можна знайти у [8]. Найкоротший шлях від вершини *i* до вершини *j* визначають за правилом:

- Відстань $d_{ij}$ отримують з матриці *D*.
- Проміжні вершини шляху від вершини *i* до вершини *j* визначають за матрицею *S*. Нехай $s_{ij} = k$, тоді отримуємо шлях (*i*, *k*), (*k*, *j*). Якщо далі $s_{ik} = k$ і $s_{kj} = j$, то весь шлях знайдено (визначені усі проміжні вершини). Інакше повторюємо процедуру визначення проміжних вершин для шляхів від вершини *i* до вершини *k* і/або від вершини *k* до вершини *j*.

Визначимо, *наприклад*, найкоротший шлях від вершини 1 до вершини 5 ($d_{15} = 12$). Оскільки $s_{15} = 4$, то отримуємо шлях (1, 4), (4, 5). Далі маємо $s_{14} = 2$ і $s_{24} = 4$, отож дугу (1, 4) замінюємо шляхом (1, 2), (2, 4). Оскільки $s_{12} = 2$, то весь шлях знайдено: (1, 2), (2, 4), (4, 5).

### ? Запитання для самоперевірки

1. Сформулюйте означення довжини найкоротшого шляху.
2. Що таке цикл від'ємної ваги?
3. Перелічіть варіанти задач про найкоротші шляхи.
4. Що таке попередник вершини? З якою метою його використовують?
5. Сформулюйте принцип оптимальності для задачі пошуку найкоротших шляхів у навантаженому орграфі.
6. Переконайтеся, що доведення леми 3.2 справедливе і для випадків, коли довжини шляхів дорівнюють $\infty$ чи $-\infty$.
7. Що таке підграфи попередників $G_\pi$?
8. Нехай $G = (V, E)$ – орієнтований граф з ваговою функцією $w: E \to R$, $s \in V$ – початкова вершина. Припустимо, що після виконання процедури `Init_Source`, за якою реалізується довільна послідовність релаксації дуг, виявилося, що $\pi[v] \neq 0$. Довести, що у цьому випадку *G* містить цикл від'ємної ваги.
9. Нехай $G = (V, E)$ – орієнтований граф з ваговою функцією $w: E \to R$, $s \in V$ – початкова вершина. Відомо, що *G* не містить циклів від'ємної ваги, досяжних з *s*. Довести, що після виконання процедури `Init_-`

`Source`, за якою реалізується довільна послідовність релаксації дуг, будь-яка вершина $v \in V$ є досяжною з $s$ у підграфі попередників $G_\pi$.

10. Нехай $G = (V, E)$ – орієнтований граф з ваговою функцією $w: E \to R$, $s \in V$ – початкова вершина. Відомо, що $G$ не містить циклів від'ємної ваги. Довести, що після виконання процедури `Init_Source` можна так реалізувати $|V|-1$ релаксацію дуг, щоб у результаті рівність $d[v] = \delta(s, v)$ виконувалася для усіх $v \in V$.
11. Нехай $G = (V, E)$ – орієнтований граф з ваговою функцією $w: E \to R$, $s \in V$ – початкова вершина. Відомо, що $G$ містить цикл від'ємної ваги, досяжний з $s$. Довести, що існує нескінченна послідовність релаксації дуг, після кожної з яких функція $d[v]$ змінюється.
12. На якій ідеї базується алгоритм Дейкстри?
13. Сформулюйте алгоритм Дейкстри. Для яких навантажених орграфів можна застосовувати алгоритм Дейкстри?
14. Доведіть, що алгоритм Дейкстри працює правильно для навантаженого орграфа, в якому дуги, які виходять з початкової вершини, будуть єдиними дугами з від'ємними вагами.
15. На якій ідеї базується алгоритм Беллмана-Форда? Сформулюйте алгоритм Беллмана-Форда.
16. Для яких навантажених орграфів можна застосовувати алгоритм Беллмана-Форда?
17. Модифікуйте алгоритм Беллмана-Форда так, щоб для вершини $v$, у якої $\delta(s, v) = -\infty$, після виконання алгоритму було встановлено правильне значення $d[v] = \delta(s, v) = -\infty$.
18. Сформулюйте алгоритм пошуку найкоротших шляхів від деякої початкової вершини $s$ до інших вершин у навантаженому ациклічному орграфі.
19. Розробіть алгоритм, який обліковуватиме загальну кількість шляхів в ациклічному орієнтованому орграфі.
20. На якій ідеї базується алгоритм Флойда-Уоршолла? Сформулюйте алгоритм Флойда-Уоршолла.
21. Для яких навантажених орграфів можна застосовувати алгоритм Флойда-Уоршолла?
22. Як можна використати результати роботи алгоритму Флойда-Уоршолла, щоб дізнатися, чи є у графі цикли від'ємної ваги?

## 🕮 Завдання для самостійної роботи

**Завдання 3.1**. На рис.3.22 зображено транспортну мережу, яка з'єднує сім міст, і відстані між ними.

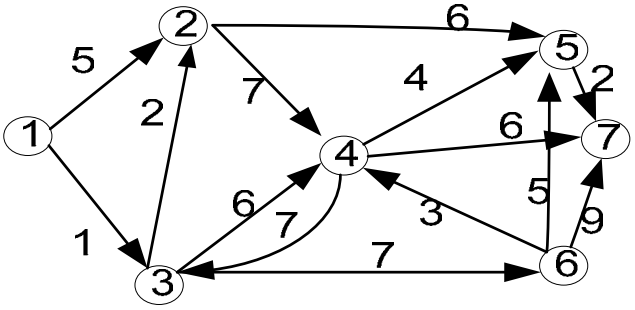



Рис. 3.22. Транспортна мережа

Відшукайте найкоротші відстані між містами: *а*) 4 і 7; *б*) 1 і 7; *в*) 1 і 6.

**Завдання 3.2**. Відшукайте найкоротші відстані від міста 1 до всіх інших міст для орграфа на рис. 3.22.

**Завдання 3.3**. Відшукайте найдовшу відстань між містами 1 і 7 для орграфа на рис. 3.22.

**Завдання 3.4**. В орграфі на рис. 3.22 вилучіть дуги (3, 2), (4, 3), (6, 4) і (6, 5), щоб утворити ациклічний орграф. Використовуючи алгоритм пошуку найкоротших шляхів у навантаженому ациклічному орграфі, відшукайте найкоротшу відстань між містами 1 і 7.

**Завдання 3.5**. Застосуйте алгоритм Флойда-Уоршолла до навантаженого орграфа на рис. 3.23. Відшукайте найкоротші відстані між містами:

*а*) 4 і 1; *б*) 1 і 5; *в*) 1 і 9, *г*) 2 і 5.

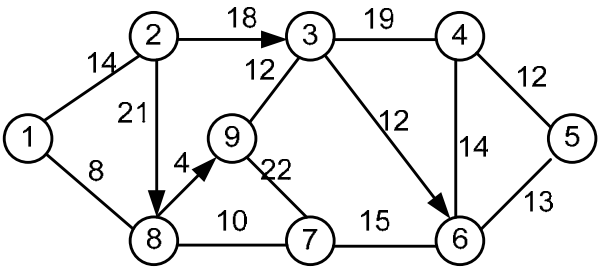



Рис. 3.23. Навантажений орграф для завдання 3.5

# 4. ДИНАМІЧНЕ ПРОГРАМУВАННЯ В ОРГРАФІ

**План викладу матеріалу**:

1. Загальні положення.
2. Задача заміни устаткування.
3. Задача оптимального розподілу коштів.
4. Задача оптимального розподілу обмеженого ресурсу.

**Ключові терміни розділу**

- *Стани системи*
- *Оптимальне керування*
- *Поетапна оптимізація*
- *Множина локальних керувань*
- *Принцип оптимальності Беллмана*
- *Адитивна цільова функція*

## 4.1. Загальні положення

Розглянемо динамічну систему, що з часом змінює свої стани $s_0, s_1, ..., s_n$, тобто в системі відбувається деякий процес, що починається зі стану $s_0$. Хоча б один зі станів $s_1, ..., s_n$ є *кінцевим* (позначимо його $s_t$) – при переході системи у такий стан процес, що в ній відбувається, завершується. Якщо кожен стан $s_j$ ( $j = \overline{1, n}$ ) описується набором значень $r$ параметрів $p_1, p_2, ..., p_r$, то систему називають *r-параметричною*. Початковий стан $s_0$ є особливим станом, отож його опис, зазвичай, відрізняється від решти станів.

Будь-який стан $s_j$ ( $j = \overline{1, n}$ ) зв'язаний з *множиною локальних керувань* $\varphi(s_j)$, кожне з яких здатне переводити систему зі стану $s_j$ у деякий інший стан $s_k$, причому обов'язково $k > j$ (розглядаються системи без зворотних зв'язків). Керування з $\varphi(s_j)$, що переводить систему зі стану $s_j$ у стан $s_k$, позначимо $u_{jk}$. Кожному керуванню $u_{jk}$ відповідають прибутки/витрати $a_{jk}$, зв'язані з його реалізацією.

Задача пошуку оптимального керування зазначеної системи полягає в тому, щоб знайти таку послідовність локальних керувань з початкового стану $s_0$ до одного з кінцевих станів $s_t$, за якої сумарна величина прибутків/витрат була б максимальною/мінімальною (таку послідовність локальних керувань називають *оптимальним керуванням* зі стану $s_0$).

Сформулюємо задачу пошуку оптимального керування динамічною системою в термінах графів. Нехай $G = (V, E)$ – орієнтова-

ний орграф з множиною вершин $V = \{s_0, s_1, ..., s_n\}$, в якому пара $\{s_j, s_k\}$ є дугою, якщо в $\varphi(s_j)$ існує керування $u_{jk}$ ($k > j$). Вага дуги $(s_j, s_k)$ дорівнює $a_{jk}$ (зазвичай, $a_{jk}>0$). Очевидно, що $G$ – ациклічний орграф.

Задача пошуку *оптимального керування* еквівалентна задачі пошуку в ациклічному орграфі $G$ шляху максимальної/мінімальної довжини, що з'єднує вершину $s_0$ з однією із кінцевих вершин $s_t$. Цей шлях назвемо *оптимальним шляхом* з вершини $s_0$.

Процедура розв'язування задачі оптимального керування методом динамічного програмування базується на принципі оптимальності Беллмана:

> розглядаючи стан $s_j$, необхідно обрати таке локальне керування $u_{jk} \in \varphi(s_j)$, що у сукупності з оптимальним керуванням зі стану $s_k$ утворить *оптимальне керування* зі стану $s_j$.

Іншими словами:

> яким би не був стан динамічної системи в результаті реалізації будь-якого числа кроків, керування на найближчому кроці необхідно обирати таким, щоб воно разом з оптимальним керуванням на всіх наступних кроках привело до максимально можливого виграшу на всіх кроках, що залишилися, включаючи і даний.

Будь-яку багатокрокову задачу ухвалення рішення можна інтерпретувати як процес зміни *стану динамічної системи* з дискретним часом. Поняття *стану* відіграє важливу роль у дискретному динамічному програмуванні і може мати різноманітну інтерпретацію.

Суть методу динамічного програмування щодо багатокрокових задач ухвалення рішень з *адитивною цільовою функцією* полягає в *поетапній оптимізації* (оптимальне рішення ухвалюється на кожному кроці). Прийняття рішення на кожному кроці, за винятком останнього, здійснюється з урахуванням усіх його можливих наслідків у майбутньому (на майбутніх кроках).

Прийняття рішення на *останньому* кроці має особливе значення, адже вибір рішення можна здійснювати "незважаючи на майбутнє". Отож у методі динамічного програмування спочатку

планують останній крок, виходячи з максимально можливої ефективності рішення.

Розглянемо застосування методу динамічного програмування в орграфі для розв'язування класичних задач *заміни устаткування*, *розподілу коштів* між підприємствами, *розподілу обмеженого ресурсу* (завантаження транспортного засобу чи задача про рюкзак).

## 4.2. Задача заміни устаткування

*Задача заміни устаткування* полягає у визначенні оптимальних термінів заміни старого устаткування новим. Унаслідок старіння устаткування зростають виробничі витрати, витрати на ремонт і обслуговування, знижуються продуктивність і ліквідна вартість. Критерієм оптимальності слугують сумарні витрати на обслуговування устаткування протягом планового періоду. Розглянемо цю задачу в спрощеній постановці. Дано:

- первісна вартість устаткування *cost*;
- ліквідна вартість устаткування *likv*(*j*), вік якого *j* років;
- затрати *zatr*(*j*) на обслуговування протягом року устаткування, що до початку річного періоду вже експлуатувалося *j* років.

Наприкінці експлуатаційного періоду наявне устаткування необхідно продати. Визначити оптимальну стратегію заміни устаткування за період *n* років, яка мінімізує сумарні витрати на обслуговування устаткування.

➢ Поточний стан цієї системи визначається двома параметрами *k* та *j*, де *k* – кількість років, які минули від початку першого придбання устаткування, *j* – вік устаткування, що експлуатується на даний момент. Стан системи зручно позначати через $s_{kj}$. *Множина локальних керувань* будь-якого стану складається з керувань:

- "*заміна*" – продаж старого устаткування і купівля нового;
- "*збереження*" – продовження експлуатації устаткування протягом наступного року.

Отже, "*заміна*" переводить систему зі стану $s_{kj}$.у стан $s_{k+1,1}$, а "*збереження*" – зі стану $s_{kj}$ у стан $s_{k+1,\,j+1}$.

Витрати на обслуговування устаткування після вибору керування у стані $s_{kj}$ визначаються такою формулою:

$$a_{kj} = \begin{cases} cost + zatr(0) - likv(j), \text{якщо "заміна";} \\ zatr(j), \quad \text{якщо "збереження".} \end{cases} \tag{4.1}$$

Додамо фіктивний стан $s_t$, у який переходить система після закінчення експлуатаційного періоду в результаті продажу наявного устаткування (перехід у стан $s_t$ може відбутися зі станів $s_{n1}$, $s_{n2}$, …, $s_{nn}$). Розглянемо задачу за таких даних: $n$ = 4, $cost$ = 2000,

| $j$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $likv(j)$ | – | 900 | 750 | 500 | 200 |
| $zatr(j)$ | 100 | 800 | 1200 | 1600 | – |

Орграф задачі заміни устаткування зображено на рис. 4.1. У вершинах орграфа вказані поруч індекси $k$ та $j$ стану $s_{kj}$ ($t$ – індекс кінцевого стану). Орграф є ациклічним.

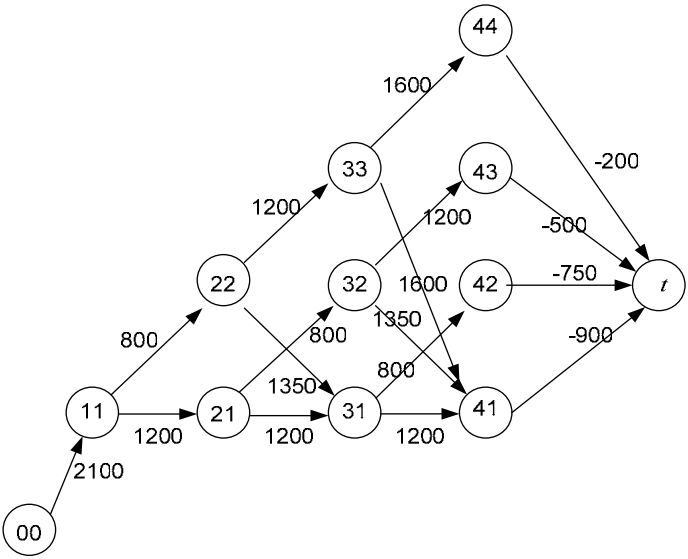



Рис. 4.1. Орграф задачі заміни устаткування

Дуги, що з'єднують стани $s_{kj}$ та $s_{k+1,\ j+1}$, відповідають керуванню "*збереження*". Дуги, що з'єднують стани $s_{kj}$ та $s_{k+1,1}$, відповідають керуванню "*заміна*". Ваги дуг відповідних керувань обчислюють згідно з (4.1) і є витратами на обслуговування устаткування після вибору керування у стані $s_{kj}$ (ваги зазначені поблизу дуг).

Продаж устаткування на останньому році експлуатації ($k = 4$) означає чистий прибуток, отож у задачі мінімізації витрат цей прибуток треба зображати від'ємним числом.

Шлях мінімальної довжини, що з'єднує вершини $s_{00}$ і $s_t$, визначатиме оптимальну стратегію заміни устаткування даної системи. Для визначення цього шляху використаємо *принцип динамічного програмування*, яке на мові теорії графів означатиме використання алгоритму визначення найкоротшого шляху в ациклічному орграфі при переміщенні від *витоку* ($s_t$) до *джерела* ($s_{00}$). Для реалізації такого варіанта алгоритму необхідно поміняти напрям усіх дуг орграфа (умовно).

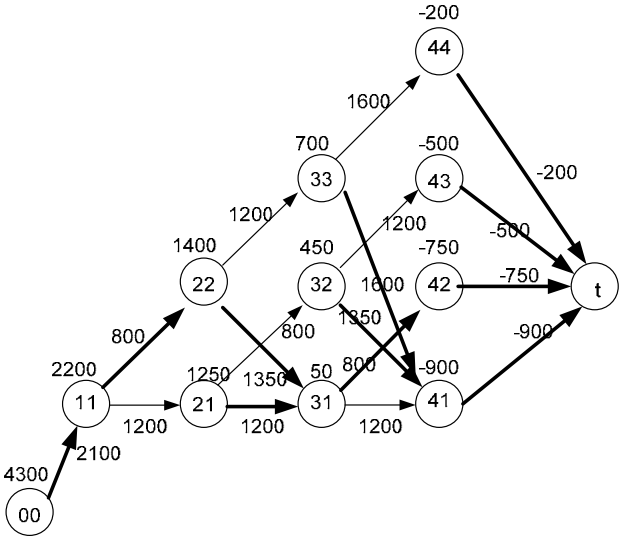



Рис. 4.2. Розв'язок задачі заміни устаткування

Отже, оптимальний шлях простягається через вершини $s_{00}$, $s_{11}$, $s_{22}$, $s_{31}$, $s_{42}$ і визначає стратегію заміни обладнання на чотири ро-

ки: “*купівля*”, “*збереження*” (після першого року), “*заміна*” (після другого року), “*збереження*” (після третього року), “*продаж*” (після четвертого року, згідно з вимогами). Мінімальні витрати на обслуговування устаткування протягом чотирьох років становитимуть 4300 одиниць грошей. ◅

### 4.3. Задача оптимального розподілу коштів

Групі з $m$ підприємств виділено додаткові кошти на реконструкцію і модернізацію виробництва. Відома матриця $A$, у якій на позиції $(i, j)$ розташована величина $a_{ij} \geq 0$, яка дорівнює приросту випуску продукції на $i$–му ($i = \overline{1, m}$) підприємстві за умови виділення йому додаткових коштів у розмірі $j$ ($j = \overline{0, n}$) грошових одиниць. Необхідно так розподілити між підприємствами загальну суму коштів у $n$ грошових одиниць, щоб сумарний приріст випуску продукції був максимальним.

➢ Поточний стан такої системи визначається двома параметрами $i$ та $j$, які означають, що $i$ підприємствам уже надано суму $j$ грошових одиниць. Стан системи зручно позначати через $s_{ij}$. Множина $\varphi(s_{ij})$ складається з локальних керувань “виділити $k$ грошових одиниць $(i + 1)$-му підприємству”, $k = 0,1, \ldots, n–j$. За такого локального керування система зі стану $s_{ij}$ переходить у стан $s_{i+1,\, j+k}$.

Розглянемо цю задачу за таких вихідних даних:

$$A = \begin{pmatrix} 0 & 10 & 18 \\ 2 & 12 & 35 \\ 3 & 14 & 29 \end{pmatrix}, \; m = 3, \; n = 2.$$

Орграф задачі зображено на рис. 4.3. У вершинах орграфа вказано поруч індекси $i$ та $j$ стану $s_{ij}$. Орграф є ациклічним. Прибуток від отриманих коштів зображають на відповідних дугах. Шлях максимальної довжини, що з’єднує вершини $s_t$ і $s_{00}$, визначатиме оптимальну стратегію розподілу коштів між підприємствами. Довжину *найдовшого шляху* з відповідної вершини до витоку зображено над відповідною вершиною. Дуги, які демонструють найдовший шлях, зображено потовщено.

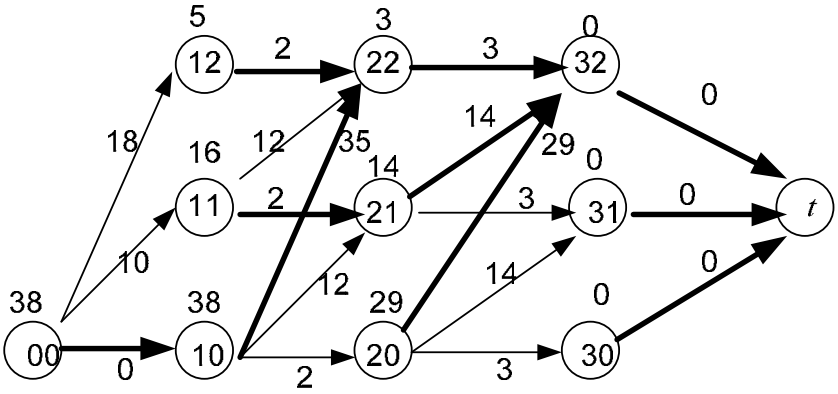



Рис. 4.3. Орграф задачі розподілу коштів

Отже, оптимальний шлях простягається через вершини $s_{00}$, $s_{10}$, $s_{22}$, $s_{32}$ і визначає оптимальну стратегію "надати усі кошти (2 гр. од.) 2-му підприємству". Максимальний приріст випуску продукції у цьому випадку становитиме 38 одиниць. ⋖

### 4.4. Задача оптимального розподілу обмеженого ресурсу

Підприємство для виробництва продуктів $m$ найменувань використовує деякий ресурс, запаси якого обмежені величиною $n$ (відоме натуральне число). Для виробництва одиниці продукту $i$-го найменування необхідно витратити $q_i$ одиниць ресурсу (відоме натуральне число, $q_i \le n$), а прибуток від цього становить $c_i$ умовних грошових одиниць. Визначення кількості виробництва $x_i$ ($i = \overline{1, m}$) продуктів $i$-го найменування, які забезпечують максимальний сумарний дохід, приводить до такої задачі:

$$\begin{cases} \sum\limits_{i=1}^{m} c_i x_i \to \max; \\ \sum\limits_{i=1}^{m} q_i x_i \le n, \quad x_i \in N \cup \{0\}, \quad i = \overline{1, m}. \end{cases} \tag{4.2}$$

➢ Поставлена задача є статичною задачею ухвалення рішень. Однак, якщо з певних причин виробництво різних продуктів необхідно здійснювати послідовно (спочатку 1-й продукт, потім 2-й і т.д.), то отримуємо багатокрокову задачу ухвалення рішень (4.2), яку і розглядатимемо далі.

Нехай $y_i$ – *сумарна* кількість використаного ресурсу при виробництві продуктів з найменуваннями від $i$ до $m$, а $f_i(y_i)$ – *максимальний прибуток*, який при цьому можна отримати. Згідно з принципом динамічного програмування отримуємо (квадратні дужки […] позначають цілу частину частки):

$$f_i(y_i) = \max_{x_i=0,1,...,\left[ n/q_i \right]} \{c_i x_i + f_{i+1}(y_{i+1})\},\ i=\overline{1,m},\ \text{де } f_{m+1}(y_{m+1})=0. \quad (4.3)$$

Задача розподілу обмеженого ресурсу еквівалентна класичній задачі про рюкзак (або задачі про завантаження):

- туристу треба визначити найнеобхідніші (найцінніші речі), які доведеться розмістити у наплічнику обмеженого об'єму;
- завантажується транспортний засіб (автомобіль, судно, літак тощо), що має обмеження на об'єм чи вантажопідйомність.

Нехай автомобіль вантажопідйомністю $n$ завантажується предметами $m$ найменувань. Для $i$-го найменування ($i=\overline{1,m}$) відомі величини: $c_i$ – прибуток від завантаження одного предмета і $q_i$ – вага одного предмета. Знайти кількості $x_i$ ($i=\overline{1,m}$) завантаження предметів $i$-го найменування, які забезпечать максимальний сумарний дохід. Відповідно до зроблених припущень отримуємо рівняння (4.3).

➢ Поточний стан зазначеної системи визначають два параметри $i$ та $j$, які означають, що предмети з $i$ найменувань ($i=\overline{1,m}$) уже завантажені з сумарною вагою $j$ одиниць. Стан системи позначатимемо через $s_{ij}$. Множина $\varphi(s_{ij})$ складається з локальних керувань "завантажити $x_{i+1}$ предметів ($i$ + 1)-го найменування", $x_{i+1}=0,1,...,\left[ n/q_{i+1} \right]$.

За такого локального керування система зі стану $s_{ij}$ переходить у стан $s_{i+1,\, j+x_{i+1}\cdot q_{i+1}}$.

Розглянемо цю задачу за таких вихідних даних ($n$ =4):

| Предмет ($i$) | 1 | 2 | 3 |
|---|---|---|---|
| Вага одного предмета ($q_i$) | 2 | 3 | 1 |
| Прибуток від завантаження одного предмета ($c_i$) | 28 | 45 | 12 |

Орграф задачі зображено на рис. 4.4. У вершинах орграфа позначено поруч індекси $i$ та $j$ стану $s_{ij}$. Орграф є ациклічним (за властивостями динамічної системи). Максимальне значення кількості завантаження першого предмета дорівнює [4/2] = 2 (тобто $x_1 = 0,\ 1,\ 2$). Аналогічно $x_2 = 0,\ 1$; $x_3 = 0,\ 1,\ 2,\ 3,\ 4$. Прибуток від завантаження відповідної кількості предмета вказують на дугах.

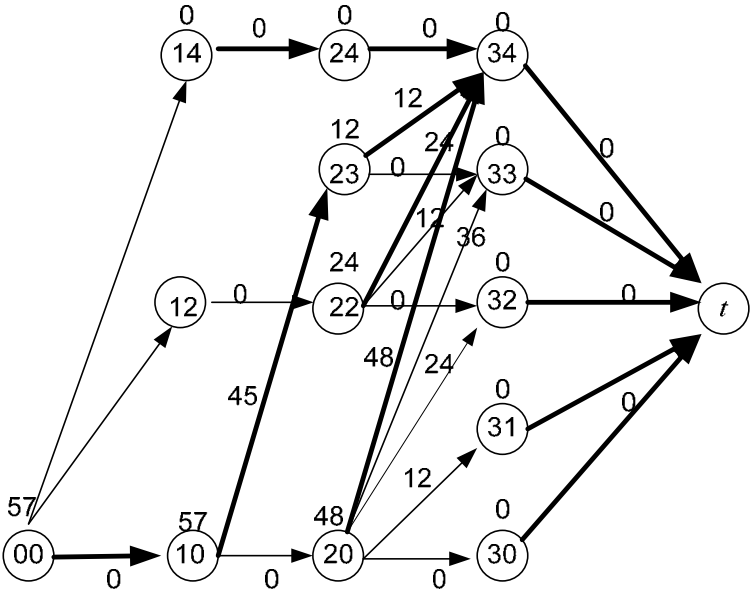



Рис. 4.4. Орграф задачі завантаження автомобіля

Шлях максимальної довжини, що з'єднує вершини $s_t$ і $s_{00}$ визначатиме оптимальну стратегію завантаження автомобіля. Довжину *найдовшого шляху* від витоку до певної вершини зображають над цією вершиною. Дуги, які складають найдовший шлях, зображені потовщеною лінією.

Оптимальний шлях перетинає вершини $s_{00}$, $s_{10}$, $s_{23}$, $s_{34}$ і визначає оптимальну стратегію "завантажити по одному предмету 2-го і 3-го найменування". Прибуток становитиме 57 гр. од. ◅

Зі збільшенням величини $n$ у задачах динамічного програмування орграф стає перевантаженим різними позначеннями, отож його візуальний розв'язок є проблематичним. Реалізувати ці задачі без використання комп'ютера (за допомогою обчислень у таблиці) можна завдяки рекурентним рівнянням. Розглянемо рівняння (4.3). Виразимо $y_{i+1}$ як функцію від $y_i$ для гарантії того, що права частина (4.3) є функцією лише від $y_i$.

Оскільки $y_{i+1}$ – *сумарна* кількість використаного ресурсу при виробництві продуктів з найменуваннями від $i$+1 до $m$, то $y_i - y_{i+1}$ – є кількістю використаного ресурсу при виробництві продукту з найменуванням $i$, тобто $y_i - y_{i+1} = q_i x_i$ (або $y_{i+1} = y_i - q_i x_i$ ). Тоді

$$f_i(y_i) = \max_{\substack{x_i=0,1,\ldots,\left[b/q_i\right] \\ y_i=0,1,\ldots,n}} \{c_i x_i + f_{i+1}(y_i - q_i x_i)\},\ i=\overline{1,m};\ f_{m+1}(x_{m+1})=0. \quad (4.4)$$

➢ Розв'яжемо задачу про оптимальне завантаження автомобіля за допомогою табличних обчислень (табл. 4.1 – 4.3) згідно з формулою (4.4).

*Таблиця 4.1.* Третій етап обчислень

| $y_3$ | $12 \cdot x_3$ | | | | | Оптимальний розв'язок | |
|---|---|---|---|---|---|---|---|
| | $x_3 = 0$ | $x_3 = 1$ | $x_3 = 2$ | $x_3 = 3$ | $x_3 = 4$ | $f_3(x_3)$ | $x_3^*$ |
| 0 | 0 | – | – | – | – | 0 | 0 |
| 1 | 0 | 12 | – | – | – | 12 | 1 |
| 2 | 0 | 12 | 24 | – | – | 24 | 2 |
| 3 | 0 | 12 | 24 | 36 | – | 36 | 3 |
| 4 | 0 | 12 | 24 | 36 | 48 | 48 | 4 |

*Таблиця 4.2*. Другий етап обчислень

| $y_2$ | $45 \cdot x_2 + f_3(y_2 - 3 \cdot x_2)$ | | Оптимальний розв'язок | |
|---|---|---|---|---|
| | $x_2 = 0$ | $x_2 = 1$ | $f_2(x_2)$ | $x_2^*$ |
| 0 | 0 + 0 = 0 | – | 0 | 0 |
| 1 | 0 + 12 = 12 | – | 12 | 0 |
| 2 | 0 + 24 = 24 | – | 24 | 0 |
| 3 | 0 + 36 = 36 | 45 + 0 = 45 | 45 | 1 |
| 4 | 0 + 48 = 48 | 45 + 12 = 57 | 57 | 1 |

*Таблиця 4.3*. Перший етап обчислень

| $y_1$ | $28 \cdot x_1 + f_2(y_1 - 2 \cdot x_1)$ | | | Оптимальний розв'язок | |
|---|---|---|---|---|---|
| | $x_1 = 0$ | $x_1 = 1$ | $x_1 = 2$ | $f_1(x_1)$ | $x_1^*$ |
| 0 | 0 + 0 = 0 | – | – | 0 | 0 |
| 1 | 0 + 12 = 12 | – | – | 12 | 0 |
| 2 | 0 + 24 = 24 | 28 + 0 = 28 | – | 28 | 1 |
| 3 | 0 + 45 = 45 | 28 + 12 = 40 | – | 45 | 0 |
| 4 | 0 + 57 = 57 | 28 + 24 = 52 | 56 + 0 = 56 | 57 | 0 |

*Вибір оптимального розв'язку*. На першому етапі $x_1^* = 0$, $y_1 = 4$.

З формули $y_2 = y_1 - 2 \cdot x_1$ отримуємо $y_2 = 4$. На другому етапі $x_2^* = 1$.

З формули $y_3 = y_2 - 3 \cdot x_2$ отримуємо $y_3 = 1$. На третьому етапі $x_3^* = 1$.

У таблиці для першого етапу необхідно отримати розв'язок лише для $y_1 = 4$. Однак таблиця налічує обчислення і для $y_1 = 0, 1. 2, 3$, які дають змогу проаналізувати чутливості розв'язків. Наприклад, якщо вантажопідйомність $n = 3$, то оптимальний розв'язок можна знайти, починаючи з $y_1 = 3$. ◅

### ? Запитання для самоперевірки

1. Сформулюйте задачу пошуку оптимального керування у динамічній системі.
2. Сформулюйте задачу заміни устаткування. Опишіть можливі стани та локальні керування відповідної динамічної системи.
3. Сформулюйте задачу розподілу коштів між підприємствами. Опишіть можливі стани та локальні керування відповідної динамічної системи.

4. Сформулюйте задачу розподілу обмеженого ресурсу. Опишіть можливі стани та локальні керування відповідної динамічної системи.
5. Виведіть рекурентні рівняння у задачі розподілу обмеженого ресурсу.

### 🕮 Завдання для самостійної роботи

**Завдання 4.1**. Групі із 4-х підприємств надають додаткові кошти. Дано матрицю $A=\begin{pmatrix} 0 & 0 & 0 & 0 \\ 10 & 12 & 11 & 16 \\ 31 & 26 & 36 & 37 \\ 42 & 36 & 45 & 46 \\ 62 & 54 & 60 & 63 \\ 76 & 78 & 77 & 80 \end{pmatrix}$, в якій на позиції $(i,k)$, $i=\overline{0,5}$, $k=\overline{1,4}$, знаходиться величина, яка дорівнює приросту випуску одиниць продукції на *k*-му підприємстві при виділенні йому додаткових *i* одиниць грошей. Розділити 5 одиниць грошей так, щоб сумарний приріст продукції був максимальним.

**Завдання 4.2**. Дано: первісна вартість устаткування *cost*; ліквідна вартість устаткування *likv*(*i*), вік якого *i* років; вартість затрат *zatr*(*i*) на обслуговування протягом одного року устаткування, щодо початку цього річного періоду вже експлуатувалося рівно *i* років. Наприкінці експлуатаційного періоду наявне устаткування необхідно продати. Визначити оптимальну стратегію заміни устаткування за період *n* років, яка мінімізує сумарні витрати на обслуговування устаткування. Варіанти завдань:

*1*. *n*=5, *cost*=5400;

| *i* | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| *likv(i)* | — | 3500 | 3000 | 2400 | 2000 | 1000 |
| *zatr(i)* | 500 | 2000 | 2100 | 2400 | 2600 | — |

*2*. *n*=5, *cost*=1100;

| *i* | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| *likv(i)* | — | 1000 | 900 | 450 | 400 | 100 |
| *zatr(i)* | 800 | 1700 | 1900 | 2900 | 4200 | — |

*3*. *n*=5, *cost*=3200;

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| *likv(i)* | — | 1400 | 700 | 500 | 400 | 200 |
| *zatr(i)* | 200 | 700 | 1500 | 3200 | 3700 | — |

*4*. *n*=5, *cost*=3000;

| *i* | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| *likv(i)* | — | 2000 | 1300 | 900 | 500 | 200 |
| *zatr(i)* | 400 | 800 | 1300 | 1400 | 2100 | — |

# 5. МАКСИМАЛЬНИЙ ПОТІК

**План викладу матеріалу**:

1. Базові поняття теорії потоків.
2. Теорема Форда-Фалкерсона.
3. Збільшувальні шляхи.
4. Метод Форда-Фалкерсона пошуку максимального потоку.
5. Залишкові мережі.
6. Зв'язок задачі максимізації потоку з іншими задачами ДО.

**Ключові терміни розділу**

- *Пропускна здатність дуг*
- *Визначення st-мережі*
- *Збільшувальний шлях*
- *Сумарна величина потоку*
- *Переріз*
- *Залишкова мережа*

## 5.1. Базові поняття теорії потоків

Графи та орграфи – математичні абстракції, які мають важливе значення, оскільки дають змогу розв'язувати велику кількість практичних задач. У цьому параграфі розширимо графові моделі розв'язування задач з таким розрахунком, щоб охопити *динамічні ситуації*, які ототожнюються з переміщенням матеріалів чи товарів різними маршрутами, протіканням речовин трубами (вода, нафта, газ тощо), передачею даних в інформаційних мережах тощо.

Розглянемо таку ідеалізовану фізичну систему. У деяких пунктах (*джерелах*) виробляють (чи видобувають) однорідну речовину, яку через систему взаємозв'язаних труб різного діаметра та різної довжини доставляють до інших пунктів (*стоків*), де її використовують. Отож трубами протікає *потік речовини* (або просто *потік*).

Напрям протікання потоку у кожній трубі – строго зафіксований і не може змінюватися. Для кожної труби відома величина *пропускної здатності*, яка визначає *максимальну* кількість одиниць потоку (тонни, літри, барелі тощо), які можна перемістити цією трубою за одиницю часу.

На початку і/або наприкінці кожної труби встановлено вентиль, який керує потоком. Це керування полягає у визначенні *величини потоку* даною трубою (тобто фактичної кількості одиниць потоку, які переміщуються цією трубою за одиницю часу).

Величину потоку і пропускну здатність труб вимірюють в одних і тих самих одиницях (наприклад, тоннах за годину, літрах за секунду, штуках за добу тощо). Очевидно, що величина потоку довільної труби не може перевищувати її пропускної здатності.

У проміжних пунктах, де труби перетинаються, відсутні резервуари для зберігання речовини, отож тут потоки перебувають у рівновазі: за одиницю часу стільки одиниць потоку з пункту вибуває, скільки до нього за цей же час прибуває (закон *збереження потоку*).

У загальному випадку трубами можна також перекачувати речовину від проміжних пунктів до джерел, від стоків до проміжних пунктів. Трубами можуть бути з'єднані джерела, отож між ними можливий обмін речовиною (те ж стосується стоків).

Якщо сумарна пропускна здатність вхідних труб *кожного* проміжного пункту дорівнює сумарній пропускній здатності вихідних труб цього пункту, то ми цілковито заповнюємо речовиною усі труби (задача оптимізації відсутня).

У протилежному випадку цілковито заповненими будуть уже не всі труби, однак речовина тече трубами, керована настроюванням вентилів так, що забезпечується закон *збереження потоку*.

З рівноваги потоку на перетинах труб інтуїтивно слідує необхідність рівноваги потоку загалом (скільки одиниць потоку з усіх джерел за одиницю часу витече, стільки усі стоки за цей же час мають використати). Вважають, що джерела/стоки можуть надіслати/використати довільну кількість речовини.

Далі буде строго доведено, що сумарна величина потоку усіма трубами, які забезпечують витікання речовини з усіх джерел, дорівнюватиме сумарній величині потоку усіма трубами, які забезпечують надходження речовини в усі стоки.

Згідно із зазначеними фактами ми зацікавлені в одержанні відповіді на таке питання: які настроювання вентилів забезпечать *максимальну величину потоку* з усіх джерел в усі стоки?

Ми можемо прямо моделювати цю ситуацію за допомогою навантаженого орграфа $G = (V, E)$. Вершини орграфа $v_1$, $v_2$, …, $v_n$ відповідають джерелам, стокам і проміжним пунктам. Дуги орграфа відповідають трубам (речовина може переміщатися тільки в од-

ному напрямі у кожній трубі). Вага дуги відповідає пропускній здатності відповідної труби.

Для спрощення дугу $(v_k, v_i)$ позначимо $(k, i)$. Вага дуги $(k, i)$ визначає її *пропускну здатність* $b_{ki}$ – максимальну кількість одиниць потоку, які можна перемістити цією дугою за одиницю часу ($b_{ki} \geq 0$).

Кількість одиниць потоку, який *фактично* переміщують дугою $(k, i)$ за одиницю часу, називають *величиною потоку* $x_{ki}$ цієї дуги. Очевидно, що $0 \leq x_{ki} \leq b_{ki}$. Якщо $x_{ki} = b_{ki}$, то дуга $(k, i)$ *–насичена* потоком.

Для кожної вершини $v \in V$ можна визначити *інтенсивність* потоку (або дивергенцію потоку – *з фізики*)

$$d(v) = \sum_{i:\,(v,i)\in E} x_{vi} - \sum_{i:\,(i,v)\in E} x_{iv}, \tag{5.1}$$

яка дорівнює різниці між сумарною величиною потоків усіма дугами, які виходять з вершини $v$, і сумарною величиною потоків усіма дугами, які закінчуються у вершині $v$.

Величина $d(v)$ може бути *додатною* (з вершини $v$ більше одиниць потоку вибуває, ніж прибуває у неї), *від'ємною* (у вершину $v$ більше одиниць потоку прибуває, ніж вибуває з неї, тобто відбувається *накопичення* потоку у вершині $v$) і *рівною нулю* (скільки одиниць потоку прибуває у цю вершину, стільки з неї і вибуває).

Вершину з додатною інтенсивністю називають *вершиною-джерелом* (або просто *джерелом*) і позначають літерою $s$. Вершину з від'ємною інтенсивністю називають *вершиною-стоком* (або просто *стоком*) і позначають літерою $t$. У загальному випадку орграф може мати *декілька* джерел і/або стоків. Вершину з нульовою інтенсивністю називають *проміжною*.

Спростимо спочатку нашу модель. Вважатимемо, що $d(v_1) > 0$, $d(v_n) < 0$, $d(v_l) = 0 \left(l = \overline{2, n-1}\right)$. Тоді в орграфі: $v_1 = s$ – єдине джерело, $v_n = t$ – єдиний стік, а $v_2, v_3, \ldots, v_{n-1}$ – проміжні вершини.

Навантажений орграф з додатними вагами, що має єдине джерело $s$ і єдиний стік $t$, називають *st-мережею*. Дугу $(k, i)$ цієї ме-

режі у напрямі від $s$ до $t$ називають *прямою* дугою; а у напрямі від $t$ до $s$ – *зворотною* дугою. Величину

$$F = d(s) = \sum_{i:\,(s,i)\in E} x_{si} - \sum_{i:\,(i,s)\in E} x_{is} \qquad (5.2)$$

називають *сумарною величиною потоку* в *st*-мережі. Задача *максимізації потоку* в *st*-мережі полягає у визначенні *максимуму* величини $F$.

Головна причина розгляду *st*-мережі полягає у тому, що вона дає змогу розв'язати безліч інших задач методом зведення.

### 5.2. Теорема Форда-Фалкерсона

Розіб'ємо множину вершин $V$ на дві підмножини $S$ і $T$, які не перетинаються ($S \subset V$, $T \subset V$, $S \cup T = V$, $S \cap T = \varnothing$), так, щоб $s \in S$ і $t \in T$. Позначимо через $(S, T)$ – усі дуги, що з'єднують $S$ і $T$, а через $(T, S)$ – усі дуги, що з'єднують $T$ і $S$.

Множину дуг $(S,T) \cup (T,S)$ називають *перерізом*, індукованим множиною S. Переріз визначає множину дуг, за умови видалення яких з орграфа *цілковито* припиняється потік від джерела до стоку та навпаки. *Пропускною здатністю* такого перерізу називають величину

$$b(S,T) = \sum_{(k,i)\in(S,T)} b_{ki}. \qquad (5.3)$$

Згідно з (5.2) величина $F$ вимірюється у джерелі $s$. За теоремою 5.1 величину $F$ можна виміряти у довільному перерізі.

**Теорема 5.1**. Для довільної підмножини вершин $S \subset V$, яка містить джерело $s$ і не містить стоку $t$, справджується співвідношення

$$F = \sum_{(k,i)\in(S,T)} x_{ki} - \sum_{(i,k)\in(T,S)} x_{ik}. \qquad (5.4)$$

➢ Для будь-якої проміжної вершини $v \in S$, згідно з (5.1), справедливий вираз:

$$0 = \sum_{i:\,(v,i)\in E} x_{vi} - \sum_{i:\,(i,v)\in E} x_{iv}. \qquad (5.5)$$

Додаючи почленно рівність (5.2) і рівності (5.5) для *усіх* проміжних вершин $v \in S$, отримуємо співвідношення (5.4). Дійсно, для довільної дуги $(r, z) \in E$ можливі такі варіанти:

– $r \in S,\ z \in S$ – величина $x_{rz}$ у сумі $\sum\limits_{i:\,(r,i)\in E} x_{ri}$ буде доданком зі знаком "+", а у сумі $\sum\limits_{i:\,(i,z)\in E} x_{iz}$ – доданком зі знаком "–" (тобто внаслідок сумування величина $x_{rz}$ знищується);

– $r \in S,\ z \in T$ (дуга належить множині $(S, T)$) – величина $x_{rz}$ буде доданком тільки у сумі $\sum\limits_{(k,i)\in(S,T)} x_{ki}$;

– $r \in T,\ z \in S$ (дуга належить множині $(T, S)$) – величина $x_{rz}$ буде доданком тільки у сумі $\sum\limits_{(i,k)\in(T,S)} x_{ik}$.

Усі розглянуті випадки узгоджуються з рівністю (5.4). ◅

**Наслідок з теореми 5.1**. $F = -d(t) = -\left( \sum\limits_{k:\,(t,k)\in E} x_{tk} - \sum\limits_{k:\,(k,t)\in E} x_{kt} \right)$.

➢ Нехай $T = \{t\}$, тоді $S = V/\{t\}$. Згідно з (5.4) маємо таке:

$$F = \sum_{(k,i)\in(S,T)} x_{ki} - \sum_{(i,k)\in(T,S)} x_{ik} = \sum_{k:\,(k,t)\in E} x_{kt} - \sum_{k:\,(t,k)\in E} x_{tk} =$$

$$= -\left( \sum_{k:\,(t,k)\in E} x_{tk} - \sum_{k:\,(k,t)\in E} x_{kt} \right). \quad ◅$$

Наслідок 5.1 відображає інтуїтивно зрозумілий факт: стікає така кількість одиниць потоку, яка виходить з джерела.

Одним з фундаментальних фактів теорії потоків в орграфах є класична теорема *про максимальний потік і мінімальний переріз* у *st*-мережі (або теорема *Форда-Фалкерсона*). *Мінімальним перерізом* серед усіх перерізів $(S,T) \cup (T,S)$, індукованих множиною $S$, є переріз з найменшою пропускною здатністю $b^* = \min\limits_{(S,T)} \big(b(S,T)\big)$.

**Теорема 5.2**. *Теорема Форда-Фалкерсона*. Сумарна величина $F$ довільного потоку від $s$ до $t$ не перевищує пропускної здатності мінімального перерізу ($F \le b^*$), причому існує потік, для якого $F = b^*$.

➢ Згідно з (5.3) і (5.4) маємо таке:

$$F = \sum_{(k,i)\in(S,T)} x_{ki} - \sum_{(i,k)\in(T,S)} x_{ik} \le \sum_{(k,i)\in(S,T)} x_{ki} \le \sum_{(k,i)\in(S,T)} b_{ki} = b(S,T).$$

Отже, сумарна величина $F$ довільного потоку від $s$ до $t$ не перевищує пропускної здатності *довільного* перерізу $st$-мережі (у тім числі й перерізу з *найменшою* пропускною здатністю: $F \le b^*$).

Якщо вдасться відшукати такий потік, що $F = b(S,T)$, то він і буде *максимальним*, а $(S,T)$ буде *мінімальним* перерізом. Доведення факту існування такого потоку слідує з теореми 5.3 та аналізу алгоритму, представленого у параграфі 5.4. ➢

### 5.3. Збільшувальні шляхи

Усі алгоритми розв'язування задачі максимізації потоку використовують проміжний алгоритм відшукання *збільшувального шляху*, який містить збільшувальні і/або зменшувальні дуги.

- *Збільшувальна* (або *аугментальна*) дуга – це дуга, величина потоку якої менша за її пропускну здатність (величину наявного потоку цією дугою можна *збільшити*). Множину/тип збільшувальних дуг позначають літерою $I$ (з англійської мови: *increase*). Тоді:

$$(k,i) \in I, \text{ якщо } 0 \le x_{ki} < b_{ki}.$$

- *Зменшувальна* дуга – це дуга, величина потоку якої є додатною (величину наявного потоку цією дугою можна *зменшити*). Множину/тип зменшувальних дуг позначають літерою $R$ (з англійської мови: *release*). Тоді:

$$(k,i) \in R, \text{ якщо } 0 < x_{ki} \le b_{ki}.$$

- Дуги, які одночасно належать як множині $I$, так і множині $R$, називають *проміжними*:

$$\big((k,i) \in I\big) \wedge \big((k,i) \in R\big), \text{ якщо } 0 < x_{ki} < b_{ki}.$$

Величину $b_{ki} - x_{ki}$ називають *резервом збільшення* потоку дугою $(k,i) \in I$, а $x_{ki}$ *– резервом зменшення* потоку дугою $(k,i) \in R$.

Розглянемо тепер, якими засобами можна переслати додаткову кількість одиниць потоку з джерела *s* у витік *t*:

1. За допомогою шляху з *s* у *t*, що складається лише зі збільшувальних дуг. На рис. 5.1 нижче кожної дуги стоїть число, яке є *резервом збільшення* потоку дугою.

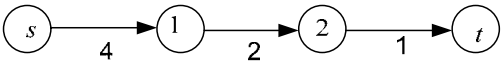



Рис. 5.1. Шлях, що містить тільки збільшувальні дуги

Цим шляхом можна збільшити потік на величину, що дорівнює $\min\{4,2,1\} = 1$. Інтенсивність проміжних вершин 1 і 2 у цьому випадку не зміниться.

2. За допомогою шляху з *t* у *s*, що складається лише зі зменшувальних дуг. На рис. 5.2 нижче кожної дуги стоїть число, яке є *резервом зменшення* потоку дугою.

Тут треба міркувати так. Можна зменшити потік кожною дугою шляху, що спричинило б зменшення зворотного потоку з *t* до *s* і, отже, збільшення чистого потоку з *s* до *t*. Максимальне зменшення потоку з *t* до *s* і, отже, збільшення потоку з *s* до *t* визначається величиною $\min\{4,\ 2,\ 3\} = 2$. Інтенсивність проміжних вершин 1 і 2 у цьому випадку не зміниться.

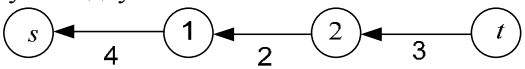



Рис. 5.2. Шлях, що містить тільки зменшувальні дуги

3. Комбінація першого і другого способу, яка полягає у побудові *збільшувального* шляху.

*Збільшувальним* (або *аугментальним*) шляхом в *st*-мережі називають простий шлях від *s* до *t без урахування орієнтації* дуг, в якому прямі дуги мають тип *I*, а зворотні – тип *R*.

Вздовж аугментального шляху можна "переслати" додатковий потік від *s* до *t* величини $\Delta$, що дорівнює *мінімальному* з резервів збільшення/зменшення, відповідно, прямих і зворотних дуг.

На рис. 5.3 зображено орграф, що має збільшувальний шлях (*s*, 1), (1, 2), (3, 2), (3, *t*). Величина потоку в цьому орграфі дорівнює 9. Позначка дуги: *величина_потоку_дуги*, [*пропускна_здатність_дуги*].

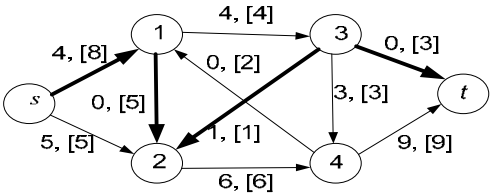



Рис. 5.3. Орграф, що містить збільшувальний шлях

Для збільшувального шляху на рис. 5.3 значення $\Delta = 1$. Збільшуючи потік прямими дугами на 1 і зменшуючи його на зворотній дузі (3, 2) на 1, отримаємо новий потік величини 10 (рис. 5.4). Інтенсивність проміжних вершин 1, 2 і 3 у цьому випадку не зміниться.

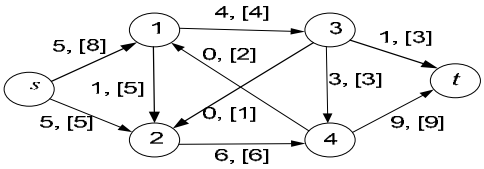



Рис. 5.4. Орграф, що містить потік величини 10

**Теорема 5.3**. Такі твердження в *st*-мережі є еквівалентними:

(I) потік від *s* до *t* максимальний;

(II) не існує збільшувального шляху;

(III) $F = b(S,T)$ для деякої множини $S \subset V$, такої, що $s \in S$ і $t \in T$.

➢ (I) $\Rightarrow$ (II). Доведення очевидне, оскільки існування такого шляху дало б змогу збільшити величину потоку, а це суперечить (I).

(II) $\Rightarrow$ (III). Зачислимо до множини $S$ ( $s \in S$ ) такі проміжні вершини:

$$\begin{array}{l} \text{якщо } \big((k,i) - \text{пряма дуга}\big) \wedge \big(v_k \in S\big) \wedge \big(x_{ki} < b_{ki}\big), \text{ то } v_i \in S, \\ \text{якщо } \big((k,i) - \text{зворотна дуга}\big) \wedge \big(v_i \in S\big) \wedge \big(x_{ik} > 0\big), \text{ то } v_k \in S. \end{array} \quad (5.6)$$

Решта вершин належатимуть підмножині $T$.

Очевидно, що переріз побудовано, оскільки $t \in T$ . Якщо б це було не так (тобто $t \in S$ ), тоді, згідно з (5.6), існував би збільшувальний шлях, а це суперечить (II).

Якщо $(k, i) \in (S,T)$, то $x_{ki} = b_{ki}$ ; інакше вершина $v_i$ належатиме до $S$, згідно з (5.6). Аналогічно, якщо $(i, k) \in (T,S)$, то $x_{ik} = 0$. Отож

$$F = \sum_{(k,i) \in (S,T)} x_{ki} - \sum_{(i,k) \in (T,S)} x_{ik} = \sum_{(k,i) \in (S,T)} b_{ki} - 0 = \sum_{(k,i) \in (S,T)} b_{ki} = b(S,T).$$

(III) $\Rightarrow$ (I), що випливає з теореми (5.2). $\blacktriangleleft$

## 5.4. Метод Форда-Фалкерсона пошуку максимального потоку

Розглянемо *метод* Форда-Фалкерсона визначення максимального потоку, який базується на теоремах 5.2 і 5.3. Ідеться про *метод*, оскільки існує декілька алгоритмів, які його реалізують.

*Метод Форда-Фалкерсона визначення максимального потоку*

*Крок 0.* Для кожної дуги орграфа $(k,i) \in E$ встановлюють $x_{ki} = 0$.

*Крок 1.* Визначають збільшувальний шлях в орграфі.

*Крок 2.* Якщо збільшувального шляху не існує, то потік *оптимальний* (*завершення* роботи). У протилежному випадку визначають *резерви збільшення*/*зменшення* потоків дугами, які утворюють збільшувальний шлях.

*Крок 3.* Обчислюють значення $\Delta$, яке дорівнює *мінімальному* з резервів збільшення/зменшення потоків дугами.

*Крок 4. Збільшують* потік прямими дугами збільшувального шляху на значення $\Delta$.

*Крок 5. Зменшують* потік зворотними дугами збільшувального шляху на значення $\Delta$.

*Крок 6.* Переходять на перший крок.

Алгоритми *визначення максимального потоку* відрізняються один від одного, здебільшого, способами відшукання збільшувального шляху (тобто реалізацією першого кроку методу Форда-Фалкерсона). Розглянемо спочатку найпростіший алгоритм відшукання збільшувального шляху, який базується на позначенні дуг і вершин орграфа [18]:

1. Визначити тип кожної дуги орграфа. Позначити вершину *s*.
2. Позначати дуги і вершини відповідно до *правил позначки* доти, доки не позначено вершину *t*, або доки позначення нових вершин стане неможливим.

   *Правила позначки*:

   – якщо вершина $x$ – позначена, вершина $y$ – не позначена, дуга $(x, y)$ – не позначена і має тип *I*, то позначають дугу $(x, y)$ та її кінець – вершину $y$;

   – якщо вершина $x$ – позначена, вершина $y$ – не позначена, дуга $(y, x)$ – не позначена і має тип *R*, то позначають дугу $(y, x)$ та її початок – вершину $y$;

   – інших випадків позначення немає.

**Теорема 5.4**. Якщо під час виконання алгоритму пошуку збільшувального шляху вершина $t$ – позначена, то в орграфа такий шлях існує. Якщо вершину $t$ позначити не вдалося, то збільшувального шляху в орграфа немає. Алгоритм завершує роботу за скінченну кількість кроків.

➢ За правилами дугу можна позначити тільки тоді, коли одну зі суміжних вершин позначено, а іншу – ні. Отже, під час виконання алгоритму не можна позначити дугу, в якої обидві вершини уже позначені, отож позначені вершини та дуги не утворюють циклів.

Внаслідок процесу позначення утворюється *дерево* з початком у джерелі *s*. Якщо при цьому позначено вершину *t*, то за теоремою 1.1 існує *єдиний простий шлях*, що з'єднує *s* і *t*. Цей шлях є *збільшувальним*, оскільки правила позначки гарантують, що прямі дуги цього ланцюга мають тип *I*, а зворотні – тип *R*.

Доведемо, якщо вершину *t* позначити не вдалося, то збільшувального шляху в орграфа немає (*від супротивного*). Нехай існує збільшувальний шлях (за означенням – це простий шлях від *s*

до $t$ без урахування орієнтації дуг). Правило позначки дає змогу досягти довільної дуги цього шляху і, зокрема, дуги, суміжної з вершиною $t$. Отож вершину $t$ буде позначено (отримали *протиріччя*).

Алгоритм завершує роботу за скінченну кількість кроків, оскільки у ньому лише по одному разу позначаються вершини та дуги, кількість яких є скінченною. ≺

Для орграфа на рис. 5.4 не вдається відшукати новий збільшувальний шлях, оскільки тільки три дуги ($s$, 1), (1, 2), (3, $t$) є збільшувальними.

**Приклад 5.1**. Визначити максимальний потік в орграфі (рис. 5.5).

➢ Для визначення максимального потоку скористаємося методом Форда-Фалкерсона, який базується на позначенні дуг і вершин орграфа під час визначення збільшувального шляху. На рис. 5.5 для усіх дуг орграфа $(k,i)\in E$ встановлено $x_{ki} = 0$ (тобто перед виконанням першого кроку всі дуги орграфа мають тип *I*). Обрані для позначення дуги зберігатимемо у стеку.

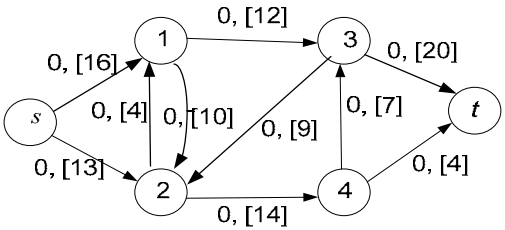



Рис. 5.5. Орграф прикладу 5.1

| Позначення вершин і дуг | Стек |
|---|---|
| $s$ | ($s$, 1), ($s$, 2) |
| $s$, ($s$, 1), 1 | (1, 3), (1, 2), ($s$, 2) |
| $s$, ($s$, 1), 1, (1, 3), 3 | (3, $t$), (1, 2), ($s$, 2) |
| $s$, ($s$, 1), 1, (1, 3), 3, (3, $t$), $t$ | (1, 2), ($s$, 2) |

Збільшувальний шлях: ($s$, 1), (1, 3), (3, $t$). Вздовж цього шляху посилаємо додатковий потік, що дорівнює min{16, 12, 20} = 12. Отримуємо орграф зі сумарною величиною потоку 12 (рис. 5.6), усі дуги якого мають тип *I*, за винятком дуги (1, 3) типу *R*.

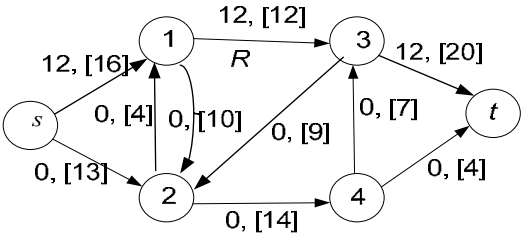



Рис. 5.6. Орграф із сумарною величиною потоку 12

| Позначення вершин і дуг | Стек |
|---|---|
| *s* | (*s*, 2), (*s*, 1) |
| *s*, (*s*, 2), 2 | (2, 4), (2, 1), (*s*, 1) |
| *s*, (*s*, 2), 2, (2, 4), 4 | (4, *t*), (4, 3), (2, 1), (*s*, 1) |
| *s*, (*s*, 2), 2, (2, 4), 4, (4, *t*), *t* | (4, 3), (2, 1), (*s*, 1) |

Збільшувальний шлях: (*s*, 2), (2, 4), (4, *t*). Вздовж цього шляху посилаємо додатковий потік, що дорівнює min{13, 14, 4} = 4. Отримуємо орграф зі сумарною величиною потоку 16 (рис. 5.7), усі дуги якого мають тип *I*, за винятком дуг (1, 3) і (4, *t*) типу *R*.

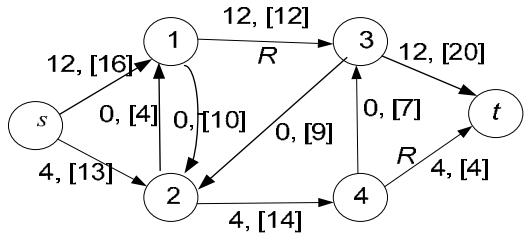



Рис. 5.7. Орграф із сумарною величиною потоку 16

| Позначення вершин і дуг | Стек |
|---|---|
| *s* | (*s*, 2), (*s*, 1) |
| *s*, (*s*, 2), 2 | (2, 4), (2, 1), (*s*, 1) |
| *s*, (*s*, 2), 2, (2, 4), 4 | (4, 3), (2, 1), (*s*, 1) |
| *s*, (*s*, 2), 2, (2, 4), 4, (4, 3), 3 | (3, *t*), (3, 2), (2, 1), (*s*, 1) |
| *s*, (*s*, 2), 2, (2, 4), 4, (4, 3), 3, (3, *t*), *t* | (3, 2), (2, 1), (*s*, 1) |

Збільшувальний шлях: (*s*, 2), (2, 4), (4, 3), (3, *t*). Вздовж цього шляху посилаємо потік, що дорівнює min{9, 10, 7, 8} = 7. Отримує-

мо орграф зі сумарною величиною потоку 23 (рис. 5.8), усі дуги якого мають тип *I*, за винятком дуг (1, 3), (4, *t*) і (4, 3) типу *R*.

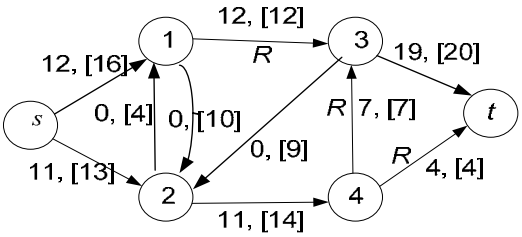



Рис. 5.8. Орграф із сумарною величиною потоку 23

| Позначення вершин і дуг | Стек |
|---|---|
| *s* | (*s*, 2), (*s*, 1) |
| *s*, (*s*, 2), 2 | (2, 4), (2, 1), (*s*, 1) |
| *s*, (*s*, 2), 2, (2, 4), 4 | (2, 1), (*s*, 1) |
| *s* | (*s*, 1) |
| *s*, (*s*, 1), 1 | (1, 2) |
| *s*, (*s*, 1), 1, (1, 2), 2 | (2, 4) |
| *s*, (*s*, 1), 1, (1, 2), 2, (2, 4), 4 | |

Вершина *t* – непозначена (потік на рис. 5.8 *максимальний*). ⮜

## 5.5. Залишкові мережі

*Залишковою мережею* (*residual network*) для певної *st*-мережі $G = (V, E)$ називають орграф $G^* = (V^*, E^*)$, для якого $V^* = V$, а дуги $E^*$ визначають так:

$$(\forall\ (k, i) \in E\colon (x_{ki} = 0) \wedge (b_{ki} > 0)) \Rightarrow (\exists\ (k, i) \in E^*\colon\ b^*_{ki} = b_{ki}),$$

$$(\forall\ (k, i) \in E\colon\ x_{ki} = b_{ki}) \Rightarrow (\exists\ (i, k) \in E^*\colon\ b^*_{ik} = b_{ik} + x_{ki}), \qquad (5.7)$$

$$(\forall\ (k, i) \in E\colon\ 0 < x_{ki} < b_{ki}) \Rightarrow ((\exists\ (i, k) \in E^*\colon\ b^*_{ik} = b_{ik} + x_{ki}) \wedge$$

$$\wedge\ (\exists\ (k, i) \in E^*\colon\ b^*_{ki} = b_{ki} - x_{ki})).$$

Дугу $(i, k)$ (чи дугу $(k, i)) \in E^*$ називають залишковою дугою. Ця дуга може і не бути дугою множини $E$ у заданій *st*-мережі (загалом, $|E^*| \le 2|E|$).

Для кожної дуги $(k, i) \in E$ ми створюємо, згідно з (5.7), одну дугу чи дві залишкові дуги у кожному напрямі:

– у напрямі потоку з вагою, яка дорівнює невикористаній пропускній здатності дуги $(k, i) \in E$;
– назустріч потоку з вагою, яка дорівнює сумі величини потоку дугою $(k, i) \in E$ і пропускної здатності дуги $(i, k) \in E$.

У жодному з цих випадків, згідно з (5.7), ми *не долучаємо* до залишкової мережі дуг, вага яких дорівнює нулю.

Якщо для кожної дуги орграфа $(k,i) \in E$ спочатку встановити $x_{ki} = 0$, то залишкова мережа, згідно з (5.7), збігатиметься із заданою *st*-мережею (вага залишкової дуги дорівнюватиме пропускній здатності відповідної дуги *st*-мережі). Наприклад, для *st*-мережі на рис. 5.5 відповідну залишкову мережу зображено на рис. 5.9.

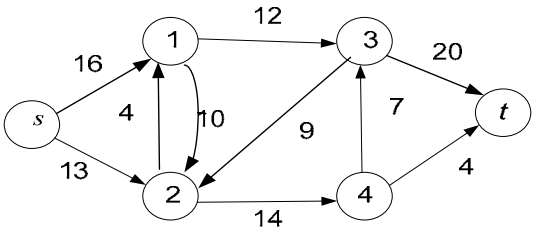



Рис. 5.9. Залишкова мережа (відповідає *st*-мережі на рис. 5.5)

Відшукання збільшувального шляху в *st*-мережі еквівалентно відшуканню орієнтованого шляху від джерела *s* до стоку *t* у відповідній залишковій мережі. Для відшукання орієнтованого шляху можна використати будь-який відомий алгоритм пошуку на графі.

Після відшукання орієнтованого шляху в залишковій мережі від джерела *s* до стоку *t* частини (Δ) пропускних здатностей дуг цього шляху "забираються" потоком, що проходить через них.

Нехай між вершинами *i* та *j* існують дуги $(i, j)$ і $(j, i)$ з пропускними здатностями, відповідно, $b_{ij}$ та $b_{ji}$. Якщо частина пропускної здатності дуги $(i, j)$ "забирається" потоком (тобто $b_{ij} - \Delta$), то, згідно із законом збереження потоку, для дуги $(j, i)$ має виконатися відповідне збільшення пропускної здатності (тобто $b_{ji} + \Delta$). Якщо між

вершинами $i$ та $j$ існує тільки одна з дуг $(i, j)$ або $(j, i)$, то для неї відповідна компенсація потоку відбувається на фіктивній дузі.

**Увага**! У залишковій мережі відображають тільки пропускні здатності дуг (невикористані та компенсаційні). Отож для вершин залишкової мережі не виконується закон збереження потоку, оскільки тут він просто не відображається.

Збільшення величини потоку вздовж орієнтованого шляху в залишковій мережі від джерела $s$ до стоку $t$ спричинює зміни у залишковій мережі: хоча б одна дуга змінює напрям або зникає.

*Алгоритм визначення максимального потоку* за допомогою залишкових мереж:

*Крок 0.* $B^*= B$, де $B$ – матриця пропускних здатностей дуг $st$-мережі.

*Крок 1.* Знайти орієнтований шлях $p$ від $s$ до $t$ у залишковій мережі, визначеній матрицею $B^*$. Якщо такого шляху не існує, то перейти на крок 3.

*Крок 2.* Нехай $b_{ij}^-$ $(b_{ji}^+)$ – пропускні здатності дуг шляху $p$, відповідно у напрямі $s \to t$ ($t \to s$) і $\Delta = \min\{b_{ij}^-\} > 0$. У матриці $B^*$ відняти $\Delta$ від усіх $b_{ij}^-$ і додати $\Delta$ до всіх $b_{ji}^+$. Перейти на 1.

*Крок 3.* Знайти максимальний потік в $st$-мережі згідно з правилом:

$$x_{ij} = \begin{cases} b_{ij} - b_{ij}^*, & \text{якщо } b_{ij} > b_{ij}^*; \\ 0, & \text{якщо } b_{ij} \le b_{ij}^*. \end{cases}$$

*Крок 4.* Завершити алгоритм.

Під час відшукання орієнтованого шляху безпосередньо у матриці $B^*$ розпочинають пошук з першого рядка (рядка $s$) та обирають наступну вершину серед тих, які з'єднано з $s$ дугами додатних ваг.

Далі розглядають рядок, що відповідає обраній вершині, і обирають наступну вершину, з'єднану з попередньою вершиною дугою додатної ваги. Процес продовжують доти, доки не буде досягнуто вершини $t$, або встановлено, що досягнути $t$ неможливо.

**Приклад 5.2**. Знайти максимальний потік в орграфі (рис. 5.5), використовуючи залишкові мережі.

➢ Матрицю пропускних здатностей дуг орграфа наведено у таблиці 5.1, а відповідну залишкову мережу зображено на рис. 5.9.

*Таблиця 5.1.* Початкова матриця $B^*$

| | $s$ | 1 | 2 | 3 | 4 | $t$ |
|---|---|---|---|---|---|---|
| $s$ | 0 | 16– | 13 | 0 | 0 | 0 |
| 1 | 0+ | 0 | 10 | 12– | 0 | 0 |
| 2 | 0 | 4 | 0 | 0 | 14 | 0 |
| 3 | 0 | 0+ | 9 | 0 | 0 | 20– |
| 4 | 0 | 0 | 0 | 7 | 0 | 4 |
| $t$ | 0 | 0 | 0 | 0+ | 0 | 0 |

Орієнтований шлях ($\Delta = 12$):

$$s \xrightarrow[16]{} 1 \xrightarrow[12]{} 3 \xrightarrow[20]{} t\,.$$

Матрицю $B^*$ після виконання кроку $2^1$ наведено у таблиці 5.2, а відповідну залишкову мережу зображено на рис. 5.10.

*Таблиця 5.2.* Матриця $B^*$ (крок $2^1$)

| | $s$ | 1 | 2 | 3 | 4 | $t$ |
|---|---|---|---|---|---|---|
| $s$ | 0 | 4 | 13– | 0 | 0 | 0 |
| 1 | 12 | 0 | 10 | 0 | 0 | 0 |
| 2 | 0+ | 4 | 0 | 0 | 14– | 0 |
| 3 | 0 | 12 | 9 | 0 | 0 | 8 |
| 4 | 0 | 0 | 0+ | 7 | 0 | 4– |
| $t$ | 0 | 0 | 0 | 12 | 0+ | 0 |

Орієнтований шлях ($\Delta = 4$):

$$s \xrightarrow[13]{} 2 \xrightarrow[14]{} 4 \xrightarrow[4]{} t\,.$$

Матрицю $B^*$ після виконання кроку $2^2$ наведено у таблиці 5.3, а відповідну залишкову мережу зображено на рис. 5.11.

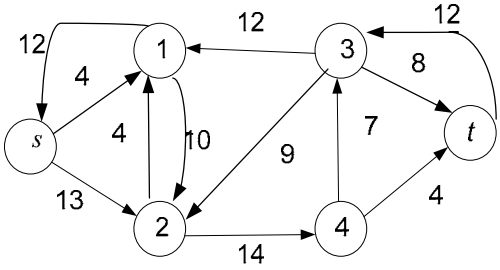



Рис. 5.10. Залишкова мережа (після виконання кроку $2^1$)

*Таблиця 5.3.* Матриця $B^*$ (крок $2^2$)

| | *s* | 1 | 2 | 3 | 4 | *t* |
|---|---|---|---|---|---|---|
| *s* | 0 | 4 | 9– | 0 | 0 | 0 |
| 1 | 12 | 0 | 10 | 0 | 0 | 0 |
| 2 | 4+ | 4 | 0 | 0 | 10– | 0 |
| 3 | 0 | 12 | 9 | 0 | 0+ | 8– |
| 4 | 0 | 0 | 4+ | 7– | 0 | 0 |
| *t* | 0 | 0 | 0 | 12+ | 4 | 0 |

Орієнтований шлях ($\Delta = 7$):

$$s \xrightarrow[9]{} 2 \xrightarrow[10]{} 4 \xrightarrow[7]{} 3 \xrightarrow[8]{} t .$$

Матрицю $B^*$ після виконання кроку $2^3$ наведено у таблиці 5.4, а відповідну залишкову мережу зображено на рис. 5.12.

*Таблиця 5.4.* Матриця $B^*$ (крок $2^3$)

| | *s* | 1 | 2 | 3 | 4 | *t* |
|---|---|---|---|---|---|---|
| *s* | 0 | 4 | 2 | 0 | 0 | 0 |
| 1 | 12 | 0 | 10 | 0 | 0 | 0 |
| 2 | 11 | 4 | 0 | 0 | 3 | 0 |
| 3 | 0 | 12 | 9 | 0 | 7 | 1 |
| 4 | 0 | 0 | 11 | 0 | 0 | 0 |
| *t* | 0 | 0 | 0 | 19 | 4 | 0 |

Якщо проаналізувати залишкову мережу на рис. 5.12, то від джерела *s* можна добратися тільки до вершини 4. Отже, орієнтованого шляху від *s* до *t* у залишковій мережі не існує. Переходимо на крок 3.

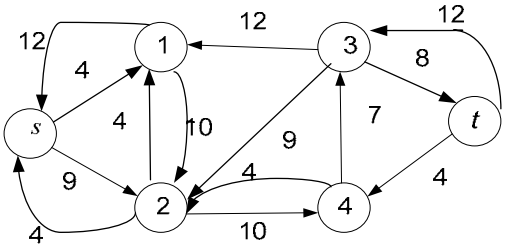



Рис. 5.11. Залишкова мережа (після виконання кроку $2^2$)

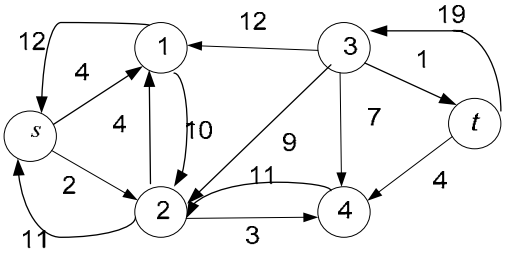



Рис. 5.12. Залишкова мережа (після виконання кроку $2^3$)

Результати виконання кроку 3 наведено у таблиці 5.5 (максимальний потік у *st*-мережі, зображеній на рис. 5.5).

*Таблиця 5.5.* Максимальний потік

| | *s* | 1 | 2 | 3 | 4 | *t* |
|---|---|---|---|---|---|---|
| *s* | 0 | 12 | 11 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 12 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 11 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 19 |
| 4 | 0 | 0 | 0 | 7 | 0 | 4 |
| *t* | 0 | 0 | 0 | 0 | 0 | 0 |

Максимальний потік у *st*-мережі, що відповідає таблиці 5.5, зображено на рис. 5.13 (відтворені тільки ті дуги, якими реально протікає потік).

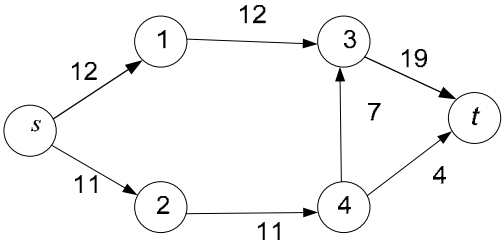



Рис. 5.13. Максимальний потік в *st*-мережі (рис. 5.5) ➢

Ми розглянули два алгоритми реалізації методу Форда-Фалкерсона відшукання максимального потоку (за допомогою позначок та залишкових мереж). Існують й інші алгоритми відшукання максимального потоку [5, 8]. Однак, дослідження того, який алгоритм найкращий, є надзвичайно складною задачею і виходить за рамки нашого підручника.

У наступному пункті розглянемо деякі задачі, розв'язок яких зводиться до розв'язку задачі відшукання максимального потоку в *st*-мережі.

## 5.6. Зв'язок задачі максимізації потоку з іншими задачами ДО

Домовимося надалі *st-мережу*, яку визначено у параграфі 5.1, називати *стандартною st*-мережею. Це нам необхідно, щоб відрізняти її від *st*-мереж, що мають додаткові обмеження чи специфічні властивості.

*1. Потік з декількома джерелами і стоками.* Дано орієнтований орграф $G=(V,E)$. Множина $V$ складається з множини джерел $s=\{v_1,...,v_l\}$, множини проміжних вершин $N=\{v_{l+1},...,v_m\}$ і множини стоків $t=\{v_{m+1},...,v_n\}$.

Розширимо орграф $G$ до орграфа $G'=(V',E')$, додавши дві фіктивні вершини $v_0$ і $v_{n+1}$ ($v_0$ – джерело; $v_{n+1}$ – стік) і фіктивні дуги $(v_0,v_j)$, $j=\overline{1,l}$ і $(v_i,v_{n+1})$, $i=\overline{m+1,n}$. Пропускні здатності фіктивних дуг: $b_{0j}=\sum\limits_{i:(j,i)\in E} b_{ji}$ ($j=\overline{1,l}$) і $b_{i,n+1}=\sum\limits_{j:(j,i)\in E} b_{ji}$ ($i=\overline{m+1,n}$).

Величину потоку у вихідному орграфі $G$ і в розширеному орграфі $G'$ визначають пропускні здатності дуг орграфа $G$. Отже, задача про максимальний потік з множини джерел $s=\{v_1,...,v_l\}$ у множину витоків $t=\{v_{m+1},...,v_n\}$ орграфа $G$ *еквівалентна* задачі про максимальний потік у стандартній *st*-мережі. Приклад зведення на рис. 5.14 (фіктивні дуги – пунктирні).

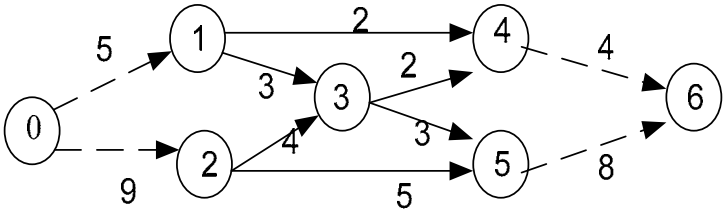



Рис. 5.14. Орграф з двома джерелами і двома витоками

*2. Потік з обмеженнями на пропускні здатності вершин*. Нехай задано деяку *st*-мережу з додатковими обмеженнями, за яких потік через кожну вершину не повинен перевищувати деякої фіксованої пропускної здатності (різної для різних вершин). Необхідно обчислити максимальний потік, що задовольняє цим додатковим обмеженням. Ця задача *еквівалентна* задачі про максимальний потік у стандартній *st*-мережі.

Дійсно, можна побудувати стандартну *st*-мережу, в якій для кожної вихідної вершини *u* долучають фіктивну вершину *u**, а пропускну здатність фіктивної дуги (*u*, *u**) встановлюють рівною пропускній здатності вершини *u*. Приклад *st*-мережі з обмеженнями на пропускні здатності вершин на рис. 5.15 (пропускну здатність вершини зазначено над нею). Зведення цієї мережі до стандартної *st*-мережі на рис. 5.16 (фіктивні дуги – пунктирні).

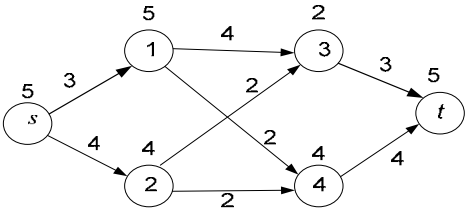



Рис. 5.15. Мережа з обмеженнями на пропускні здатності вершин

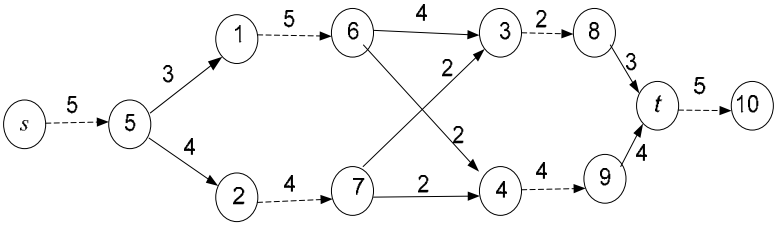



Рис. 5.16. Зведення мережі (рис. 5.15) до стандартної *st*-мережі

*3. Задача про максимальний потік для неорієнтованих st-мереж.* Ця задача більше, ніж стандартна задача, відповідає обраній моделі трубопроводу для транспортування рідин: вона допускає випадок, коли рідина може протікати через труби в обох напрямах.

Задача про максимальний потік для неорієнтованих *st*-мереж *зводиться* до задачі про максимальний потік для стандартних (орієнтованих) *st*-мереж.

Дійсно, можна побудувати стандартну *st*-мережу з тими ж вершинами, де кожному ребру вихідної мережі відповідатимуть дві дуги з пропускними здатностями, які дорівнюватимуть пропускній здатності цього ребра.

Довільний потік в орієнтованій мережі безпосередньо відповідатиме потоку такої ж величини, що протікає в неорієнтованій мережі. Міркуємо так.

Нехай в орієнтованій мережі через дугу $(u, v)$ протікає потік величиною $r$, а через ребро $(v, u)$ протікає потік величиною $z$. Якщо $r > z$, то можна пропустити потік величиною $r - z$ ребром $[u, v]$ неорієнтованої мережі у напрямі вершини $v$; у протилежному випадку це буде потік величиною $z - r$ ребром $[u, v]$ у напрямі вершини $u$.

*4. Задача про допустимі потоки.* Дано орієнтований орграф $G = (V, E)$, дуги якого мають додатні ваги (*пропускні здатності*). Нехай кожна вершина орграфа також має вагу, яку можна розглядати як *запас* за *додатного* значення цієї ваги, чи як *потребу* – за *від'ємного* значення. Сума ваг вершин орграфа дорівнює *нулю*. Приклад такого орграфа наведено на рис. 5.17.

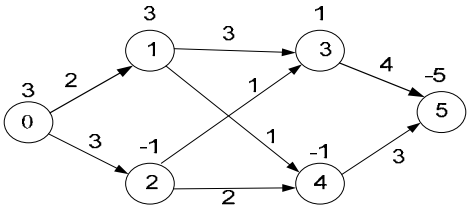



Рис. 5.17. Орграф задачі про допустимі потоки

Вершини з запасом відповідають складам у задачі про розподіл запасів, вершини з потребами – роздрібним торговим точкам, а дуги – дорогам на маршрутах вантажних машин.

Потік називають *допустимим*, якщо різниця між величиною потоку, що надходить у кожну вершину, і величиною потоку, що з неї витікає, дорівнює вазі цієї вершини. Постає *задача відшукання* допустимих потоків для заданого орграфа.

Задача про допустимі потоки відповідає на таке питання: чи можна відшукати такий спосіб доставки товарів, за якого повсюдно забезпечується відповідність запасів і потреб. Можливий допустимий потік в орграфі (рис. 5.17) зображено на рис. 5.18.

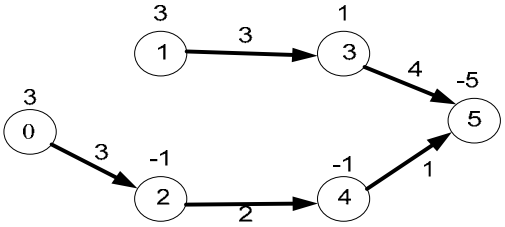



Рис. 5.18. Допустимий потік в орграфі (рис. 5.17)

Задача про допустимий потік *зводиться* до задачі про максимальний потік в *st*-мережі. Для цього необхідно долучити до орграфа фіктивну вершину-джерело *s*, з якої виходитимуть дуги у кожну вершину із запасом (вага дуги дорівнюватиме запасу відповідної вершини), і фіктивну вершину-стік *t*, в яку веде дуга з кожної вершини з потребами (вага дуги дорівнюватиме абсолютній величині потреби відповідної вершини). На рис. 5.19 проілюстровано приклад такого зведення для орграфа на рис. 5.17.

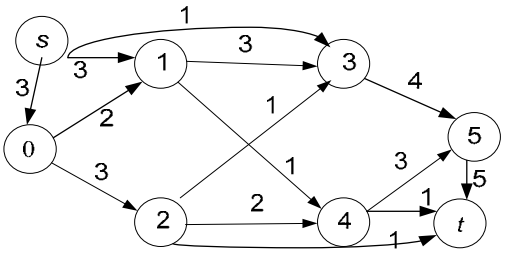



Рис. 5.19. Зведення до *st*-мережі орграфа на рис. 5.17

У початковому орграфі міститься допустимий потік тоді і тільки тоді, коли усі дуги, що виходять із джерела, і всі дуги, що ведуть у стік, заповнені так, що в *st*-мережі утворюється максимальний потік.

*5. Задача про максимальний потік як задача лінійного програмування* (ЛП). Задачу *максимізації* потоку в *st-мережі* ($v_0$– джерело, $v_n$ – стік) як задачу ЛП формулюють так: знайти значення $x_{ij}$

для усіх $(i, j) \in E$, які *максимізують* величину $f = \sum_{j=1}^{n} x_{0j} = \sum_{i=0}^{n-1} x_{in}$ за обмежень: $0 \le x_{ij} \le b_{ij}$, $i, j = \overline{0, n}, i \ne j$; $\sum_{i=0}^{n-1} x_{ik} - \sum_{j=1}^{n} x_{kj} = 0$, $k = \overline{1, n-1}$.

## ? Запитання для самоперевірки

1. Що таке джерело і стік потоку?
2. Що таке пропускна здатність дуги?
3. Що таке величина потоку дугою?
4. Сформулюйте означення сумарної величини потоку.
5. Що таке інтенсивність потоку у вершині орграфа?
6. Сформулюйте закон збереження потоку.
7. Що таке переріз?
8. Сформулюйте означення пропускної здатності перерізу.
9. Як вимірюють сумарну величину потоку для довільного перерізу.
10. Що таке збільшувальна/зменшувальна дуга?
11. Що таке збільшувальний (аугментальний) шлях?
12. Сформулюйте і доведіть теорему Форда-Фалкерсона.
13. Опишіть алгоритм пошуку аугментального шляху.
14. Опишіть метод Форда-Фалкерсона пошуку максимального потоку.
15. Що таке залишкова мережа?
16. Опишіть алгоритм визначення максимального потоку за допомогою залишкових мереж.
17. Опишіть зведення задачі про максимальний потік з декількома джерелами і стоками до задачі про максимальний потік в *st*-мережі.
18. Опишіть зведення задачі про максимальний потік з обмеженнями на пропускні здатності вершин до задачі про максимальний потік в *st*-мережі.
19. Опишіть зведення задачі про максимальний потік для неорієнтованих *st*-мереж до задачі про максимальний потік в *st*-мережі.
20. Опишіть зведення задачі про допустимі потоки до задачі про максимальний потік у *st*-мережі.
21. Опишіть зведення задачі про максимальний потік до задачі лінійного програмування.

## 🕮 Завдання для самостійної роботи

**Завдання 5.1**. Туристичному бюро необхідно у певний день організувати переліт туристів з міста *s* у місто *t*. Прямих рейсів із *s* в *t* немає. Однак можна скористатись проміжними містами *a*, *b*, *c*, *d*. Схема можливих перельотів: $e_1=(s, a)$, $e_2=(s, c)$, $e_3=(a, b)$, $e_4=(c, d)$, $e_5=(d, a)$, $e_6=(b, c)$, $e_7=(b, t)$, $e_8=(d, t)$. Для кожного маршруту $e_i$ задано вагу $c(e_i)$ – число вільних місць (рейси, на які вільних місць немає, не зазначено). Необхідно вибрати таку стратегію бронювання квитків, за якої вдається організувати переліт максимальної кількості туристів. Варіанти завдань:

1. $c(e_1)=15,\ c(e_2)=4,\ c(e_3)=12,\ c(e_4)=8,\ c(e_5)=5, c(e_6)=3,$ $c(e_7)=7,\ c(e_8)=10$.
2. $c(e_1)=30,\ c(e_2)=8,\ c(e_3)=12,\ c(e_4)=20,\ c(e_5)=10,\ c(e_6)=8,$ $c(e_7)=14,\ c(e_8)=20$.
3. $c(e_1)=20,\ c(e_2)=6,\ c(e_3)=8,\ c(e_4)=15,\ c(e_5)=7,\ c(e_6)=4,$ $c(e_7)=11,\ c(e_8)=15$.
4. $c(e_1)=21,\ c(e_2)=7,\ c(e_3)=9,\ c(e_4)=16,\ c(e_5)=9,\ c(e_6)=6,$ $c(e_7)=13,\ c(e_8)=17$.
5. $c(e_1)=22,\ c(e_2)=8,\ c(e_3)=10,\ c(e_4)=17,\ c(e_5)=10,\ c(e_6)=7,$ $c(e_7)=14,\ c(e_8)=18$.

**Завдання 5.2**. Відшукати максимальний потік і величину потоку кожною дугою для орграфа на рис. 5.19.

**Завдання 5.3**. Відшукати максимальний потік і величину потоку кожною дугою для орграфа на рис. 5.20.

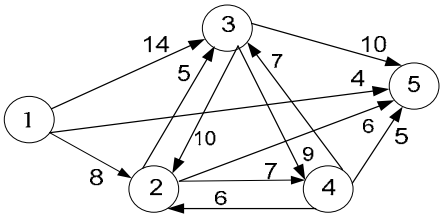



Рис. 5.20. Орграф завдання 5.3

# 6. ПОТІК МІНІМАЛЬНОЇ ВАРТОСТІ

**План викладу матеріалу**:

1. Загальні положення.
2. Відшукання потоку мінімальної вартості.
3. Зв'язок задачі про потік мінімальної вартості з іншими задачами ДО.

**Ключові терміни розділу**

- *Потік мінімальної вартості*
- *Вартість потоку дугою*
- *Максимальний потік мінімальної вартості*
- *Розподільна мережа*
- *Сумарна вартість потоку*
- *Модифікація визначення залишкової мережі*

## 6.1. Загальні положення

Задача максимізації потоку може мати декілька рішень. Ця обставина дає право поставити запитання, чи можна ввести *додаткові* критерії з метою вибору одного з цих рішень? Можливо, з тих чи інших причин перевагу необхідно надати рішенню, яке використовує найменше число дуг, чи рішенню, що передбачає побудову найкоротших шляхів тощо.

Ці задачі складніші, ніж стандартна задача про максимальний потік, і відомі у літературі як задачі про *потік мінімальної вартості* (*mincost flow problem*).

Як і у випадку задачі про максимальний потік, існують різні еквівалентні способи постановки задачі про потік мінімальної вартості (*моделі*). Найчастіше у літературі описують модель *максимального потоку з мінімальною вартістю* (*mincost-maxflow model*) і модель *розподільної мережі* (*distribution network*).

Модель *максимального потоку з мінімальною вартістю* базується на *st*-мережі, в якій кожній дузі $(i, j)$ з пропускною здатністю $b_{ij}$ відповідає невід'ємна *вартість дуги* $c_{ij}$ проходження *одиниці потоку* цією дугою.

Вартість потоку дугою $(i, j)$ дорівнює добуткові $c_{ij} \cdot x_{ij}$, де $x_{ij}$ – величина потоку цією дугою. *Сумарна вартість потоку* (чи просто *вартість потоку*) дорівнює сумі вартостей потоку кожною дугою. Необхідно знайти такий *максимальний потік*, вартість якого є *найменшою* порівняно з вартостями інших максимальних потоків.

Модель *розподільної мережі* використовують, здебільшого, під час розгляду систем, що здійснюють розподіл товарів. Вершини орграфа, що відповідають складам, мають додатні ваги (*запаси* товарів). Вершини, що відповідають роздрібним торговим точкам, мають від'ємні ваги (*потреби* товарів). Пропускні здатності дуг відповідають кількості та вантажопідйомності автомобілів, що курсують на відповідних маршрутах. Природною інтерпретацією вартості дуги може слугувати вартість перевезення одиниці товару відповідним маршрутом.

Нагадаємо, що потік є допустимим, якщо різниця між величиною потоку, що надходить у кожну вершину, і величиною потоку, що з неї витікає, дорівнює вазі цієї вершини. Задача полягає у тому, щоб у *розподільній мережі* відшукати *допустимий потік мінімальної вартості*.

Очевидно, що задача про максимальний потік мінімальної вартості і задача про допустимий потік мінімальної вартості у розподільній мережі *еквівалентні*. Це безпосередньо випливає із задачі 4 параграфа 5.6 (вартості дуг, які виходять з фіктивної вершини $s$ чи входять у фіктивну вершину $t$, вважають рівними нулю). Як наслідок цієї еквівалентності термін *задача про потік мінімальної вартості* вживають для позначення обох задач.

## 6.2. Відшукання потоку мінімальної вартості

Для відшукання потоку мінімальної вартості передусім розширюють визначення *залишкових мереж* з тим, щоб відобразити вартості дуг, а саме: зворотним дугам приписують відповідні від'ємні вартості, а вартості прямих дуг не змінюють.

**Теорема 6.1**. Максимальний потік має мінімальну вартість тоді і тільки тоді, коли його залишкова мережа не містить орієнтованого циклу сумарної від'ємної вартості.

➢ *Необхідність* (від супротивного). Дано максимальний потік мінімальної вартості. Припустимо, що залишкова мережа цього потоку містить цикл сумарної від'ємної вартості $c$, а $\Delta$ – величина мінімальної пропускної здатності дуг цього циклу.

Змінимо потік, додаючи $\Delta$ до величини потоку в дугах, які відповідають дугам циклу з додатною вартістю в залишковій ме-

режі (прямі дуги), і віднімаючи $\Delta$ від величини потоку в дугах, які відповідають дугам циклу з від'ємною вартістю в залишковій мережі (зворотні дуги). Ці зміни не роблять впливу на різницю між величиною потоку, що надходить у кожну вершину, і величиною потоку, що з неї витікає. Однак ці зміни спричинюють *зменшення вартості потоку* на величину $c \cdot \Delta$, оскільки $c \cdot \Delta < 0$. Отримали *суперечність* (за припущенням маємо потік мінімальної вартості).

*Достатність* (від супротивного). Нехай існує максимальний потік *без циклів сумарної від'ємної вартості*, вартість $F$ якого не є мінімальною. Розглянемо будь-який максимальний потік з мінімальною вартістю $M$. Для перетворення $F$ у $M$ необхідно *зменшувати* вартість потоку за рахунок зміни величини потоку дуг деякого орієнтованого циклу. Однак цього не можна зробити, оскільки немає циклів сумарної від'ємної вартості (протиріччя з умовою $F > M$). ≺

Теорема 6.1 дає змогу сформулювати простий узагальнений алгоритм відшукання потоку мінімальної вартості (*алгоритм викреслювання циклів*):

*Крок 0*. Знайти максимальний потік.

*Крок 1*. У залишковій мережі знайти довільний орієнтований цикл сумарної від'ємної вартості. Якщо такого циклу не існує, то перейти на крок 3.

*Крок 2*. Змінити потік, додаючи $\Delta$ до величини потоку в дугах, які відповідають дугам циклу з додатною вартістю в залишковій мережі (прямі дуги), і віднімаючи $\Delta$ від величини потоку в дугах, які відповідають дугам циклу з від'ємною вартістю в залишковій мережі (зворотні дуги). Перейти на крок 1.

*Крок 3*. Користуючись останньою залишковою мережею, побудувати орграф, що містить максимальний потік мінімальної вартості. Завершити алгоритм.

Узагальнений алгоритм викреслювання циклів допускає декілька різних реалізацій, оскільки методи відшукання початкового максимального потоку і відшукання циклів сумарної від'ємної вартості не описано.

**Приклад 6.1**. Знайти потік мінімальної вартості в орграфі на рис. 6.1, використовуючи узагальнений алгоритм викреслювання циклів. Вартості дуг зображено в округлих дужках.

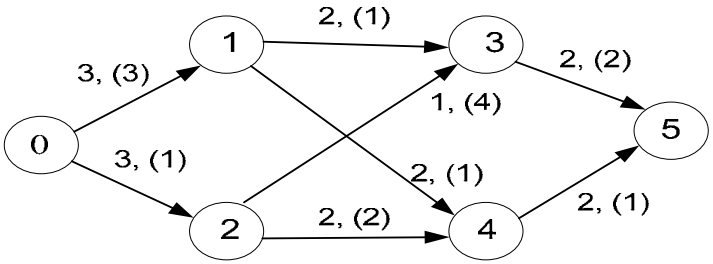



Рис. 6.1. Орграф задачі про потік мінімальної вартості

➢ Знайдемо максимальний потік в орграфі, використовуючи залишкові мережі. Завершальну залишкову мережу максимального потоку сумарної величини 4 та вартості 22 зображено на рис. 6.2, а відповідний орграф з максимальним потоком – на рис. 6.3.

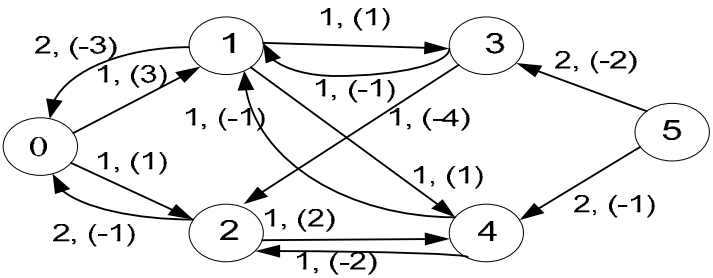



Рис. 6.2. Залишкова мережа максимального потоку вартості 22

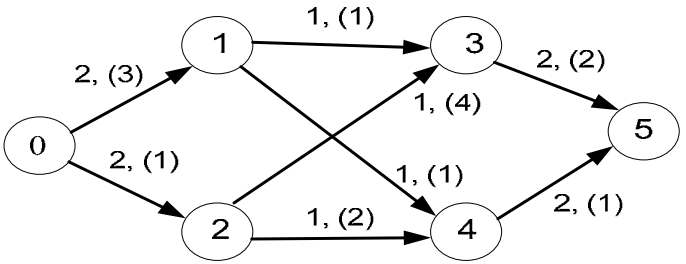



Рис. 6.3. Орграф з максимальним потоком вартості 22

У залишковій мережі на рис. 6.2 існують такі орієнтовані цикли сумарної від'ємної вартості:

$$4 \xrightarrow[1,(-1)]{} 1 \xrightarrow[2,(-3)]{} 0 \xrightarrow[1,(1)]{} 2 \xrightarrow[1,(2)]{} 4;$$

$$3 \xrightarrow[1,(-4)]{} 2 \xrightarrow[2,(-1)]{} 0 \xrightarrow[1,(3)]{} 1 \xrightarrow[1,(1)]{} 3;$$

$$3 \xrightarrow[1,(-4)]{} 2 \xrightarrow[1,(2)]{} 4 \xrightarrow[1,(-1)]{} 1 \xrightarrow[1,(1)]{} 3.$$

Обираємо цикл $4 \xrightarrow[1,(-1)]{} 1 \xrightarrow[2,(-3)]{} 0 \xrightarrow[1,(1)]{} 2 \xrightarrow[1,(2)]{} 4$, дуги якого мають мінімальну пропускну здатність $\Delta = 1$. Змінимо потік на рис. 6.3, додаючи $\Delta = 1$ до величини потоку в дугах (0, 2) і (2, 4), і віднімаючи $\Delta = 1$ від величини потоку в дугах (0, 1) і (1, 4).

Орграф максимального потоку сумарної величини 4 та вартості 21 зображено на рис. 6.4, а відповідну залишкову мережу – на рис. 6.5.

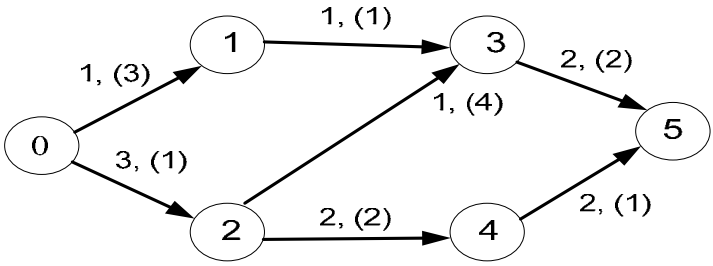



Рис. 6.4. Орграф з максимальним потоком вартості 21

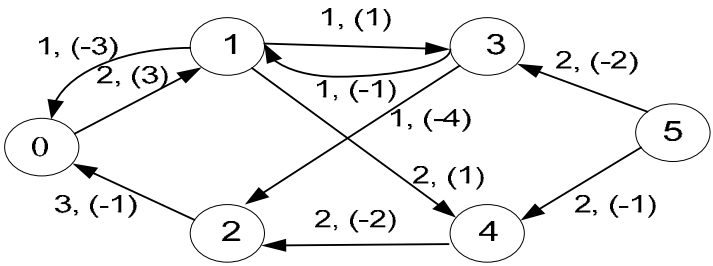



Рис. 6.5. Залишкова мережа максимального потоку вартості 21

У залишковій мережі на рис. 6.5 існує орієнтований цикл сумарної від'ємної вартості $3 \xrightarrow[1,(-4)]{} 2 \xrightarrow[3,(-1)]{} 0 \xrightarrow[2,(3)]{} 1 \xrightarrow[1,(1)]{} 3$, дуги якого мають мінімальну пропускну здатність $\Delta = 1$. Змінимо потік на рис. 6.4, додаючи $\Delta = 1$ до величини потоку в дугах (0, 1) і (1, 3), і віднімаючи $\Delta = 1$ від величини потоку в дугах (0, 2) і (2, 3).

Орграф максимального потоку сумарної величини 4 і вартості 20 зображено на рис. 6.6, а відповідну залишкову мережу – на рис. 6.7.

У залишковій мережі на рис. 6.7 орієнтованого циклу сумарної від'ємної вартості не існує. Отож, орграф на рис. 6.6 зображає *максимальний потік* сумарної величини 4 і *мінімальної вартості* 20.

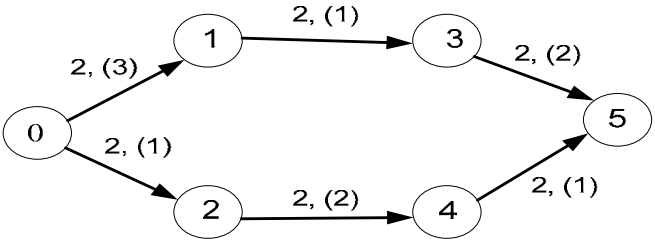



Рис. 6.6. Орграф з максимальним потоком вартості 20

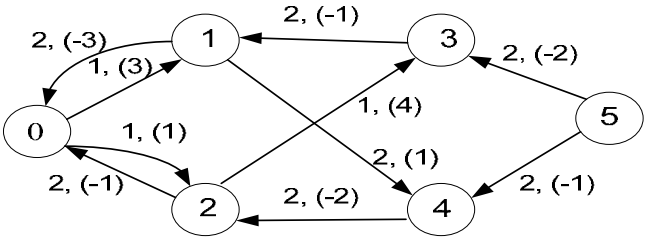



Рис. 6.7. Залишкова мережа максимального потоку вартості 20 ◅

Під час розв'язування прикладу 1.22 ми для більшої наочності водночас використовували орграфи та відповідні залишкові мережі. Однак можна так переформулювати алгоритм *викреслювання циклів*, щоб використовувати тільки залишкові мережі (самостійно).

## 6.3. Зведення узагальненої транспортної задачі до задачі про потік мінімальної вартості

У четвертому розділі першої частини цього підручника ми розглядали постановку та розв'язання стандартної *транспортної задачі* (ТЗ). В *узагальненій ТЗ*, окрім пунктів постачання однорідних товарів (джерел) і пунктів споживання цих товарів (стоків), ще наявні *перевалочні пункти* (проміжні вершини), в яких товари перевантажують з одних транспортних засобів на інші.

Зазвичай вважають, що на перевалочних пунктах не може відбуватися накопичення товарів (скільки товарів за одиницю часу надійшло, стільки товарів за одиницю часу буде відправлено). Якщо ж перевалочний пункт має склади, то на ньому може відбуватися накопичення товарів (у цьому випадку його умовно зачислюють до пунктів споживання).

У стандартній ТЗ перевезення товарів здійснюється між будь-якими пунктами постачання та будь-якими пунктами споживання. На відміну від стандартної ТЗ, в узагальненій ТЗ перевезення товарів може також відбуватися між окремими пунктами постачання чи між окремими пунктами споживання.

Окрім цього, в узагальненій ТЗ між окремими пунктами постачання, перевалочними пунктами і пунктами споживання може не існувати сполучення (доріг), а на пропускні здатності наявних доріг накладають обмеження (тобто пропускні здатності доріг відповідають кількості та вантажопідйомності транспортних засобів, що курсують цими дорогами).

Узагальнену ТЗ називають *закритою*, якщо сумарна величина попиту товарів $B$ усіма пунктами споживання дорівнює сумарній величині пропозиції товарів $B$ усіх пунктів постачання. Якщо умова закритості не виконується, то вводять фіктивний пункт постачання/споживання (див. розділ 4 частини 1 цього підручника).

У *закритій* узагальненій ТЗ необхідно знайти такий спосіб перевезення товарів сумарної величини $B$ від пунктів постачання до пунктів споживання з *можливим* використанням перевалочних пунктів, який *мінімізує загальну вартість* перевезення товарів транспортною мережею; щодо цього необхідно враховувати обмеження

на пропускні здатності дуг і на величини попиту і пропозиції вершин.

Обмеження, які накладають на пропускні здатності доріг, можуть вплинути на існування допустимих розв'язків узагальненої ТЗ. На відміну від стандартної ТЗ, *умова закритості* узагальненої ТЗ ще *не гарантує* існування хоча б одного допустимого розв'язку.

У такому випадку логічно знайти *максимально можливу* сумарну величину товарів, які можна перевезти від пунктів постачання до пунктів споживання з *можливим* використанням перевалочних пунктів, яка *мінімізує загальну вартість* перевезення цих товарів транспортною мережею.

Аналізуючи вищесказане, можна зробити висновок, що модель узагальненої ТЗ більше відповідає умовам реальних транспортних мереж, ніж модель стандартної ТЗ. Однак розв'язати узагальнену ТЗ набагато важче, ніж стандартну ТЗ.

Для розв'язування узагальненої ТЗ розроблено спеціальний симплексний метод у мережі [13, 16, 17]. Однак найчастіше узагальнену ТЗ зводять до задачі про потік мінімальної вартості.

Очевидно, що узагальненій ТЗ відповідає *модель спеціальної розподільної мережі* (рис. 6.8), в якій вершини орграфа, що відповідають джерелам, мають додатні ваги (*запаси* товарів), а вершини, що відповідають стокам – від'ємні ваги (*потреби* товарів). Ваги проміжних вершин дорівнюють нулю (скільки товарів за одиницю часу надійшло, стільки за одиницю часу й буде відправлено). Вартістю дуги слугує вартість перевезення одиниці товару відповідним маршрутом. Позначка дуги: **пропускна_здатність, (вартість_дуги)**.

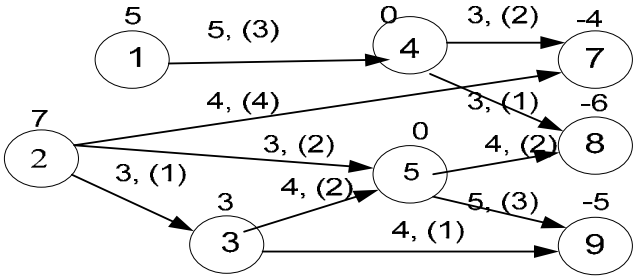



Рис. 6.8. Узагальнена ТЗ (модель спеціальної розподільної мережі)

Орграф відповідної задачі про потік мінімальної вартості зображено на рис. 6.9.

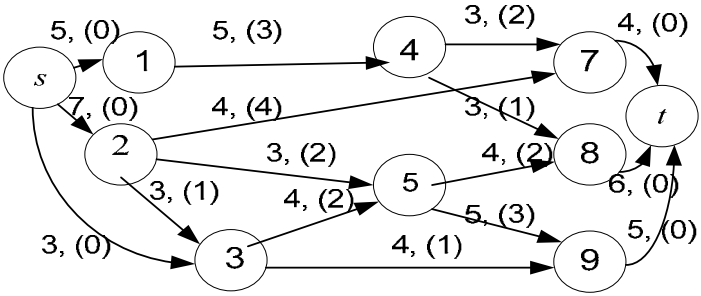



Рис. 6.9. Орграф задачі про потік мінімальної вартості (для узагальненої ТЗ на рис. 6.8)

Інколи пропускна здатність дуги має обмеження з обох боків (тобто $l_{ij} \le x_{ij} \le u_{ij}$). Нижню межу пропускної здатності дуги $l_{ij}$ можна видалити з обмежень підстановкою $x'_{ij} = x_{ij} - l_{ij}$. Для нового потоку $x'_{ij}$ верхньою межею пропускної здатності дуги $(i, j)$ буде величина $b_{ij} = u_{ij} - l_{ij}$, а нижньою – число 0 (тобто $0 \le x'_{ij} \le b_{ij}$). У цьому випадку вагу вершини $i$ зменшують на величину $l_{ij}$, а вагу вершини $j$ збільшують на величину $l_{ij}$.

**Приклад 6.2**. Компанія забезпечує зерном із трьох зерносховищ три птахофабрики. Пропозиція зерносховищ – 100, 200 і 50 тисяч тонн зерна, а попит птахофабрик – 150, 80 і 120 тисяч тонн, відповідно. Компанія може транспортувати зерно залізничним транспортом, за винятком трьох маршрутів, де використовують автомобільний транспорт. На рис. 6.10 проілюстровано можливі маршрути між зерносховищами і птахофабриками.

Дуги (1, 4), (3, 4) і (4, 6) відповідають автомобільним маршрутам, які мають верхні та нижні границі пропускних здатностей. Інші дуги відповідають залізничному транспорту, пропускні здатності якого практично не обмежені. Вартість транспортування однієї тонни зерна в умовних грошових одиницях подано поблизу

кожної дуги в округлих дужках. У мережі на рис. 6.10 необхідно видалити з обмежень на пропускні здатності дуг нижні межі.

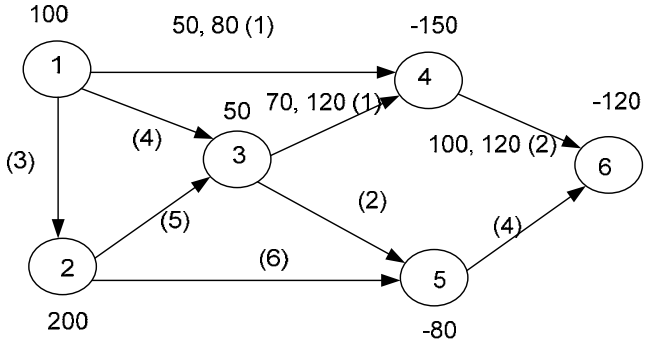



Рис. 6.10. Транспортна мережа задачі про зерносховища і птахофабрики

➢ На рис. 6.11 проілюстровано мережу після вилучення нижніх меж пропускних здатностей дуг.

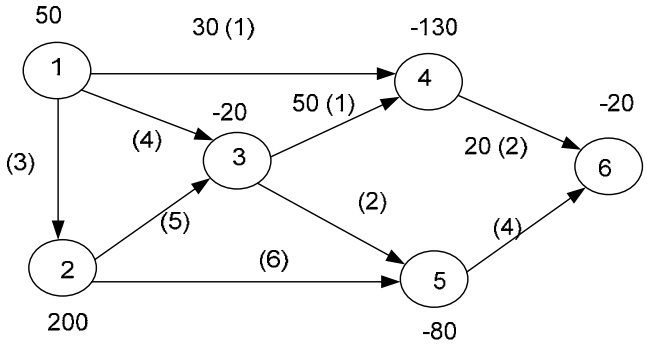



Рис. 6.11. Мережа без нижніх меж

## ? Запитання для самоперевірки

1. Сформулюйте задачу про потік мінімальної вартості.
2. Опишіть модель розподільної мережі.
3. Сформулюйте узагальнену транспортну задачу.

## 🕮 Завдання для самостійної роботи

**Завдання 6.1**. Розв’яжіть задачу про потік мінімальної вартості (рис. 6.9).

**Завдання 6.2**. Розв’яжіть задачу про потік мінімальної вартості (рис. 6.11).

# 7. КАЛЕНДАРНЕ ПЛАНУВАННЯ У МЕРЕЖАХ

**План викладу матеріалу**:

1. Головні визначення і терміни.
2. Метод критичного шляху (CPM).
3. Система PERT.
4. Аналіз та оптимізація мережевого графіка.

**Ключові терміни розділу**

- *Комплекс робіт* (*проект*)
- *Календарне планування*
- *Графічне представлення робіт*
- *Упорядкування мережі*
- *Ранній термін здійснення події*
- *Ранній термін початку роботи*
- *Пізній термін початку роботи*
- *Повний резерв часу роботи*
- *Оцінки виконання роботи*
- *Коефіцієнт напруження роботи*
- *Структурне планування*
- *Оперативне керування*
- *Мережа "роботи – події"*
- *Критичний шлях у мережі*
- *Пізній термін здійснення події*
- *Ранній термін закінчення роботи*
- *Пізній термін закінчення роботи*
- *Вільний резерв часу роботи*
- *Оцінки виконання проекту*
- *Оптимізація "час – вартість"*

## 7.1. Головні визначення і терміни

*Планування і керування у мережах* (ПКМ) – система обчислювальних і графічних методів, організаційних та контрольних заходів на основі мережевої моделі щодо планування і керування певним комплексом робіт.

Під *комплексом робіт* (*комплексом операцій* чи *проектом*) розуміють будь-яке завдання, для виконання якого необхідно здійснити велику кількість різноманітних робіт. Ці роботи залежні одна від іншої так, що виконання певних робіт не може розпочатися раніше, ніж виконані деякі інші роботи.

Планування і керування на мережах налічує такі етапи:

1. *Структурне планування*. Проект розбивають на чітко визначені роботи (операції), для яких визначають тривалості та пріоритети їхнього виконання.
2. *Календарне планування*. На основі аналізу робіт будують мережевий графік, що відображає взаємозв'язки між ними. Ви-

значають моменти початку й завершення кожної роботи й інші часові характеристики мережевого графіка. Оптимізують мережеву модель.

3. *Оперативне керування*. Використовують мережевий графік для формування звітів про виконання проекту. При цьому мережева модель може оперативно корегуватися з метою розробки нового календарного плану для частини робіт, які ще треба завершити.

Існує два способи опису проекту на рівні структурного планування: табличний і графічний. Розглянемо їх у прикладі 7.1.

**Приклад 7.1**. У таблиці 7.1 визначено роботи, їхню тривалість, а також пріоритети виконання робіт (передування робіт) деякого проекту. Графік виконання робіт проекту зображено на рис. 7.1.

➢ *Таблиця 7.1*. Перелік робіт проекту

| Робота | Роботи, які передують | Тривалість роботи (у днях) |
|---|---|---|
| *A* | – | 5 |
| *B* | – | 6 |
| *C* | *B* | 4 |
| *D* | *A, C* | 3 |

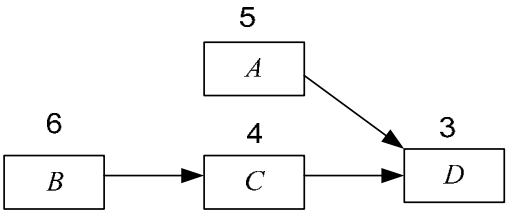



Рис. 7.1. Графік виконання робіт проекту ➢

Предметом розгляду в цьому розділу є *календарне планування*. Результати структурного планування вважатимемо відомими, а питання оперативного керування виходять за межі ДО.

*Мережева модель* – план виконання комплексу взаємозв'язаних робіт, заданого у специфічній формі, графічне зображення якої називають *мережевим графіком* (або *мережею*). Головною рисою мережевої моделі є чітке визначення усіх *часових* взаємозв'язків робіт. Головними елементами мережевої моделі є *події* та *роботи*.

Термін *робота* використовують у ПКМ у широкому значенні. По-перше, це *дійсна робота* – протяжність процесу в часі, що вимагає затрат ресурсів. По-друге, це о*чікування* – протяжність процесу в часі, що не вимагає затрат праці (наприклад, процес сушіння виробу після фарбування, твердіння бетону тощо). По-третє, це *залежність* (або *фіктивна робота*) – логічний зв'язок між двома чи декількома роботами (подіями), який не вимагає затрат праці, матеріальних ресурсів чи часу. Фіктивна робота вказує на те, що можливість однієї роботи безпосередньо залежить від результатів іншої. Тривалість фіктивної роботи дорівнює нулю.

*Подія* – це момент завершення деякого процесу, що відображає окремий етап виконання проекту. Подія може бути результатом окремої роботи чи сумарним результатом декількох робіт. Подія може здійснитися тільки тоді, коли закінчаться усі роботи, що їй передують. Наступні роботи можуть початися тільки тоді, коли подія здійсниться.

Подія *не має тривалості* (здійснюється миттєво). Кожна подія мережевої моделі має бути цілковито, чітко і всебічно визначена. Формулювання події має містити у собі опис результатів усіх робіт, які безпосередньо їй передують. Серед подій мережевої моделі виділяють *початкову* і *завершальну* події. Початкова подія не має робіт і подій, що їй передують. Завершальна подія не має наступних робіт і подій. Події на мережевому графіку зображаються *вершинами*, а роботи – *дугами*.

**Приклад 7.2**. На рис. 7.2 зображено мережевий графік для проекту з прикладу 7.1.

➢

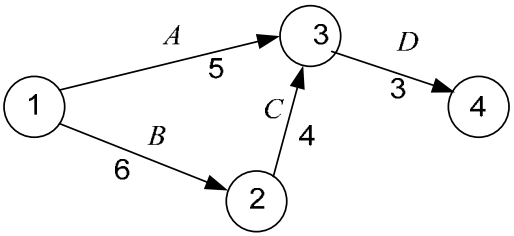



Рис. 7.2. Мережевий графік проекту (приклад 7.1)

З мережевого графіка на рис. 7.2 випливає, що роботи *A* і *B* виконуються незалежно одна від одної після здійснення початкової події 1; робота *C* – після здійснення події 2 (виконана робота *B*); а робота *D* – після настання події 3 (виконані роботи *A* і *C*). ⮜

У мережі на рис. 7.2 поблизу кожної дуги проставлено назву роботи та її тривалість. Під час наступних перетворень мережі назву роботи не вказують, оскільки це надлишкова інформація, адже роботу ідентифікує відповідна дуга: роботу *A* – дуга (1,3); роботу *B* –дуга (1, 2); роботу *C* – дуга (2, 3); роботу *D* – дуга (3, 4).

**Приклад 7.3**. Побудувати мережевий графік для проекту, перелік робіт якого наведено у таблиці 7.2.

*Таблиця 7.2*. Перелік робіт проекту

| Робота | Роботи, які передують | Тривалість роботи (у днях) |
|---|---|---|
| *A* | – | 5 |
| *B* | – | 6 |
| *C* | *B* | 4 |
| *D* | *A*, *C* | 3 |
| *E* | *C* | 4 |
| *F* | *C* | 5 |
| *G* | *D*, *E*, *F* | 3 |

➢ Мережевий графік на базі таблиці 7.2 зображено на рис. 7.3.

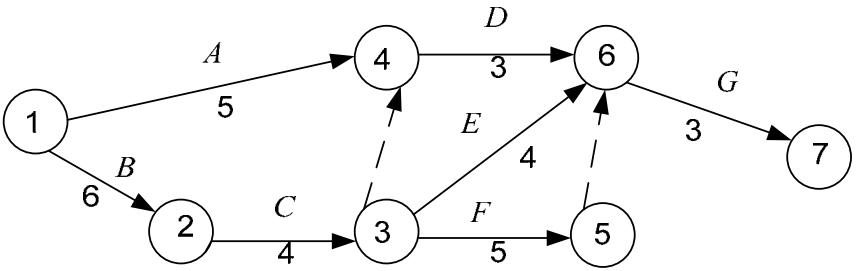



Рис. 7.3. Мережевий графік проекту (приклад 7.3)

У мережевому графіку на рис. 7.3 використано дві фіктивні роботи (3, 4) і (5, 6), їх зображено пунктиром. Ці роботи не вимагають часу на виконання і використовуються лише для того, щоб правильно відобразити зв'язки між роботами. ⮜

Мережевий графік, фактично, є *ациклічною st-мережею*, оскільки серед подій (вершин) мережевої моделі виділяють *початкову* і *завершальну* події (вершини) та існує чіткий порядок виконання робіт і кожна робота може виконуватися лише один раз. Вага дуги відповідає тривалості відповідної роботи.

При побудові мережевого графіка необхідно дотримуватися таких правил, які забезпечують утворення ациклічної *st*-мережі:

1. Не може бути вершин (за винятком завершальної вершини), з яких *не виходить* жодна дуга (робота).
2. Не може бути вершин (за винятком початкової вершини), в які *не входить* хоча б одна дуга.
3. У мережі не може бути *петель* і *циклів*. При виникненні циклу (у складних мережах це виявляється лише за допомогою комп'ютера) необхідно повернутися до початкових даних і шляхом уточнення переліку робіт домогтися його усунення.
4. Будь-які дві вершини мають зв'язуватися однією дугою.
5. У мережі має бути лише одна початкова й одна завершальна події. Якщо це не так, то домогтися бажаного можна шляхом уведення фіктивних подій і робіт.

Перші системи, що використовували мережеві графіки, були застосовані в США наприкінці 50-х років і одержали назви: CPM (Critical-Path Method, *метод критичного шляху*) і PERT (Program Evaluation & Review Technique, *метод оцінки й огляду програми*). Система CPM була вперше застосована при керуванні будівельними роботами, а система PERT – при розробці ракетних систем "Поларіс".

Систему CPM орієнтовано на розв'язування *детермінованих* задач планування на мережах, в яких тривалість кожної роботи є величиною відомою та сталою.

Систему PERT орієнтовано на аналіз проектів, в яких тривалість усіх чи деяких робіт визначити точно не вдається. Систему PERT, зазвичай, використовують під час проектування і впровадження нових систем, у яких багато робіт не мають аналогів.

## 7.2. Метод критичного шляху (CPM)

Для застосування *методу критичного шляху* попередньо необхідно упорядкувати мережевий графік. *Упорядкування* мережевого графіка полягає у такому розташуванні подій і робіт, за якого для будь-якої роботи подія, яка їй передує, розташована лівіше і має менший номер порівняно з подією, яка завершує цю роботу. Тобто в упорядкованому мережевому графіку всі роботи-стрілки спрямовані зліва направо: від подій з меншими номерами до подій з більшими номерами.

Умови упорядкування мережевого графіка відповідають умовам нумерування вершин ациклічного графа (теорема 1.2). Алгоритм нумерування вершин мережевого графіка (*подій*), який наведено у доведенні теореми 1.2, задовольняє вищевказані умови упорядкування мережевого графіка.

Одне з найважливіших понять мережевого графіка – поняття шляху. *Шлях* – будь-яка послідовність робіт, у якій кінцева подія кожної роботи збігається з початковою подією наступної за нею роботи. Серед різних шляхів мережевого графіка виділяють *повний шлях L* – будь-який шлях, початок якого збігається з початковою подією мережі, а кінець – із завершальною.

Найтриваліший повний шлях у мережевому графіку називають *критичним*. Критичними називають також роботи й події, розташовані на цьому шляху. Для визначення критичного шляху використовують алгоритм обчислення *найдовшого шляху* в навантаженому ациклічному орграфі (§ 3.5).

Критичний шлях має особливе значення у системі ПКМ, оскільки роботи цього шляху визначають час завершення проекту загалом. Для скорочення тривалості проекту необхідно, передусім, скорочувати тривалості робіт, що лежать на критичному шляху.

Розглянемо спочатку *параметри подій*. Будь-яка подія може наступити тільки за умови виконання усіх робіт, що їй передують. Отож *ранній* $t_p(j)$ *термін* здійснення $j$-ої події визначається тривалістю максимального шляху, що передує цій події:

$$t_p(j)=\max_{L_{nj}} t(L_{nj}), \qquad (7.1)$$

де $L_{nj}$ – довільний шлях від початкової події до $j$-ої події мережі.

Якщо подія $j$ має декілька попередніх шляхів, а, отже, декілька попередніх подій $i$, то ранній термін здійснення події $j$ зручно обчислювати за формулою:

$$t_p(j) = \max_{i \in \Gamma^+(j)} (t_p(i) + t_{ij}), \tag{7.2}$$

де $t_{ij}$ – час виконання роботи $(i, j)$; $\Gamma^+(j)$ – множина номерів вузлів (*подій*) мережі, з яких виходять дуги (роботи), що входять у вузол $j$.

Аналіз формул (7.1), (7.2) засвідчує, що для визначення *ранніх термінів* здійснення подій можна використати алгоритм відшукання *найдовшого шляху* в ациклічному орграфі (§ 3.5). Під час реалізації цього алгоритму знаходять максимальні шляхи від початкової події до іншої довільної події.

Затримка здійснення події $j$ стосовно свого раннього терміну *не відобразиться* на терміні здійснення завершальної події доти, доки сума терміну здійснення цієї події й тривалості максимального з *наступних* за нею шляхів не перевищить довжини критичного шляху. Отож *пізній термін* $t_n(j)$ здійснення $j$-ої події дорівнює:

$$t_n(j) = t_{кр} - \max_{L_{jз}} t(L_{jз}), \tag{7.3}$$

де $L_{iз}$ – довільний шлях, що йде за $j$-ою подією, тобто шлях від $j$-ої події до завершальної події мережі.

Якщо подія $j$ має декілька наступних шляхів, а, отже, декілька наступних подій $i$, то пізній термін здійснення події $j$ зручно знаходити за формулою:

$$t_n(j) = \min_{i \in \Gamma^-(j)} \{t_n(i) - t_{ji}\}, \tag{7.4}$$

де $t_{ji}$ – час виконання роботи $(j, i)$; $\Gamma^-(j)$ – множина номерів вузлів (*подій*) мережі, у які входять дуги (роботи), що виходять з вузла $j$.

*Резерв часу* $R(j)$ $j$-ої події визначається як різниця між пізнім і раннім термінами її здійснення:

$$R(j) = t_n(j) - t_p(j). \tag{7.5}$$

Резерв часу події демонструє, на який допустимий період часу можна затримати настання цієї події, не збільшуючи термін виконання комплексу робіт.

Критичні події резервів часу не мають, оскільки будь-яка затримка у здійсненні події, що лежить на критичному шляху, спричинить таку ж затримку в здійсненні завершальної події. Отож визначаючи ранній термін настання завершальної події мережі, ми тим самим визначаємо довжину критичного шляху, а виявляючи події з *нульовими* резервами часу, визначаємо його топологію.

**Приклад 7.4**. Побудувати мережевий графік для проекту, перелік робіт якого наведено у таблиці 7.3. Визначити у цьому графіку критичний шлях і часові параметри подій.

*Таблиця 7.3*. Перелік робіт проекту

| Робота | Зміст роботи | Роботи, які передують | Тривалість роботи (у тижнях) |
|---|---|---|---|
| *A* | Підготувати проект | – | 5 |
| *B* | Визначити орендарів | – | 6 |
| *C* | Підготувати проспект | *A* | 4 |
| *D* | Вибрати підрядника | *A* | 3 |
| *E* | Підготувати документацію | *A* | 1 |
| *F* | Отримати дозвіл на будівництво | *E* | 4 |
| *G* | Побудувати павільйони | *D, F* | 14 |
| *H* | Заключити контракти з орендарями | *B, C* | 12 |
| *I* | Вселити орендарів у павільйони | *G, H* | 2 |

➢ Під час визначення ранніх термінів здійснення подій рухаємося за графіком у напрямі зліва направо і використовуємо формулу (7.2). Довжина критичного шляху дорівнює ранньому терміну здійснення завершальної події. Для початкової події ранній термін здійснення дорівнює 0.

Під час визначення пізніх термінів здійснення подій рухаємося за графіком у зворотному напрямі (тобто справа наліво) і використовуємо формулу (7.4). Для завершальної події пізній термін здійснення події дорівнює його ранньому терміну.

Мережевий графік на базі таблиці 7.3 зображено на рис. 7.4. Позначка вершини (події): **ранній_термін, пізній_термін**.

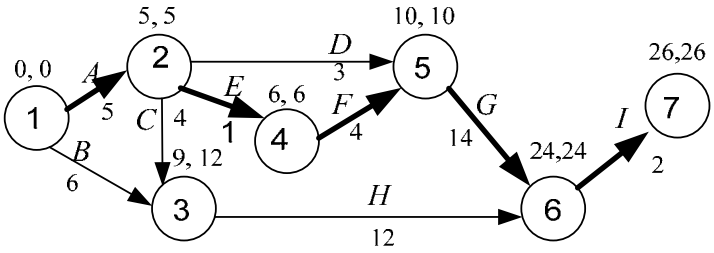



Рис. 7.4. Мережевий графік проекту (приклад 7.4)

Критичний шлях тривалості 26 зображено потовщеними лініями. Критичні події: 1, 2, 4, 5, 6, 7. Критичні роботи: *A*, *E*, *F*, *H*, *I*. ◅

Тепер перейдемо до *параметрів робіт*. Окрема робота може розпочатись (і завершитись) у ранні, пізні або інші проміжні терміни. Очевидно, що *ранній термін* $t_{рп}(i, j)$ *початку* роботи $(i, j)$ збігається з раннім терміном настання початкової події $i$, тобто

$$t_{рп}(i, j)=t_p(i). \tag{7.6}$$

*Ранній термін* $t_{рз}(i, j)$ *завершення* роботи $(i, j)$ визначає формула

$$t_{рз}(i, j)=t_p(i)+t_{ij}. \tag{7.7}$$

Робота не може закінчитись після допустимого найпізнішого терміну своєї кінцевої події $j$, тому *пізній термін* $t_{пз}(i, j)$ *завершення* роботи $(i, j)$ визначається співвідношенням

$$t_{пз}(i,\ j)=t_n(j). \tag{7.8}$$

а *пізній термін* $t_{пп}(i, j)$ *початку* цієї роботи — співвідношенням

$$t_{пп}(i, j)=t_n(j)-t_{ij}. \tag{7.9}$$

*Резерв часу* $R(L)$ *шляху* $L$ визначається як різниця між довжиною критичного шляху і довжиною даного шляху:

$$R(L)=t_{кр}-t(L). \tag{7.10}$$

Він демонструє, наскільки у сумі можна збільшити тривалості усіх робіт, що належать цьому шляхові. Якщо затягнути виконання ро-

біт, що лежать на цьому шляху, на час більше ніж $R(L)$, то критичний шлях переміститься на шлях $L$. Звідси випливає, що кожна робота на ділянці $L$, що не збігається з критичним шляхом, має резерв часу.

Для всестороннього аналізу робіт з метою наступної оптимізації використовують поняття чотирьох різновидів *резервів часу робіт*: повний резерв часу, частковий резерв часу першого роду, частковий резерв часу другого роду (або вільний резерв часу) і незалежний резерв часу. У багатьох випадках достатньо поняття *повного* і *вільного* резервів часу робіт, які й буде нами розглянуто.

*Повний резерв часу* $R_п(i, j)$ роботи $(i, j)$ демонструє, на скільки можна збільшити час виконання цієї роботи:

$$R_п(i, j)=t_п(j)-t_р(i)-t_{ij}. \tag{7.11}$$

Роботи, що лежать на *критичному шляху* (так само, як і критичні події), *резервів часу не мають*.

*Вільний резерв часу* $R_в(i, j)$ роботи $(i, j)$ – це частина повного резерву часу, на яку можна збільшити тривалість роботи, не змінивши при цьому раннього терміну її кінцевої події (або ранніх термінів початку наступних робіт):

$$R_в(i, j) = t_р(j) - t_р(i) - t_{ij}. \tag{7.12}$$

Легко довести, що

$$R_в(i, j) = R_п(i, j) - R(j). \tag{7.13}$$

Сформулюємо *правило* про можливі терміни виконання довільної некритичної роботи $(i, j)$ з часом виконання $t_{ij}$:

- якщо $R_в(i, j) = R_п(i, j)$, то зазначену роботу можна виконати на будь-якому проміжку довжиною $t_{ij}$, що цілковито належить проміжкові $(t_р(i), t_п(j))$, без порушення термінів виконання проекту;
- якщо $R_в(i, j) < R_п(i, j)$, то початок роботи $(i, j)$ можна зсунути на величину $\Delta \le R_в(i, j)$ відносно $t_р(i)$, без порушення ранніх термінів початку наступних робіт.

Отже, вільний резерв часу можна використати на збільшення тривалості зазначеної та попередньої робіт без порушення резерву часу наступних робіт.

**Приклад 7.5**. Визначити часові параметри робіт для мережевого графіка, зображеного на рис. 7.4. Дайте відповідь на такі запитання:

1. На скільки можна відкласти роботу *C*, *B*, *E* і *H*, щоб це не порушило термінів виконання проекту?
2. На скільки можна відкласти роботу *C*, *B*, *E* і *D*, щоб це не порушило ранніх термінів початку наступних робіт?

➢ Запишемо часові параметри робіт у табл. 7.4.

*Таблиця 7.4*. Часові параметри робіт

| Робота | Дуга $(i, j)$ | $t_{ij}$ | Терміни робіт і резерви часу | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | $t_{рп}$ | $t_{рз}$ | $t_{пп}$ | $t_{пз}$ | $R_п$ | $R_в$ |
| *A* | (1, 2) | 5 | 0 | 5 | 0 | 5 | 0 | 0 |
| *B* | (1, 3) | 6 | 0 | 6 | 6 | 12 | 6 | 3 |
| *C* | (2, 3) | 4 | 5 | 9 | 8 | 12 | 3 | 0 |
| *D* | (2, 5) | 3 | 5 | 8 | 7 | 10 | 2 | 2 |
| *E* | (2, 4) | 1 | 5 | 6 | 5 | 6 | 0 | 0 |
| *F* | (4, 5) | 4 | 6 | 10 | 6 | 10 | 0 | 0 |
| *G* | (5, 6) | 14 | 10 | 24 | 10 | 24 | 0 | 0 |
| *H* | (3, 6) | 12 | 9 | 21 | 12 | 24 | 3 | 3 |
| *I* | (6, 7) | 2 | 24 | 26 | 24 | 26 | 0 | 0 |

*Відповіді*. 1. Можна відкласти (на тижні): *C* – 3, *B* – 6, *E* – 0, *H* – 3.
2. Можна відкласти (на тижні): *C* – 0, *B* – 3, *E* – 0, *D* – 2. ◅

## 7.3. Система PERT

Під час визначення тимчасових параметрів мережевого графіка дотепер передбачалося, що час виконання кожної роботи чітко визначено. Таке припущення реально виконується нечасто.

Найчастіше тривалість роботи заздалегідь невідома і може приймати певне можливе значення з деякою ймовірністю. Тобто тривалість роботи $t(i, j)$ є випадковою величиною, що характеризується своїм законом розподілу з такими числовими характеристиками, як *середнє значення* $\bar{t}(i, j)$ і *дисперсія* $\sigma^2(i, j)$.

Аналіз великої кількості статистичних даних (хронометражі часу реалізації окремих робіт, нормативні дані і т.д.) засвідчує, що для оцінки середньої тривалості робіт найпридатніший β-*розподіл* [10, 15, 17].

Для визначення числових характеристик $\bar{t}(i, j)$ і $\sigma^2(i, j)$ цього розподілу для роботи $(i, j)$ на підставі опитування відповідальних виконавців проекту й експертів визначають три часові оцінки:

- *оптимістичну оцінку* $t_o(i, j)$ тривалості роботи $(i, j)$;
- *песимістичну оцінку* $t_n(i, j)$ тривалості роботи $(i, j)$;
- *найімовірнішу оцінку* $t_{ні}(i, j)$ тривалості роботи $(i, j)$.

Припущення про β-розподіл тривалості роботи $(i, j)$ дає змогу отримати такі оцінки її числових характеристик:

$$\bar{t}(i, j) = \frac{t_o(i, j) + 4t_{ні}(i, j) + t_n(i, j)}{6}; \qquad (7.14)$$

$$\sigma^2(i, j) = \left[\frac{t_n(i, j) - t_o(i, j)}{6}\right]^2. \qquad (7.15)$$

Якщо фахівцям складно оцінити $t_{ні}(i, j)$, тоді використовують спрощену (і менш точну) оцінку середньої тривалості роботи $(i, j)$ на підставі лише двох часових оцінок $t_o(i, j)$ *і* $t_n(i, j)$:

$$\bar{t}(i, j) = \frac{2t_o(i, j) + 3t_n(i, j)}{5}. \qquad (7.16)$$

За значенням $\bar{t}(i, j)$ і $\sigma^2(i, j)$ можна визначати часові параметри мережевого графіка й оцінити їхню надійність. Наприклад, за досить великої кількості робіт, що належать шляху $L$, і за виконання деяких узагальнених умов можна застосувати центральну граничну теорему Ляпунова, на підставі якої можна стверджувати, що загальна довжина шляху $L$ має нормальний закон розподілу із середнім значенням $\bar{t}(L)$ і дисперсією $\sigma^2(L)$, які обчислюють за формулами:

$$\bar{t}(L) = \sum_{i,j} \bar{t}(i, j); \quad \sigma^2(L) = \sum_{i,j} \sigma^2(i, j). \qquad (7.17)$$

Нехай мережевий графік на рис. 7.4 зображає мережу з випадковими тривалостями робіт і числа над роботами-стрілками задають середні значення $\bar{t}\,(i, j)$ тривалості відповідних робіт, обчислені за формулою (7.14) або (7.16). Відомі усі дисперсії $\sigma^2(i, j)$, обчислені за формулою (7.15).

У цьому випадку часові параметри мережевого графіка: довжина критичного шляху, ранні та пізні терміни здійснення подій, резерви часу подій тощо – будуть такі ж, як і визначені у прикладах 7.4 і 7.5. Однак тепер вони є *середніми* значеннями відповідних випадкових величин: середньою довжиною критичного шляху $\bar{t}_{кр}$, середнім значенням раннього терміну завершення події $\bar{t}_p\,(i)$ тощо.

Важливим моментом аналізу мереж з випадковими тривалостями робіт стає оцінка ймовірності того, що термін виконання проекту $\bar{t}_{кр}$ не перевищить заданого директивного терміну $T$. Вважаючи $\bar{t}_{кр}$ випадковою величиною, що має нормальний закон розподілу, отримаємо:

$$P(\bar{t}_{кр} \le T) = \frac{1}{2} + \frac{1}{2}\Phi\left(\frac{T - \bar{t}_{кр}}{\sigma_{кр}}\right), \tag{7.18}$$

$$\sigma_{кр} = \sqrt{\sigma_{кр}^2}\,, \tag{7.19}$$

де $\Phi(z)$ –інтеграл ймовірностей Лапласа ($z = (T - \bar{t}_{кр}) / \sigma_{кр}$.

Якщо значення $P(\bar{t}_{кр} \le T)$ є малим (наприклад, меншим за 0,3), то небезпека зриву заданого терміну виконання комплексу робіт є великою, необхідно вживати додаткових заходів (перерозподіл ресурсів у мережі, перегляд складу робіт і подій тощо). Якщо значення $P(\bar{t}_{кр} \le T)$ є значним (наприклад, більшим за 0,8), то з достатнім ступенем надійності можна прогнозувати виконання проекту у встановлений термін.

Розглянуті дотепер мережі були *детермінованими*, хоча роботи у них могли характеризуватися не тільки детермінованими, але й випадковими тривалостями.

Водночас існують проекти, коли на деяких етапах той або інший комплекс наступних робіт залежить від невідомого заздалегідь результату. Який із цих комплексів робіт фактично виконуватиметься, можна передбачити лише з деякою ймовірністю. Мережі, що відображають такі проекти, називають *стохастичними*.

Наприклад, може бути передбачено декілька варіантів продовження дослідження залежно від отриманих дослідних даних або кілька варіантів будівництва підприємств різної потужності з обробки сировини залежно від результатів розвідки запасів цієї сировини.

У свою чергу стохастичні мережі можуть характеризуватися детермінованими або випадковими тривалостями.

### 7.4. Аналіз та оптимізація мережевого графіка

Після знаходження критичного шляху й резервів часу робіт та оцінки ймовірності виконання проекту в заданий термін необхідно здійснити всебічний аналіз мережевого графіка і вжити заходів щодо його *оптимізації*.

Аналіз мережевого графіка починається з аналізу топології мережі, що налічує контроль побудови мережевого графіка, доцільності вибору робіт, ступеня їхньої відокремленості.

Визначити ступінь важкості виконання у термін кожної групи робіт некритичного шляху можна за допомогою *коефіцієнта напруження* робіт $K_н$ роботи $(i, j)$, який визначає така формула:

$$K_н(i, j) = \frac{t(L_{\max}) - t'_{\text{кр}}}{t_{\text{кр}} - t'_{\text{кр}}}, \qquad (7.20)$$

де $t(L_{\max})$ – довжина максимального шляху, що проходить через роботу $(i, j)$; $t'_{кр}$ – довжина відрізка цього шляху, що збігається з критичним шляхом.

Коефіцієнт напруження може змінюватися у межах від 0 до 1 (для робіт критичного шляху). Чим ближче до 1 коефіцієнт напруження $K_н(i, j)$, тим складніше виконати дану роботу у встановлений термін. Чим ближче $K_н(i, j)$ до нуля, тим більшим резервом володіє максимальний шлях, що проходить через дану роботу.

**Приклад 7.6**. Знайти коефіцієнт напруження роботи (1, 4) для мережевого графіка (рис. 7.5).

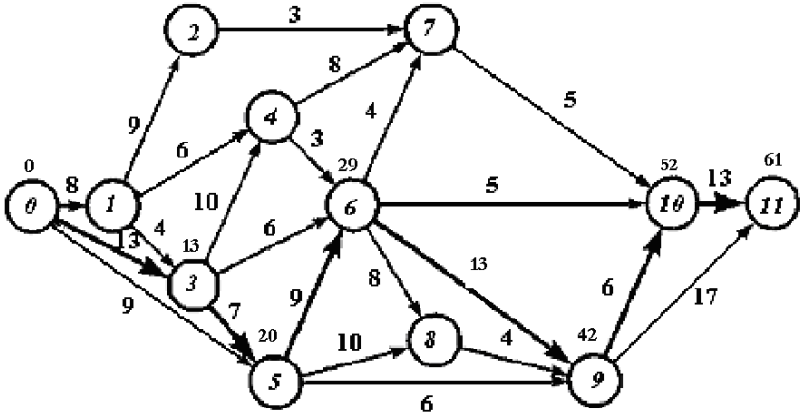



Рис. 7.5. Мережевий графік

➢ Критичний шлях тривалості 61 (одиниць часу): 0→1→3→5→→6→9→10→11. Максимальний шлях, що проходить через роботу (1, 4) – це шлях $L_4$ (0→1→4→6→9→10→11) з тривалістю $t(L_{\max})=t(L_4)=49$ (одиниць часу). Максимальний шлях $L_4$, збігається із критичним на відрізку 6→9→10→11 тривалістю 13+6+13=32 (одиниць часу). Використовуючи формулу (7.20), визначимо:

$$K_н(1,4)=\frac{49-32}{61-32}=\frac{17}{29}\approx 0,59. \qquad \prec$$

Обчислені коефіцієнти напруження дають змогу додатково класифікувати роботи за зонами. Залежно від величини $K_н(i, j)$ вирізняють три зони: *критичну зону* ($K_н(i, j) > 0,8$); *підкритичну зону* ($0,6 \le K_н(i, j) < 0,8$) і *резервну зону* ($K_н(i, j) < 0,6$).

***Оптимізація*** *мережевого графіка* – процес поліпшення організації виконання комплексу робіт з урахуванням терміну його виконання. Оптимізацію здійснюють з метою скорочення довжини критичного шляху, вирівнювання коефіцієнтів напруження робіт, раціонального використання ресурсів.

Передусім відшукують заходи щодо скорочення тривалості робіт, що перебувають на критичному шляху. Цього досягають:

- шляхом перерозподілу усіх видів ресурсів із менше напружених зон у зони, що поєднують найнапруженіші роботи;
- шляхом скорочення трудомісткості критичних робіт завдяки передачі частини робіт на інші шляхи, що мають резерви часу;
- паралельним виконанням робіт критичного шляху;
- зміною складу робіт і структури мережі.

Часто оптимізацію мережевого графіка реалізують методом "*час – вартість*", припускаючи, що зменшення тривалості роботи пропорційно зростанню її вартості. Тривалість роботи $(i, j)$ може перебувати в межах $a(i, j) \le t(i, j) \le b(i, j)$, де $a(i, j)$ – мінімально можлива тривалість роботи $(i, j)$, яку тільки можна здійснити в умовах розробки; $b(i, j)$ – нормальна тривалість роботи $(i, j)$.

При цьому вартість $c(i, j)$ роботи $(i, j)$ набуває значення у межах від $c_{\min}(i, j)$ до $c_{\max}(i, j)$, де $c_{\min}(i, j)$ – нормальна тривалість роботи, $c_{\max}(i, j)$ – екстрена тривалість роботи.

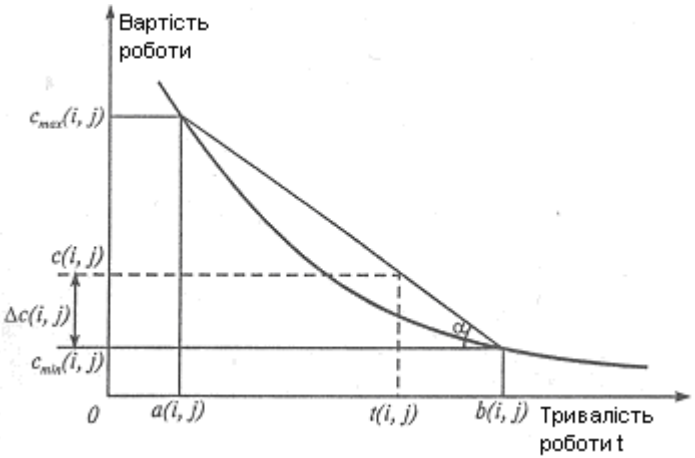



Рис. 7.6. Визначення витрат на прискорення роботи $(i, j)$

Використовуючи апроксимацію за прямою (рис. 7.6), можна легко знайти зміну вартості роботи $\Delta c(i, j)$ при скороченні її тривалості на величину $\Delta c(i, j) = [b(i, j) - t(i, j)]h(i, j)$. Величина $h(i, j)$, яка показує *витрати на прискорення роботи* $(i, j)$ (порівняно з нормальною тривалістю) *на одиницю часу*, дорівнює тангенсу кута $\alpha$ нахилу апроксимуючої прямої:

$$h(i, j) = \operatorname{tg}\alpha = \frac{c_{\max}(i, j) - c_{\min}(i, j)}{b(i, j) - a(i, j)}.$$

Оптимізація мережевого графіка з урахуванням вартості використовує резерви часу робіт. Тривалість кожної роботи, що має резерв часу, збільшують доти, доки не буде вичерпано цей резерв або доки не буде досягнуто значення $b(i, j)$. У цьому випадку початкова вартість виконання проекту $C = \sum_{i,j} c(i, j)$ зменшиться на величину:

$$C = \sum_{i,j} \Delta c(i, j) = \sum_{i,j} [b[i, j] - t(i, j)]h(i, j).$$

Тривалість кожної роботи $t(i, j)$ доцільно збільшити на величину такого резерву часу, щоб не змінити ранні (очікувані) терміни настання усіх подій мережі, тобто на величину вільного резерву часу $R_в(i, j)$.

**Приклад 7.7**. Здійснити оптимізацію мережевого графіка (рис. 7.5). Граничні значення тривалостей робіт $a(i, j)$ та $b(i, j)$, їхньої вартості $c(i, j)$, коефіцієнти витрат на прискорення робіт $h(i, j)$ наведено в таблиці 7.6.

*Примітки*: 1. У таблиці 7.6 представлено параметри лише тих робіт, які мають вільний резерв часу.

2. Вартості $c(i, j)$ інших робіт: $c(0,1) = 50$; $c(0,3) = 45$; $c(1,2) = 82$; $c(3,4) = 55$; $c(3,5) = 72$; $c(5,6) = 30$; $c(6,7) = 26$; $c(6,9) = 75$; $c(6,8) = 42$; $c(9,10) = 35$; $c(10,11) = 10$ (умовних одиниць).

*Таблиця 7.6.* Умови та розв'язання прикладу 7.7

| № з/п | Робота $(i, j)$ | Тривалість роботи (діб) | | | $R_в(i, j)$ | $c(i, j)$ | $h(i, j)$ | $\Delta C$ |
|---|---|---|---|---|---|---|---|---|
| | | $a(i, j)$ | $t(i, j)$ | $b(i, j)$ | | | | |
| 1 | (0,5) | 5 | 9 | 14 | 11 | 60 | 8 | 40 |
| 2 | (1,4) | 4 | 6 | 10 | 9 | 28 | 4 | 16 |
| 3 | (1,3) | 3 | 4 | 6 | 1 | 37 | 12 | 12 |
| 4 | (2,7) | 2 | 3 | 7 | 13 | 86 | 6 | 24 |
| 5 | (3,6) | 4 | 6 | 9 | 10 | 92 | 10 | 30 |
| 6 | (4,7) | 3 | 8 | 14 | 2 | 48 | 5 | 10 |
| 7 | (4,6) | 1 | 3 | 6 | 3 | 64 | 12 | 36 |
| 8 | (5,8) | 5 | 10 | 18 | 7 | 15 | 1 | 7 |
| 9 | (5,9) | 3 | 6 | 12 | 16 | 86 | 7 | 42 |
| 10 | (6,10) | 2 | 5 | 10 | 14 | 44 | 5 | 25 |
| 11 | (7,10) | 1 | 5 | 15 | 10 | 74 | 4 | 40 |
| 12 | (8,9) | 2 | 4 | 8 | 1 | 20 | 3 | 3 |
| 13 | (9,11) | 11 | 17 | 23 | 2 | 40 | 4 | 8 |
| | | | | | Разом | 694 | — | 293 |

➢ Ненульові значення вільних резервів часу робіт ( $R_в(i, j)$ ), а також результати оптимізації мережі наведено у таблиці 7.7. У таблиці 7.7 підкреслено ті роботи, вільні резерви часу яких цілковито використані на збільшення їхньої тривалості.

Вартість первісного варіанта мережевого графіка (первісного плану) дорівнює сумі вартостей усіх робіт, у тім числі робіт, що не мають резервів і не включені у таблицю 7.7:

$$C = 694 + 50 + 45 + \ldots + 35 + 10 = 1216 \text{ (умовних одиниць).}$$

Вартість нового плану дорівнює $C - \Delta C = 1216 - 293 = 923$ (умовних одиниць), тобто зменшилася майже на 25%. Новий оптимізований мережевий графік зображено на рис. 7.7. Легко переко-

натися у тому, що з'явилися нові критичні шляхи довжиною $t_{кр} = 61$, наприклад:

$$0 \to 1 \to 3 \to 4 \to 7 \to 10 \to 11;$$
$$0 \to 3 \to 5 \to 8 \to 9 \to 11;$$
$$0 \to 1 \to 3 \to 4 \to 6 \to 7 \to 10 \to 11;$$
$$0 \to 3 \to 5 \to 6 \to 8 \to 9 \to 11 \text{ і т. д.}$$

Можна проілюструвати, що у новому варіанті мережевого графіка (рис. 7.7) з 64-х повних шляхів 28 – критичні. Якби верхні границі тривалостей робіт передбачали цілковите використання резерву часу всіх робіт, представлених у таблиці 7.6, то в новому плані усі повні шляхи були б критичними.

Отже, у результаті оптимізації мережі ми прийшли до плану, що дає змогу виконати комплекс робіт у термін $t_{кр} = 61$ за мінімальної його вартості $C = 923$.

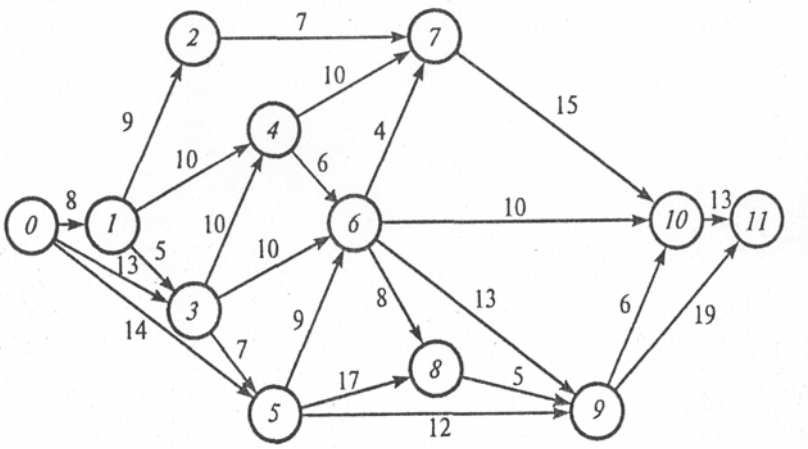



Рис. 7.7. Новий варіант мережевого графіка

Зауважимо, що при лінійній залежності вартості робіт від їхньої тривалості задачу побудови оптимального мережевого графіка можна сформулювати як задачу лінійного програмування, у

якій необхідно мінімізувати вартість виконання проекту при двох групах обмежень.

Перша група обмежень засвідчує, що тривалість кожної роботи має знаходитися у заданих межах. Друга група обмежень вимагає, щоб тривалість будь-якого повного шляху мережевого графіка не перевищувала встановленого директивного терміну виконання проекту. Однак вирішувати такі задачі класичними чи методами лінійного програмування, зазвичай, неефективно.

### ? Запитання для самоперевірки

1. Що таке комплекс робіт?
2. Чим відрізняється структурне планування від календарного?
3. Які головні завдання структурного планування?
4. Як описують проект на рівні структурного планування?
5. Які головні завдання календарного планування?
6. Які завдання вирішує оперативне керування?
7. Що таке подія?
8. Як позначають подію на мережевому графіку?
9. Які види робіт виділяють у ПКМ?
10. Опишіть правила побудови мережі “*роботи – події*”.
11. У чому полягає упорядкування мережі?
12. Що таке шлях у мережі?
13. Що таке критичний шлях у мережі? Як його знайти?
14. Що таке ранній термін здійснення події?
15. Що таке пізній термін здійснення події?
16. Що таке резерв часу події?
17. Що таке ранній термін початку роботи?
18. Що таке пізній термін початку роботи?
19. Що таке ранній термін закінчення роботи?
20. Що таке пізній термін закінчення роботи?
21. Що таке повний резерв часу роботи?
22. Що таке вільний резерв часу роботи?
23. Для чого використовують повний резерв часу роботи?
24. Для чого використовують вільний резерв часу роботи?
25. Що таке коефіцієнт напруження роботи?

26. У чому полягає оптимізація мережевого графіка?
27. Які заходи приймаються щодо скорочення тривалості робіт, що перебувають на критичному шляху?
28. До чого прирівнюють витрати на прискорення роботи?
29. Опишіть метод оптимізації проекту "*час – вартість*".

## 🕮 Завдання для самостійної роботи

**Завдання 7.1**. Необхідно зробити дерев'яну шухляду (роботу виконує один чоловік). Для цього доведеться розмітити дошки відповідно до розмірів шухляди (15 хв.); розрізати дошки (12 хв.); склеїти частини шухляди (40 хв.); прибити до кришки шухляди петлі (8 хв.); почекати, доки шухляда висохне, і витерти її (15 хв.); петлі (із кришкою) прибити до шухляди (10 хв.). У дужках зазначено тривалість робіт.

Побудувати мережевий графік. Знайти час виконання комплексу робіт, часові характеристики подій і робіт.

**Завдання 7.2**. Необхідно замінити колесо машини (роботу виконують два чоловіка). Дістати з багажника домкрат і інструменти (40 с); зняти диск із колеса (30 с); звільнити колесо (50 с); поставити домкрат під машину (26 с); підняти машину (20 с); з багажника взяти запасне колесо (25 с); зняти гайки і колесо (20 с); установити запасне колесо на вісь (10 с); загвинтити (не сильно) гайки на осі (15 с); опустити машину і зібрати домкрат (25 с); поставити домкрат назад у багажник (10 с); загвинтити гайки на осі до кінця (12 с); покласти пошкоджене колесо й інструменти у багажник (40 с); поставити на місце диск колеса (10 с). У дужках зазначено тривалість робіт.

Побудувати мережевий графік. Знайти час виконання комплексу робіт, тимчасові характеристики подій і робіт.

**Завдання 7.3**. Для мережевого графіка (рис. 7.8) знайти усі повні шляхи, критичний шлях; розрахувати ранні та пізні терміни здійснення подій, початки і закінчення робіт; визначити резерви часу повних шляхів і подій, резерви часу (повний та вільний) робіт і коефіцієнти напруження робіт.

**Завдання 7.4**. Як зміниться термін виконання проекту (рис. 7.8), резерви часу робіт і подій, коефіцієнти напруження робіт, якщо збільшити тривалість роботи $t(9,10)$ на величину: а) $R_n(9,10)$; б) $R_в(9,10)$.

**Завдання 7.5**. Як зміниться термін виконання проекту (рис. 7.8), резерви часу робіт і подій, коефіцієнти напруження робіт, якщо тривалість кожної

роботи $t(i, j)$ збільшити на величину: а) $R_п(i, j)$ б) $R_в(i, j)$. Для випадків а), і б) знайти усі критичні шляхи мережевого графіка.

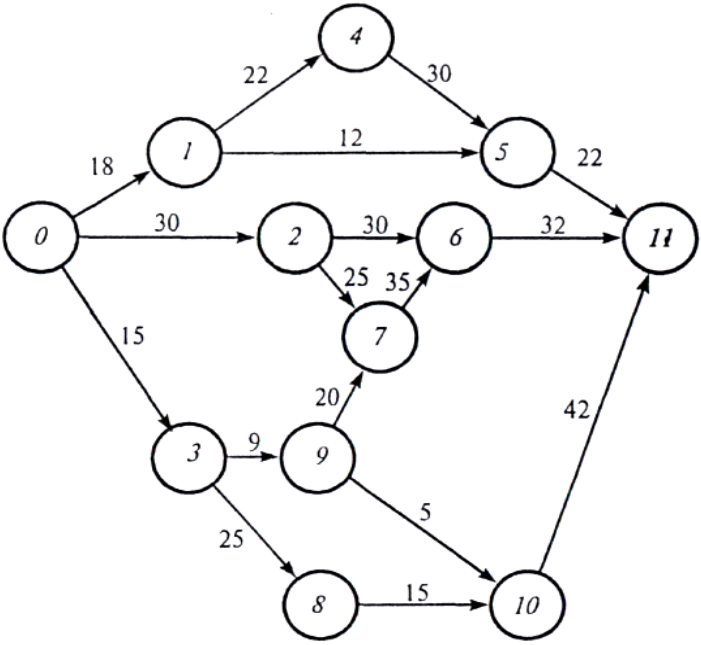



Рис. 7.8. Мережевий графік (завдання 7.3 – 7.5)

**Завдання 7.6**. У таблиці 7.6 зазначено оцінки часу виконання робіт мережевого графіка, надані відповідальними виконавцями та експертами.

*Таблиця 7.6*. Умови завдання 7.6

| № з/п | Робота $(i, j)$ | Оцінка часу виконання роботи, доби | | |
|---|---|---|---|---|
| | | оптимістична $t_o(i, j)$ | песимістична $t_п(i, j)$ | найімовірніша $t_{ні}(i, j)$ |
| *1* | *2* | *3* | *4* | *5* |
| 1 | (1, 2) | 5 | 9 | 6 |
| 2 | (1, 3) | 2 | 7 | 5 |
| 3 | (1, 4) | 4 | 10 | 8 |

*Закінчення табл. 7.6*

| *1* | *2* | *3* | *4* | *5* |
|---|---|---|---|---|
| 4 | (3, 4) | 9 | 14 | 11 |
| 5 | (2, 5) | 7 | 13 | 10 |
| 6 | (4, 5) | 1 | 4 | 3 |

Необхідно:

а) побудувати мережевий графік;

б) визначити очікувані значення тривалості робіт;

в) визначити критичний шлях та його довжину.

Вважаючи, що час критичного шляху розподілений за нормальним законом, знайти ймовірність того, що термін виконання комплексу робіт не перевищить 17 діб.

**Завдання 7.7**. У таблиці 7.7 зазначено оцінки часу виконання і вартості робіт.

*Таблиця 7.7*. Умови завдання 7.7

| № з/п | Робота $(i, j)$ | Тривалість роботи (доба) | | $h(i, j)$ | $C = (i, j)$ при $t(i, j) = b(i, j)$ |
|---|---|---|---|---|---|
| | | $a(i, j)$ | $b(i, j)$ | | |
| *1* | *2* | *3* | *4* | *5* | *6* |
| 1 | (0, 1) | 10 | 20 | 6 | 35 |
| 2 | (0,2) | 12 | 32 | 3 | 50 |
| 3 | (1,2) | 2 | 12 | 3 | 15 |
| 4 | (1,3) | 2 | 7 | 8 | 10 |
| 5 | (2,7) | 2 | 7 | 3 | 10 |
| 6 | (3,4) | 16 | 26 | 2 | 50 |
| 7 | (3,5) | 8 | 13 | 6 | 15 |
| 8 | (4,6) | 12 | 22 | 4 | 40 |

*Закінчення табл. 7.7*

| *1* | *2* | *3* | *4* | *5* | *6* |
|---|---|---|---|---|---|
| 9 | (5,6) | 20 | 25 | 4 | 30 |
| 10 | (6,7) | 8 | 13 | 5 | 25 |
| 11 | (7,8) | 6 | 11 | 9 | 20 |
| | | | | Разом | 300 |

За даними таблиці 7.7 необхідно:

а) побудувати мережний графік;

б) визначити критичний шлях і вартість проекту за мінімально можливих значень тривалості усіх робіт;

в) план з максимальними значеннями тривалості усіх робіт і, відповідно, мінімальною вартістю проекту.

# СПИСОК ЛІТЕРАТУРИ


1. *Ахо А., Хопкрофт Дж., Ульман Дж.* Структуры данных и алгоритмы. – М.: Издательский дом "Вильямс", 2003.
2. *Бартіш М. Я.* Методи оптимізації: Навчальний посібник. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2006.
3. *Вентцель Е. С.* Исследование операций. Задачи, принципы, методология: Учеб. пособие для вузов. – М.: Дрофа, 2004.
4. *Волков И. К., Загоруйко Е. А.* Исследование операций: Учебник. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2002.
5. *Глибовець М. М.* Основи комп'ютерних алгоритмів. – К.: Вид. дім "КМ Академія", 2003.
6. *Зайченко Ю. П.* Дослідження операцій: Підручник. – К.: ЗАТ "ВІПОЛ", 2000.
7. *Катренко А. В.* Дослідження операцій: Підручник. – Львів: "Магнолія Плюс", 2004.
8. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНО, 1999.
9. *Костевич Л. С.* Математическое программирование: Информ. технологии оптимальных решений: Учеб. пособие. – Мн.: Новое знание, 2003.
10. *Кремер Н. Ш., Путко Б. А., Тришин И. М., Фридман М. Н.* Исследование операций в экономике: Учеб. пособие. – М.: ЮНИТИ, 2004.
11. *Макконнелл Дж.* Основы современных алгоритмов: Учеб. пособие. – – М.: Техносфера, 2004.
12. *Машина Н. І.* Математичні методи в економіці: Навчальний посібник. – К.: Центр навчальної літератури, 2003..
13. *Наконечний С. І., Савіна С. С.* Математичне програмування: Навчальний посібник. – К.: КНЕУ, 2003.
14. *Нікольський Ю. В., Пасічник В. В., Щербина Ю. М.* Дискретна математика: Підручник. – К.: Видавнича група BHV, 2007.
15. *Сеньо П. С.* Випадкові процеси: Підручник. – Львів: Компакт-ЛВ, 2006.
16. *Седжвик Р.* Фундаментальные алгоритмы на С++. Алгоритмы на графах. – СПб.: ООО "ДиаСофтЮП", 2002.
17. *Таха Х.* Введение в исследование операций, 7-е издание. – М.: Издательский дом "Вильямс", 2005.
18. *Черняк А. А., Новиков В. А., Мельников О. И., Кузнецов А. В.* Математика для экономистов на базе MathCad: Учеб. пособие. – СПб.: БХВ–Петербург, 2003.

Навчальне видання

Бартіш Михайло Ярославович
Дудзяний Ігор Михайлович

# Дослідження операцій
# Частина 2. Алгоритми оптимізації на графах

Редактор *І. Лоїк*
Технічний редактор *С. Сеник*
Комп'ютерне макетування *І. Дудзяний*

Підп. до друку 29.06.2007 р. Формат 60×84/16. Папір друк.
Друк на різогр. Гарнітура Times. Умовн. друк. арк.6,9. Обл.-вид. арк. 7,3.
Тираж 250 прим. Зам.

Видавничий центр Львівського національного університету імені Івана Франка.
79000 Львів, вул. Дорошенка, 41