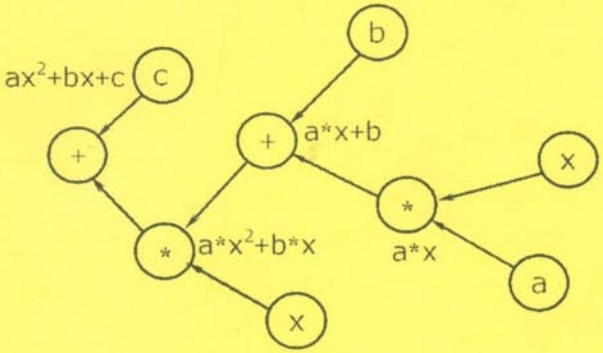
В. М. Дубовой
О. Д. Никитенко

СПЕЦІАЛЬНІ РОЗДІЛИ МАТЕМАТИКИ



Міністерство освіти і науки України
Вінницький національний технічний університет

В.М. Дубовой
О.Д. Никитенко

СПЕЦІАЛЬНІ РОЗДІЛИ МАТЕМАТИКИ

Затверджено Вченою радою Вінницького національного технічного університету як навчальний посібник для студентів напряму підготовки "Системна інженерія" всіх спеціальностей. Протокол № 11 від 2 липня 2007р.

Вінниця ВНТУ 2007

УДК 681.518
Д 54

Рецензенти:

Р.Н. Квєтний, доктор технічних наук професор
В.М. Лисогор, доктор технічних наук професор
В.О. Поджаренко, доктор технічних наук професор

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України

Дубовой В.М., Никитенко О.Д.

Д54 **Спеціальні розділи математики.** Навчальний посібник. – Вінниця: ВНТУ, 2007. – 165 с.

В посібнику узагальнюються питання математичних основ комп'ютерних технологій управління. Матеріал посібника ґрунтується на загальному курсі математики, але більшість його розділів може вивчатися самостійно. Посібник розроблений у відповідності з планом кафедри та програмами дисциплін "Спеціальні розділи математики", "Дискретна математика", "Математична статистика".

УДК 681.518

ЗМІСТ

ВСТУП

Математична підготовка відіграє дуже важливу роль у професійній кваліфікації спеціаліста з комп'ютеризованих систем управління і автоматики (СУА). Це зумовлено широким застосуванням комп'ютерних технологій як в процесі проектування СУА, так і в самих системах для реалізації складних алгоритмів управління.

Класичний курс математики, що з невеликими модифікаціями викладається студентам всіх інженерних спеціальностей, є недостатнім для сучасного фахівця СУА. Тому метою даного посібника є поглиблення математичної підготовки в напрямках математичних основ комп'ютерних технологій управління.

Спрямованість посібника на підготовку студентів за напрямом "Комп'ютеризовані системи управління і автоматики" зумовлює включення до посібника окремих питань щодо реалізації математичних методів у комп'ютерних програмах.

Матеріал посібника ґрунтується на загальному курсі математики, але більшість його розділів може вивчатися самостійно. Разом з тим слід зауважити, що обмежений обсяг посібника не дає можливості викласти весь матеріал у повному обсязі, з необхідними в курсі математики доведеннями і обґрунтуваннями. Він є лише методичним путівником по складному просторі сучасних математичних методів системної інженерії.

При підготовці посібника автори ставили за мету не тільки роз'яснити основні математичні поняття системної інженерії, але й підкреслити їх тісний взаємний зв'язок, який ховається за зовнішньою різницею задач та термінів.

1. МНОЖИНИ

Теорія множин - розділ математики, в якому вивчаються загальні властивості множин. Теорія множин лежить в основі більшості математичних дисциплін; вона вплинула на розуміння предмета самої математики.

Даний розділ підготовлений з використанням матеріалів [1, 2, 3, 18, 21].

1.1. Основні означення

Множина – це сукупність елементів, об'єднаних у відповідності до деякого правила, які знаходяться у певних відношеннях між собою та з іншими елементами, що не належать множині. Це не є в повному змісті логічним означенням поняття множини, а всього лише поясненням. Множина - це, мабуть, найширше поняття математики й логіки.

Сучасна теорія множин ґрунтується на ідеях і роботах Кантора. «Множество есть многое, мыслимое нами как единое» (Г. Кантор).

Множини позначаються: $A=\{a_1, a_2, \ldots, a_n\}$.

Множина, що не містить жодного елемента, називається ***пустою множиною*** і позначається $\varnothing$.

Виділяють декілька основних типів **відношень множин та їх елементів:**

1) ***відношення належності*** елемента множині позначається: $a_i \in A$, $b_j \notin A$;

2) якщо всі елементи множини B є елементами множини A, то множина B називається ***підмножиною множини*** A, а множина A називається ***універсальною множиною*** відносно множини B. Якщо множина B належить множині A, то має місце ***відношення включення:*** $B \subset A$;

3) ***відношення тотожності*** $A=B$ (якщо множини містять однакові елементи). Одночасне виконання співвідношення $A \subset B$ й $B \subset A$

можливо тільки при $A=B$. І навпаки $A=B$, якщо $A \subset B$ й $B \subset A$. Це може служити визначенням рівності двох множин через відношення включення.

Покриття універсальної множини – це сукупність її підмножин, для якої $U = \bigcup_i A_i$.

Якщо ж сукупність підмножин покриття множини A така, що $A_i \cap A_j = \varnothing$ при $i \neq j$, то сукупність $\{A_1, ..., A_n, ...,\}$ називається ***розбиттям*** множини A, а підмножини A, — ***класами*** цього розбиття, $i = 1, 2, ..., n, ...$

У множинах не визначене відношення порядку, тобто, якщо є множина $\{1,2,3\}$, то вона тотожна множині $\{3,1,2\}$ і $\{1,3,2\}$.

Потужністю або ***кардинальним числом*** множини A називають кількість елементів множини A, і позначають $|A|$ або $\#A$.

Множини A і B називаються ***рівнопотужними,*** якщо між їх елементами існує взаємно однозначна відповідність. Рівнопотужні множини часто називають ***еквівалентними.*** Це означає, що відношення множин рефлексивні (кожна множина рівнопотужна сама собі), симетричні (якщо A рівнопотужна B, то й B рівнопотужна A) і транзитивні (якщо A рівнопотужна B й B рівнопотужна C, то A рівнопотужна C).

Множина підмножин універсальної множини називається ***степенем множини***. Степінь множини $P(A)=2^n$, де n – потужність множини (кількість елементів).

Множини поділяються на скінченні та нескінченні. ***Скінченна множина*** - це така множина, яка містить скінченну кількість елементів, тобто має скінченну потужність. ***Нескінченна множина*** - це така множина, яка містить нескінченну кількість елементів, тобто має нескінченну потужність.

Розглянемо множину $C = \{(a, b) \mid a \in A,\ b \in B\}$, яка складається з усіх попарних добутків елементів A і B. Ця множина називається ***декартовим добутком множин*** A і B та позначається $A \times B$. Якщо множини A і B скінченні і складаються відповідно з m і n елементів, то очевидно, що C

складається з mn елементів. Елементи декартового добутку називають ще ***кортежами.***

1.2. Способи задавання множини

Виділяють такі способи задання множини:

- переліком елементів;
- за допомогою правила належності елементів множині;
- за допомогою породжувальної процедури.

Множину A можна задати простим перерахуванням його елементів $A = \{a_1, a_2, \ldots, a_n\}$. Наприклад, специфікація задає множину деталей виробу, каталог — множина книг у бібліотеці. Але цей спосіб непридатний для задання нескінченних множин і навіть у випадку кінцевих множин часто практично не реалізується.

Інший спосіб задання множини полягає у виборі елементів x, які мають певну властивість $P(x)$ (умова від x). Звичайно $P(x)$ — це висловлення, у якому щось стверджується про x, або деяка функція змінної x. Множина, задана за допомогою форми $P(x)$, позначається як $X = \{x|P(x)\}$, або $X = \{x : P(x)\}$, причому $a \in \{x|P(x)\}$, якщо $P(a)$ істинне. Висловлення $P(x)$ є правилом належності елементів множині.

В задачах, які розв'язуються за допомогою комп'ютерної техніки, множини задають переважно за допомогою деякого алгоритму генерування їх елементів (вказують ***породжувальну процедуру*** або ***породжувальний предикат).*** Прикладом такої породжувальної процедури є алгоритм генерування простих чисел.

1.3. Операції над множинами

Операції над множинами (ще називають ***теоретико-множинними операціями***) звичайно ілюструють графічно за допомогою так званих ***діаграм Вейча*** (рис. 1.1). На цих діаграмах множини-аргументи зображуються у вигляді областей площини, а результат виконання операції — у вигляді заштрихованої області.

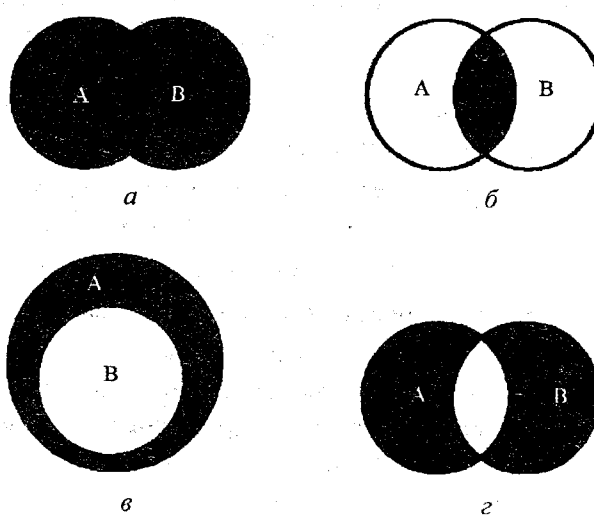
При формальному зображенні операцій використовуються символи $\Leftrightarrow$, $\exists x$, $\forall x$, $\Rightarrow$, які надалі будуть служити для скорочення виразів *"тоді і тільки тоді, коли"*, *"існує x такий, що"*, *"для всякого x"* і *"слідує"* або *"випливає"*, відповідно.

Основні операції над множинами:

1) ***об'єднання множин*** (рис.1.1,а): $A \cup B$.

Об'єднанням множин A і B називається множина, яка складається з тих і тільки тих елементів, які входять до складу хоча б однієї з цих множин. Одержана множина позначається $A \cup B$, тобто $A \cup B = \{a \mid a \in A \text{ або } a \in B\}$.

Результатом операції є елементи, які належать або множині А, або множині B.



а *б* *в* *г*

Рис 1.1 Діаграми Вейча:
а — діаграма для $A \cup B$; *б* — діаграма для $A \cap B$;
в — діаграма для A'; *г* — діаграма для $A \oplus B$

Приклади:

1. Нехай $A = \{1, 2, 3\}$, $B = \{1, 3, 4, 6\}$; тоді $A \cup B = \{1, 2, 3, 4, 6\}$.

2. Нехай $П$ — множина всіх парних натуральних чисел, а $Н$ — множина всіх непарних натуральних чисел; тоді $П \cup Н = N$, де N — множина всіх натуральних чисел.

Об'єднання сукупності $\{M_x\}, x \in X$ позначають $\bigcup_{x \in X} M_x$.

Об'єднанням сукупності множин (рис.1.1, *а*) називається множина, що складається із всіх елементів, які належать хоча б одній з множин даної сукупності, тобто

$$\alpha \in \bigcup_{x \in X} M_x \Leftrightarrow \exists x \in X : \alpha \in X$$

2) ***переріз множин*** (рис.1.1, *б*): $A \wedge B$

Перерізом множин A і B називається множина, яка складається з елементів, що входять до складу як множини A, так і множини B. Одержана множина позначається, $A \cap B$ тобто $A \cap B = \{a \mid a \in A \; i \; a \in B\}$. Якщо $A \cap B = 0$, то множини A і B називаються такими, що не перерізаються.

Приклади:

Нехай $A, B, П, Н$ — множини з попереднього прикладу; тоді:

$A \cap B = \{1, 3\}$;

$П \cap Н = \varnothing$;

$N \cap Н = Н$;

$N \cap П = П$.

3) ***різниця множин*** $A \backslash B$

Різницею множин A і B називається множина $B \setminus A = \{a \mid a \in B \; i \; a \notin A\}$. Очевидно, що $B \setminus A = B \setminus (A \cap B)$. Якщо $A \subseteq B$, то $B \setminus A$ називається ***доповненням*** множини A в множині B (рис.1.1, *в*): і позначається A'_B або просто A', коли B можна визначити із контексту.

Доповнення множини (передбачає наявність універсальної множини) $\overline{A} = U \setminus A$

Приклади:

1. Нехай $A = \{1, 2, 3\}$, $B = \{1, 3, 4, 5\}$. Тоді

$$B \setminus A = \{1, 3, 4, 5\} \setminus \{1, 2, 3\} = \{4, 5\} = B \setminus (A \cap B) = \{1, 3, 4, 5\} \setminus \{1, 3\} = \{4, 5\}$$

2. Множина $N \setminus П = Н$, тобто $N \setminus П$ являє собою множину всіх непарних натуральних чисел. Навпаки, $N \setminus Н = П$.

4) ***диз'юнктивна сума $A \oplus B$ (симетрична різниця)*** (рис.1.1, *г*)

Симетричною різницею множин A *і* B називається множина $A \div B = (A \setminus B) \cup (B \setminus A)$. Це є множина всіх елементів, що належать або A, або B (але не обом разом). Наприклад, $\{1, 2, 3\} \oplus \{2, 3, 4\} = \{1, 4\}$. Диз'юнктивна сума виходить об'єднанням елементів множин за винятком тих, які зустрічаються двічі.

З наведеної діаграми для операції $A \div B$ очевидним чином випливає така рівність: $A \oplus B = (A \cup B) \setminus (A \cap B)$.

Введені операції об'єднання, перерізу і доповнення задовольняють певні властивості.

Властивості операцій

1) ідемпотентність	$A \cup A = A$; $A \cap A = A$	(1.1)
2) комутативність	$A \cup B = B \cup A$; $A \cap B = B \cap A$	(1.2)
3) асоціативність	$A \cup (B \cup C) = (A \cup B) \cup C$; $A \cap (B \cap C) = (A \cap B) \cap C$	(1.3)
4) дистрибутивність	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$;	
	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	(1.4)
5) поглинання	$(A \cap B) \cup A = A$; $(A \cup B) \cap A = A$	(1.5)
6) властивість нуля	$A \cup \varnothing = A$; $A \cap \varnothing = \varnothing$	(1.6)
7) властивість одиниці	$A \cup U = U$; $A \cap U = A$	(1.7)
8) інволютивність	$\overline{\overline{A}} = A$	(1.8)
9) повнення	$A \cup \overline{A} = U$; $A \cap \overline{A} = \varnothing$	(1.9)
10) закони Де-Моргана	$\overline{A \cap B} = \overline{A} \cup \overline{B}$; $\overline{A \cup B} = \overline{A} \cap \overline{B}$	(1.10)

Користуючись цими тотожностями можна спрощувати складні вирази, які містять у собі множини, аналогічно тому, як проводяться спрощення виразів в елементарній алгебрі. Розглянемо приклади.

Приклади:

1. $(A \cap B \cap C) \cup (A' \cap B \cap C) \cup B' \cup C =$
   $= [(A \cup A') \cap B \cap C] \cup B' \cup C' = (U \cap B \cap C) \cup B' \cup C' =$
   $= (B \cap C) \cup B' \cup C' = (B \cap C) \cup (B \cap C)' = U$
2. $(A \cap B \cap C \cap D') \cup (A' \cap C) \cup (B' \cap C) \cup (C \cap D =)$
   $(A \cap B \cap C \cap D') \cup [(A' \cup B' \cup D) \cap C] =$
   $= [(A \cap B \cap D') \cup (A \cap B \cap D')'] \cap C = U \cap C = C)$
3. $(A \cap B')' \cup B = (A' \cup B'') \cup B = A' \cup (B \cup B) = A' \cup B$

1.4. Реалізація операцій з множинами у програмах

Використання теорії множин є зручним способом розв'язання багатьох задач комбінаторного та графового типу. Найзручніші засоби для застосування множин у програмах надає мова програмування Паскаль.

1.4.1. Визначення множини

У мові Паскаль існує декілька способів задавання множини

а) *type*

TsetA= set of <перераховувальний тип (char, integer, word, byte ...)>;

Var X:TsetA;

Y: set of 1..3;

б) *const c: set of byte=(0,1,2,3,4);*

Означення типу даних "множина" відповідає заданню поняття "універсальна множина". Опис змінної множинного типу означає вказування, елементи якої універсальної множини *можуть* належати (але не обов'язково належать) цій змінній.

Пуста множина позначається [ ].

1.4.2. Операції

Мова Паскаль передбачає можливість виконання алгебраїчних і логічних операцій над множинами:

- - об'єднання +
- - переріз *
- - доповнення \
- порівняння: перевірка, чи є множина *A* підмножиною множини *B* (*A<B, A>B*);
- перевірка тотожності: *A=B*;
- перевірка належності елемента множині: ***a* in *A*** (це єдина операція, в якій один операнд – множина, а інший – елемент множини);
- присвоєння: *A:=B* (якщо множини однотипні), або *A:=* **[1..5]**.

1.4.3. Внутрішнє подання множин

Спосіб внутрішнього (у пам'яті комп'ютера) подання множин у програмі залежить від мови програмування. При поданні множини у

програмі на Паскалі кожному елементу множини відповідає один біт. Цей біт дорівнює 1, якщо елемент належить множині, і 0, якщо не входить до неї. Довжина бітової послідовності, що зображує множину, відповідає потужності універсальної множини.

Таке внутрішнє подання множини відповідає поняттю характеристичної функції.

Характеристичною функцією множини $X \subset U$ називають функцію X_X, яка дорівнює 1 на елементах X і 0 на решті елементах U. Операції над підмножинами множини U відповідають операціям з їх характеристичними функціями. Зокрема, перерізу множин відповідає добуток характеристичних функцій: $X_{A\cap B}(u) = X_A(u)X_B(u)$. Доповненню (до U) відповідає функція (1-X).

Потужність множини можна записати як суму значень її характеристичної функції:

$$|X| = \sum_u X_X(u)$$

Об'єднання $A_1 \cup .... \cup A_n$ можна записати як доповнення до перерізу доповнень множин A_i

$$X_{A_1 \cup \ldots \cup A_n} = 1 = (1 - X_{A_1}) \ldots (1 - X_{A_n})$$

1.5. Нечіткі множини

Теорія нечітких множин є узагальненням й переосмисленням найважливіших напрямків класичної теорії множин.

Підхід до формалізації поняття нечіткої множини полягає в узагальненні поняття належності. У звичайній теорії множин існує кілька способів задання множини. Одним з них є задання за допомогою розглянутої вище характеристичної функції. Особливістю цієї функції є бінарний характер її значень. У теорії нечітких множин характеристична функція називається функцією належності, а її значення - ступенем належності елемента нечіткій множині. Ступінь належності може приймати проміжні значення від 0 до 1.

1.5.1. Основні визначення

Нехай V— множина (класична). Нечітка множина A задається своєю функцією належності елемента x нечіткій множині A:

$$\mu_A : V \to [0; 1]$$

Якщо μ_A приймає лише значення $\{0, 1\}$ то множина — класична, якщо вона може приймати проміжні значення, то така множина є нечіткою.

Носій нечіткої множини A — це

$$\text{supp}A = \{x \in V \mid \mu_A > 0\}.$$

Множина рівня α (де $\alpha \in [0;1]$) це:

$$A_\alpha = \{x \in V \mid \mu_A \geq \alpha\}.$$

Тоді $$\text{supp}A = \bigcup_{\alpha>0} A_\alpha .$$

Пуста множина має $\mu_0(x) = 0$, а універсальна множина - $\mu_V(x) = 1$.

Можна казати, що $\mu_A(x)$ - це ступінь належності елемента x до множини A.

Якщо елементи універсальної множини є раціональними числами, тобто $V=R$, то її нечіткі одноелементні підмножини називають нечіткими числами.

Нечіткою множиною A називається сукупність пар

$$\left\{\left\langle x, \mu^A(x)\right\rangle\right\},$$

де $\mu_A(x)$ - функція належності, тобто $x \in X$ і $\mu^A : X \to [0; 1]$.

Нехай, наприклад,

$$U=\{a, b, c, d, e\},$$

$$A = \left\{\langle a,0\rangle, \langle b,0,1\rangle, \langle c,0,5\rangle, \langle d,0,9\rangle, \langle e,1\rangle\right\}.$$

Це означає, що елемент a не належить множині A, елемент b належить їй в малому ступені, елемент c більш-менш належить, елемент d належить значною мірою, e є елементом множини A.

1.5.2. Операції над нечіткими множинами

Над нечіткими множинами визначені операції, причому вони визначені так, щоб в окремому випадку, коли множина є чіткою, операції переходять у звичайні операції теорії множин, тобто операції над нечіткими множинами узагальнюють відповідні операції над звичайними множинами. При цьому узагальнення може бути реалізовано різними

способами, через що будь-якій операції над звичайними множинами може відповідати кілька операцій у теорії нечітких множин.

1) доповненням нечіткої множини A називається нечітка множина $\neg A$, функція належності якої дорівнює:

$$\mu^{\neg A}(x)=1-\mu^{A}(x),\ \forall x\in X. \tag{1.11}$$

2) перерізом двох нечітких множин A і $B\subseteq X$ називається нечітка множина $A\cap B$, функція належності якої дорівнює:

$$\mu^{A\cap B}(x)=\mu^{A}(x)\wedge\mu^{B}(x),\ \forall x\in X, \tag{1.12}$$

де $\wedge$ - знак операції мінімуму.

3) об'єднанням двох нечітких множин A і $B\subseteq X$ називається нечітка множина $A\cup B$, функція належності якої дорівнює:

$$\mu^{A\cup B}(x)=\mu^{A}(x)\vee\mu^{B}(x),\ \forall x\in X, \tag{1.13}$$

де $\vee$ - знак операції максимуму.

1.6. Множини з додатковими властивостями

На практиці застосування поняття множини найчастіше використовуються множини з певними властивостями їх елементів і підмножин. Множина із заданими на ній відношеннями чи законами композицій елементів називається простором. Конкретизація множин, властивостей відношень та законів композиції приводить до різних типів просторів: метричних і топологічних, лінійних та евклідових, нормальних і ймовірнісних тощо. Розглянемо деякі з них.

У математиці термін "простір" вживається для позначення множини з деякою додатковою структурою. Залежно від цієї додаткової структури елементи простору можуть називатися "точками", "векторами", "подіями" і т.п.. Підмножина простору називається "підпростором" якщо на цій підмножині є структура такого ж типу.

Топологічний простір - основний об'єкт вивчення топології. Поняття топологічного простору можна розглядати як узагальнення поняття геометричної фігури, у якому ми відволікаємося від властивостей на зразок розміру або точного положення частин фігури в просторі, і

зосереджуємося тільки на взаємному розташуванні частин. Топологічні простори виникають природно майже у всіх розділах математики.

Нехай дана множина X. Система T її підмножин називається ***топологією на X***, якщо виконані такі властивості:

1) об'єднання довільного сімейства множин, що належать T, належить T.

2) перерізання кінцевого сімейства множин, що належать T, належить T.

3) X і $\varnothing$ належать T.

Множина X разом із заданою на ній топологією T називається ***топологічним простором***. Множини, що належать T, називаються відкритими множинами.

Лінійний (векторний) простір — основне поняття лінійної алгебри, абстрактне узагальнення множини всіх векторів на площині чи в просторі з операціями додавання векторів та множення вектора на скаляр. Елементи абстрактного лінійного простору називаються ***векторами***, але не робиться ніяких припущень стосовно природи чи походження цих елементів.

Незалежно від природи елементів векторного простору, їх додавання і множення на скаляр відповідають звичайним правилам шкільної алгебри.

У довільному векторному просторі не визначені операції скалярного чи векторного добутку векторів.

Лінійний простір X називається ***нормованим***, якщо кожному елементу цього простору поставлено у відповідність строго визначене дійсне число - ***норму***, яка позначається $\|x\|$ і виконуються властивості:

$$\|x\|>=0;\ \|x\|=0 \Leftrightarrow x=0 \quad x\in X \tag{1.14}$$

$$\|a*x\|=|a|*\|x\| \quad x\in X \quad a\in R \tag{1.15}$$

$$\|x+y\|\le\|x\|+\|y\| \quad x, y\in X \tag{1.16}$$

Повний нормований простір називається ***банаховим простором***.

Для будь-якого нормованого простору E існує банаховий простір $\widetilde{E}$ такий, що 1) $E\subset\widetilde{E}$; 2) $\forall f\in E$, $\|f\|_E=\|f\|_{\widetilde{E}}$; 3) множина E щільна в $\widetilde{E}$. Простір $\widetilde{E}$ називається поповненням простору E.

Комплексний гільбертів простір (на честь Давида Гільберта) — це топологічний векторний простір H, у якому визначена операція скалярного добутку і який є повним відносно відповідної топології. Поняття гільбертового простору є узагальненням поняття скінченновимірного евклідового простору. Типовий гільбертів простір — нескінченновимірний і складається із функцій.

Метричним простором називається множина, у якій визначена відстань між будь-якою парою елементів.

Метричний простір M є множиною точок з функцією відстані (також називається метрикою) $d : M \times M \to R$ (де R позначає множину реальних чисел). Для будь-яких точок x, y, z з M ця функція повинна відповідати таким умовам (аксіомам метрики):

$d(x, y) \geq 0$

$d(x, y) = 0 \Leftrightarrow x = y$.

$d(x, y) = d(y, x)$ (симетрія)

$d(x, z) \leq d(x, y) + d(y, z)$ (нерівність трикутника).

Ці аксіоми відображають інтуїтивне поняття відстані. Наприклад, відстань повинна бути додатною; відстань від x до y така ж, як і від y до x. Нерівність трикутника означає, що пройти від x до z можна коротше, або хоча б не довше, ніж спочатку пройти від x до y, а потім від y до z.

Евклідів простір - це n-вимірний метричний простір, характеристики якого неформально можна вважати узагальненнями звичних та досліджуваних Евклідом дво- та тривимірних просторів.

Нехай декартові координати в тривимірному просторі такі, що точці P відповідають три її координати (x_1, x_2, x_3), а точці Q - координати (y_1, y_2, y_3). Тоді, якщо квадрат довжини прямолінійного відрізка, що з'єднує P та Q, дорівнює: $l^2 = (x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2$, то такий простір називають ***евклідовим простором***, а декартові координати з такими властивостями називають ***евклідовими координатами***.

Узагальнюючи на випадок n вимірів, отримаємо

$$l^2 = (x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2 = \sum_{k=1} (x_k - y_k)^2 .$$

Наведений вигляд функції для евклідового простору має назву ***евклідової метрики.***

Ймовірнісний простір - поняття, введене А. Н. Колмогоровим у 30-х роках XX століття для формалізації поняття ймовірності, що дало початок бурхливому розвитку теорії ймовірностей як строгої математичної дисципліни.

Ймовірнісний простір - це трійка (Ω, F, P),

де Ω — - довільна множина, елементи якої називаються елементарними подіями;

F - сігма-алгебра підмножин Ω, які називають (випадковими) подіями;

P - ймовірнісна міра або ймовірність, така що $P(\Omega)=1$.

Нехай $U=\{E_1, E_2, E_n\}$ — універсальна множина елементарних подій, і кожній елементарній події $E_i \in U$ відповідає елементарна ймовірність $P(E_i), i=1,2,\ldots,n$. Оскільки $P(U)=1$, то повинно виконуватись співвідношення

$$P(E_1)+P(E_2)+\ldots+P(E_n)=1.$$

Контрольні питання і завдання для самостійної роботи

1. Що таке множина?
2. Які існують способи задання множин?
3. Які основні операції виконуються над множинами?
4. Скільки існує підмножин в n-елементній множині?
5. Дайте означення нечіткої множини та основних операцій над ними.
6. Що таке простір, наведіть основні види просторів та їх ознаки.
7. Дано множини A={1, 2, 3, 4, 5, 6}, B={2, 3, 5, 8, 0}. Знайти A∩B, A∪B, A\B, A⊕B та нарисувати відповідні діаграми Вейча.
8. Чи існують такі множини A, B, C, що $A\cap B\neq\varnothing$, $A\cap C\neq\varnothing$, $(A\cap B)\setminus C=\varnothing$?
9. Описати словами кожну з множин:

   а) $\{x\in \mathrm{N} \mid x$ ділиться на 2 і x ділиться на 3$\}$;

   б) $\{x \mid x\in A \ i \ x\in B\}$;

   в) $\{x \mid x\in A \ i \ x\notin B\}$;

   г) $\{(x, y)\in D^2 \mid x^2+y^2=1\}$;

   д) $\{(x, y)\in D^2 \mid y=2x \ i \ y=3x\}$.

10. Довести тотожності: $A \cup A' = U$; $A \setminus (B \cup C) = (A \setminus B) \cap (A \setminus C)$;
$A \setminus (B \cup C) = (A \setminus B) \setminus C$; $A \setminus (B \cap C) = (A \setminus B) \cup (A \setminus C)$;
$A \setminus (B \setminus C) = (A \setminus B) \cup (A \cap C)$; $(A \setminus B) \setminus C = (A \setminus C) \setminus (B \setminus C)$;
$A \cap B = A \setminus (A \setminus B)$; $A \cup B = A \cup (B \setminus A)$; $(A' \cup B) \cap A = A \cap B$;
$A \cap (B \setminus A) = \varnothing$; $(A \cup B) \setminus C = (A \setminus C) \cup (B \setminus C)$
11. Довести співвідношення:
$A \cup B \subseteq C \Leftrightarrow A \subseteq C$ *і* $B \subseteq C$;
$A \subseteq (B \cap C) \Leftrightarrow A \subseteq B$ *і* $A \subseteq C$;
$A \cap (B \setminus C) = (A \cap B) \setminus (A \cap C) = (A \cap B) \setminus C$;
$(A \cap B) \cup (A \cap B') = (A \cup B) \cap (A \cup B') = A$;
$A = B \Leftrightarrow A \cup B = B \Leftrightarrow A \cap B = A \Leftrightarrow A \setminus B = \varnothing$ *і* $A' \cup B = U$,
де U — універсальна множина, а A' — доповнення множини A в U.
12. Скласти програму мовою Pascal з використанням множин для підрахунку кількості і складання списку голосних літер у тексті, що зберігається у текстовому файлі.
13. Скласти програму мовою Pascal, яка утворює множину, що складається з перших 100 парних натуральних чисел.
14. Є група студентів, які позначаються літерами, для простоти, від A до F. З них дівчата мають імена від A до D, хлопці мають імена від E до F. З них комп'ютерщики: C, E, A; преферансисти: від E до B. Скласти програму, яка дає відповідь на такі питання:
    - Хто займається і програмуванням, і преферансом?
    - Хто з дівчат не займається нічим?
    - У кого з хлопців і дівчат є спільні інтереси?
15. Скласти програму мовою Pascal з використанням множини, яка розбиває текст, що зберігається у текстовому файлі, на слова, і виводить їх в стовпчик на екран. Переносів у файлі немає.

Рекомендована література

1. Сигорский В.П. Математический аппарат инженера.– К.: «Техніка», 1975. - 786 с.
2. Капітонова Ю.В. та ін. Основи дискретної математики. – К.: Наукова думка, 2002. - 579 с.
3. Лапа В.Г. Математические основы кибернетики. – К.: Вища школа, 1974. - 452 с.

4. Корн Г., Корн Т. Справочник по математике. - М.: Наука, 1974.
5. Александров П.С. Введение в теорию множеств и общую топологию. – М.: Наука, 1977.
6. Бурбаки Н. Начала математики. Ч 1. Основные структуры анализа. Кн. 1. Теория множеств. – М.: Мир, 1965. – 456 с.
7. Йех Т. Теория множеств и метод форсинга. – М.: Мир, 1973. – 150 с.
8. Кантор Г. Труды по теории множеств. – М.: Наука, 1985. – 431 с.
9. Крон П. Дж. Теория множеств и континуум гипотеза. – М.: Мир, 1969. – 347 с.
10. Куратовский К., Мостовский А. Теория множеств. – М.: Мир, 1970. – 416 с.
11. Лавров И.А., Максимова Л.Л. Задачи по теории множеств, математической логике и теории алгоритмов. – М.: Наука, 1984. – 224 с.
12. Манин Ю.И. Доказуемое и недоказуемое. – М.: Советское радио, 1979. – 168 с.
13. Барвайс Дж.Справочная книга по математической логике. Ч. 2. Теория множеств. – М.: Наука, 1982. – 376 с.
14. Френкель А.А., Бар-Хиллел И. Основания теории множеств. – М.: Мир, 1966. – 556 с.
15. Хаусдорф Ф. Теория множеств. – М.-Л.: ОНТИ, 1937.
16. Мостовский А. Конструктивные множества и их приложения. – М.: Мир, 1973. – 256 с.
17. Аверкин А.Н. и др. Нечеткие множества в моделях управления и искусственного интеллекта/ Под ред. Поспелова Д.А. – М.: Наука, 1986. –312 с.
18. Бондарев В.М. и др. Основы программирования. – Харьков: Фолио, 1997.
19. Дубовой В.М. Моделювання систем контролю та керування. – Вінниця: ВНТУ, 2005. – 175 с.
20. Кофман А. Введение в теорию нечетких множеств. – М.: Радио и связь, 1982. – 432 с.
21. Коршунов Ю.М. Математические основы кибернетики. – М.: Энергия, 1972. – 376 с.
22. Нечеткие множества и теория возможностей. Последние достижения: Пер. с англ./ Под ред. Р. Р. Ягера. – М.: Радио и связь, 1986. – 408 с.
23. Новиков Ф.А. Дискретная математика для программистов. - СПб.: Питер, 2001.

2. КОМБІНАТОРИКА

Комбінаторика – це розділ математики, який вивчає методи розрахунку кількості підмножин, які можна утворити за допомогою вибірки елементів певної базової множини за певним правилом. Оскільки комбінаторика має справу із скінченними множинами, то її часто називають *теорією скінченних множин.*

Даний розділ підготовлений з використанням матеріалів [1, 2, 3, 7, 8].

2.1. Основи комбінаторики

Відповідно до предмета розгляду комбінаторики, підмножини, що утворюються за правилами вибірки, називаються ***комбінаторними об'єктами***. Таким чином головна задача комбінаторики – підрахунок кількості комбінаторних об'єктів.

2.1.1. Основні поняття комбінаторики

Коротко зупинимось на таких поняттях, як *невпорядкована* і *впорядкована* множина, які фігурують в означеннях багатьох комбінаторних конструкцій.

Для невпорядкованої множини важливим є лише якісний склад її елементів. Наприклад, якщо $A=\{a,b,c\}$, $B=\{b,a,c\}$, то $A=B$. Якщо множина впорядкована, тобто додатково вказано, який елемент вважається першим, другим, ..., останнім (задано порядок елементів), то важливими є як якісний склад елементів, так і їхній порядок. Для множини M, в якій задано порядок елементів, тобто вона є впорядкованою, будемо використовувати позначення $\overline{M}$. Дві впорядковані множини вважаються рівними, якщо вони збігаються як за якісним складом, так і за порядком своїх елементів.

2.1.2. Комбінаторні об'єкти без повторень

Формули для підрахунку кількості комбінаторних об'єктів випливають з правил суми та добутку.

Правило суми: якщо об'єкт „*a*" може бути вибраний *p* способами, а об'єкт „*b*" – *q* способами, то об'єкт „*a або b*" може бути вибраний $p+q$ способами.

Правило добутку: якщо об'єкт „*a*" може бути вибраний *p* способами, а об'єкт „*b*" – *q* способами, то об'єкт „*a і b*" може бути вибраний $p \cdot q$ способами.

Основні типи комбінаторних об'єктів: сполучення, перестановка, розміщення.

Сполучення – вибірка *k* елементів з загальної множини *n* елементів. Сполучення не враховують порядок елементів. Кількість можливих сполучень позначається C_n^k (літера *C* від англійського слова "*Combination*" - "комбінація").

Число різних *k*- елементних підмножин *n*- елементної множини становить

$$C_n^k = \frac{n!}{k!(n-k)!} \tag{2.1}$$

де $n! = n \cdot (n-1) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$ - факторіал числа *n* (добуток всіх цілих чисел від 1 до *n*, читається *n*-факторіал).

Перестановка - вибірка *n* елементів із множини потужністю *n* з врахуванням порядку слідування. Кількість можливих перестановок позначається P_n. Для будь-якого *n*

$$P_n = n! \tag{2.2}$$

Розміщення – вибірка *k* елементів з загальної множини потужністю *n* елементів з врахуванням порядку слідування. Кількість можливих розміщень позначається A_n^k.

Очевидно, з означення випливає

$$A_n^k = C_n^k \cdot P_n \tag{2.3}$$

Кількість упорядкованих *k*-елементних підмножин *n*-елементної множини, всі *k* елементів якої різні, становить

$$A_n^k = k! C_n^k = \frac{n!}{(n-k)!} = n \cdot (n-1) \cdot \ldots \cdot (n-k+1). \tag{2.4}$$

2.1.3. Комбінаторні об'єкти з повтореннями

Нехай впорядковується (задається порядок елементів) множина M потужності n, яка не є різноелементною і містить k_1 елементів 1-го типу, k_2 елементів 2-го типу, ..., k_s елементів s-го типу (очевидно, $k_1+k_2+...+k_s = n$), причому елементи однакового типу вважаються *невідрізнимими.* Одержану при цьому впорядковану множину M^* називають ***перестановкою з n елементів з повтореннями.***

Через наявність однакових елементів не всі перестановки з повторенням будуть різними (відрізнимими), тому виникає питання про визначення кількості лише відрізнимих перестановок.

Потужність множини всіх *відрізнимих* перестановок з n елементів з повторенням позначається символом $C_n^{(k1,k2,...,ks)}$.

$$C_n^{(k_1,k_2,...k_s)} = \frac{n!}{k_1!k_2!...k_s!}. \qquad (2.5)$$

Якщо з них вибрати будь-які k елементів (k =0,1,2,...), не обов'язково різних типів і додатково довільним чином впорядкувати, то одержана при цьому впорядкована множина потужності k називається ***розміщенням з n елементів по k з повтореннями.***

Потужність множини всіх розміщень з n елементів по k з повторенням позначається $\overline{A_n^k}$. Оскільки елементи вибираються з повторенням, то на відміну від символу A_n^k, де завжди $k \le n$, в $\overline{A_n^k}$ може бути $k > n$ (наприклад, $\overline{A_2^{100}}$).

Для $\overline{A_n^k}$ існує формула

$$\overline{A_n^k} = n^k. \qquad (2.6)$$

Розміщення з n елементів по k з повторенням є узагальненням відповідного розміщення без повторення, оскільки запас елементів кожного з n типів вичерпується одним елементом.

Прикладом розміщень з повторенням є телефонні номери.

Нехай маємо елементи n типів (елементи кожного однакового типу вважаються *невідрізнимими,* а їхній запас - невичерпним). Якщо з них вибирають будь-які k елементів (k=0,1,2,...) не обов'язково різних типів, причому порядок самих елементів *ігнорується,* то одержана при цьому

невпорядкована множина потужності k називається ***сполученням з n елементів по k з повтореннями.***

Потужність множини всіх сполучень з n елементів по k з повторенням позначається символом $\overline{C_n^k}$. Задача про число комбінацій з n елементів по k з повторенням еквівалентна задачі про число перестановок з $n+k-1$ елементів з повторенням

$$\overline{C_n^k} = C_{n+k-1}^k. \qquad (2.7)$$

2.2. Генерування комбінаторних об'єктів

Для розв'язання практичних задач виникає необхідність генерування комбінаторних об'єктів, тобто практичного здійснення вибірки елементів деякої множини за певним правилом.

Генерування комбінаторних об'єктів використовується у програмуванні для розв'язання задач перебору варіантів, наприклад, перебору можливих шляхів проходження маршруту між початковим і кінцевим пунктом призначення, перебору можливих способів групування колективів тощо.

2.2.1. Генерування перестановок

Найчастіше використовується генерування перестановок.

Для розв'язання задач генерування перестановок розроблені алгоритми:

1) метод простого перебору (рекурсивний алгоритм);

2) лексикографічний (антилексикографічний) метод. Лексикографічним називають порядок, за яким сортують слова за алфавітом – спочатку за першою літерою, потім за другою і так далі. Антилексикографічний порядок – сортування з кінця;

3) метод перестановок сусідніх елементів (алгоритм Джонсона і Троттера). При такому алгоритмі кожні дві сусідні комбінації відрізняються рівно на два елементи;

4) метод мінімальної кількості транспозицій

та інші.

Реалізація алгоритмів генерування перестановок мовою Паскаль наведена у додатку.

2.2.2. Генерування підмножин

Генерування підмножин ґрунтується на внутрішньому поданні множин у комп'ютері. Для того, щоб згенерувати підмножину у програмі на Паскалі, необхідно надати значення 1 бітам, які відповідають тим елементам множини, що належать підмножині, і 0 – тим, що не належать їй.

Для генерування підмножини в Паскалі використовується абсолютна адресація.

```
Var
   S: set of 1..8;
   b: byte  absolute S;
Begin
 for b:=0 to 255 do...
End.
```

При абсолютній адресації байт b розташовується у пам'яті в тому ж місці, що і множина *S*. Тому зміна значення b призводить до зміни розташування одиниць і нулів в його коді, отже і до зміни складу множини *S*. У наведеному фрагменті алгоритму у циклі перебираються всі можливі значення *b*, отже генеруються і всі можливі підмножини множини *S*.

Якщо множина *S* містить більше елементів, то для генерування підмножин необхідно вибирати такий тип *b*, який „перекривав" би у пам'яті всю множину. Найбільше ціле дане longint має 32 біти. При більшій потужності множини доводиться використовувати вже масив байтів і штучну процедуру перебору значень, аналогічну процесорній операції додавання з переносом.

2.2.3. Генерування сполучень

Найпростіший спосіб генерування сполучень – це генерування всіх можливих підмножин і перевірка кількості елементів у згенерованій множині.

```
Const c
...
for b:=0 to 255 do
begin
   k:=0;
   for i:=1 to 255 do
     begin
       if i in S then inc(k);
       if k=c then ...
     end
end;
```

2.2.4. Генерування розміщень

Алгоритм генерування розміщень природно складається з двох процедур:

- генерування сполучення із заданою кількістю елементів;
- перестановка елементів згенерованої підмножини.

Відповідні алгоритми розглянуті вище.

2.2.5. Генерування розбиттів множини

Нагадаємо, що *розбиття множини* - це подання його у вигляді об'єднання довільної кількості підмножин, які попарно не перетинаються.

Найпростіший спосіб генерування розбиттів множини X – ітераційний. На першому кроці генерується підмножина U_1 множини X. Потім знаходять доповнення $X' = X \setminus U_1$ і генерують підмножину U_2. Так продовжують доки чергове доповнення не стане пустою множиною.

Контрольні питання і завдання для самостійної роботи

1. Яка головна задача комбінаторики?
2. Чим відрізняються впорядкована та невпорядкована множина?
3. Наведіть основні формули для знаходження сполучення, перестановки та розміщення для комбінаторних об'єктів без повторень.
4. Наведіть означення та розрахункову формулу для сполучення з n елементів по k з повторенням.

5. Наведіть означення та розрахункову формулу для розміщення з n елементів по k з повторенням.
6. Наведіть означення перестановки з n елементів з повторенням.
7. Як реалізується генерування комбінаторних об'єктів: перестановок, підмножин, сполучень, розміщень?
8. Скількома способами на першості світу з футболу можуть розподілитися медалі, якщо у фінальній частині грають 24 команди?
9. На кафедрі працює сім викладачів. Скількома способами можна скласти комісію з трьох чоловік для прийому "хвостів"?
10. В шаховому турнірі брали участь 30 чоловік, і кожні два шахісти зіграли між собою лише один раз. Скільки партій було зіграно в турнірі?
11. Скільки слів із п'яти букв можна скласти, якщо $X = \{a, b, c, d\}$ і буква a зустрічається в слові не більше двох разів, буква b — не більше одного разу і буква c — не більше трьох разів?
12. Скільки різних слів можна скласти перестановкою букв у слові "чачача"?
13. Скільки існує різних n- знакових телефонних номерів?
14. Група студентів налічує 13 чоловік. Перерахувати кількість варіантів послідовності здавання іспиту.
15. Група студентів складається з 12 чоловік. Підрахувати кількість варіантів, згідно з якими студенти сидять за партами.
16. У групі налічується 7 хлопців і 6 дівчат. Підрахувати кількість варіантів пар.
17. Скласти програму для підрахунку кількості перестановок, сполучень та розміщень для довільних значень n та k.
18. Комп'ютерною мережою, яка складається з 10 комп'ютерів, користується 25 користувачів. Кожен користувач зберігає свою інформацію на 2-х з 10 комп'ютерах. Підрахувати:
    1. Кількість папок на кожному комп'ютері, виділених для користувачів.
    2. Кількість варіантів закріплення комп'ютерів за користувачами.

Рекомендована література

1. Бушмакін В.М. та ін. Комбінаторика. – Львів: Видавництво НУ

"Львівська політехніка", 2002. – 196 с.
2. Липский В. Комбинаторика для программистов. - М.: Мир, 1988.
3. Новиков Ф.А. Дискретная математика для программистов. - СПб.: Питер, 2001.
4. Виленкин Н.Я. Комбинаторика. - М.: Наука, 1969.
5. П.Риордан Дж. Введение в комбинаторный анализ. - М.: Иностранная литература, 1963.
6. Сачков В.Н. Введение в комбинаторные методы дискретной математики. - М.: Наука, 1977.
7. Глонь О.В., Дубовой В.М., Мітюшкін Ю.І. Комп'ютеризовані системи керування. – Вінниця: ВНТУ, 2005. – 157 с.
8. Дубовой В.М. Моделювання систем контролю та керування. – Вінниця: ВНТУ, 2005. – 175 с.
9. Седжвик Р. Фундаментальные алгоритмы на С. Алгоритмы на графах: Пер. с англ. – СПб.: ДиаСофтЮП, 2003. – 480 с.

3. ГРАФИ

Теорія графів надає в розпорядження інженера виключно зручний апарат для моделювання структурних властивостей систем і відношень між об'єктами найрізноманітнішої природи. Завдяки наочності і простоті цей апарат в останній час завоював визнання і широко використовується в науково-технічній літературі.

Даний розділ підготовлений з використанням матеріалів [1 - 5, 11 - 13, 16].

3.1. Основи теорії графів

Виникнення теорії графів пов'язують з іменем Ейлера, який у 1736 р. не тільки розв'язав популярну на той час головоломку про кенігсберзькі мости, а й знайшов критерій існування в графі спеціального маршруту (ейлерового циклу). Довгий час цей результат залишався єдиним результатом теорії графів, і лише в середині XIX ст., переважно зусиллями Кірхгофа та Келлі, були одержані нові результати в теорії графів.

Хоча теорія графів виникла більше двох століть тому, але її інтенсивний розвиток припадає лише на останні 50—60 років. Цей розвиток завдячує широкому застосуванню графів у теорії автоматів, теорії проектування, економіці, хімії, біології тощо.

Граф – це наочне графічне зображення взаємозв'язку елементів деякої множини об'єктів. Таке дуже загальне означення наштовхує на думку, що методи і алгоритми теорії графів можуть використовуватись для розв'язування дуже великої кількості задач. Адже поняття множини – це фундамент сучасної математики, а це означає, що практично будь-яка математична модель може бути подана у термінах графів. Теорія графів дуже багата на алгоритми розв'язання найрізноманітніших задач. Питання полягає лише в доцільності такого подання – може існує більш адекватний і ефективний шлях розв'язання конкретної задачі?

При створенні графової моделі слід в першу чергу визначитися, у якому просторі вона створюється. В залежності від природи елементів множини та їх зв'язків графи можуть розглядатися у різних просторах:

- в геометричному просторі – наприклад, карта автомобільних доріг, план розташування комп'ютерів у будинку;

- в просторі станів (у часі) – алгоритм (відображує зміну та зв'язок станів комп'ютера), мережевий графік (відображує зміну та зв'язок станів технологічного процесу);
- в просторі відношень – комп'ютерна мережа (відображує інформаційний зв'язок комп'ютерів), "любовний трикутник" (відображує відносини між людьми), схема взаємодії підрозділів підприємства.

3.1.1. Основні елементи графів

Граф $G\{V, E\}$ складається з двох множин – множини V об'єктів (***вершин***, вузлів) і множини E зв'язків (***ребер***).

Односторонні зв'язки зображуються направленими ребрами, які називають ***дугами***. Граф з дугами називають ***орієнтованим*** або орграфом. Якщо хоча б одне ребро графа має орієнтацію, то граф є орієнтованим.

Часто граф описується відображенням $\Gamma(x_1)=\{x_2\}$. Відображення точки x_1 – набір точок, з якими точка x_1 пов'язана ребрами, причому ребра виходять з точки x_1. Для рис.3.1 $\Gamma(x_3)=\{x_1, x_4\}$.

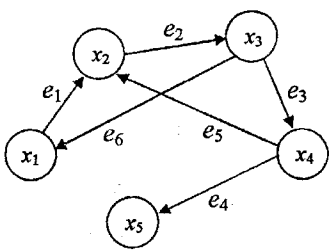


Рис. 3.1. Приклад графа

Дуги називаються ***суміжними***, якщо вони мають спільні кінцеві вершини. Вершини x_i та x_j називаються суміжними, якщо в графі має місце дуга $a(x_i, x_j)$ або $a(x_j, x_i)$ при цьому вершини називаються ***інцидентними*** цьому ребру, а ребро - інцидентним цим вершинам.

Орієнтованим ланцюгом називається послідовність дуг, в якій кожна дуга починається з вершини, у якій закінчується попередня дуга.

Простим орієнтованим ланцюгом називається ланцюг, в якому кожна вершина використовується не більше одного разу.

Петля – це ребро, в якому початкова і кінцева точки збігаються.

Шлях – послідовність суміжних ребер, причому кінцева точка попереднього ребра є початковою точкою наступного ребра.

Шлях e_1, e_2... e_q називається замкненим, якщо початкова вершина ребра e_1 та кінцева вершина ребра e_q збігаються. Такий шлях називають ***циклом***.

Контур - замкнений шлях в орграфі; контур називається простим або елементарним, якщо жодна вершина в ньому не зустрічається двічі.

Породженим підграфом графа $G(X, E)$ називається граф, вершини якого задовольняють умову $X_s \subset X$ і при цьому зберігаються всі дуги, що сполучають вершини X_s.

3.1.2. Види графів

Граф називається ***зваженим***, якщо кожному його ребру приписана певна числова характеристика (графи, які не характеризуються вагою ребер називаються ***незваженими***).

Наприклад, для графів в евклідовому просторі вагою є відстань між вершинами (довжина ребра), для графів в просторі станів – час переходу з одного стану в інший, для графів в просторі логічних зв'язків – "інтенсивність" зв'язку.

Граф називається ***повним***, якщо кожні дві різні його вершини з'єднані одним і тільки одним ребром. Для задання повного графа досить знати число його вершин. Граф, що не є повним, можна перетворити в повний з тими ж вершинами, додавши відсутні ребра.

Вершини в графі можуть відрізнятися одна від одної тим, скільком ребрам вони належать.

Степенем вершини називається число ребер графа, яким належить ця вершина. *Вершина* називається *парною*, якщо її степінь - число парне, і називається *непарною*, якщо її степінь - число непарне.

Два графа незважених G_1 і G_2 називаються ***ізоморфними***, якщо існує взаємно однозначна відповідність між множинами їх вершин, яка має властивість: число ребер, що з'єднують будь-які дві вершини в G_1, дорівнює числу ребер, що з'єднують відповідні вершини в G_2.

3.1.3. Операції над графами

На практиці часто зустрічаються графи, які будуються з деякого початкового графа за допомогою вилучення однієї з його вершин або

одного з його ребер. Існують і інші можливі перетворення графів, які розглядаються як операції над графами. Оскільки граф визначається як сукупність двох пов'язаних множин (вершин і ребер), то операції над графами формулюються переважно через відповідні операції над множинами. Розглянемо основні операції над графами і деякі їх властивості.

1. Операція вилучення ребра

Нехай $G=(V,E)$ — граф і $e\in E$ — деяке його ребро. Говорять, що граф $G_1=G-e$ одержано з графа G внаслідок операції вилучення ребра e, якщо $G_1=(V,E\setminus\{e\})$. Отже, кінці ребра e не вилучаються з множини V.

Неважко показати, що для довільних ребер e і e_1 графа G виконується така тотожність: $(G-e)-e_1=(G-e_1)-e$.

Отже, якщо виконується підряд кілька операцій вилучення ребра, то результат не залежить від порядку вилучення ребер з графа (рис. 3.2).

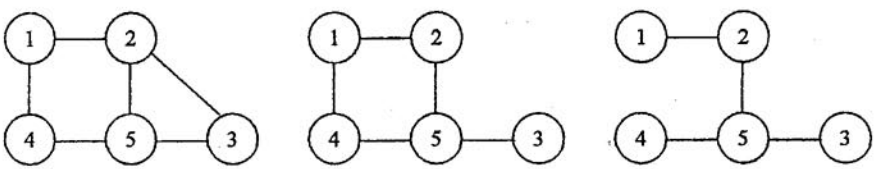


Рис. 3.2. Вилучення ребра: а) граф G; б) граф $G\setminus\{(2,3)\}$;в) граф $(G\setminus\{(2,3)\})\setminus\{(1,4)\}$.

2. Операція вилучення вершини

Нехай $G=(V,E)$ і $v\in V$. Говорять, що граф $G_1=G-v$ одержаний з графа G внаслідок операції вилучення вершини v, якщо вершина v вилучена з V, а з E вилучені всі ребра, інцидентні з вершиною v.

Неважко переконатися, що операція вилучення вершини не залежить від порядку, в якому вилучаються вершини з графа (рис.3.3).

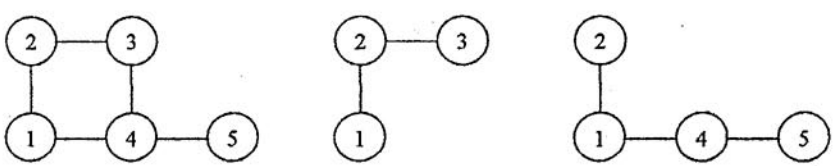


Рис. 3.3. Вилучення вершини: *а*- граф G; *б*-граф $G-\{4\}$; *в*-граф $G-\{3\}$.

Операції вилучення ребра, вершини і переходу до підграфа — це операції, за допомогою яких можна з початкового графа одержувати графи з меншим числом вершин і ребер.

Видалення вершини v_0 з графа $G(V,E)$ зводиться до знаходження доповнення графа $G_1(V_1,E_1)=G(V,E)\setminus G_0\left(\{v_0\},\left\{\forall_i[e_{i0},e_{0i}]\right\}\right)$.

3. Операція введення ребра

Якщо $u,v\in V$ і $(u,v)\notin E$ в графі $G=(V,E)$, то граф $G+e=(V,E\cup\{e\})$, де $e=(u,v)$.

Внаслідок комутативності операції об'єднання множин можна стверджувати, що послідовність операцій введення ребер у граф G не залежить від порядку, в якому ці ребра вводяться в граф G. Іншими словами, справедлива тотожність

$$\forall e,e_1\in E\quad((G+e)+e_1=(G+e_1)+e).$$

4. Операція введення вершини в ребро

Нехай (u,v) — деяке ребро графа G. *Введенням вершини* w *в ребро* (u,v) називається операція, внаслідок якої одержуємо два ребра (u,w) і (w,v), а ребро (u,v) при цьому вилучається з графа G.

5. Операція об'єднання графів

Граф F називається об'єднанням графів $G=(V,E)$ *і* $H=(V_1,E_1)$, якщо $F=(V\cup V_1,E\cup E_1)$. Граф F позначається $G\cup H$. Об'єднання графів $F=G\cup H$ називається *диз'юнктивним,* якщо $V\cap V_1=\varnothing$.

Безпосередньо з означення операції об'єднання графів випливає, що $(\forall G,H)(G\cup H=H\cup G)$.

Операція диз'юнктивного об'єднання графів дає можливість розглянути ще один важливий тип графів.

Граф є ***зв'язним,*** якщо його не можна подати у вигляді диз'юнктивного об'єднання двох підграфів, і ***незв'язним*** — у протилежному випадку.

Отже, всякий незв'язний граф можна зобразити у вигляді диз'юнктивного об'єднання скінченного числа зв'язних підграфів. Кожний із таких зв'язних підграфів називається ***компонентом зв'язності.***

Зв'язний регулярний граф степеня 2 є ***циклічним графом.*** Циклічний граф з n вершинами позначається C_n. Циклічний граф C_6 наведено на рис. 3.4.

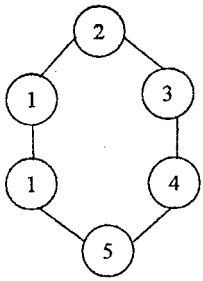


Рис. 3.4. Циклічний граф

6. Добуток графів

Добутком графів $G=(V,E)$ *і* $H=(V_1,E_1)$ називається граф $F=G\times H$, у якого $V=V\times V_1$, а E визначається таким чином: вершини $(u,\ u_1)$ і (v,v_1) суміжні в F тоді і тільки тоді, коли $u=v$, а u_1 і v_1 суміжні в H або $u_1=v_1$, а u і v суміжні в G. Приклад наведений на рис.3.5.

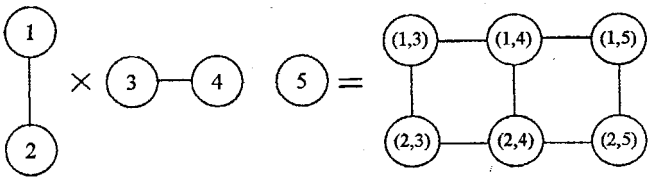


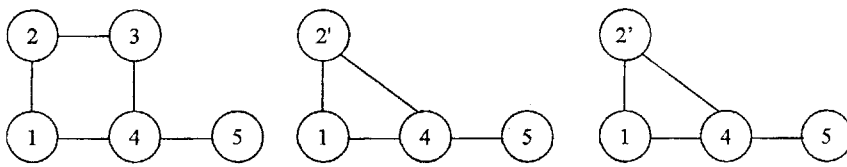
Рис. 3.5. Добуток графів

7. Ототожнення (злиття) вершин

Якщо $G=(V,E)$ — граф, u, v — дві його вершини, причому вершина v суміжна з вершинами $\{v_1,\ldots,v_l\}$, а вершина u суміжна з вершинами $\{u_1,\ldots,u_k\}$, то граф $H\left(V\setminus\{u,v\}\cup u',\ E\setminus\left\{a_u,\{u_i\},a_v,\{v_j\}\right\}\cup\left\{a_{u'},\{u_i\},a_{u'},\{v_j\}\right\}\right)$, одержаний приєднанням нової вершини u' до множини вершин замість вилучених вершин u, v і множини ребер $\{(u',u_i)\},\ \{(u',v_j)\}$ $(i=1,2,\ldots,k,$

$j = 1, 2, \ldots, l)$ замість вилучених $\{(u, u_i)\}, \{(v, v_j)\} (i = 1, 2, \ldots, k, j = 1, 2, \ldots, l)$, називається графом, одержаним із G ототожненням вершин u і v.

Операція стягування ребра (u, v) в графі $G=(V,E)$ означає ототожнення вершин u і v в графі G. Граф G називається графом, який *стягується до графа* H, якщо H можна одержати з G за допомогою деякої послідовності операцій стягування ребра.



а б в

Рис. 3.6. Стягування ребра: *а* — граф G; *б* — ототожнення вершин 2 і 3 в G; *в* — стягування ребра (2, 3) в G.

8. Операція роздвоєння (розщеплення) вершини

Нехай v — деяка з вершин графа G. Розіб'ємо множину суміжних з нею вершин довільним чином на дві частини — M і P, а потім виконаємо таке перетворення графа G: вилучимо вершину v разом з інцидентними їй ребрами і введемо дві нові вершини u і w разом з ребром, яке з'єднує ці вершини, вершину u з'єднаємо ребром з кожною вершиною множини M, а вершину w — з кожною вершиною з множини P. Одержаний граф позначимо G' і будемо вважати, що він одержаний з графа G внаслідок роздвоєння (розщеплення) вершини v (рис. 3.7).

9. Операція з'єднання графів

Нехай $G_1=(V_1,E_1)$ і $G_2=(V_2,E_2)$ — два графи, у яких множини вершин V_2 і V_1, не перетинаються, тобто $V_2 \cap V_1 = \varnothing$. *Операція з'єднання* графів G_2 і G_1, полягає в тому, що множини V_2 і V_1, об'єднуються, а потім з'єднуються ребрами: кожна вершина графа G_1 з кожною вершиною графа G_2

$$G = G_1 \overline{\cup} G_2 = G(V_1 \cup V_2, E_1 \cup E_2 \cup \{E(v_1, v_2) : v_1 \in V_1, v_2 \in V_2\}).$$

Очевидно, що операція з'єднання графів може бути виражена у вигляді добутку (суперпозиції) операції об'єднання графів G_2 і G_1, та послідовності операцій введення ребра.

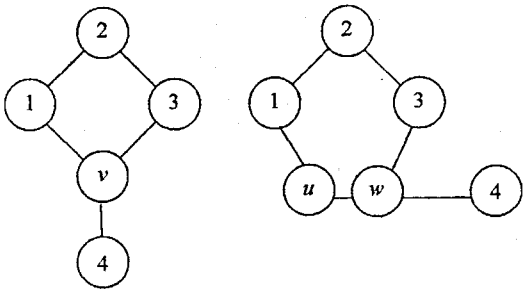


Рис. 3.7. Роздвоєння вершини v

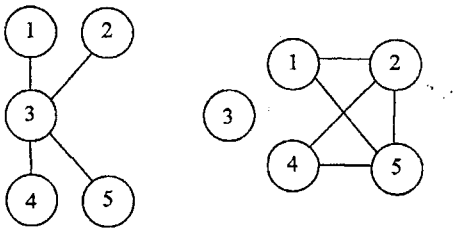
10. Перетин графів

Якщо дано граф $G_1=(V_1,E_1)$ і граф $G_2=(V_2,E_2)$, то перетин графів $G=G_1\cap G_2=G(V_1\cap V_2,E_1\cap E_2)$.

11. Операція доповнення (знаходження різниці) графів

Нехай $G=(V,E)$ — граф. *Доповненням* G^* графа G називається граф з множиною вершин V, в якому дві вершини суміжні тоді і тільки тоді, коли вони не суміжні в графі G (рис. 3.8). Звідси випливає, що коли граф G має n вершин, то граф G^* можна побудувати, вилучивши з повного графа всі ребра, які належать G (граф G вважається підграфом графа K_n).

Очевидно також, що доповнення повного графа є пустим графом і, навпаки, доповнення пустого графа є повним графом. Неважко довести, що доповнення регулярного графа є регулярним графом.



а б

Рис. 3.8. Доповнення графа: а – граф G; б – доповнення графа G

Якщо у топологічному просторі графів визначити метрику, то операція доповнення графів дозволить розглядати числову характеристику різниці графів з однаковою потужністю множини вершин. Одним з можливих способів чисельного оцінювання різниці може бути використання поняття *топологічної ентропії*, яке визначене у теорії топологічних просторів.

12. Додавання вершини

Додавання вершини v_0 до графа $G_1=(V_1,E_1)$ зводиться до з'єднання графів $G(V,E)=G_1(V_1,E_1)\cup G_0(\{v_0\},\phi)$;

Формалізовані в аналітичному вигляді операції над графами використовуються переважно для теоретичних досліджень структурних моделей та доведення правильності результатів алгоритмів, що призначені для розв'язання задач на графах.

3.2. Способи опису графів

Графічне зображення графа є зручним і наочним для людини, але графові моделі переважно використовуються в комп'ютерних алгоритмах. Оскільки комп'ютер оперує з числами, то для комп'ютерної обробки зручнішим є не графічне, а умовне числове подання графа.

Для опису графів використовуються різноманітні матриці та списки.

Найпоширеніші:

- матриця суміжності;
- матриця інциденції;
- списки пар вершин.

Матриця інциденції є головним з теоретичної точки зору способом опису графа, оскільки показує зв'язок між вершинами і ребрами і є просто табличною формою подання відповідності $G=(V,E)$, яка власне і є графом.

Якщо граф має n вершин і m ребер, то матриця інциденції має розмір $(n\times m)$. Якщо ребро графа ненаправлене, відповідний елемент матриці дорівнює 1. Якщо ребро (дуга) виходить з вершини, відповідний елемент матриці дорівнює 1. Якщо ребро (дуга) входить в вершину, відповідний елемент матриці дорівнює -1. Якщо немає ребра, яке пов'язує дві вершини, відповідний елемент матриці дорівнює 0

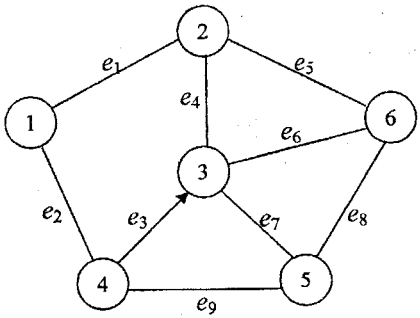


Рис. 3.9. Приклад графа

Матриця, яка описує граф на рис.3.9, має вигляд

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
v_1	1	1	0	0	0	0	0	0	0
v_2	1	0	0	1	1	0	0	0	0
v_3	0	0	-1	1	0	1	1	0	0
v_4	0	1	1	0	0	0	0	0	1
v_5	0	0	0	0	0	0	1	1	1
v_6	0	0	0	0	1	1	0	1	0

Матриця суміжності для графа, який має n вершин, буде мати розмірність $(n \times n)$. Елемент матриці $A[i,j]=1$, якщо є ребро або дуга з вершини i в вершину j і $A[i,j]=0$, якщо немає такого ребра.

Якщо $A[i,j]=A[j,i]=1$, то між вершинами i та j знаходиться ребро. Якщо $A[i,j] \neq A[j,i]$, то між вершинами i та j є дуга.

Матриця суміжності неорієнтованого графа симетрична відносно головної діагоналі. Якщо граф орієнтований (коли хоча б одне ребро графа є дугою), то матриця суміжності буде несиметричною.

Матриці суміжностей — досить природний спосіб задання графа в пам'яті ЕОМ. Якщо граф має n вершин, то для його задавання необхідно мати масив розміром n^2. У тому випадку, коли матриця симетрична або симетрична і головна її діагональ нульова, то це скорочує необхідний об'єм пам'яті до $1/2 \cdot n \cdot (n-1)$ елементів. Часто буває так, що багато елементів матриці суміжностей нульові і більша частина зайнятої пам'яті стає

зайвою. Незважаючи на це, багато задач на графах при такому заданні розв'язується досить просто і ефективно. Але існує цілий ряд задач, які розв'язуються більш ефективно, якщо задавати графи в пам'яті ЕОМ іншими способами, наприклад списками.

Матриця суміжності для графа на рис.3.9

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Задання графа у вигляді списків можна виконати різними способами. Один з них — задання графа *списками суміжностей*. У цьому випадку з кожною вершиною графа u зв'язується список вершин $L(u)$, суміжних з цією вершиною.

Другий спосіб спискового задання графа — це *таблиця списків суміжностей* . Для рис.3.9:

e_1 : 1-2; e_2 : 1-4; e_3 : 4-3; e_4 : 2-3; e_5 : 2-6; e_6 : 3-6; e_7 : 3-5; e_8 : 5-6; e_9 : 4-5

Вибір того чи іншого способу задання графа залежить від алгоритму розв'язання задачі.

Для опису зважених графів використовуються *матриці вагів*. Особливістю застосування матриці вагів є неоднозначний запис ваги відсутніх ребер. Якщо, наприклад, матриця вагів використовується для пошуку найкоротшого шляху в графі, то відсутнім ребрам приписується нескінченна вага, а якщо найдовшого – то нульова.

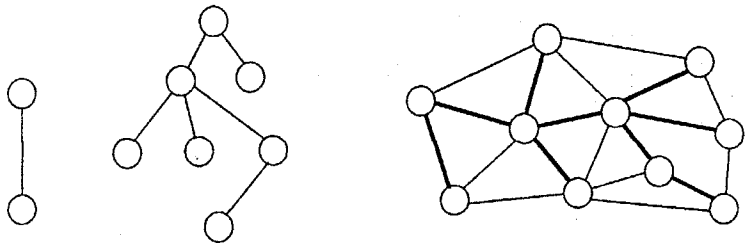
Операції над графами змінюють розмір матриць, що подають графи, тому вони є нелінійними з точки зору перетворень матриць. Використовують також ізоморфні перетворення графів, які змінюють тільки нумерацію вершин.

3.3. Дерева графів і доповнення

Найпоширеніший різновид графів — дерева. Процес розподілу карток при каталогізації, складання словників і енциклопедій, розподіл

устаткування по підприємствах, цехах і ділянках і багато аналогічних задач описуються графами типу дерев.

Найпростіше дерево має тільки одне ребро (рис.3.10,а). Дерево з n вершинами має (n-1) ребер.



а) б) в)

Рис.3.10. Приклади дерев: а) найпростіше, б) кореневе, в) кістякове дерево, виділене у графі

Зв'язний ациклічний граф називається (неорієнтованим) ***деревом***. Дерево називається ***кореневим,*** якщо в ньому виділена вершина, яка називається ***коренем*** (рис.3.10,б).

Кістяковим деревом графа G називається підграф графа G, який містить усі вершини графа (але, можливо, не всі ребра) і є деревом (рис.3.10,в). Інші назви - *кістяк, стягуюче дерево, каркас*. Мінімальне кістякове дерево – кістякове дерево, яке має мінімальну сумарну вагу (довжину) ребер.

Безпосередньо з означення дерева випливає, що граф є деревом тоді і тільки тоді, коли будь-які дві його вершини зв'язані лише одним ланцюгом. Звідси випливають властивості дерева:

1. Якщо T - дерево і u - його кінцева вершина, то граф $(T - u)$ — дерево. Дійсно, граф $(T - u)$ — підграф дерева T, для якого виконуються всі умови дерева.

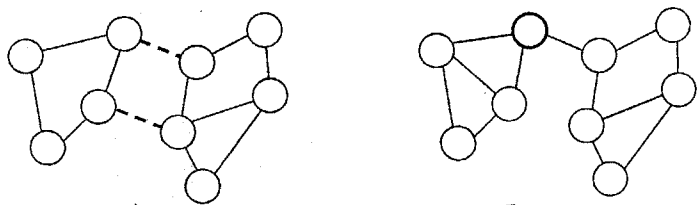
2. Всяке непусте дерево має щонайменше дві кінцеві вершини і одне кінцеве ребро.

3. Ребро зв'язного графа називається *суттєвим,* якщо його вилучення веде до порушення зв'язності цього графа. В дереві кожне ребро суттєве.

3.4. Розрізи, суграфи

При розв'язуванні багатьох задач на графах доводиться розбивати граф на частини. Прикладом таких задач є розбиття складної принципової схеми електронного пристрою на блоки. У теорії графів такі розбиття називають розрізами.

Розріз - множина ребер зв'язного графа, видалення яких приводить до незв'язного графа (рис. 3.11,а).



а) б)

Рис. 3.11. Розрізи: а) розрізаючі ребра;
б) розрізаюча вершина

Розрізаюча вершина - вершина x графа G така, що після її видалення множина $V(G)\backslash\{x\}$ розбивається на непусті підмножини V' і V'', що не перерізаються і між якими немає ребер графа G (рис. 3.11,б).

Простий розріз - розріз, у якого будь-яка власна підмножина елементів не є розрізом.

Суграф - частина графа, утворена видаленням з вихідного графа деяких ребер. Кількість вершин графа й суграфа однакова, тобто суграфом $G'(X', U')$ графа $G(X, U)$ називають граф, для якого $X'=X$, $U'\subset U$ (рис. 3.11,б).

Цикломатичне число — ізоморфна характеристика графа G

$$\lambda(G) = m(G) - n(G) + \chi(G),$$

де $n(G)$ — кількість його вершин, $m(G)$ — кількість ребер, $\chi(G)$ - кількість компонент.

Основні властивості цикломатичного числа:

- $\lambda(G) \geq 0$;
- $\lambda(G) = 0$ тоді і тільки тоді, коли граф не містить циклів;
- при $\lambda(G) > 0$ на G можна видалити $\lambda(G)$ ребер таким чином, щоб суграф, який залишиться, не мав циклів і мав попередню кількість компонент; будь-який суграф, отриманий із G шляхом видалення

меншої кількості ребер, містить цикли.

Будь-який суграф G', який задовольняє умови

$$\chi(G') = \chi(G), \quad m(G') = m(G) - \lambda(G), \quad \lambda(G') = 0,$$

називається ***каркасом графа G***, а видалені ребра хордами G (відносно G'). Кожна компонента каркаса є деревом, яке містить всі вершини відповідної компоненти графа G.

3.5. Мережі. Потоки в мережах

Мережа – це технічний об'єкт, яким розповсюджуються потоки. Прикладами таких мереж є водогінна мережа, комп'ютерна мережа, мережа автомобільних доріг тощо.

Потік – це кількість речовини, енергії або інформації, яка проходить через переріз за одиницю часу.

У математичному розумінні мережею називають зважений граф, у якому задані ***потоки*** $v(x,y)$ і ***пропускні здатності*** ребер $c(x,y)$ - максимальні кількості потоку, які можуть проходити через ребра від витоку до стоку, де x – початкова вершина ребра, y – кінцева вершина.

Значення потоків визначаються функцією потоку $f(x,y)$, причому

$$f(x,y) \le c(x,y). \tag{3.1}$$

В мережі існує три типи вузлів:

- *виток* S, з якого виходить більше потоку, ніж входить до нього;
- *сток* T, в який входить більше потоку, ніж виходить з нього;
- *проміжні вузли*, в які скільки виходить потоку, стільки ж і входить.

Приклад мережі показаний на рис.3.12. На рисунку кожне ребро охарактеризоване пропускною спроможністю (перша цифра) і величиною потоку (друга цифра). Наведена мережа має один виток, два стоки і 9 проміжних вузлів.

Для мережі можна записати певні співвідношення, які повинні задовольняти потоки через ребра:

1) головна умова існування мережі

$$V_{ij} \le C_{ij}, \quad i, j = 1..N. \tag{3.2}$$

2) умова балансу потоків у проміжному вузлі (сума потоків, які входять з усіх інших вершин в j-ту вершину дорівнює сумі потоків, що виходять з j вершини)

$$\sum_{\substack{i=1 \\ i \neq j \\ j \neq S \\ j \neq T}}^{N} V_{ij} = \sum_{\substack{i=1 \\ i \neq j}}^{N} V_{ji} . \qquad (3.3)$$

3) умова витоку (сума всіх потоків від витоку до всіх інших вершин більша ніж сума всіх потоків, що входять від і-тих вершин до витоку)

$$\sum_{\substack{i=1 \\ i \neq S}}^{N} V_{Si} > \sum_{\substack{i=1 \\ i \neq S}}^{N} V_{iS} . \qquad (3.4)$$

4) умова стоку (сума всіх потоків до стоку від усіх інших вершин більша ніж сума всіх потоків, що виходять зі стоку)

$$\sum_{\substack{i=1 \\ i \neq T}}^{N} V_{Ti} < \sum_{\substack{i=1 \\ i \neq T}}^{N} V_{iT}$$

5) умова збереження кількості потоку в мережі

$$\sum_{\substack{i=1 \\ i \neq S}}^{N} (V_{Si} - V_{iS}) = \sum_{\substack{i=1 \\ i \neq T}}^{N} (V_{iT} - V_{Ti})$$

Дослідженням мереж присвячено багато спеціальної літератури.

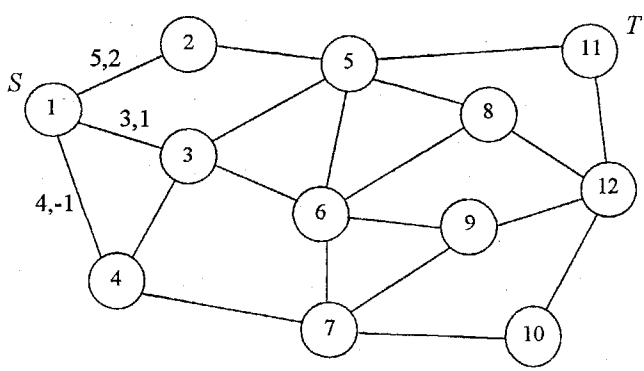


Рис. 3.12. Приклад мережі

3.6. Алгоритми на графах

Як вже зазначалося, графи широко використовуються для розв'язання різноманітних задач з використанням комп'ютерів. Це підтверджується тим, що задачам на графах присвячена половина фундаментальної праці Р.Седжвіка "Фундаментальні алгоритми на С".

Основні задачі, які вирішуються на графах:

1) пошук шляху в графі – будь-якого або найкоротшого (для незважених графів найкоротший шлях – шлях, який складається з мінімальної кількості ребер; для зважених графів – це шлях з мінімальною сумарною довжиною ребер).

2) пошук циклів в графі: будь-яких, мінімальної та максимальної довжини.

3) пошук гамільтонового циклу (циклу, який проходить через всі вершини графа) та ейлерового циклу (циклу, який проходить через всі ребра графа).

4) побудова дерев графа, тобто виділення таких вершин і ребер, які утворюють підграф типу дерево, в тому числі кістякового дерева.

5) пошук критичного шляху у графі (шляху максимальної довжини);

6) задача про максимальний потік: знайти такий розподіл потоків у мережі, який забезпечить максимальний сумарний потік з витоку до стоку.

Розглянемо найпоширеніші алгоритми розв'язання цих задач. Запис алгоритмів мовою Паскаль наведений у додатку.

3.6.1. Пошук шляху в незваженому графі

Основні способи пошуку шляху у незважених графах:

1) способи перебору варіантів;
2) пошук в глибину;
3) пошук в ширину.

Переборні методи

Переборні методи основані на геперуванні перестановок вершин і перевірці, чи є згенерована перестановка необхідним шляхом.

Пошук шляху в графі в глибину

Головний інструмент алгоритму пошуку в глибину – стек.

Стек – структура даних, яка діє за правилом: останнє занесене дане першим вилучається (LIFO – last input – first output). Схема стека показана на рис.3.13.

Використовуються 2 способи програмної реалізації стека:

1) використання динамічних структур даних (у випадку, коли максимальна кількість елементів у стеку заздалегідь невідома);

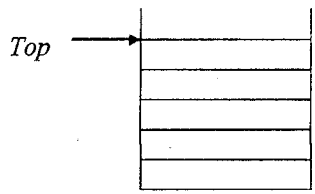


Рис. 3.13. Схема стека

2) імітація стека за допомогою масиву (у випадку, коли максимальна кількість елементів у стеку заздалегідь відома).

На рис.3.14 показана схема імітації стека за допомогою масиву.

Програмна реалізація стека визначається відображенням правил занесення та вилучення даних у відповідних програмних процедурах.

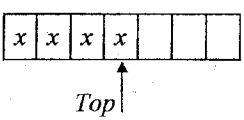


Рис. 3.14. Імітація стека за допомогою масиву

Процедура занесення даного в стек
(*x* – дане, яке заноситься в стек)

```
Procedure InSteck (x: byte);
  begin
   inc(Top);
   steck[Top]:=x;
  end;
```

Процедура вилучення даного зі стеку

```
Procedure DelSteck;
  begin
   dec(Top);
  end;
```

Операція перевірки, чи не є стек пустим

```
Function FreeSteck:  boolean;
   begin
     FreeSteck:=(Top=0);
    end;
```

Ініціалізація стека

```
Const N=8;  {кількість вершин графа, максимальна довжина стека}
Var
  steck : array [1..N] of byte;
  Top:  byte;
```

Алгоритм пошуку шляху в глибину ґрунтується на таких кроках:

1) починаючи від початкової вершини, будується шлях у графі, використовуючи ребра з мінімальним номером кінцевої вершини. Послідовність вершин заноситься у стек;

2) якщо на цьому шляху не зустрінеться кінцева вершина, тоді зі стеку вилучається остання занесена вершина і береться ребро з більшим номером кінцевої вершини;

3) якщо невикористаних ребер немає, знову зі стеку вилучається вершина (робиться крок назад);

4) якщо стек став пустим, а шлях не знайдено, то його немає взагалі.

Переваги та недоліки методу пошуку в глибину: цей метод дозволяє знайти шлях у незваженому графі, знайдений шлях не є найкоротшим, але метод у середньому забезпечує найбільшу швидкість.

Пошук шляху у графі в ширину

Цей метод дозволяє знайти найкоротший шлях у незваженому графі.

Основний засіб для здійснення пошуку в ширину – структура даних *черга.*

Черга – структура даних, яка діє за правилом: перший занесений першим вилучається (FIFO: first input – first output).

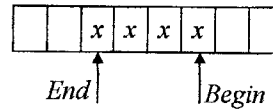


Рис. 3.15. Імітація стека за допомогою масиву

Недоліки:

- програма громіздка;
- великі затрати пам'яті (на кожен елемент виділяється 16 байт).

Перевага:

- не потрібно знати кількість елементів черги.

Найбільша можлива довжина черги дорівнює кількості вершин графа.

Існує 2 способи організації черги за допомогою масиву:

1) при вилученні елемента з черги, всі елементи черги пересуваються до початку масиву(недолік – багато зайвих операцій);

2) робота з чергою як з циклічним списком.

Процедура ініціалізації черги

```
Procedure Init;
begin
  E:=0;
  B:=0;
  end;
```

Процедура додавання елемента в чергу

```
Procedure Dod (x: integer);
begin
  if (E=0) and (B=0) then
    begin
      inc (E);
      inc (B);
      Q[1]:=x;
    end
else if E<N then
    begin
```

```
        inc (E);
        Q[E+1]:=x;
      end
else Q[E+1]:=x;
end;
```

Процедура вилучення елемента з черги

```
Procedure Del;
 begin
   if B=N then B:=1 else inc(B);
 end;
```

Алгоритм пошуку шляху в ширину ґрунтується на таких кроках:

1) вершини при пошуку в ширину заносяться в *чергу*. Елемент черги складається з двох частин: номера вершини та номера елемента черги, звідки до цієї вершини потрапили;

2) до черги заноситься початкова вершина, а у другу частину елемента – 0 (ознака початку шляху);

3) розглядаються *всі* вершини, суміжні з початковою, і заносяться до черги;

4) якщо серед розглянутих вершин немає кінцевої – пересунути вказівник початку черги і розглянути вершини, суміжні з тією, на яку вказує вказівник;

5) дійшовши таким чином до кінцевої вершини, пройти шлях у зворотному порядку, використовуючи другу частину елементів черги.

3.6.2. Пошук шляху в зваженому графі

Якщо в графі враховуються ваги ребер, то очевидно, це ставить задачу пошуку шляху з мінімальною або максимальною сумарною вагою.

Пошук шляху в зваженому графі методом перебору

Метод перебору є найпростішим і найнеефективнішим. Він дозволяє знайти розв'язок у будь-якому випадку (якщо розв'язок взагалі існує), але кількість варіантів, які необхідно перебрати, найчастіше надто велика.

Пошук шляху в зваженому графі методом гілок та границь

Метод дозволяє скоротити кількість варіантів у порівнянні з методом перебору. Застосовується в задачах, де критерій пошуку є монотонно зростаючою функцією кількості ребер.

Ідея методу: знаходиться перший шлях, що задовольняє умову задачі. При пошукові іншого шляху з додаванням до нього чергового ребра перевіряємо, чи не став вже цей частковий шлях довшим за початковий (порівнюємо з границею). Якщо став, то далі нарощувати його немає сенсу – він вже заздалегідь гірший, і всі наступні варіанти (гілки), що мають своїм початком цей частковий шлях, не розглядаються.

Алгоритм Дейкстри дозволяє знайти найкоротші шляхи від заданої вершини до кожної з решти вершин графа.

Ідея алгоритму: якщо безпосередній шлях від вершини V_i до V_j більший за шлях $V_i \rightarrow V_k \rightarrow V_j$, то шлях $V_i \rightarrow V_j$ заміняють на шлях $V_i \rightarrow V_k \rightarrow V_j$.

Побудова мінімального кістякового дерева

Прикладом використання задачі побудови мінімального кістякового дерева є прокладання телефонної мережі між заданими пунктами.

Найчастіше для побудови мінімального кістякового дерева використовується алгоритм Пріма-Краскала. Ними доведено, що жадібний алгоритм дозволяє знайти мінімальне кістякове дерево. Це алгоритм, який на кожному етапі розв'язання задачі обирає найкращий варіант з точки зору критерію задачі.

Алгоритм Пріма-Краскала. Ідея алгоритму: спочатку вибирають найкоротше ребро, а його кінцеві вершини включають до дерева – це початок дерева. Потім приєднують до нього найкоротше ребро з тих, що виходять з вершин, які належать до дерева, а входять у вершину, яка не належить до дерева. І так далі, поки всі вершини не увійдуть до складу дерева.

Задача може мати декілька розв'язків, якщо є ребра однакової довжини.

Задача про максимальний потік

Найчастіше для розв'язання задачі використовується алгоритм Форда-

Фолкерсона:

1)якщо мережа має декілька витоків та декілька стоків, то вона перетворюється в мережу з одним витоком і одним стоком шляхом введення уявних витоку і стоку;

2)початок розв'язання задачі полягає у тривіальному розв'язку (коли всі потоки рівні 0);

3)виконується збільшення потоку за допомогою "збільшувальних ланцюгів". "Збільшувальний ланцюг" – це така послідовність вершин і ребер, яка починається з витоку і закінчується стоком, і в усіх ребрах на цьому шляху потік менший за пропускну здатність, тобто може бути збільшений.

Доведено, що алгоритм буде сходитись, тобто потоки будуть наближатись до максимального значення, якщо процес збільшення починати з найкоротшого "збільшувального ланцюга".

4)виконується збільшення потоку у "збільшувальному ланцюзі" на величину:

а) якщо граф неорієнтований

$$\min (C_{ij}-V_{ij}) \text{ для } V_{ij} > 0,$$
$$\min (C_{ij}+V_{ij}) \text{ для } V_{ij} < 0.$$

б) якщо граф орієнтований, потрібно враховувати, що напрямок дуги змінити не можна.

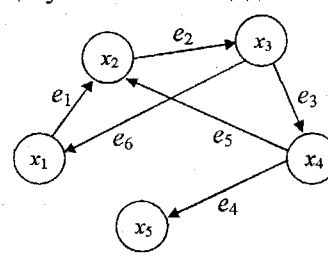
Оскільки для забезпечення збіжності алгоритму потрібно починати з найкоротшого збільшувального ланцюга, то в основу перебору ланцюгів кладуть метод пошуку в ширину, який не закінчується пошуком першого найкоротшого шляху, а продовжується далі.

Контрольні питання і завдання для самостійної роботи

1. Дайте означення: а)шляху, орієнтованого ланцюга; б) петлі, контуру; в) повного графа, ізоморфного графа.
2. Дайте означення: а) ейлерового шляху, ейлерового циклу; б) гамільтонового шляху, гамільтонового циклу; в) цикломатичного числа; г) розрізу в графі, суграфа.
3. Визначте та графічно інтерпретуйте операції над графом: а) вилучення ребра та вершини; б) введення ребра та введення вершини в ребро; в) об'єднання графів, добуток (перетин) графів; г) ототожнення (злиття)

вершин, стягування ребра; д) роздвоєння (розщеплення) вершини, операція з'єднання графів; г) доповнення (знаходження різниці) графів, додавання вершини.

4. Який граф називається: а) зв'язним; б) орієнтованим?
5. Які властивості мають зв'язні, ейлерові та гамільтонові графи?
6. Поясніть призначення різноманітних матриць та списків, які використовуються для опису графів.
7. Наведіть означення дерева та кістякового дерева.
8. Дайте означення: а) розрізу, розрізаючої вершини; б) суграфа та цикломатичного числа з його основними властивостями.
9. Охарактеризуйте різні типи вузлів у мережах та наведіть основні співвідношення, які повинні задовольняти потоки через ребра
10. Назвіть основні задачі, які вирішуються на графах.
11. Охарактеризуйте та поясніть суть алгоритмів: а) перебору варіантів; б) пошуку в глибину; в) пошуку в ширину.
12. Коротко характеризуйте основні алгоритми пошук шляху в зваженому графі.
13. Побудуйте матриці суміжності та інцидентності для наведеного графа.



14. Знайдіть гамільтонів та ейлерів ланцюг для графа з попереднього прикладу.
15. Дано матрицю суміжності. Перетворити її на список ребер та зобразити відповідний граф.
16. 8 дипломатів треба розсадити за столом переговорів. Кожен дипломат може сидіти лише поряд з декількома з решти. Сумісність дипломатів задається матрицею. Скласти програму розміщення дипломатів методом перебору.
17. На комп'ютері необхідно виконати N завдань, які вимагають T_1, T_2, ... T_N часу. Одночасно комп'ютер може виконувати K завдань. Скласти програму створення плану виконання завдань.

18. Локальна комп'ютерна мережа складається з N сегментів. У кожному сегменті 1 сервер та К робочих станцій. Сервери зв'язані між собою. Структура зв'язків задається матрицею суміжності. Скласти програму знаходження маршруту передавання інформації між двома робочими станціями а) методом перебору; б) методом пошуку в глибину; в) методом пошуку в ширину.
19. Поштар розносить пошту. Скласти програму знаходження для нього найкоротшого маршруту. Відстані від поштового відділення до будинків та між будинками задаються матрицею. Використати метод гілок та границь.
20. Турист, виїжджаючи з дому, хоче якнайшвидше дістатися до морського курорту. Відстані між містами у атласі автомобільних доріг задаються таблицею. Скласти програму знаходження найкоротшого маршруту.

Рекомендована література

1. Дубовой В.М. Моделювання систем контролю та керування. - Вінниця: ВНТУ, 2005. – 175 с.
2. Сигорский В.П. Математический аппарат инженера. – К.: Техніка, 1975 – 786 с.
3. Капітонова Ю.В. та інш. Основи дискретної математики. – К.: Наукова думка, 2002. – 579 с.
4. Лапа В.Г. Математические основы кибернетики. – К.: Вища школа, 1974. - 452 с.
5. Бондарев В.М. и др. Основы программирования. – Харьков: Фолио, 1997.
6. Кристофидес Н. Теория графов. - М., 1978.
7. Кузин Л.Т. Основы кибернетики. Основы кибернетических моделей. - М., 1977.
8. Основы моделирования сложных систем / Под ред. И.В.Кузьмина. - К.: Вища школа, 1981.
9. Питерсон Дж. Теория сетей Петри и моделирование систем. - М.: Мир, 1984.
10. Свами М.Н., Тхуласираман К. Графы, сети и алгоритмы. - М.: Мир, 1984.

11. Глонь О.В., Дубовой В.М., Мітюшкін Ю.І. Комп'ютеризовані системи керування. – Вінниця: ВНТУ, 2005. – 157 с.
12. Дубовой В.М., Квєтний Р.Н. Програмування комп'ютеризованих систем управління та автоматики. – Вінниця: ВДТУ, 1997.
13. Дубовой В.М., Квєтний Р.Н. Програмування персональних комп'ютерів систем керування. – Вінниця: ВДТУ, 1999.
14. Йенсен П., Барнес Д. Потоковое программирование. - М.: Радио и связь: 1984.
15. Коршунов Ю.М. Математические основы кибернетики. – М.: Энергия, 1972. – 376 с.
16. Липский В. Комбинаторика для программистов. – М.: Мир, 1988. – 195 с.
17. Новиков Ф.А. Дискретная математика для программистов. - СПб.: Питер, 2001.
18. Питерсон Дж. Теория сетей Петри и моделирование систем. - М.: Мир, 1984.
19. Седжвик Р. Фундаментальные алгоритмы на С. Алгоритмы на графах: Пер. с англ. – СПб.: ДиаСофтЮП, 2003. – 480 с.
20. Орэ О. Теория графов. - М.: Наука, 1980.

4. ВИПАДКОВІ ПРОЦЕСИ

Теорія випадкових процесів є одним з найважливіших розділів математики для фахівців з систем управління і автоматики, оскільки переважна більшість процесів у САУ мають стохастичну складову.

Даний розділ підготовлений з використанням матеріалів [1 - 3, 9, 16, 17, 19-21, 25, 33].

4.1. Випадкові величини і випадкові процеси

Одним з основних об'єктів, які вивчає наука про випадкове, є ***випадкова величина.*** Випадкова величина розглядається в ймовірнісному просторі (див. розділ 1.6), який визначається трійкою (Ω, F, P), де Ω — - довільна множина, елементи якої називаються елементарними подіями; F - сігма-алгебра підмножин Ω, які називають (випадковими) подіями; P - ймовірнісна міра або ймовірність, така що $P(\Omega)=1$.

Випадковий процес – це випадкова величина, яка змінюється у часі, або випадкова функція часу. Відповідно, випадковий процес характеризується четвіркою (Ω, F, P, t).

На відміну від детермінованих процесів, протікання яких визначено однозначно, випадковий процес показує зміну в часі фізичної системи, яку неможливо передбачити наперед. Кількісно випадковий процес описується функцією часу, яка в будь-який момент часу може приймати різні значення з заданим розподілом ймовірностей.

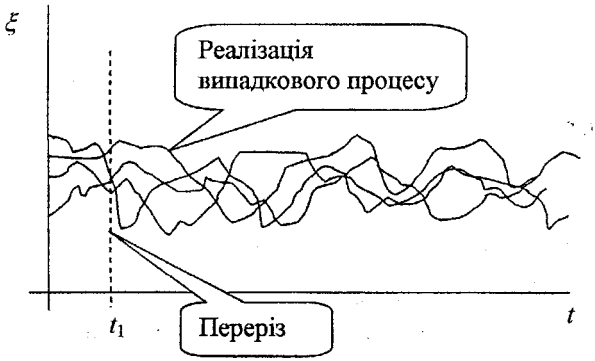


Рис. 4.1. Реалізації випадкового процесу

Детермінований процес має одну реалізацію, функція часу має вигляд $S(t)$. Випадкова функція часу визначається сукупністю функцій часу і законами, що характеризують властивості сукупності. Кожна з функцій цієї сукупності називається реалізацією випадкової функції і позначається $\xi^{(k)}(t)$, де $k \in N$.

В залежності від того, належать можливі значення часу t і реалізація $\xi(t)$ дискретній множині чисел чи відрізку дійсної осі, розрізняють 4 ***типи випадкових процесів***:

1) випадковий процес загального типу, тобто t і $\xi(t)$ можуть приймати будь-які значення на відрізку дійсної осі (рис.4.2,а);

2) випадковий процес з дискретними значеннями, коли t - неперервна, $\xi(t)$ - дискретна (рис.4.2,б);

3) випадкова послідовність, коли t - дискретна, $\xi(t)$ - неперервна;

4) дискретна випадкова послідовність, коли і t, і $\xi(t)$ - дискретні.

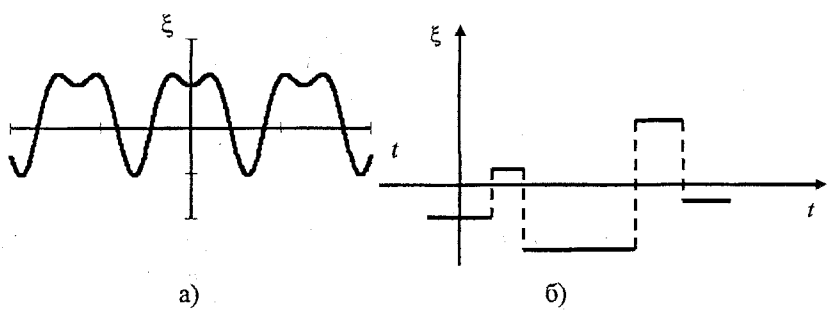


Рис. 4.2. Види випадкових процесів

4.2. Характеристики випадкових процесів

Розглянемо N реалізацій випадкової функції, виділимо з цього числа n_1 реалізацій, значення яких в момент часу t_1 менші числа x_1. При достатньо великому числі N відносна частка $\frac{n_1(x_1,t_1)}{N}$ буде статистично стійкою, тобто при зростанні кількості реалізацій буде залишатися рівною одному сталому числу. Це число називається ймовірністю того, що при $t=t_1$ випадкова функція $\xi(t)$ знаходиться нижче рівня x_1. Вказана ймовірність, як і число n_1, залежить від фіксованого моменту часу та від обраного рівня,

тобто це функція двох змінних $F_1(x_1,t_1)=P\{\xi(t_1)\le x_1\}$, яку називають ***функцією розподілу ймовірностей випадкового процесу***.

Похідна від функції розподілу є головною характеристикою випадкового процесу $\frac{dF_1(x_1,t_1)}{dx_1}=f_1(x_1,t_1)$ і називається ***щільністю ймовірності або розподілом випадкового процесу***. Типові щільності ймовірності випадкових процесів наведені у табл. 4.1 і табл. 4.2.

Для повної характеристики випадкового процесу необхідно використовувати багатовимірну диференціальну функцію розподілу ймовірності. Процес у кожний наступний момент часу повинен розглядатися як залежний від значень процесу у всі попередні моменти.

Візьмемо деякі фіксовані значення t_1, t_2, ..., t_n, ... аргументу t випадкового процесу $\xi(t)$. Тоді отримаємо сукупність випадкових величин $\xi(t_1)$, $\xi(t_2)$, ..., $\xi(t_n)$, що є його перетинами в цих точках. Якщо зафіксованих значень багато і відстані між ними при цьому будуть взяті невеликі, то закономірності випадкового процесу $\xi(t)$ досить точно визначаються сумісним розподілом ймовірностей цих випадкових змінних. Сумісні розподіли, отриманих таким чином випадкових змінних, називаються ***скінченновимірними розподілами*** даного випадкового процесу $\xi(t)$.

Візьмемо два фіксовані довільні значення t_1, t_2 аргументу t. Нехай при цьому перетини випадкового процесу $\xi(t)$ є випадковими змінними $\xi(t_1)$, $\xi(t_2)$. Тоді сумісний розподіл $f(x_1, x_2; t_1, t_2)$ випадкового вектора $(\xi(t_1), \xi(t_2))$ є функцією двох аргументів. Маючи *двовимірний закон розподілу* $f(x_1, x_2)$ випадкового процесу $\xi(t)$, можна знайти і його одновимірний закон розподілу $f(x_1)$

$$f(x_1)=\int_{-\infty}^{+\infty} f(x_1,x_2)dx_2 \qquad (4.1)$$

Розглянемо два випадкових процеси $x(t)$ та $y(t)$. Якщо величини X та Y пов'язані, то ймовірність отримання ними пари значень $P(x,y)$. Розглядаючи випадкову величину x як причину, а y – як наслідок, можна розглядати умовну ймовірність $P(y/x=x_1)$.

Має місце ***теорема Байєса***

$$P(x,y)=P(x)\cdot P(y/x)=P(y)\cdot P(x/y).$$

Аналогічно для неперервних випадкових величин:

$$f(x,y)=f(x)\cdot f(y/x)=f(y)\cdot f(x/y).$$

Таблиця 4.1

Типові розподіли ймовірності дискретних процесів

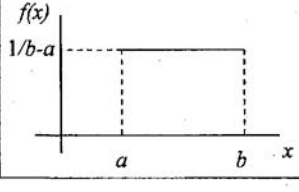
Назва	Вираз	Математичне сподівання	Дисперсія
1	2	3	4
Вироджений	$\begin{cases} 0, x \le a, \\ 1, x < a \end{cases}$	a^k	0
Бернуллі	$\begin{cases} 0, x \le 0, \\ 1-p, 0 < x \le 1 \\ 1, x > 1 \end{cases}$	p	$p(1-p)$
Біноміальний	$\begin{cases} \sum_{k=1}^{l} C_n^k p^k (1-p)^{n-k}, l < x \le l+1; \\ 1, x > n; 0, x \le 0, \end{cases}$	np	$np(1-p)$
Паскаля (біноміальний від'ємний)	$\begin{cases} \sum_{k=1}^{l} C_{r+k-1}^k p^r (1-p)^k, l < x \le l+1; \\ 1, x > n; 0, x \le 0 \end{cases}$	$r(1-p)/p$	$r(1-p)/p^2$
Геометричний	$p(1-p)^k$, $0<p<1$	$(1-p)/p$	$(1-p)/p^2$

Продовження табл. 4.1

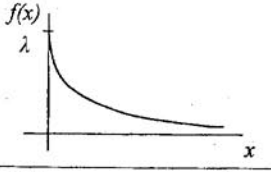
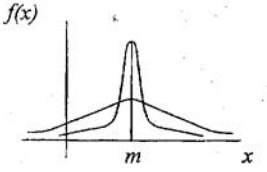
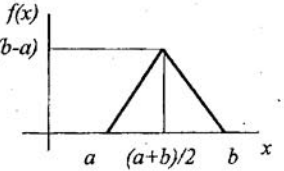
1	2	3	4
Гіпер-геометричний	$\frac{C_{Np}^{k} C_{N(1-p)}^{n-k}}{C_{N}^{n}}$, $0<p<1$	np	$\frac{N-n}{N-1} np(1-p)$
Пуассона	$\frac{\lambda^{k}}{k!} e^{-\lambda}$, $\lambda>0$	λ	λ
Логарифмічний	$-\frac{(1-p)^{k}}{k \ln p}$, $0<p<1$	$\frac{1-p}{p \ln p}$	$-\frac{1-p}{p^{2} \ln p}\left[1+\frac{1-p}{\ln p}\right]$
Бореля-Таннера	$\frac{r}{(k-r)!} k^{k-r-1} e^{-\alpha k} \alpha^{k-r}$, $(0<\alpha<1)$	$r/(r-\alpha)$	$\alpha r/(1-\alpha)^{3}$

Таблиця 4.2

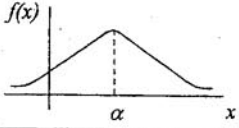
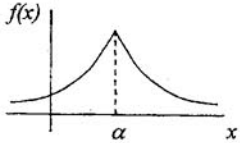
Типові розподіли ймовірності неперервних процесів

Назва	Вираз	Математичне сподівання	Дисперсія	Графічний вигляд
1	2	3	4	5
Рівномірний	$f(x)=\frac{1}{b-a}$	$\frac{a+b}{2}$	$\frac{(b-a)^{2}}{12}$	

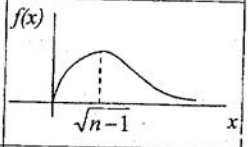
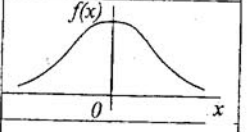
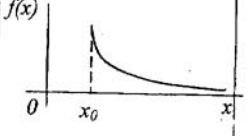
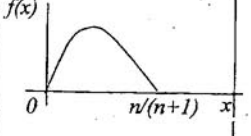
Продовження табл. 4.2

1	2	3	4	5
Показниковий (експоненціальний)	$\begin{cases}\lambda e^{-\lambda x}, x \geq 0 \\ 0, x < 0\end{cases}$	$\frac{k!}{\lambda^k}$	$\frac{1}{\lambda^2}$	
Нормальний (Гауса)	$\frac{1}{\sqrt{2\pi}\sigma} e^{-(x-m)^2/2\sigma^2}, x \in (-\infty, \infty$	m	σ^2	
Трикутний (Сімпсона)	$\begin{cases}\frac{2}{b-a} - \frac{2}{(b-a)^2}\lvert a+b-2x\rvert, \\ x \in [a,b] \\ 0, x \notin [a,b]\end{cases}$	$\frac{4}{(b-a)^2(k+1)(k+2)} \times \times \left[a^{k+2} + b^{k+2} - 2\left(\frac{a+b}{2}\right)^{k+2}\right]$	$\frac{(b-a)^2}{24}$	
Гамма	$\begin{cases}\frac{\lambda^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\lambda x}, x > 0 \\ 0, x \leq 0\end{cases}$	$\frac{\alpha(\alpha+1)\ldots(\alpha+k-1)}{\lambda^k}$	$\frac{\alpha}{\lambda^2}$	При різних значеннях α та λ набуває різного вигляду

Продовження табл. 4.2

1	2	3	4	5
Бета	$\begin{cases}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}, x\in[0,1]\\ 0, x\notin[0,1]\end{cases}$	$\frac{\Gamma(\alpha+k)\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\alpha+\beta+k)}$	$\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$	При різних значеннях α та β набуває різного вигляду
Коші	$\frac{1}{\pi}\cdot\frac{\lambda}{\lambda^2+(x-\alpha)}$	----------	----------	
Лапласа (подвійний експоненціальний)	$\frac{\lambda}{2}e^{-\lambda\lvert x-\alpha\rvert}, x\in(-\infty,\infty)$	$\alpha^{2k+1}(2k+1)!;$ $[\frac{\alpha^{2k}}{(2k)!}+\frac{\alpha^{2(k-1)}}{2(k-1)!\lambda^2}+\ldots$ $+\frac{1}{\lambda^{2k}}](2k)!$	$\frac{2}{\lambda^2}$	
χ^2	$\begin{cases}\frac{1}{2^{\alpha/2}\Gamma(\alpha/2)}x^{\alpha/2-1}e^{-x/2}, x>0\\ 0, x\le 0\end{cases}$	$\alpha(\alpha+2)\ldots[\alpha+2(k-1)]$	2α	При різних значеннях α набуває різного вигляду

Продовження табл. 4.2

1	2	3	4	5
χ	$\begin{cases} \dfrac{1}{2^{\alpha/2}\Gamma(\alpha/2)} x^{\alpha-1} e^{-x/2}, x>0 \\ 0, x \le 0 \end{cases}$	$\dfrac{2^{k/2}\Gamma((\alpha+k)/2)}{\Gamma(\alpha/2)}$	$\alpha + (2-\sqrt{2}) \cdot \cdot \left[\dfrac{\Gamma((\alpha+1)/2)}{\Gamma(\alpha/2)}\right]^2$	
Стьюдента (t-розподіл)	$\dfrac{\Gamma((\alpha+1)/2)}{\sqrt{\alpha\pi}\Gamma(\alpha/2)}\left(1+\dfrac{x^2}{\alpha}\right)^{-(\alpha+1)/2}$	$\dfrac{\alpha^k \Gamma(\alpha/2-k)\Gamma(k+1/2)}{\sqrt{\pi}\Gamma(\alpha/2)}$, $2k<\alpha$	$\begin{cases} \alpha/(\alpha-2), \alpha>2 \\ \infty, \alpha \le 2 \end{cases}$	
Парето	$\begin{cases} \dfrac{\alpha}{x_0}\left(\dfrac{x_0}{x}\right)^{\alpha+1}, x>x_0 \\ 0, x \le x_0 \end{cases}$	$\dfrac{\alpha}{\alpha-k} x_0^k, k<\alpha$	$\begin{cases} \dfrac{\alpha}{(\alpha-1)(\alpha-2)} x_0^2, \\ \alpha>2 \\ \infty, \alpha \le 2 \end{cases}$	
Шермана	$\begin{cases} \sum\limits_{m=1}^{n} m b_m x^{m-1}, x \in \left[0, \dfrac{n}{n+1}\right] \\ 0, x \notin \left[0, \dfrac{n}{n+1}\right] \end{cases}$	$\left(1-\dfrac{1}{n+1}\right)^{n+1}$	$\dfrac{2n^{n+2}+n(n-1)^{n+2}}{(n=2)(n+1)^{n+2}} - \left(1-\dfrac{1}{n+1}\right)^{2(n+1)}$	

Відповідно багатовимірна диференціальна функція розподілу ймовірності може бути знайдена за рекурсивною формулою

$$f[x_1(t_1), x_2(t_2), x_3(t_3)\ldots]=f[x_1(t_1)]\cdot f[x_2(t_2)/\,x_1]\cdot f[x_3(t_3)/\,x_1, x_2]$$

У багатьох випадках для *характеристики випадкового процесу* достатньо знати окремі числові характеристики розподілів ймовірності, які називаються моментами:

1) початкові моменти n-го порядку

$$M_n\{\xi(t)\} = m_n(t) = \int_{-\infty}^{+\infty} x^n f(x,t)dx$$

Найчастіше використовується перший початковий момент (математичне сподівання)

$$m_1(t) = \int_{-\infty}^{+\infty} xf(x,t)dx$$

або для процесу з дискретним часом

$$m_1(t) = \frac{\sum_{i=1}^{n} x_i(t)}{n}$$

Кожен випадковий процес при фіксованому значенні аргументу є деякою випадковою змінною, а невипадкова функція - постійною величиною. Тому легко довести такі *властивості математичного сподівання випадкового процесу*:

- математичне сподівання невипадкової функції $\varphi(x)$ є цією ж функцією, тобто:

$$E(\varphi(t))= \varphi(t); \qquad (4.2)$$

- якщо множником є невипадкова функція, то її можна виносити за знак математичного сподівання, тобто:

$$E(\varphi(t)\,\xi(t))= \varphi(t)\cdot E(\xi(t))= \varphi(t)\cdot E_\xi(t) \qquad (4.3)$$

де $\varphi(t)$ - невипадкова функція, а $\xi(t)$ - випадковий процес;

- математичне сподівання суми випадкових процесів $\xi(t)$ та $\eta(t)$ рівне сумі їх математичних сподівань, тобто:

$$E_{\xi+\eta}(t)= E_\xi(t)+ E_\eta(t) \qquad (4.4)$$

- якщо при всіх значеннях аргументу $t \in T$ перетини випадкових процесів $\xi(t)$ та $\eta(t)$ є незалежними випадковими змінними, то математичне сподівання добутку цих процесів рівне добутку їх математичних сподівань, тобто:

$$E_{\xi\cdot\eta}(t)= E_\xi(t)\cdot E_\eta(t) \qquad (4.5)$$

2) центральні моменти

$$M\{[\xi(t)-m_1(t)]^n\}=D_n(t)=\int_{-\infty}^{+\infty}[x-m_1(t)]^n f(x,t)dx$$

Найчастіше використовується другий центральний момент (дисперсія)

$$D_2(t)=\int_{-\infty}^{+\infty}[x-m_1(t)]^2 f(x,t)dx=\sigma_x^2(t)$$

або для процесу з дискретним часом

$$\sigma_x^2(t)=\frac{\sum_{i=1}^{n}[x_i(t)-m_1(t)]^2}{n-1}$$

Арифметичний корінь з дисперсії випадкового процесу називається *середнім квадратичним відхиленням, флуктуацією, стандартом* цього процесу.

Властивості дисперсії випадкового процесу:

- дисперсія випадкового процесу є невід'ємнозначною функцією, тобто:

$$D(\xi(t))=D_\xi(t)\geq 0; \tag{4.6}$$

- дисперсія невипадкової функції $\varphi(t)$ дорівнює нулю, тобто:

$$D(\varphi(t))=0; \tag{4.7}$$

- додавання невипадкової функції $\varphi(t)$ до випадкового процесу $\xi(t)$ не змінює дисперсії цього процесу, тобто:

$$D(\xi(t)+\varphi(t))=D_\xi(t); \tag{4.8}$$

- якщо множником є невипадкова функція, то її, піднісши до квадрату, можна виносити за знак дисперсії, тобто:

$$D(\varphi(t)\cdot\xi(t))=(\varphi(t))^2\cdot D_\xi(t), \tag{4.9}$$

де $\xi(t)$ - випадковий процес;

- якщо, при всіх значеннях аргументу $t_i\in T$ перетини випадкових процесів $\xi(t)$ та $\eta(t)$ є незалежними випадковими змінними, то дисперсія суми цих процесів рівна сумі їх дисперсій, тобто:

$$D_{\xi+\eta}(t)=D_\xi(t)+D_\eta(t) \tag{4.10}$$

3) другий змішаний початковий момент (кореляційна функція)

$$B_{x_1x_2}(t_1,t_2)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}x_1x_2f_2(x_1,x_2,t_1,t_2)dx_1dx_2$$

4) другий змішаний центральний момент (коваріаційна функція)

$$R_{x_1x_2}(t_1,t_2)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}[x_1-m_1(t_1)][x_2-m_2(t_2)]f_2(x_1,x_2,t_1,t_2)dx_1dx_2$$

Примітка. Властивості коваріаційних і кореляційних функцій розглянемо у підрозділі.4.4.

Основні класи випадкових процесів

Основною ознакою, за якою класифікуються випадкові процеси, є властивість їх скінченновимірних розподілів. При такій класифікації, в першу чергу, звертається увага на природу утворення процесу, динаміку його зміни при зміні твірного параметра, на внутрішні зв'язки між перетинами цього процесу та їх вид. Тому перейдемо безпосередньо до аналізу випадкових процесів за цими основними ознаками.

Процеси з незалежними приростами

Нехай $t_0<t_1, \ldots,<t_n$, а $\eta_0=\xi(t_0),\eta_1=\xi(t_1)-\xi(t_0),\ldots,\eta_n=\xi(t_n)-\xi(t_{n-1})$, де $\xi(t)$ - випадковий процес.

Якщо всі випадкові змінні $\xi(t_0), \eta_1, \eta_2, \ldots, \eta_n$ незалежні, то випадковий процес $\xi(t)$ називається ***процесом з незалежними приростами.***

Прикладом такого процесу є ***броунівський рух (вінерівський процес).***

Якщо $T=\{0, 1, 2,\ldots\}$, то знаючи розподіли випадкових змінних $\xi(t_0), \eta_1, \eta_2, \ldots$, можемо визначити сумісний розподіл будь-якої множини випадкових змінних $\xi(t)$. Дійсно, $\xi(t_i)=\xi(t_0)+\eta_1+\ldots+\eta_i$, $(i=0,1,2,\ldots)$. Отже, в цьому випадку процес з незалежними приростами є просто послідовністю незалежних випадкових змінних $\xi(t_0), \eta_1, \eta_2, \ldots$.

Серед процесів з незалежними приростами можна виділити ***загальні пуассонівські процеси***. Це процеси з незалежними приростами, в яких всі перетини $\xi(t_i)$ є випадковими змінними, розподіленими за законом Пуассона з параметром $\lambda(t_i)$, які є додатнозначними функціями.

Процеси з незалежними значеннями

Випадковий процес називається ***процесом n-го порядку,*** якщо він повністю визначається своїми функціями скінченновимірних розподілів $F_{(n)}(x,t)$ порядку n, але не визначається функціями скінченновимірних розподілів порядку $n-1$.

Випадковий процес $\xi(t)$ називається ***процесом з незалежними значеннями, цілком випадковим процесом,*** якщо випадкові величини $\xi(t_1)$, $\xi(t_2)$, ... є взаємонезалежними для кожної скінченної множини $t_1, t_2, \ldots$ значень аргументу з його області визначення T.

З означення випливає, що всі скінченновимірні функції розподілу цілком випадкового процесу визначаються одновимірними функціями розподілу відповідних його перетинів таким чином:

$$f_{t_1,\ldots,t_n}(x_1,x_2,\ldots,x_n)=\prod_{i=1}^{n} f_{t_i}(x_i) \tag{4.11}$$

тобто, такий процес є процесом першого порядку. Якщо перетини цілком випадкового процесу є дискретними випадковими змінними, то цей процес повністю визначається умовною ймовірністю $P_{t_1}(x_1)=P(\xi(t_1)=x_1)$, а коли його перетини є неперервними випадковими змінними – то умовною густиною $f_{t_1}(x_1)$.

Процеси з незалежними значеннями розглядаються, в основному, для дискретного випадку. Прикладами цілком випадкових процесів є: послідовність випробувань в схемі Бернуллі, випадковий відбір у статистиці тощо.

Марковські випадкові процеси

Випадковий процес $\xi(t)$ називається ***марковським***, або ***процесом без післядії,*** якщо ймовірність будь-якої події, пов'язаної з протіканням цього процесу в майбутньому, не змінюється, якщо ми, знаючи його теперішній стан, враховуємо додатково інформацію про поведінку цього процесу в минулому.

Детальніше марковські процеси розглянуто нижче.

Мартингали

Однією з основних характеристик випадкового процесу є його математичне сподівання. Ця характеристика визначається через математичне сподівання перетинів процесу. Тому природу випадкового процесу часто можна ефективно описати, акцентуючи увагу на еволюції цієї величини в процесі зростання значень його аргументу.

Випадковий процес $\xi(t)$ називається *мартингалом,* якщо для будь-яких значень часу $t_1<t_2<\ldots<t_{n-1}$ умовне математичне сподівання перетину $\xi(t_{n+1})$ при умові, що в ці моменти часу він перебував у станах $x_1, x_2, \ldots, x_n$, відповідно, дорівнює значенню процесу в момент часу t_n при всіх допустимих значеннях $x_1, x_2, \ldots, x_n$, тобто:

$$E(\xi(t_{n+1}) \mid \xi(t_1)=x_1, \ \ldots, \ \xi(t_n)=x_n)=x_n \tag{4.12}$$

Отже, мартингали є процесами близькими до марковських випадкових процесів.

4.3. Поняття стаціонарності та ергодичності

Процеси, в яких статистичні характеристики, отримані при усередненні на основі множини реалізацій, не змінюються у часі називають ***стаціонарними***.

Процеси, в яких не зберігаються статистичні характеристики, називаються ***нестаціонарними***.

Розділяють два типи стаціонарності процесів:

процеси стаціонарні в "цілому" (закон розподілу ймовірності не залежить від часу) Випадковий процес $\xi(t)$ називається *стаціонарним у вузькому розумінні*, якщо функція розподілу $f_n(x_1,x_2,...,x_n,t_1,t_2,...,t_n)$ довільного порядку n не змінюється при будь-якому зсуві τ всієї групи точок $t_1,\ldots,t_n$ вздовж осі часу. Тобто, для будь-якого n

$$f_n(x_1,x_2,...,x_n,t_1,t_2,...,t_n)=f_n(x_1,x_2,...,x_n,t_1+\tau,t_2+\tau,...,t_n+\tau);$$

- процеси частково стаціонарні (не залежать від часу тільки окремі моменти розподілів). *В широкому розумінні*, випадкові **стаціонарні процеси** – це процеси, в яких середнє значення і дисперсія не залежать від часу, а кореляційна функція залежить лише від різниці часу $\tau = t_2 - t_1$.

Для нормальних процесів поняття стаціонарності і в широкому, і в строгому розумінні збігаються.

Для стаціонарного випадкового процесу характерно:

1) одновимірна функція розподілу має один і той же вигляд в будь-який момент часу

$$f_1(x_1,t+\tau)=f_1(x);$$

2) двовимірна функція розподілу залежить лише від різниці часу

$$f_2(x_1,x_2,t_1,t_2)=f(x_1,x_2,t_2-t_1);$$

3) кореляційна функція залежить лише від τ

$$B(\tau)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}x_1x_2f(x_1,x_2,\tau)dx_1dx_2,$$

де τ - різниця часу.

Випадковий процес називається ***ергодичним***, якщо будь-яка його ймовірнісна характеристика, одержана усередненням за множиною реалізацій, з ймовірністю, близькою до 1, дорівнює часовому середньому, яке одержано усередненням за достатньо великий проміжок часу з однієї реалізації.

Загальна ергодична теорема

Якщо математичне сподівання випадкового процесу $\xi(t)$ є постійною величиною і його кореляційна функція задовольняє умову

$$\lim_{T\to\infty}\frac{1}{T^2}\int_a^b\int_a^b B_\xi(t_1,t_2)dt_1dt_2=0, \qquad (4.13)$$

то середнє значення цього випадкового процесу при необмеженому розширенні інтервалу осереднення $[a,b]$ прямує в середньому квадратичному до математичного сподівання цього процесу, і навпаки, якщо границя в середньому квадратичному середнього значення стохастичного процесу з постійним математичним сподіванням, при необмеженому розширені інтервалу осереднення, дорівнює математичному сподіванню цього процесу, то його кореляційна функція задовольняє умову (4.13).

Стаціонарні випадкові процеси, для яких виконується ергодична теорема, називаються ергодичними. Стаціонарний процес $\xi(t)$ ергодичний, якщо для будь-якого $\varepsilon>0$ існує таке T_0, що

$$|B_\xi(\tau)|<\varepsilon \text{ при } |\tau|>T_0 \qquad (4.14)$$

тобто, якщо модуль кореляційної функції цього процесу прямує до нуля при $|\tau|\to\infty$. Умова (4.14) є достатньою умовою того, що стаціонарний випадковий процес $\xi(t)$ ергодичний.

4.4. Кореляційний аналіз

Найважливішим питанням аналізу різноманітних процесів є дослідження причинно-наслідкового зв'язку між ними. У математиці розглядають переважно два види зв'язку:

- ***функціональний зв'язок*** – однозначна відповідність елементів однієї множини елементам іншої множини. Між x та y існує функціональна залежність, якщо кожному можливому значенню x відповідає однозначно визначене значення y;

- ***статистичний зв'язок*** – має місце тоді, коли елементи однієї множини відповідають елементам іншої множини з певною ймовірністю. Статистичний зв'язок полягає в тому, що одна випадкова змінна реагує на зміну другої зміною свого закону розподілу ймовірності.

Кореляційний аналіз розглядає питання глибини статистичного зв'язку між випадковими процесами. Кореляційний аналіз ґрунтується на визначенні другого змішаного моменту двох випадкових процесів, зсунутих один від

одного на певний інтервал часу. Найчастіше серед цих процесів один є причиною, а інший наслідком.

Нагадаємо, що існує два види других змішаних моментів:

- коваріаційна функція

$$B_{xy}(t_1,t_2)=\int_{-\infty}^{+\infty}[x(t_1)-m_x]\cdot[y(t_2)-m_y]\cdot f(x,y)dxdy;$$

- кореляційна функція

$$R_{xy}(t_1,t_2)=\int_{-\infty}^{+\infty}x(t_1)y(t_2)f(x,y)dxdy.$$

Між кореляційною і коваріаційною функціями існує зв'язок

$$B_{xy}(t_1,t_2)=\int_{-\infty}^{+\infty}[x(t_1)-m_x]\cdot[y(t_2)-m_y]\cdot f(x,y)dxdy=$$

$$=\int_{-\infty}^{+\infty}[x(t_1)y(t_2)-x(t_1)m_y-m_xy(t_2)+m_xm_y]\cdot f(x,y)dxdy= \qquad (4.15)$$

$$=R_{xy}(t_1,t_2)-m_xm_y$$

З означення кореляційної функції та властивостей випадкового процесу отримуємо такі властивості:

1. *Кореляційна функція симетрична відносно своїх аргументів, тобто при перестановці аргументів вона не змінюється:*

$$R_\xi(t_1,t_2)=R_\xi(t_2,t_1) \qquad (4.16)$$

2. *Кореляційна функція випадкового процесу $\xi(t)$ при рівності її аргументів є дисперсією цього процесу, тобто, якщо $t=t_1=t_2$, то*

$$R_\xi(t,t)=D_\xi(t) \qquad (4.17)$$

3. *Додавання до випадкового процесу $\xi(t)$ невипадкової функції $f(t)$ не змінює його кореляційної функції, тобто, якщо $\eta(t)=\xi(t)+f(t)$, то*

$$R_\eta(t_1,t_2)=R_\xi(t_1,t_2) \qquad (4.18)$$

зокрема, для центрованого випадкового процесу $(\xi(t)-m(t))$

$$E\{[\xi(t_1)-m(t_1)][\xi(t_2)-m(t_2)]\}=B(t_1,t_2)-m(t_1)m(t_2)$$

4. *При множенні випадкового процесу $\xi(t)$ на невипадкову функцію $\varphi(t)$, його кореляційна функція множиться на $\varphi(t_1)$ $\varphi(t_2)$, тобто, якщо $\eta(t)=\xi(t)\cdot\varphi(t)$, то*

$$R_\eta(t_1,t_2)=R_\xi(t_1,t_2)\cdot\varphi(t_1)\cdot\varphi(t_2) \qquad (4.19)$$

5. *Модуль кореляційної функції випадкового процесу є не більшим середнього геометричного дисперсій відповідних перетинів цього процесу, тобто*

$$|R_\xi(t_1, t_2)| \le \sqrt{D_\xi(t_1) \cdot D_\xi(t_2)} \qquad (4.20)$$

6. *Кореляційна функція є парною* $R(\tau)=R(-\tau)$

Враховуючи (4.15), з властивостей кореляційної функції можна сформулювати властивості коваріаційною функції.

Передумови використання кореляційного аналізу:

- змінні величини повинні бути випадковими;
- випадкові величини повинні мати спільний нормальний розподіл.

Взаємна кореляційна функція двох стохастичних процесів $\xi(t)$ і $\eta(t)$ оцінює ступінь залежності їх пар всіх можливих перетинів. Значення її виражається в квадратних одиницях. Тому часто ступінь цієї ж залежності виражають безрозмірною характеристикою, так званою ***нормованою взаємною кореляційною функцією (коефіцієнтом кореляції):***

$$\rho_{\xi\eta}(t_1,t_2) = \frac{B_{\xi\eta}(t_1,t_2)}{\sqrt{B_\xi(t_1,t_2)} \cdot \sqrt{B_\eta(t_1,t_2)}} = \frac{B_{\xi\eta}(t_1,t_2)}{\sqrt{D_\xi(t_1) \cdot D_\eta(t_2)}} \qquad (4.21)$$

Нормована взаємна кореляційна функція має властивості:

1. *Коефіцієнт кореляції не змінюється, якщо аргументи і випадкові процеси поміняти місцями, тобто*

$$\rho_{\xi\eta}(t_1, t_2) = \rho_{\eta\xi}(t_2, t_1). \qquad (4.22)$$

2. *Від додавання невипадкових функцій до випадкових процесів їх коефіцієнт кореляції не змінюється, тобто якщо* $\xi(t)$, $\eta(t)$, $\tau(t)$, $\theta(t)$ - *стохастичні процеси;* $f(t)$, $g(t)$- *невипадкові функції і* $\tau(t)= \xi(t)+f(t)$, $\theta(t)=\eta(t)+g(t)$, *то*

$$\rho_{\tau\theta}(t_1, t_2) = \rho_{\xi\eta}(t_1, t_2),$$

3. *При множенні випадкових процесів* $\xi(t)$ *і* $\eta(t)$ *на невипадкові функції* $f(x)$ *та* $g(x)$, *відповідно, їх коефіцієнт кореляції множиться на добуток* $f(x) \cdot g(x)$ *цих невипадкових функцій, при відповідних значеннях аргументів, тобто*

$$\rho_{\tau\theta}(t_1, t_2) = f(t_1) \cdot g(t_2) \cdot \rho_{\xi\eta}(t_1, t_2), \qquad (4.23)$$

де $\tau(t)= \xi(t) \cdot f(t)$, $\theta(t)=\eta(t) \cdot g(t)$.

4. *Модуль коефіцієнта кореляції не більший за одиницю, тобто*

$$|\rho_{\xi\eta}(t_1,t_2)| \le 1 \qquad (4.24)$$

Для двох стаціонарних і стаціонарно зв'язаних випадкових процесів кореляційна функція непарна $B_{\xi\eta}(\tau) \neq B_{\xi\eta}(-\tau)$

Для стаціонарних випадкових процесів взаємні коваріаційна та кореляційна функції не залежать від моментів часу t_1 та t_2, а лише від їх різниці:

$$R_{xy}(\tau) = \int_{-\infty}^{+\infty} [x(t) - m_x] \cdot [y(t+\tau) - m_y] \cdot f(x,y) dxdy$$

$$B_{xy}(\tau) = \int_{-\infty}^{+\infty} x(t) y(t+\tau) f(x,y) dxdy$$

Для ергодичних процесів усереднення можна виконувати не за множиною реалізацій, а у часі, тоді:

$$R_{xy}(\tau) = \frac{1}{T} \int_0^T [x(t) - m_x] \cdot [y(t+\tau) - m_y] dt$$

$$B_{xy}(\tau) = \frac{1}{T} \int_0^T x(t) y(t+\tau) dt$$

Вибіркова коваріація для процесу з дискретним часом

$$K_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

Вибірковий коефіцієнт кореляції

$$r = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n \sigma_x \sigma_y}$$

Типові кореляційні функції показані на рис.4.3.

1. Якщо коефіцієнт кореляції $\rho > 0$, то залежність між змінними прямо пропорційна.

2. Якщо коефіцієнт кореляції $\rho < 0$, то залежність між змінними обернено пропорційна.

3. Якщо величини незалежні, то $\rho = 0$.

Якщо $\rho = 0$, то випадкові величини є некорельовані, але це не означає, що вони незалежні, між ними може існувати нелінійна залежність. Таким чином залежні випадкові процеси можуть бути некорельовані.

Процеси є незалежними, якщо:

$$f(x,y) = f(x) \cdot f(y), \text{ або } f(y/x) = f(y).$$

Для характеристики повної залежності використовують коефіцієнт детермінації

$$D = 1 - \rho^2.$$

Коефіцієнт кореляції характеризує лінійний зв'язок між випадковими процесами, але не всі залежності є лінійними.

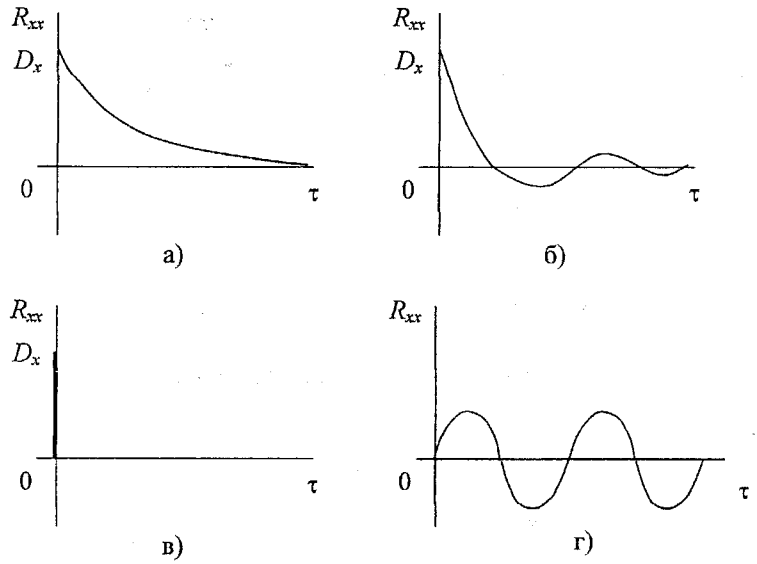


Рис. 4.3. Типові кореляційні функції: а) процес загального типу; б) процес з періодичною складовою; в) некорельований (цілком випадковий) процес; г) детермінований періодичний процес

Використовують також показник $\beta_{xy} = \frac{R_{xy}}{\sigma_x^2}$ - коефіцієнт, який оцінює міру чутливості однієї змінної до іншої. За його допомогою будується і перевіряється модель зв'язку одної залежної змінної (ендогенної) і більш незалежних (екзогенних) змінних.

Частковий коефіцієнт кореляції. Основні поняття кореляційного аналізу, введені для двовимірної моделі, можна поширити на багатовимірний випадок.

Нехай є багатовимірна нормальна сукупність, яка складається з m процесів X_1, X_2, ..., X_j..., X_m. У цьому випадку взаємозалежність між процесами можна описати *кореляційною матрицею*, яка складається з парних коефіцієнтів кореляції

$$Q_m = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1m} \\ \rho_{21} & 1 & \cdots & \rho_{2m} \\ \cdots & \cdots & \rho_{jk} & \cdots \\ \rho_{m1} & \rho_{m2} & \cdots & 1 \end{bmatrix} \quad (4.25)$$

Як і у двовимірному випадку, для оцінки коефіцієнта кореляції необхідно оцінити математичні сподівання й дисперсії. У багатовимірному кореляційному аналізі маємо m математичних сподівань і m дисперсій, а також $m(m—1)/2$ парних коефіцієнтів кореляції. Таким чином, потрібно зробити оцінку $2m+m(m-1)/2$ параметрів.

У випадку багатовимірної кореляції залежності між процесами більш різноманітні й складні, ніж у двовимірному випадку. Однією кореляційною матрицею не можна повністю описати залежності між процесами. Для цього використовується поняття часткового коефіцієнта кореляції *1-го порядку*.

Нехай вихідна сукупність складається з m процесів. Можна вивчати залежності між двома з них при фіксованому значенні l процесів з m-2 що залишилися. У цьому випадку маємо частковий коефіцієнт кореляції l-го порядку.

Розглянемо більш докладно структуру часткових коефіцієнтів кореляції на прикладі системи з трьох процесів X_1, X_2, X_3. Ця система дозволяє вивчити часткові коефіцієнти кореляції тільки 1-го порядку, тому що не можна фіксувати більше одного процесу. Частковий коефіцієнт кореляції першого порядку для процесів X_1 і X_2 при фіксованому значенні X_3 виражається через парні коефіцієнти кореляції й має вигляд

$$\rho_{12.3} = (\rho_{12} - \rho_{13}\rho_{23}) / \sqrt{(1-\rho_{13}^2)-(1-\rho_{23}^2)} \quad (4.26)$$

Частковий коефіцієнт кореляції, так само як і парний коефіцієнт кореляції, змінюється від -1 до +1. У загальному вигляді, коли система складається з m *процесів*, частковий коефіцієнт кореляції *1-го* порядку може бути знайдений з *кореляційної матриці*. Розглядається підматриця порядку $l+2$, складена з елементів матриці Q_m, які відповідають індексам коефіцієнта часткової кореляції.

Наприклад, кореляційна матриця системи з п'яти процесів має вигляд

$$Q_5 = \begin{bmatrix} 1 & \rho_{12} & \rho_{13} & \rho_{14} & \rho_{15} \\ \rho_{21} & 1 & \rho_{23} & \rho_{24} & \rho_{25} \\ \rho_{31} & \rho_{32} & 1 & \rho_{34} & \rho_{35} \\ \rho_{41} & \rho_{42} & \rho_{43} & 1 & \rho_{45} \\ \rho_{51} & \rho_{52} & \rho_{53} & \rho_{54} & 1 \end{bmatrix}$$

Для визначення часткового коефіцієнта кореляції другого порядку, наприклад $\rho_{12.45}$ слід використати підматрицю четвертого порядку, викресливши з вихідної матриці Q_5 третій рядок і третій стовпець, тому що процес X_3 не розглядають.

У загальному вигляді формулу часткового коефіцієнта кореляції l-го порядку (l=m—2) можна записати у вигляді:

$$\rho_{jk\cdot 1,2,\ldots,m} = Q_{jk} / (Q_{jj} Q_{kk})^{1/2} \qquad (4.27)$$

де Q_{jk} — алгебраїчні доповнення до елемента ρ_{jk} кореляційної матриці Q_m;

Q_{jj} і Q_{kk} — алгебраїчні доповнення до елементів ρ_{jj} і ρ_{kk} кореляційної матриці Q_m.

Очевидно, що вираз (4.26) є окремим випадком виразу (4.27).

Множинний коефіцієнт кореляції. Часто важливо оцінити зв'язок одного з процесів з усіма іншими. Це можна зробити за допомогою *множинного*, або *сукупного,* коефіцієнта кореляції

$$\rho_{j\cdot 1,2,\ldots,m} = \sqrt{1 - |Q_m| / Q_{jj}} \qquad (4.28)$$

де $|Q_m|$-визначник кореляційної матриці Q_m;

Q_{jj} — алгебраїчне доповнення до елемента ρ_{jj}.

Квадрат коефіцієнта множинної кореляції $\rho^2_{j\cdot 1,2,\ldots m}$ називається *множинним коефіцієнтом детермінації.* Коефіцієнти множинної кореляції й детермінації — величини додатні, приймають значення в інтервалі (0, 1).

4.5. Регресійний аналіз

Основна задача регресійного аналізу – вивчення виду залежності між двома випадковими процесами, з яких один є причиною, а інший наслідком.

Функція регресії випадкової змінної y відносно x (функція регресії y за x) – це умовне математичне сподівання $M(Y/X)$ випадкової змінної y, яка розглядається як функція певного класу, тобто

$$\bar{y}(x) = \varphi(x, b_1, b_2, \ldots, b_m) \qquad (4.29)$$

Вид залежності $\varphi(x)$ обирають, виходячи з візуальної оцінки характеру розташування точок на полі кореляції; досвіду попередніх досліджень тощо.

Крива регресії – емпіричне рівняння регресії, вона є відповідною оцінкою функції регресії.

Лінійний регресійний аналіз вивчає лише ті види залежності $\varphi(x)$, які лінійні за оцінюваними параметрами, хоча можуть бути нелінійні за змінною X.

Важливе місце в лінійному регресійному аналізі належить «нормальній регресії». Вона має місце, якщо зробити припущення, що закон розподілу випадкової величини Y є нормальним. За нормальної регресії є можливість оцінити значимість оцінок коефіцієнтів регресії, а також побудувати довірчий інтервал для коефіцієнтів регресій і умовного математичного сподівання.

Розглянемо найпростіший випадок лінійного регресійного аналізу, коли залежність $\varphi(x)$ лінійна і за оцінюваними параметрами, і за змінними. Двовимірна модель залежності y від x:

$$y = a + \beta x + e,$$

де a – константа, яка відображає значення y при x=0;

β - коефіцієнт регресії;

e – помилка, значення завади.

У лінійному регресійному аналізі умовне математичне сподівання $E(Y|X=x)$ подають у вигляді:

$$E(Y \mid X = x) = \varphi(x) = \beta_0 + \beta_1 x \qquad (4.30)$$

Оцінці в цьому виразі підлягають параметри β_0 та β_1, які називають коефіцієнтами регресії, а також $\sigma^2_{зал}$ - залишкова дисперсія.

Залишкова дисперсія може слугувати для оцінки точності підбору виду функції регресії, повноти набору факторів, включених в аналіз.

Оцінки параметрів моделі (4.30) β_0 і β_1, позначимо b_0 й b_1. Оцінку залишкової дисперсії $\sigma^2_{зал}$ позначимо $s^2_{зал}$. Підставивши у формулу (4.30) замість параметрів їх оцінки, одержимо рівняння регресії $\bar{y}(x) = b_0 + b_1 x_1$ Оцінки параметрів функції регресії знаходять, використовуючи метод найменших квадратів з умови мінімуму суми квадратів відхилень фактичних значень процесу y_i від обчислених за рівнянням регресії $y(x_i)$:

$$Q = \sum_{i=1}^{n} [y_i - y(x_i)]^2 = \min$$

$$\text{або } Q = \sum_{i=1}^{n}[y_i - b_0 - b_1 x_i)]^2 = \min$$

Складемо систему нормальних рівнянь:

- перше рівняння

$$\frac{\partial Q}{\partial b_0} = 2\sum_{i=1}^{n}[y_i - b_0 - b_1 x_i] = 0 \text{ або } nb_0 + b_1 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i ,$$

- друге рівняння

$$\frac{\partial Q}{\partial b_1} = 2\sum_{i=1}^{n} x_i[y_i - b_0 - b_1 x_i] = 0 \text{ або } b_0 \sum_{i=1}^{n} x_i - b_1 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i .$$

Отже,

$$\begin{cases} nb_0 + b_1 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \\ b_0 \sum_{i=1}^{n} x_i + b_1 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i \end{cases} \tag{4.31}$$

Розв'язуючи систему (4.31) відносно b_0 й b_1, знайдемо оцінки параметрів β_0 і β_1;

$$b_1 = \frac{\sum_{i=1}^{n} x_i y_i - \frac{1}{n}\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2 / n}, \tag{4.32}$$

$$b_0 = \left(\sum_{i=1}^{n} y_i\right)/n - b_1\left(\sum_{i=1}^{n} x_i\right)/n = \bar{y} - b_1 \bar{x}. \tag{4.33}$$

Розв'язуючи рівняння (4.31)-(4.33), отримуємо параметри регресії. Тоді функція регресії має вигляд:

$$Y \text{ за } X\text{: } E(Y/X = x) = E(y) + \rho\frac{\sigma_y}{\sigma_x}[X - E(x)]$$

$$X \text{ за } Y\text{: } E(X/Y = y) = E(x) + \rho\frac{\sigma_x}{\sigma_y}[Y - E(y)],$$

де $\rho\frac{\sigma_x}{\sigma_y}$ та $\rho\frac{\sigma_y}{\sigma_x}$ - коефіцієнти регресії.

Залишається одержати оцінку параметра $\sigma^2_{зал}$. Маємо

$$s^2_{зал} = \frac{1}{n-2}\sum_{i=1}^{n}[y_i - y(x_i)]^2 \qquad (4.34)$$

де n — кількість спостережень.

Підставивши в (4.34) замість $y(x_i)$ його математичне сподівання з рівняння регресії знаходимо залишкову дисперсію

$$\sigma^2_{зал} = 1 - \rho^2$$

Оцінки, отримані за способом найменших квадратів, мають мінімальну дисперсію в класі лінійних оцінок.

На практиці часто передумови кореляційного аналізу порушуються: один з процесів виявляється величиною невипадковою або процеси не мають сумісного нормального розподілу. Однак статична залежність між ними існує. Для вивчення зв'язку між процесами в цьому випадку існує загальний показник залежності, оснований на показнику мінливості – загальній (повній) дисперсії.

Повною називається дисперсія величини відносно її математичного сподівання. Так, для процесу Y це $\sigma^2_Y = E[Y - E(Y)]^2$. Дисперсію σ^2_Y можна розкласти на дві складові, одна з яких характеризує вплив процесу X на Y, друга - вплив інших факторів. Очевидно, чим менший вплив інших факторів, тим тіснішій зв'язок, тим більше наближається він до функціонального. Представимо σ^2_Y в такому вигляді:

$$\sigma^2_Y = E[E(Y \mid X = x) - E(Y)]^2 + E[Y - E(Y \mid X = x)]^2 \qquad (4.35)$$

Перший доданок позначають $\delta^2_{Y/X}$. Це дисперсія функції регресії відносно математичного сподівання процесу, вона вимірює вплив процесу X на Y.

Другий доданок позначають $\sigma^2_{Y/X}$. Це дисперсія процесу Y відносно функції регресії чи залишкова дисперсія, вона вимірює вплив на Y інших факторів.

Щільність зв'язку зручно оцінювати в одиницях загальної дисперсії σ^2_Y, тобто розглядати відношення $\{M[\bar{y}(x) - M(Y)]^2 / \sigma^2_Y\}$. Цю величину позначають $\eta^2_{TY|x}$ і називають теоретичним кореляційним відношенням. Таким чином,

$$\eta^2_{TY|x} = 1 - \sigma^2_{Y|x} / \sigma^2_Y \qquad (4.36)$$

З рівняння видно, що значення $\eta^2_{TY|x}$ завжди знаходиться в проміжку (0,1).

Розглянемо випадок, коли залежність нелінійна за змінними x_i. Принцип знаходження коефіцієнтів той же — метод найменших квадратів.

У загальному випадку нелінійної залежності між змінними Y і X зв'язок може виражатися багаточленом *k-го* степеня від x:

$$M(Y \mid X = x) = \beta_0 + \beta_1 x + \beta_2 x^2 + ... + \beta_k x^k .$$

Коефіцієнти регресії визначають за принципом найменших квадратів. Система нормальних рівнянь має вигляд

$$\begin{cases} b_0 n + b_1 \sum_{i=1}^{n} x_i + b_2 \sum_{i=1}^{n} x_i^2 + ... + b_k \sum_{i=1}^{n} x_i^k = \sum_{i=1}^{n} y_i \\ b_0 \sum_{i=1}^{n} x_i + b_1 \sum_{i=1}^{n} x_i^2 + b_2 \sum_{i=1}^{n} x_i^3 + ... + b_k \sum_{i=1}^{n} x_i^{k+1} = \sum_{i=1}^{n} y_i x_i \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ b_0 \sum_{i=1}^{n} x_i^k + b_1 \sum_{i=1}^{n} x_i^{k+1} + b_2 \sum_{i=1}^{n} x_i^{k+1} ... + b_k \sum_{i=1}^{n} x_i^{2k} = \sum_{i=1}^{n} y_i x_i^{2k} \end{cases}$$

Обчисливши коефіцієнти системи, її можна розв'язати будь-яким відомим способом.

4.5.1. Перевірка значимості рівняння регресії

Оцінити значимість рівняння регресії — значить установити, чи відповідає математична модель, що виражає залежність між Y та X, експериментальним даним. Для оцінки значимості перевіряють "нульову" гіпотезу: $\beta_1=0$. Якщо вона не відкидається, то вважають, що між Y та X немає зв'язку (або зв'язок нелінійний). Для перевірки нульової гіпотези використовують основне положення дисперсійного аналізу про розбивку суми квадратів на доданки. Загальна сума квадратів відхилень результативного процесу $Q = \sum_{i=1}^{n} (y_i - \bar{y})^2$ розкладається на $Q_1=\delta^2_{Y/X}$ (суму, що характеризує вплив процесу X) і $Q_2=\sigma^2_{Y/X}$ (залишкову суму квадратів, що характеризує вплив неврахованих факторів). Очевидно, чим менший вплив неврахованих факторів, тим краще математична модель відповідає експериментальним даним, тому що варіація Y в основному пояснюється впливом процесу X.

Для перевірки нульової гіпотези обчислюють статистику $F=(Q_1/Q_{зал})\cdot(k_2/k_1)$, що має розподіл Фішера-Снедекора з $k_1=1$, $k_2=n-2$ ступенями свободи (n - число спостережень). За рівнем значимості α й числом ступенів

свободи k_1 і k_2 знаходять за таблицями F-розподілу критичне значення $F_{(\alpha,k_1,k_2)}$, яке задовольняє умову $P(F>F_{(\alpha,k_1,k_2)})\geq\alpha$. Якщо $F>F_{(\alpha,k_1,k_2)}$, нульову гіпотезу відкидають і рівняння вважають значимим. Якщо $F<F_{(\alpha,k_1,k_2)}$, то немає підстав відкидати нульову гіпотезу.

4.5.2. Багатовимірний регресійний аналіз

У випадку, якщо зміни процесу визначаються дією сукупності інших процесів, має місце багатовимірний регресійний аналіз.

Нехай результативний процес Y, а незалежні фактори (x_1, x_2, ..., x_m). Для багатовимірного випадку передумови регресійного аналізу можна сформулювати в такий спосіб: Y — незалежні випадкові величини із середнім $M(Y|X=x_1,x_2,...,x_m)$ і постійною дисперсією $\sigma^2_{зал}$; x_1 x_2, ..., x_m — лінійно незалежні вектори $x_1(x_{11}$ x_{12}, ..., $x_{1n})$, ... $x_m(x_{m1}$, x_{m1}, ..., $x_{nm})$. Розглянемо модель вигляду

$$M(Y|X=x_1\,x_2,\ldots,x_m)=\beta_0+\beta_1 x_1+\ldots+\beta_m x_m$$

Оцінці підлягають параметри β_0, β_1,... β_m і залишкова дисперсія. Замінивши параметри їх оцінками, запишемо рівняння регресії

$$\bar{y}(x)=b_0+b_1x_1+\ldots+b_mx_m \qquad (4.37)$$

Коефіцієнти в цьому виразі знаходять методом найменших квадратів.

Вихідними даними для обчислення коефіцієнтів b_1, b_2 ..., b_m є вибірка з багатовимірної сукупності, яка подається звичайно у вигляді матриці X і вектора $У$. Як і у двовимірному випадку, складають систему нормальних рівнянь

$$\begin{cases} nb_0+b_1\sum\limits_{i=1}^{n}x_{1i}+\ldots+b_m\sum\limits_{i=1}^{n}x_{mi}=\sum\limits_{i=1}^{n}y_i \\ b_0\sum\limits_{i=1}^{n}x_{1i}+b_1\sum\limits_{i=1}^{n}x_{1i}x_{1i}+\ldots+b_m\sum\limits_{i=1}^{n}x_{1i}x_{mi}=\sum\limits_{i=1}^{n}y_ix_{1i} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ b_0\sum\limits_{i=1}^{n}x_{mi}+b_1\sum\limits_{i=1}^{n}x_{1i}x_{mi}+\ldots+b_m\sum\limits_{i=1}^{n}x_{mi}x_{mi}=\sum\limits_{i=1}^{n}y_ix_{mi} \end{cases} \qquad (4.38)$$

яку можна розв'язати будь-яким способом, відомим з лінійної алгебри.

Використовуючи формули лінійної алгебри, запишемо остаточні вирази для параметрів:

$$b_j=\sum_{i=1}^{n}C_{ij}^{-1}A_{yi},\ b_0=\bar{y}-b_1\bar{x}_1-\ldots-b_m\bar{x}_m \qquad (4.39)$$

Оцінкою залишкової дисперсії $\sigma^2_{зал}$ є

$$s^2_{зал} = \left\{ \sum_{i=1}^{n} [y_i - y(x_i)]^2 \right\} / (n - m - 1),$$

де y_i, — виміряне значення результативної ознаки;

$y(x_i)$ - значення результативної ознаки, обчислене за рівнянням регресії.

Останнім часом все більшого поширення набуває один з нових розділів статистичного аналізу – ***факторний аналіз***. Спочатку цей метод розроблявся для пояснення розмаїття кореляцій між вихідними параметрами. Дійсно, результатом кореляційного аналізу є матриця коефіцієнтів кореляцій.

Подальша розробка факторного аналізу показала, що цей метод може бути з успіхом застосований в задачах групування та класифікації об'єктів. Факторний аналіз може бути використаний як самостійний метод дослідження, так і разом з регресійним аналізом. В цьому випадку для набору залежних змінних знаходять узагальнені фактори, які потім входять в регресійний аналіз як змінні. Такий підхід дозволяє скоротити кількість змінних в регресійному аналізі, зменшити вплив помилок, видалити корельованість змінних.

4.6. Спектральні характеристики випадкових процесів

Канонічний розклад випадкового процесу

Випадковий процес $\xi(t)$ є такою функцією дійсного аргументу t, що при кожному фіксованому значенні його аргументу значення цієї функції є якимось конкретним, але наперед невідомим, причому кожне можливе в цьому випадку значення реалізується лише з певною імовірністю. Для спрощення аналізу випадкових процесів їх подають через простіші процеси цього ж класу. Найпростішими вважаються такі випадкові процеси, в яких залежність від часу виражається невипадковими функціями часу, а вся випадковість зосереджена у параметрах та коефіцієнтах таких функцій. Окремі, або і всі коефіцієнти та параметри такого виразу, є випадковими величинами.

Процес вигляду $\xi(t)=\eta \cdot \varphi(t)$, де η - випадкова величина, $\varphi(t)$ - невипадкова функція, називається ***елементарним випадковим процесом*** або елементарною випадковою функцією.

Всі можливі реалізації випадкового процесу відрізняються від графіка функції $x=\varphi(t)$ лише розтягом чи стиском його в η разів. Зокрема, можливі випадки $\eta=1$, $\eta=0$.

Характеристики елементарного процесу $\xi(t)$ обчислюються дуже просто

• математичне сподівання такого процесу $\xi(t)= \eta\cdot\varphi(t)$ дорівнює

$$E(\xi(t)) = \varphi(t)\, E(\eta) \tag{4.40}$$

• дисперсія

$$D(\xi(t))= (\varphi(t))^2\, D(\eta) \tag{4.41}$$

• кореляційна функція

$$K_\xi(t_1,t_2)= \varphi(t_1)\cdot \varphi(t_2)\cdot D(\eta) \tag{4.42}$$

Зокрема, якщо математичне сподівання випадкової величини η дорівнює нулю, то математичне сподівання елементарного випадкового процесу $\xi(t)=\eta\cdot\varphi(t)$ тотожно дорівнює нулю.

Реалізації випадкового процесу здійснюють випадкові коливання навколо графіка його математичного сподівання. Тому виникає задача зображення випадкового процесу $\xi(t)$ у вигляді

$$\xi(t) = E\xi(t) + \sum_n \eta_n \cdot \varphi_n(t) \tag{4.43}$$

де η_n- некорельовані випадкові величини з математичними сподіваннями рівними нулю,

$\varphi_n(t)$- деякі звичайні (не випадкові) функції.

Кожне зображення випадкового процесу у вигляді (4.43) називається ***канонічним розкладом*** цього процесу. В окремих випадках сума в (4.43) скінченна. Випадкові величини η_n називаються коефіцієнтами канонічного розкладу процесу, а функції $\varphi_n(t)$ - координатними функціями цього розкладу.

Нехай випадковий процес $\xi(t)$ має зображення (4.43). Тоді, використовуючи властивості кореляційної функції, отримаємо:

$$R_\xi(t_1,t_2) = \sum_n D_\eta \cdot \varphi(t_1)\,\varphi(t_2), \tag{4.44}$$

де D_η, - дисперсії випадкових величин η_n, відповідно.

Кожне зображення кореляційної функції випадкового процесу у вигляді (4.44) називається канонічним розкладом цієї функції. Отже, на основі канонічного розкладу (4.43) випадкового процесу маємо, канонічний розклад (4.44) його кореляційної функції. Має місце і обернене твердження. З розкладу (4.44) при $t_1= t_2 = t$ одержуємо і розклад дисперсії випадкового процесу:

$$D_\xi(t) = K_\xi(t_1,t_2) = \sum_n D_\eta \cdot \varphi_n(t)^2 \tag{4.45}$$

З означень похідної та інтеграла від випадкового процесу випливає, що

$$\xi'(t)=\eta\cdot\varphi'(t) \tag{4.46}$$

та

$$\int_0^t \xi(\tau)d\tau = \eta \cdot \int_0^t \varphi(\tau)d\tau \qquad (4.47)$$

Випадковий процес $\xi(t)$ може бути також зображено у вигляді:

$$\xi(t) = E_\xi(t) + \int_D \eta(\lambda) \cdot \varphi(\lambda)d\lambda \qquad (4.48)$$

де $\eta(\lambda)$ - білий шум параметра λ,

$\varphi(t, \lambda)$ - відповідна невипадкова функція аргументу t та параметра λ.

Кожен такий вираз (4.48) називається ***інтегральним канонічним зображенням*** випадкового процесу $\xi(t)$. Інтегральне канонічне зображення (4.48) випадкового процесу $\xi(t)$ цілком аналогічне його канонічному розкладу (4.43). Тут нескінченно малі некорельовані елементарні випадкові процеси $\eta(\lambda)\varphi(t,\lambda)d\lambda$ відіграють роль некорельованих елементарних процесів в розкладі (4.43). Тому функції $\varphi(t,\lambda)$ при різних фіксованих значеннях параметра λ в області інтегрування D називаються координатними функціями інтегрального канонічного зображення даного процесу. Вони є функціями аргументу t.

Сукупність коефіцієнтів $\eta(\lambda)$ утворюють ***спектр*** процесу у ***базисі координатних функцій*** $\varphi(t,\lambda)$.

Якщо базисом розкладу є система гармонічних функцій, то відповідний спектр є спектром Фур'є.

Спектральні характеристики випадкових процесів відрізняються від спектральних характеристик детермінованих процесів. Оскільки перетворення Фур'є можна застосовувати лише для періодичних процесів, які обов'язково є детермінованими, то для самого випадкового процесу перетворення Фур'є не застосовують, але його застосовують для перетворення усереднених (статистичних) характеристик цього випадкового процесу, тому що усереднена характеристика не є випадковою.

Для таких процесів використовують поняття ***спектральної щільності потужності*** або ***енергетичний спектр.***

Розглянемо невипадкову функцію $x_T(t)$, що визначається як

$$x_T(t) = \begin{cases} x(t), & 0 < t \le T \\ 0, & t \le 0, t > T \end{cases} \qquad (4.49)$$

При відомому кінцевому T функція задовольняє умову абсолютної інтегрованості. Цей процес має кінцеву енергію, а отже, і *спектральну щільність енергії* $|S_{XT}(jw)|^2$ де

$$S_{XT}(j\omega) = \int_{-T/2}^{T/2} x_T(t)e^{-j\omega t}dt \qquad (4.50)$$

За теоремою Релея середня потужність за період T визначається формулою

$$P_{XT} = \frac{1}{T}\int_{-T/2}^{T/2}[x_T(t)]^2 dt = \int_0^{\infty}\frac{|S_{XT}(j\omega)|^2}{\pi T}d\omega \qquad (4.51)$$

Знайдемо *спектральну щільність* потужності процесу $x_T(t)$:

$$G_T(\omega) = \frac{|S_{XT}(j\omega)|^2}{\pi T} \qquad (4.52)$$

Усереднивши цю функцію за множиною реалізацій і спрямувавши $T\to\infty$, визначимо ***спектральну щільність потужності випадкового процесу X***:

$$G_{XX}(\omega) = \lim_{T\to\infty} M\left\{\frac{1}{T}\left|\int_{-T/2}^{T/2} x_T(t)e^{-j\omega t}dt\right|^2\right\} \qquad (4.53)$$

Очевидно, $G_{XX}(\omega)$ — невід'ємна парна функція, що обмежує площу, рівну потужності:

$$P_X = \frac{1}{2\pi}\int_{-\infty}^{\infty} G_{XX}(\omega)d\omega \qquad (4.54)$$

Спектральна щільність потужності й автокореляційна функція позв'язані перетвореннями Вінера-Хінчина, аналогічними перетворенням Фур'є

$$R_{XX}(\tau) = \frac{1}{2\pi}\int_{-\infty}^{\infty} G_{XX}(\omega)e^{j\omega\tau}d\omega = \frac{1}{\pi}\int_0^{\infty} G_{XX}(\omega)\cos(\omega\tau)d\omega \qquad (4.55)$$

$$G_{XX}(\omega) = \int_{-\infty}^{\infty} R_{XX}(\tau)e^{-j\omega\tau}d\tau = 2\int_0^{\infty} R_{XX}(\tau)\cos(\omega\tau)d\tau \qquad (4.56)$$

Якщо спектр однобічний, тобто визначений тільки в області позитивних частот, то

$$R_{XX}(\tau) = \frac{2}{\pi}\int_0^{\infty} G_{XX}(\omega)\cos(\omega\tau)d\omega \qquad (4.57)$$

$$G_{XX}(\omega) = 4\int_0^{\infty} R_{XX}(\tau)\cos(\omega\tau)d\tau \qquad (4.58)$$

Отже, спектральна щільність потужності являє собою функцію, що характеризує розподіл енергії по осі частот: на основі її можуть бути визначені кореляційні й енергетичні характеристики випадкового процесу.

Зв'язок автокореляційних функцій і спектральних щільностей потужності для найпоширеніших видів випадкових процесів наведений в табл. 4.3.

Як енергетична характеристика взаємодії випадкових процесів служить взаємна спектральна щільність потужності $G_{XY}(w)$, яка означена аналогічно

спектральної щільності потужності одиничного випадкового процесу, і пов'язана, відповідно до теореми Вінера-Хінчина, з $R_{XY}(\tau)$ співвідношеннями перетворень Фур'є:

$$R_{XY}(\tau)=\frac{2}{\pi}\int_0^{\infty}G_{XX}(\omega)\cos(\omega\tau)d\omega=\frac{1}{2\pi}\int_{-\infty}^{\infty}G_{XY}(\omega)e^{j\omega\tau}d\omega$$

$$G_{XX}(\omega)=2\int_0^{\infty}R_{XY}(\tau)\cos(\omega\tau)d\tau=2\int_{-\infty}^{\infty}R_{XY}(\tau)e^{-j\omega\tau}d\tau \quad (4.61)$$

Таблиця 4.3

Енергетичні характеристики найпоширеніших видів випадкових процесів

Характеристика процесу	Спектральна щільність потужності	Автокореляційна функція	Дисперсія
1	2	3	4
Білий шум	c^2	$c^2\delta^2$	∞
Результат проходження білого шуму через ідеальний фільтр нижніх частот	c^2 при $\lvert\omega\rvert\le\omega_0$ 0 при $\lvert\omega\rvert>\omega_0$	$\frac{c^2}{\pi\tau}\sin\omega_0\tau$	$\frac{c\omega_0}{\pi}$
Результат проходження білого шуму через *RC*-фільтр	$\frac{2\alpha c^2}{\alpha^2+\omega^2}$	$c^2\exp(-\alpha\lvert\tau\rvert)$	c^2
Результат проходження білого шуму через гаусівський низькочастотний фільтр	$c^2\sqrt{\frac{\pi}{\alpha}}\exp\left(-\frac{\omega^2}{4\alpha}\right)$	$c^2\exp(-\alpha\tau^2)$	c^2
Результат проходження білого шуму через ідеальний смуговий фільтр	0 при $\lvert\omega\rvert<\omega_1$ c^2 при $\omega_1\le\lvert\omega\rvert\le\omega_2$ 0 при $\lvert\omega\rvert>\omega_2$	$\frac{c^2}{\pi\tau}(\sin\omega_2\tau-\sin\omega_1\tau)$	$\frac{c^2}{\pi}(\omega_2-\omega_1)$
Результат проходження білого шуму через коливальний контур	$\alpha c^2\left[\frac{1}{\alpha^2+(\beta+\omega)^2}+\frac{1}{\alpha^2+(\beta-\omega)^2}\right]$	$c^2\cos\beta\tau\exp(-\alpha\lvert\tau\rvert)$	c^2

Продовження табл. 4.3

1	2	3	4
Результат проходження білого шуму через подвійний *RC*-фільтр	$\frac{4\alpha^3 c^2}{(\alpha^2+\omega^2)^2}$	$c^2(1+\alpha\|\tau\|\exp(-\alpha\|\tau\|)$	c^2
Результат проходження білого шуму через потрійний *RC*-фільтр	$\frac{16\alpha^5 c^2}{3(\alpha^2-\omega^2)^3}$	$c^2[1-\alpha\|\tau\|+(\alpha^2\tau^2)/3]\times$ $\times\exp(-\alpha\|\tau\|)$	c^2
Періодичний випадковий процес із випадковою фазою рівномірно розподіленою від $-\pi$ до π, фіксованою частотою ω_0 і амплітудою c	$\pi c^2\delta(\omega-\omega_0)$	$\frac{c^2}{2}\cos\omega_0\tau$	$\frac{c^2}{2}$
Сума n процесів вигляду, зазначеного в попередньому прикладі	$\sum_{K=i}^{n}\pi c_K^2\delta(\omega-\omega_{0K})$	$\frac{1}{2}\sum_{K=i}^{n}c_K^2\cos\omega_{0K}\tau$	$\frac{1}{2}\sum_{K=i}^{n}c_K^2$
Періодичний випадковий процес із випадковими амплітудою c (відомо $M[c^2]$), фазою, рівномірно розподіленою від $-\pi$ до π й фіксованою частотою ω_0	$\pi M[c^2]\,\delta(\omega-\omega_0)$	$\frac{1}{2}M[c^2]\cos\omega_0\tau$	$\frac{1}{2}M[c^2]$
Періодичний випадковий процес із випадковими амплітудою (відомо $M[c^2]$), частотою (закон розподілу $f_\omega(\omega)$) і фазою, рівномірно розподіленою від $-\pi$ до π	$\pi M[c^2]f_\omega(\omega)$	$\frac{1}{2}M[c^2]\int_{-\infty}^{\infty}f_\omega(\omega)\cos\omega\tau d\omega$	$\frac{1}{2}M[c^2]$

4.7. Марковські процеси

Процес, в якому наступне значення залежить лише від одного попереднього і не залежить від більш ранніх значень, називається ***марковським процесом***.

Марковський випадковий процес, заданий на дискретній множині, область значень якого в кожному перетині є однією й тією ж скінченною чи зліченною множиною називається ***ланцюгом Маркова з дискретним часом*** або дискретним ланцюгом Маркова. Дискретні процеси в природі, економіці, в біологічних популяціях, соціальні процеси дуже часто є ланцюгами Маркова. Більшість з них мають всі типові ознаки, характерну структуру, що дає можливість вивчати їх з єдиної точки зору. Ланцюги Маркова досить поширені в системах масового обслуговування та в теорії запасів.

Приклад походження марковських процесів – процеси з випадковим приростом: $x(t)=U(t)+\xi(t)$, де $U(t)$ - детермінована частина, $\xi(t)$ - випадкова частина. Таким чином, оскільки електромагнітні завади є випадковими, некорельованими і адитивними, то переважна більшість електричних сигналів в електронних схемах є марковськими процесами, тому марковські процеси є одним з головних об'єктів, які вивчаються в теорії випадкових процесів.

Наведене вище описове означення марковського випадкового процесу можна формалізувати таким чином.

Випадковий процес $\xi(t)$ називається ***марковським процесом,*** якщо для всіх $t_1, t_2, \ldots$ з області його визначення T таких, що $t_1< t_2,$, і всіх заданих значеннях $\xi(t)$ цього процесу при $t\le t_1$, умовний розподіл ймовірності $\xi(t_2)$ залежить лише від $\xi(t_1)$.

Отже, якщо відоме значення $\xi(t)$ деякого процесу і значення $\xi(t+\Delta t)$ не залежить від $\xi(t-\delta)$ при всіх $\Delta t>0$,$\delta>0$, то це марковський процес.

Тому n-вимірна щільність ймовірностей марковського випадкового процесу подається через одновимірну та відповідні умовні щільності таким чином:

$$p(x_1,\ldots,x_{n-1};t_1,\ldots,t_{n-1})=p_1(x_1;t_1)\cdot p_2(x_1;t_1 \mid x_2;t_2)\cdot p_2(x_3;t_3 \mid x_2;t_2)\ldots p_2(x_n;t_n \mid x_{n-1};t_{n-1}) \quad (4.62)$$

Умовна щільність розподілу виражається формулою

$$p_2(x_k;t_k \mid x_{k-1};t_{k-1})=\frac{p_2(x_{k-1};x_k \mid t_{k-1};t_k)}{p_1(x_{k-1};t_{k-1})} \quad (4.63)$$

Отже з (4.62) маємо:

$$p(x_1,\ldots,x_n;t_1,\ldots,t_n)=\frac{p_2(x_1,x_2;t_1,t_2)\cdot p_2(x_2,x_3;t_2,t_3)\cdot\ldots\cdot p_2(x_{n-1},x_n;t_{n-1},t_n)}{p_1(x_2;t_2)\cdot p_1(x_3;t_3)\cdot\ldots\cdot p_1(x_{n-1};t_{n-1})} \quad (4.64)$$

З (4.64) випливає, що n-вимірну щільність розподілу ймовірності марковського випадкового процесу завжди можна визначити, якщо відома його двовимірна щільність розподілу, тобто такий процес достатньо повно характеризується своїм двовимірним законом розподілу.

Функція $P(s, x; t, A)$, яка є ймовірністю того, що процес $\xi(t)$ перебуваючи у момент s у стані x, на момент часу t буде знаходитися в одному зі станів множини станів A, називається *марковською перехідною функцією*, а окремі її значення - ***перехідними ймовірностями*** цього процесу, тобто,

$$P(s, x; t, A)= P(\xi(t) \in A \mid \xi(s) = x) \quad (4.65)$$

Функцію $F(s, x; t, y)= P(\xi(t) <y \mid \xi(s) = x)$ також називають марковською перехідною функцією тому, що за її значеннями функція $P(s, x; t, A)$ відтворюється однозначно.

Очевидно, перехідна функція дискретнозначного процесу (кількість станів його скінченна чи зліченна) визначається таким чином:

$$P(s,x;t,A)= \sum_{y\in A} P(s,x;t,y) \quad (4.66)$$

а абсолютно неперервного -

$$P(s,x;t,A)= \int_A p(s,x;t,y)dy \quad (4.67)$$

де $p(s, x; t, y)$ щільність розподілу випадкової змінної $\xi(t)$ при умові $\xi(s) = x$.

Марковський процес $\xi(t)$, де $a\le t\le b$. можемо задати функцією розподілу $F_a(x)$ його станів $\xi(a)$ у початковий момент та його перехідною функцією $F(t_1,x_1;t_2,x_2)$.

4.7.1.Опис та зображення ланцюгів Маркова

Для спрощення викладу, стани $E_к$ випадкового процесу будемо ототожнювати з їхніми індексами. Тоді однокрокову перехідну ймовірність процесу зі стану i в якому він перебував у момент часу t_n (на n-ому кроці), у стан j в момент часу t_{n+1} (у наступний момент часу, на $(n+1)$-ому кроці) позначатимемо

$$p_{i,j}^{n,n+1} = P(\xi_{n+1} = j \mid \xi = i),$$

де $\sum_{j=1} p_{i,j}^{n,n+1} = 1, \quad i=1,\ldots 2,\ldots$

Марковський випадковий процес, заданий на дискретній множині, область значень якого в кожному перетині є однією й тією ж скінченною чи зліченною множиною, називається дискретним ланцюгом Маркова.

Схему можливих реалізацій *ланцюга Маркова з дискретним часом* зображено на рис 4.4. Тут вертикальними відрізками прямих зображені перетини цього процесу, а точками на них - позначені стани ланцюга. Відрізками горизонтальних прямих зображені можливі переходи зі стану у стан, дивлячись зліва направо.

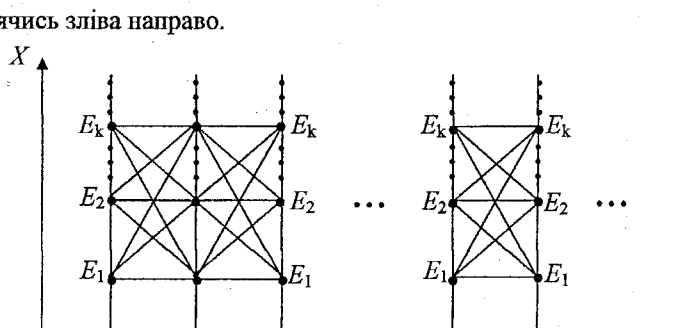


Рис. 4.4. Реалізації дискретного ланцюга Маркова

Ланцюг Маркова називається *однорідним*, якщо його однокрокові перехідні ймовірності $p_{i,j}^{n,n+1}$ не залежать від часу (не залежать від n), тобто $p_{i,j}^{n,n+1} = p_{ij}$ для всіх $n = 0, 1, 2, \ldots$. Перехідні ймовірності p_{ij} називаються *стаціонарними перехідними ймовірностями.*

Далі ланцюги Маркова будемо вважати однорідними. Ймовірності p_{ij} переходу зі стану E_i у стан E_j за один крок (одну одиницю часу), як правило, об'єднують в матрицю

$$P = \begin{pmatrix} p_{11}p_{12}\cdots p_{1k}\cdots \\ p_{21}p_{22}\cdots p_{2k}\cdots \\ \cdots\cdots\cdots\cdots \\ p_{k1}p_{k2}\cdots p_{kk}\cdots \\ \cdots\cdots\cdots\cdots \end{pmatrix}$$

Квадратна матриця з невід'ємними елементами, сума яких у кожному рядку дорівнює одиниці, називається *стохастичною.*

Стан $E_к$ називається *досяжним* зі стану E_j, якщо існує таке ціле невід'ємне число n, що з ймовірністю відмінною від нуля, система може перейти зі стану E_j, в стан $E_к$ тобто, $p_{jk}^{(n)} > 0$.

Стан E_j ланцюга Маркова називається *зворотним*, якщо система, яка в початковий момент перебувала в цьому стані, обов'язково повернеться колинебудь в цей же стан E_j.

Початковий стан ланцюга Маркова задається ймовірностями перебування у кожному стані множини A. Ці ймовірності утворюють стохастичний вектор $a=(a_1,a_2,\dots)$.

Нехай ланцюг Маркова заданий стохастичним вектором $a=(a_1,a_2,\dots)$ та стохастичною матрицею

$$P=\begin{pmatrix} p_{11}p_{12}\cdots \\ p_{21}p_{22}\cdots \\ \cdots\cdots \end{pmatrix}$$

Методом математичної індукції можна довести, що ймовірності перебування у кожному стані на $n+1$ кроці утворюють стохастичний вектор

$$p(n+1)=p(n)\cdot P=a\cdot P^{n+1} \qquad (4.68)$$

4.7.2. Марковські процеси загального вигляду

В загальному випадку будемо вважати, що випадковий процес $\xi(t)$ в кожен момент часу може перейти із стану E_k, у якому він перебував безпосередньо перед цим моментом, у будь-який стан E_m цього процесу, і що такий процес є неоднорідним за часом. Позначимо $P_{km}(\tau,\ t)$ умовну ймовірність того, що випадковий процес в момент часу t знаходиться в стані E_m при умові, що в попередній момент часу $\tau\cdot(\tau< t)$ він знаходився в стані E_k.

Перехід зі стану E_k в якому перебував процес у момент τ, у стан E_m в момент часу t, очевидно, відбувається через деякий проміжний стан E_i у проміжний між моментами τ та t момент часу s. З іншого боку, в марковських випадкових процесах перехідна ймовірність $p_{im}(s,\ t)$ переходу із стану E_i в стан E_m не залежить від попереднього стану E_i. Множина станів $\{E_k\}_{k=0}^{\infty}$ зліченна.

Отже, перехідні ймовірності загального марковського процесу задовольняють таке ***рівняння Колмогорова- Чепмена***:

$$p_{km}(\tau,t)=\sum_i P_{ki}(\tau,s)\cdot P_{im}(s,t) \qquad (4.69)$$

де всі

$$P_{kj}(\tau,t) \geq 0 \quad \text{і} \quad \sum_j P_{kj}(\tau,t) = 1 \tag{4.70}$$

і навпаки, кожний розв'язок рівняння (4.69), який задовольняє умови (4.70), є перехідними ймовірностями деякого марковського процесу загального вигляду.

4.7.3. Узагальнене рівняння Маркова

Опишемо марковський процес функцією $F(t, x; \tau, y)$, яка є ймовірністю того, що в момент часу τ випадковий процес $\xi(t)$ прийме значення не більше за y, якщо в попередній момент часу t ($t < \tau$) він набув значення $\xi(t) = x$. В такому процесі розвиток його протягом часу, що передував моменту t, не впливає на його перебіг в наступні моменти часу і тому не змінює функції $F(t, x; \tau, y)$. Функція $F(t, x; \tau, y)$ має всі властивості функції розподілу.

Нехай три будь-які моменти часу t, s, τ такі, що $t<s<\tau$. Згідно з властивостями функції $F(t, x; s, z)$, процес $\xi(t)$ із стану x, в якому він перебував у момент t з ймовірністю $d_z F(t, x; s, z)$, перейде в момент часу s в один із станів в інтервалі $(z, z+dz)$, а, згідно з означенням функції $F(t, x; \tau, y)$, якщо в момент s процес знаходився у стані z, то він з ймовірністю $F(s, z; \tau, y)$ у момент τ перейде у стан не більший y. Тому,

$$F(t,x;\tau,y) = \int_{-\infty}^{+\infty} F(s,z;\tau,y) \cdot d_z F(t,x;s,z) \tag{4.71}$$

Рівняння (4.71) називається узагальненим рівнянням Маркова.

Врахувавши, що

$$F(t,x;\tau,y) = \int_{-\infty}^{y} f(t,x;\tau,z)dz$$

узагальнене рівняння Маркова (4.71) можна записати у такому вигляді:

$$f(t,x;\tau,y) = \int_{-\infty}^{+\infty} f(s,z;\tau,y) \cdot f(t,x;s,z)dz \tag{4.72}$$

4.8. Диференціальні рівняння Колмогорова

Нехай неперервний випадковий процес $\xi(t)$ задовольняє умову неперервності:

$$\lim_{\Delta t \to 0} \frac{1}{\Delta t} \cdot \int_{|y-x| \geq 0} d_y F(t - \Delta t, x; \tau, y) = 0 \tag{4.73}$$

Тоді, якщо існують частинні похідні

$$\frac{\partial F(t,x;\tau,y)}{\partial x} \text{ та } \frac{\partial^2 F(t,x;\tau,y)}{\partial x^2},$$

то $F(t, x; \tau, y)$ такого процесу є розв'язком першого рівняння Колмогорова

$$a(t,x) \overset{df}{=} \lim_{\Delta t \to 0} \frac{1}{\Delta t} \int_{|y-x|<\delta} (y-x) d_y F(t-\Delta t, x; t, y) \tag{4.74}$$

$$b(t,x) \overset{df}{=} \lim_{\Delta t \to 0} \frac{1}{\Delta t} \int_{|y-x|<\delta} (y-x)^2 d_y F(t-\Delta t, x; t, y) \tag{4.75}$$

$$\frac{\partial F(t,x;\tau,y)}{\partial t} = -a(t,x) \cdot \frac{\partial F(t,x;\tau,y)}{\partial x} - \frac{1}{2} \cdot b(t,x) \cdot \frac{\partial^2 F(t,x;\tau,y)}{\partial x^2} \tag{4.76}$$

де

$$a(t,x) \overset{df}{=} \lim_{\Delta t \to 0} \frac{1}{\Delta t} \int_{|y-x|<\delta} (y-x) d_y F(t-\Delta t, x; t, y) \tag{4.77}$$

$$b(t,x) \overset{df}{=} \lim_{\Delta t \to 0} \frac{1}{\Delta t} \int_{|y-x|<\delta} (y-x)^2 d_y F(t-\Delta t, x; t, y) \tag{4.78}$$

Рівняння Колмогорова виконується також і для щільності розподілу ймовірностей

$$\frac{\partial p(t,x;\tau,y)}{\partial \tau} = -\frac{\partial}{\partial y}(a(\tau,y) \cdot p(t,x;\tau,y)) + \frac{\partial^2}{\partial y^2}(b(\tau,y) \cdot p(t,x;\tau,y)) \tag{4.79}$$

Друге рівняння Колмогорова (4.79) отримали раніше фізики Планк і Фоккер при побудові теорії дифузії, тому часто його називають рівнянням Фоккера-Планка-Колмогорова.

Рівняння Колмогорова (4.76) та (4.79) використовуються по-різному, в залежності від мети, поставленої при дослідженні неперервного випадкового процесу $\xi(t)$. Зокрема рівняння дають можливість знайти розподіл ймовірностей такого процесу, виходячи із загального опису його властивостей, що виражається у попередньому задаванні конкретного вигляду функцій $a(t, x)$ та $b(t, x)$.

4.9. Перетворення випадкових процесів

В системах управління та автоматики практично всі процеси можна

вважати випадковими, тобто крім детермінованої складової, вони містять і випадкову складову. Всі складові систем можна розглядати як ***перетворювачі*** таких випадкових процесів. В теорії перетворень випадкових процесів розглядається задача: відомі характеристики випадкового процесу на вході перетворювача, знайти відповідні характеристики на виході перетворювача. Для розв'язання цієї задачі *всі перетворювачі поділяються на види*:

1 поділ: з одним вхідним процесом або з багатьма вхідними процесами;

2 поділ: лінійні або нелінійні;

3 поділ: статичні або динамічні.

Лінійними називаються перетворювачі, які описуються лінійними алгебраїчними або диференціальними рівняннями.

Статичні перетворювачі – це такі, в яких результат перетворення залежить лише від поточного значення вхідного процесу.

Динамічні перетворювачі – це такі, в яких результат перетворення залежить як від поточного, так і від попередніх значень процесу.

Статичні перетворювачі описуються алгебраїчними рівняннями, а динамічні – диференціальними.

Як відомо, найпоширенішими характеристиками випадкового процесу є щільність розподілу ймовірностей його значень і кореляційна (коваріаційна) функція. На жаль, знайти для будь-якого перетворювача аналітичними методами обидві ці характеристики можна лише у крайніх випадках. Найчастіше розраховується лише одна з них, а щодо іншої висловлюються певні припущення.

4.9.1. Перетворення випадкового процесу у статичному одновходовому перетворювачі

Узагальнена схема нелінійного статичного перетворювача показана на рис.4.5.

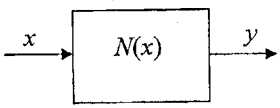


Рис. 4.5. Одновходовий нелінійний статичний перетворювач

Нехай процес X характеризується щільністю розподілу ймовірностей $f_x(x)$, процес $Y - f_y(y)$. Тоді

$$f_y(y) = \dot{N}^{-1}(y) \cdot f_x[N^{-1}(y)] \qquad (4.80)$$

де $N^{-1}(y)$ - функція зворотного перетворення, якщо вона існує;

$\dot{N}^{-1}(y)$ - похідна від неї, якщо вона існує.

Наприклад, при нормальному розподілі вхідного процесу і квадратичному перетворенні маємо

$$f_x(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{x^2}{2\sigma^2}};$$

$$y=N(x)=x^2$$

Тоді

$$x=N^{-1}(y)=\sqrt{y}$$

$$\dot{N}^{-1}(y)=\frac{1}{2\sqrt{y}},$$

звідки

$$f_y(y)=\frac{1}{2\sqrt{y}}\cdot\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(\sqrt{y})^2}{2\sigma^2}}=\frac{1}{2\sigma\sqrt{2\pi y}}e^{-\frac{y^2}{2\sigma^2}}$$

Вираз (4.80) справедливий зокрема для лінійного статичного перетворювача $y=ax+b$. Тоді $N^{-1}(y)=\frac{y-b}{a}$ і $\dot{N}^{-1}(y)=\frac{1}{a}$, звідки

$$f_y(y)=\frac{1}{a}f_x\left(\frac{y-b}{a}\right) \qquad (4.81)$$

У загальному випадку аналітичний розрахунок кореляційної функції результату нелінійного перетворення неможливий. Але, якщо відома двовимірна щільність розподілу ймовірностей вхідного процесу $f_X(x_1,t_1;x_2,t_2)$, то застосовуючи двічі перетворення (4.80), знаходимо двовимірну щільність розподілу ймовірностей вихідного процесу:

$$f_{XY}(y_1,t_1;x_2,t_2)=\dot{N}^{-1}(y_1)\cdot f_X\left[N^{-1}(y_1),t_1;x_2,t_2\right]$$

$$f_Y(y_1,t_1;y_2,t_2)=\dot{N}^{-1}(y_2)\cdot f_{XY}\left[y_1,t_1;N^{-1}(y_2),t_2\right],$$

звідки може бути знайдена кореляційна функція $R(t_1,t_2)$.

4.9.2. Перетворення випадкового процесу у двовходовому статичному перетворювачі

Двовходові статичні перетворювачі виконують операції додавання, віднімання, множення, ділення, а також складніші обчислення функцій двох

аргументів. Узагальнене зображення двовходового статичного перетворювача показане на рис.4.6.

Аналітичний розрахунок результатів такого перетворення можливий лише в окремих випадках. За основу такого розрахунку візьмемо характеристики суми випадкових процесів, оскільки всі інші операції можуть бути зведені до операції знаходження суми (як, наприклад, це робиться у комп'ютерних процесорах).

Сумою випадкових процесів $\xi(t)$ та $\eta(t)$ називається такий процес $\tau(t)$, всі перетини якого, при кожному значенні аргументу t з області його визначення, є сумою перетинів цих процесів при тому ж значенні аргументу t.

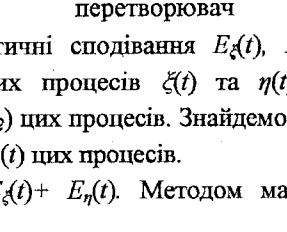


Рис.4.6. Двовходовий статичний перетворювач

Нехай відомі математичні сподівання $E_\xi(t)$, $E_\eta(t)$, кореляційні функції $R_\xi(t_1,t_2)$, $R_\eta(t_1,t_2)$ випадкових процесів $\xi(t)$ та $\eta(t)$, відповідно та взаємна кореляційна функція $R_{\xi\eta}(t_1,t_2)$ цих процесів. Знайдемо математичне сподівання та кореляційну функцію суми $\tau(t)$ цих процесів.

Як відомо, $E_{\xi+\eta}(t) = E_\xi(t) + E_\eta(t)$. Методом математичної індукції легко довести, що:

$$E\left(\sum_{i=1}^{n}\xi_i\right) = \sum_{i=1}^{n}E(\xi_i). \tag{4.82}$$

Кореляційна функція суми двох стохастичних процесів:

$$K_\tau(t_1,t_2) = K_\xi(t_1,t_2) + K_\eta(t_1,t_2) + K_{\xi\eta}(t_1,t_2) + K_{\eta\xi}(t_1,t_2) \tag{4.83}$$

тобто, кореляційна функція суми двох стохастичних процесів дорівнює сумі кореляційних функцій доданків та їх взаємних кореляційних функцій. Зокрема, якщо процеси $\xi(t)$ та $\eta(t)$ незалежні, то кореляційна функція суми двох некорельованих процесів дорівнює сумі їх кореляційних функцій, тобто

$$K_{\xi+\eta}(t_1,t_2) = K_\xi(t_1,t_2) + K_\eta(t_1,t_2). \tag{4.84}$$

Методом математичної індукції можна довести, що для будь-якого $n \geq 2$ кореляційна функція суми $\tau_{(t)} = \sum_{i=1}^{n}\xi_i(t)$ випадкових процесів $\xi_i(t)$ $(i = \overline{1,n})$ дорівнює сумі їх кореляційних функцій та сумі всіх можливих взаємних кореляційних функцій пар цих стохастичних процесів, тобто

$$K_\tau(t_1,t_2)=\sum_{i,j=1}^{n} K_{\xi_i\xi_j}(t_1,t_2) \tag{4.85}$$

Зокрема, якщо всі випадкові процеси $\xi_i(t)$ $(i=\overline{1,n})$ попарно незалежні, то кореляційна функція їх суми

$$K_\tau(t_1,t_2)=\sum_{i=1}^{n} K_{\xi_i}(t_1,t_2). \tag{4.86}$$

Дисперсія суми двох незалежних процесів дорівнює сумі їх дисперсій, тобто, якщо процеси $\xi(t)$ та $\eta(t)$ незалежні, то

$$D_{\xi+\eta}(t)=D_\xi(t)+D_\eta(t). \tag{4.87}$$

Для нормальних процесів дисперсія суми може бути знайдена і при наявності кореляції

$$D_{\xi+\eta}(t)=D_\xi(t)+D_\eta(t)-2\cdot\rho_{\xi,\eta}(t)\cdot\sqrt{D_\xi(t)\cdot D_\eta(t)}$$

Якщо процеси x_1 та x_2 функціонально пов'язані, у випадку зростаючої функціональної залежності $D_y=(\sigma_{x1}+\sigma_{x2})^2$ (оскільки $\rho_{x1x2}=1$); у випадку спадної функціональної залежності $D_y=(\sigma_{x1}-\sigma_{x2})^2$ (оскільки $\rho_{x1x2}=-1$)

Аналогічне співвідношення виконується для спектральних щільностей потужності, оскільки перетворення Фур'є є лінійним.

4.9.3. Перетворення випадкового процесу у лінійному динамічному одновходовому перетворювачі

Динамічний перетворювач описується диференціальним рівнянням. Лінійні перетворювачі систем управління переважно описуються звичайними лінійними диференціальними рівняннями зі сталими коефіцієнтами. В загальному випадку таке диференціальне рівняння можна подати у вигляді:

$$\begin{aligned} &a_n\frac{d^n y(t)}{dt^n}+a_{n-1}\frac{d^{n-1}y(t)}{dt^{n-1}}+\ldots+a_1y(t)+a_0= \\ &=b_m\frac{d^m x(t)}{dt^m}+b_{m-1}\frac{d^{m-1}x(t)}{dt^{m-1}}+\ldots+b_1x(t)+b_0 \end{aligned} \tag{4.88}$$

Диференціювання стохастичного процесу

Про збіжність послідовності випадкових змінних можна говорити лише в ймовірнісному розумінні. Похідною ж функції $x(t)$ якщо вона існує, називають границю

$$\lim_{\Delta t \to 0} \frac{x(t+\Delta) - x(t)}{\Delta t}$$

Класичне означення похідної все ж таки можна застосовувати в тому випадку, якщо є підстави вважати диференційовними всі реалізації $x(t)$ випадкового процесу $\xi(t)$. В загальному ж випадку, похідну стохастичного процесу $\xi(t)$ можна означити як границю випадкової змінної $\eta_\Delta(t)$.

Послідовність випадкових величин $\xi_1, \xi_2, \ldots, \xi_n$ називається збіжною в середньому квадратичному до випадкової величини ξ, якщо математичне сподівання квадрата різниці $(\xi_n - \xi)$ прямує до 0 при $n \to \infty$

$$\lim_{n \to \infty} E(\xi_n - \xi)^2 = 0$$

Випадковий процес $\xi(t)$ при фіксованому значенні аргументу є деякою випадковою змінною ξ_t. Тому можна говорити і про збіжність в середньому квадратичному при $t \to t_0$ випадкових змінних ξ_t - перетинів цього випадкового процесу до деякої випадкової змінної η_{t_0}:

$$\eta_{t_0} = \lim_{t \to t_0} \xi_t$$

тобто,

$$\lim_{t \to t_0} E(\xi_t - \eta_{t_0})^2 = 0,$$

а якщо t_0 послідовно приймає всі значення з області визначення випадкового процесу $\xi(t)$, то мова йде про збіжність в середньому квадратичному випадкового процесу $\xi(t+\Delta t)$ при $\Delta t \to 0$ до деякого випадкового процесу $\eta(t)$:

$$\eta(t) = \lim_{\Delta t \to 0} \xi(t + \Delta t)$$

тобто,

$$\lim_{\Delta t \to 0} E(\xi(t + \Delta t) - \eta(t))^2 = 0$$

Похідною випадкового процесу $\xi(t)$ називається такий випадковий процес $\eta(t)$, що

$$\lim_{\Delta t \to 0} E\left(\left(\frac{\xi(t + \Delta t) - \xi(t)}{\Delta t} - \eta(t)\right)^2\right) = 0 \qquad (4.89)$$

Якщо такий процес $\eta(t)$ існує, то випадковий процес $\xi(t)$ називається диференційованим. Похідну процесу $\xi(t)$ позначають

$$\xi'(t) = \frac{d}{dt}\xi(t) = \frac{d\xi(t)}{dt}$$

Отже, $\eta_\Delta(t) = \xi'(t)$ є границею в середньому квадратичному послідовності $\eta_\Delta(t)$:

$$\xi'(t) = \lim_{\Delta t \to 0} \frac{\xi(t+\Delta t) - \xi(t)}{\Delta t} \tag{4.90}$$

Похідна $\xi'(t)$ стохастичного процесу $\xi(t)$ є випадковим процесом. Якщо він є диференційованим, то його похідна $\tau(t)$ називається другою похідною випадкового процесу $\xi(t)$:

$$\tau(t) = \frac{d}{dt}(\xi'(t)) = \frac{d^2\xi(t)}{dt^2} = \xi''(t) \text{ і т.д.} \tag{4.91}$$

Ґрунтуючись на визначенні похідної випадкового процесу, можна довести, що математичне сподівання та кореляційна функція похідної матимуть вигляд:

$$E(\xi'(t)) = (E(\xi(t)))' \tag{4.92}$$

$$R_{\xi'}(t_1, t_2) = \frac{\partial^2 R_\xi(t_1, t_2)}{\partial t_1 \partial t_2} \tag{4.93}$$

Інтегрування випадкових процесів

Нехай випадковий процес $\xi(t)$ перетворюється інтегруванням:

$$\eta(s) = \int_a^b f(s,t) \cdot \xi(t) dt \tag{4.94}$$

де $f(s,t)$ - деяка невипадкова функція (ядро перетворення).

Традиційне поняття інтеграла не може бути застосованим до випадкового процесу з тих же причин, що і поняття похідної.

Випадковий процес $\xi(t)$ називається інтегрованим на проміжку $[a,b]$ з ядром $f(s,t)$, якщо існує такий стохастичний процес $\eta(s)$, що

$$\lim_{\max \Delta t_k \to 0} E\left(\left(\sum_k f(s,t_k) \cdot \xi(t_k) \cdot \Delta t_k - \eta(s)\right)^2\right) = 0 \tag{4.95}$$

незалежно від розбиття точками $t_1, t_2, \ldots, t_n$ проміжку $[a,b]$ на частинки $\Delta t_k = t_k - t_{k-1}$. Випадковий процес $\eta(s)$ називається інтегралом на проміжку $[a,b]$ від випадкового процесу $\xi(t)$ з вагою функції $f(s,t)$, і позначається

$$\eta(s) = \int_a^b f(s,t) \cdot \xi(t) dt \tag{4.96}$$

Якщо існує скінченне математичне сподівання і кореляційна функція випадкового процесу $\xi(t)$, а вагова функція $f(s,t)$ на проміжку $[a,b]$ обмежена, то

$$E_\eta(s) = \int_a^b f(s,t) \cdot E_\xi(t) dt \qquad (4.97)$$

$$R_\eta(s_1,s_2) = \int_a^b \int_a^b f(s_1,t_1) \cdot f(s_2,t_2) \cdot R_\xi(t_1,t_2) dt_1 dt_2 \qquad (4.98)$$

На основі визначень похідної і інтегралу від випадкового процесу можуть бути знайдені деякі характеристики перетворювачів, які описуються диференціальними рівняннями (4.88).

Для лінійних динамічних перетворювачів найчастіше диференціальні рівняння зображують у вигляді операторного (за Лапласом) зображення

$$Y(p) = W(p) \cdot X(p) \text{ Ё} \qquad (4.99)$$

де $W(P)$ – передаточна функція.

Відповідна схема зображена на рис.4.7.

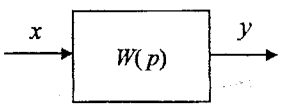


Рис.4.7. Лінійний динамічний перетворювач

Оригіналом у часовому просторі для передаточної функції є ***імпульсна перехідна функція перетворювача*** – реакція лінійного динамічного перетворювача на вхідний імпульс у вигляді δ-функції

$$W(P) \leftrightarrow g(\tau)$$

Відповідно оригіналом до операторного зображення (4.99) є ***інтеграл Дюамеля***:

$$y(t) = \int_0^t x(t-\tau) g(\tau) d\tau \qquad (4.100)$$

Очевидно інтеграл Дюамеля відповідає загальному поняттю інтегрування (4.94), ядром якого є імпульсна перехідна функція перетворювача.

Тоді на основі (4.97) і (4.98) можна записати:

$$E[y(t)] = \int_0^t E[x(t-\tau)] g(\tau) d\tau$$

$$R_{xy}(\tau) = \int_0^\tau R_{xx}(\tau - t) g(t) dt \qquad (4.101)$$

$$R_y(\tau) = \int_0^{\tau}\int_0^{\tau} R_x(\tau - t_1 - t_2)g(t_1)g(t_2)dt_1dt_2$$

Враховуючи зв'язок між спектральною щільністю потужності і кореляційною функцією, можна записати аналогічне перетворення спектральних щільностей

$$S_{xy}(\omega) = S_{xx}(\omega)\cdot W(j\omega) \qquad (4.102)$$
$$S_y(\omega) = S_x(\omega)\cdot |W(j\omega)|$$

де ω – кутова частота.

Існуючі методи визначення результату перетворення випадкових процесів лінійними динамічними перетворювачами дають можливість визначити в першу чергу математичне сподівання, дисперсію та коваріаційну функцію. Визначення функції розподілу ймовірностей та інших моментів набагато складніше, але найчастіше воно і не потрібне, оскільки внаслідок центральної граничної теореми результат перетворення повинен мати функцію розподілу, яка наближається до нормальної. ***Центральна гранична теорема*** стверджує, що закон розподілу ймовірностей суми великої кількості випадкових величин, серед яких немає таких, які б значно переважали інші, буде нормальним, незалежно від законів розподілення доданків. Оскільки вираз (4.100) фактично є сумою, то для більшості імпульсних перехідних функцій умови центральної граничної теореми виконуються.

4.10. Операторний метод

Операторний метод моделювання перетворень стохастичних даних дозволяє наближено оцінити закон розподілу вихідного сигналу $f_y(y)$, якщо відомі закони розподілу вхідних сигналів $fx_1(x_1)$, $fx_2(x_2)$ та їх перший і другий моменти, включаючи взаємну кореляційну функцію $R_{X1X2}(x_1,x_2)$.

Нехай вхідні дані X_1 і X_2 розподілені за законами $f_{X1}(x_1)$, $f_{X2}(x_2)$ та їх взаємна кореляційна функція – $R_{x1x2}(\tau)$. Диференціальний закон розподілу вихідного даного $f_y(y)$ може бути знайдений як ***інтегральний оператор*** вигляду:

$$f_Y(y) = \Phi_{XY}(f_X(\bar{x}), A, \mathrm{W}) = \int_{-\infty}^{+\infty}\cdots\int_{-\infty}^{+\infty} f_X(\bar{x})\varphi(x, y, \mathrm{A}, \mathrm{W})\mathrm{d}\bar{x} \qquad (4.103)$$

де Φ_{XY} – інтегральний оператор,

n – кількість вхідних величин,

A та W - параметри алгебраїчного й інтегрально-диференціального перетворень.

Вираз ***ядра оператора*** $\varphi(x,y)$ для нелінійної алгебраїчної операції ґрунтується на формулі (4.80) нелінійного перетворення випадкового процесу і рівнянні регресії. Вираз ядра для інтегрально-диференціального перетворення ґрунтується на поданні такого перетворення інтегралом Дюамеля (4.100). Опис операторів для основних типів операцій наведений у таблиці 4.4.

Таблиця 4.4.

Інтегральні оператори перетворення щільності розподілу

Операція	Оператор	Параметри
$y = N(x)$	$f_y = \int\limits_{\Omega_x} f_x(x)\delta[y - N(x)]dx$	Ω_x - область визначення f_x δ - дельта-функція Дірака
$y = N(x_1, x_2)$	$f_y = \int\limits_{\Omega_{X1}} \int\limits_{\Omega_{X2}} \int\limits_{-\infty}^{+\infty} f_{x_1}(x_1) f_{x_2}(x_2)\delta[y - N(x_1, x_2)]\delta[x_2 + ax_1 + b\xi + c]d\xi dx_1 dx_2$	$a = -r_{x_1x_2}\sqrt{\frac{D_{x_2}}{D_{z_1}}}$ $b = -\sqrt{1 - r^2_{x_1x_2}}$ $c = r_{x_1x_2} m_{x_1}\sqrt{\frac{D_{x_2}}{D_{z_1}}} + (\sqrt{1 - r^2_{x_1x_2}} - 1)m_{x_2}$
$y = \int\limits_0^t x(t-\tau)g(\tau)d\tau$	$f_y = \int\cdots \int\limits_{\Omega_X} \prod\limits_{i=1}^{n} f_x(x_i - m_x)\delta[y - (1-a)m_y - a\sum\limits_{i=1}^{n} x_{n-1}(t - i\tau)\cdot g_0(i\tau)]dx_1 \ldots dx_n$	$\tau = \frac{S_{xx\max}}{D_x}$ $T_{np} = \frac{\pi W_{\max}}{\int\limits_0^\infty W(\omega)d\omega}$ $g_0(i\tau) = \int\limits_{i\tau}^{(i+1)\tau} g(\tau)d\tau$ $n = ent\left[\frac{T_{np}}{\tau}\right]$ $a = \sqrt{\frac{D_y}{\tau D_x \sum\limits_{i=1}^{n} g_0(i\tau)}}$

Контрольні питання і завдання для самостійної роботи

1. Чим відрізняються випадкові величини і випадкові процеси?
2. Визначте типи випадкових процесів та основні характеристики типових дискретних та неперервних процесів.
3. Визначте властивості математичного сподівання та дисперсії випадкового процесу.
4. Що таке стаціонарність та ергодичність? Дайте формулювання загальної ергодичної теореми.
5. Сформулюйте властивості кореляційної функції.
6. Сформулюйте властивості коефіцієнта кореляції.
7. Для чого використовується кореляційне відношення?
8. Для чого використовується кореляційна матриця?
9. Чим відрізняються функціональні та статистичні зв'язки?
10. Що таке регресія? Лінійна та нелінійна регресії.
11. Як виконується оцінка значущості коефіцієнтів регресії?
12. Чому для характеристики випадкових процесів замість спектрів використовуються спектральні щільності потужності?
13. Що таке "марковський процес"?
14. Що таке "ланцюги Маркова"?
15. Що характеризує узагальнене рівняння Маркова?
16. Що описують диференціальні рівняння Колмогорова?
17. Характеристики основних перетворень випадкового процесу.
18. Операторний метод. Оцінка законів розподілу імовірностей.
19. Перетворення спектральної щільності у лінійних та нелінійних системах.
20. Дана матриця вибірних парних коефіцієнтів кореляції розмірності m=4 (n=50):

$$\begin{bmatrix} 1 & -0,85 & -0,62 & -0,21 \\ -0,85 & 1 & -0,53 & 0,34 \\ -0,62 & 0,53 & 1 & 0,46 \\ -0,21 & 0,34 & 0,46 & 1 \end{bmatrix}$$

Обчислити оцінки частинних коефіцієнтів кореляції першого та другого порядків.

21. Визначити оцінки множинних коефіцієнтів кореляції та детермінації першої ознаки з усіма іншими, використовуючи дані попереднього завдання.

22. За даними, наведеними у таблиці, підрахувати оцінки параметрів двовимірної моделі; записати рівняння регресії.

Таблиця 4.5

	Y							
X	1—2	2—3	3—4	4—5	5—6	6—7	7—8	8—9
10—20	4	5	0	0	0	0	0	0
20—30	1	3	1	0	0	0	0	0
30—40	2	3	6	5	3	1	0	0
40—50	0	5	9	19	8	7	2	1
50—60	0	1	2	7	16	9	4	2
60—70	0	0	1	5	6	4	2	2
70—80	0	0	0	0	0	0	1	3

23. За даними таблиці 4.5, оцінити параметри рівняння регресії. Передбачається, що зв'язок визначається формулою $M(Y \mid X = x) = \varphi(x) = \beta_0 + \beta_1 x$

24. За даними, наведеними у таблиці 4.6, визначити коефіцієнти в рівнянні регресії. Зв'язок між змінними має вигляд: $M(Y \mid X = x) = \varphi(x) = \beta_0 + \frac{\beta_1}{x}$

Таблиця 4.6

	Y							
X	4 - 6	6 - 8	8 - 10	10 - 12	12 - 14	14 - 16	16 - 18	18 - 20
3,4—3,8							2	3
3,8—4,2				1	1	3	2	1
4,2—4,6			1	4	2	2		
4,6—5,0		3	1	3	3			
5,0—5,4		3	2					
5,4—5,8	2	2	2					
5,8—6,2	3		1					
6,2—6,6	3	1						

Рекомендована література

1. Сигорский В.П. Математический аппарат инженера. – К.: Техніка, 1977. – 768 с.
2. Сеньо П.С. Випадкові процеси. – Львів: Компакт-ЛВ, 2006. – 288 с.
3. Маликов В.Т., Дубовой В.М., Кветный Р.Н., Исматуллаев П.Р. Анализ измерительных информационных систем. – Ташкент: Фан, 1984. – 176 с.
4. Бартлетт М.С.. Введение в теорию случайных процессов. – М.: ИЛ, 1958. - 384 с.
5. Вентцель Е.С., Овчаров Л.А.. Теория случайных процессов и ее инженерные приложения. – М.: Наука, 1991. - 384 с.
6. Гмурман В.Е. Теория вероятностей и математическая статистика. – М.: Высшая школа, 1977. - 479 с.
7. Гнеденко Б. В. Курс теории вероятностей. — М.: УРСС, 2001. — 448 с.
8. Карлин С. Основы теории случайных процессов. – М.: Мир, 1971. - 536 с.
9. Коваленко И.Н., Кузнецов Н.Ю., Шуренков В.М. – Случайные процессы. – К.: Наукова думка, 1983. - 368 с.
10. Пугачев В.С. Теория случайных функций. – М.: Физматгиз, 1960. - 883 с.
11. Розанов Ю.А. Случайные процессы: Краткий курс. – М.:Наука. 1979. - 183 с.
12. Сеньо П.С. Теорія ймовірностей та математична статистика. – К.: ЦУЛ, 2004. - 445 с.
13. Скороход А.В. Лекції з теорії випадкових процесів. – К.: Либідь, 1990. - 168 с.
14. Лившиц Н.А., Пугачев В.Н. Вероятностный анализ систем автоматического управления. – М.: Советское радио, 1963.
15. Зюко А.Г. Теория передачи сигналов. – М.: Связь, 1980.
16. Харкевич А.А. Спектры и анализ. – М.: Физматгиз, 1962.
17. Глонь О.В., Дубовой В.М., Мітюшкін Ю.І. Комп'ютеризовані системи керування. – Вінниця: ВНТУ, 2005. – 157с.
18. Гриншпан Л.А. Методы анализа стохастических сетевых моделей вычислительных систем. - Минск: Наука и техника, 1988.
19. Дубовой В.М. Програмування систем моделювання інформаційних процесів. - К.: ІСДО України, 1994. - 68с.
20. Дубовой В.М. Моделювання систем контролю та керування. – Вінниця: ВНТУ, 2005. – 175 с.

21. Дубовой В.М., Квєтний Р.Н. Програмування комп'ютеризованих систем управління та автоматики. – Вінниця: ВДТУ, 1997.
22. Евсиков Ю.А., Чапурский В.В. Преобразование случайных процессов в радиотехнических устройствах. - М.: Высшая школа, 1977.
23. Жовинский А.Н., Жовинский В.Н. Инженерный экспресс-анализ случайных процессов. - М.: Энергия, 1979.
24. Коваленко И.Н. Расчет вероятностных характеристик систем. - К.: Техніка, 1982.
25. Корн Г., Корн Т. Справочник по математике. – М.: Наука, 1974.
26. Коршунов Ю.М. Математические основы кибернетики. – М.: Энергия, 1972. – 376 с.
27. Кузин Л.Т. Основы кибернетики. Основы кибернетических моделей. - М., 1977.
28. Левин Б.Р. Статистическая радиотехника. - М.: Советское радио, 1966.
29. Иванова В.М., Калинина В.Н. и др. Математическая статистика. – М.: Высш.шк., 1981. - 371с.
30. Основы моделирования сложных систем. / Под ред. И.В.Кузьмина. - К.: Вища школа, 1981.
31. Прохоров Ю.В., Розанов Ю.А. Теория вероятностей. – М.: Наука, 1973.
32. Пугачев В.С. и др. Основы статистической теории автоматических систем. - М.: Машиностроение, 1974.
33. Пугачев В.С. Теория вероятностей и математическая статистика. - М.: Наука, 1979. – 326 с.
34. Тихонов В.И. Статистическая радиотехника. - М.: Сов. радио, 1966.

5. ФОРМАЛЬНІ АЛГЕБРАЇЧНІ СИСТЕМИ

Основою сучасних підходів до математичного моделювання є формальні алгебраїчні системи. За їх зовнішньою абстрактністю ховається цілком конкретне уявлення про єдність математичного опису різноманітних об'єктів реального світу. На цьому ґрунтується ідея програмування на алгоритмічних мовах, оскільки програма є формальною моделлю.

Даний розділ підготовлений з використанням матеріалів [3, 7, 12].

5.1. Поняття формальної системи

Алгеброю називають математичну систему, яка складається з *множини об'єктів*, *переліку операцій*, які виконуються над цими об'єктами, і *множини аксіом*, яким повинні відповідати ці операції. Решта всіх закономірностей алгебри виводиться з цих аксіом у вигляді теорем.

Формальна система – це сукупність засобів опису алгебри та способів отримання висновків в межах цієї алгебри.

Склад формальної системи:

1) алфавіт – сукупність символів, за допомогою яких записуються всі вирази формальної системи;
2) позначення об'єктів формальної системи;
3) позначення операцій формальної системи;
4) правила утворення формул в цій формальній системі;
5) правила перетворення формул (правила висновку).

Розглянемо детальніше основні елементи формальних систем.

5.1.1. Універсальні алгебри

Універсальною Ω- алгеброю (або просто алгеброю) називається система $G(A, \Omega)$, яка складається з деякої непустої множини A ***(основна множина алгебри*** або ***носій алгебри)*** і множини визначених на A операцій $\Omega = \{\omega_1^{k_1}, \omega_2^{k_2}, ..., \omega_n^{k_n}, ...\}$ ***(сигнатура алгебри),*** де k_i — арність (ar) операції ω_i, $k_i \in \mathrm{N}$, $ar(\omega_i) = k_i$, $i = 1, ..., n, ...$ Операції з множини Ω називаються ***основними операціями алгебри.*** *Арність* функції – це

кількість її аргументів (це узагальнення терміну "бінарна операція" – операція з двома операндами).

Нехай $G=(A,\Omega)$ — довільна алгебра, $\omega\in\Omega$, $ar(\omega)=n$ і $A'\subseteq A$. Підмножина A' називається ***замкнутою*** відносно операції ω, якщо для довільних $a_1,...,a_n$ із A' виконується умова $\omega(a_1,...,a_n)\in A'$. Система (A',Ω) називається ***підалгеброю*** алгебри (A, Q), якщо $A'\subseteq A$ і A' замкнута відносно будь-якої основної операції алгебри $G=(A,\Omega)$.

Алгебра називається ***скінченною,*** якщо її посій має скінченне число елементів.

Якщо алгебра G скінченна і складається з n елементів, то її називають ***алгеброю порядку*** n.

Універсальні алгебри $G=(A,\Omega)$ і $Q=(B,\Omega')$ називаються алгебрами одного ***типу,*** якщо між елементами сигнатур Ω і Ω' можна встановити таку взаємно однозначну відповідність, при якій всяка операція ω із Ω і відповідна їй операція ω' із Ω' будуть мати одну і ту ж арність. Отже, можна вважати, що в алгебрах одного типу заданий один і той же опис операцій.

Алгебра $G=(A,\Omega)$ називається ***гомоморфною*** алгебрі $Q=(B,\Omega)$ того ж типу, що і алгебра G, якщо існує відображення $h: A\to B$, таке, що для всіх елементів $a_1,...,a_n$ із A і будь-якої n-арної операції ω із Ω справедлива рівність

$$h(\omega(a_1,...,a_n)=\omega(h(a_1),...,h(a_n)). \qquad (5.1)$$

При цьому відображення h називається ***гомоморфізмом.*** Якщо h— взаємно однозначне відображення алгебри G на алгебру Q, то воно називається ***ізоморфізмом,*** а алгебри G ***і*** Q— ***ізоморфними*** $(G\sim Q)$.

5.1.2. Абсолютно вільні алгебри

Алгебра $T(\Omega, X, Eq)$ називається ***абсолютно вільною,*** коли всякий елемент із $T(\Omega, X)$ єдиним способом виражається через елементи алфавіту X і символи нульарних операцій із Ω.

Якщо Eq містить співвідношення асоціативності, співвідношення скорочення $(x\cdot x^{-1}=x^{-1}\cdot x=e)$ і співвідношення, пов'язані з одиницею

$(x \cdot e = e \cdot x = x)$, то $T(\Omega, X, E_q)$ називається ***вільною групою.*** Одиницею групи $T(\Omega, X, Eq)$ є пусте слово e. Тотожні співвідношення дозволяють записати довільне слово з $T(\Omega, X, Eq)$ у вигляді нескорочуваного слова $x_{i_1}^{n_1} x_{i_2}^{n_2} \ldots x_{i_k}^{n_k}$.

5.1.3. Поняття σ-алгебри

Система F підмножин множини Ω називається ***σ-алгеброю***, якщо вона разом з будь-якою послідовністю множин $A_1, \ldots, A_n$ містить також їх об'єднання $\bigcup_{n=1}^{\infty} A_n$.

Властивості σ-алгебри:

1) $\Omega \in F$ (Ω - одиниця в σ-алгебрі)

1а) $\varnothing \in F$

2) якщо $A_1, \ldots, A_n, \ldots \in F$, то $\bigcup_{i=1}^{\infty} \in F$ (інакше кажучи, замкнуто стосовно зліченних об'єднань)

2а) якщо $A_1, \ldots, A_n, \ldots \in F$, то $\bigcap_{i=1}^{\infty} \in F$, то (інакше кажучи, замкнуто стосовно зліченних перетинань)

3) якщо $A, B \in F$, то $A \setminus B \in F$.

5.2. Групи, кільця, класи, простори

5.2.1. Алгебраїчні системи

Існують типові математичні системи, які розглядають загальні закономірності алгебр різних типів. Такий загальний розгляд алгебр допомагає у з'ясовуванні та доведенні властивостей нових алгебр.

Визначаючи на деякій множині S один або два закони композиції й наділяючи їх певними властивостями, а також задаючи структуру множини стосовно законів композиції (наявність нейтрального елемента і симетричних множин), одержуємо різні алгебраїчні системи, (структури або моделі), наведені у таблиці 5.1.

Таблиця 5.1

Алгебраїчні системи

Назва алгебраїчних систем	Перший закон (адитивний)				Другий закон (мультиплікативний)			
	Властивості		Елементи		Властивості		Елементи	
	Асоціатив-ність	Комутатив-ність	Нейтральн-ий	Симетричн-ий	Асоціатив-ність	Комутатив-ність	Нейтральн-ий	Симетричн-ий
Півгрупа (моноїд)	*							
Абелева (комутативна) півгрупа	*	*						
Півгрупа з нулем (одиницею)	*		*					
Абелева півгрупа з нулем (одиницею)	*	*	*					
Група	*		*	*				
Абелева (комутативна) група	*	*	*	*				
Асоціативне кільце	*	*	*	*	*			
Абелеве (комутативне) кільце	*	*	*	*	*	*		
Кільце з одиницею (унітарне кільце)	*	*	*	*	*		*	
Абелеве кільце з одиницею	*	*	*	*	*	*	*	
Тіло	*	*	*	*	*		*	*
Поле (комутативне тіло)	*	*	*	*	*	*	*	*

Примітки: 1. Другий закон композиції (якщо він визначений) є дистрибутивним ліворуч і праворуч щодо першого закону.

2. Симетричні елементи стосовно другого закону визначені для всіх елементів, крім нейтрального стосовно першого закону (нуля).

Так, група - це наділена асоціативним законом множина, що містить нейтральний елемент і симетрична відносно цього закону. Якщо, крім того, закон композиції комутативний, то групу називають абелевою (комутативною).

У всякій групі співвідношення (рівняння) $a \perp x = b$ і $y \perp a = b$ допускають єдиний розв'язок $x = \bar{a} \perp b$ (частка праворуч) і $y = b \perp \bar{a}$ (частка ліворуч). Має місце також співвідношення $\overline{(a \perp b)} = \bar{a} \perp \bar{b}$ або $-(a+b) = -b - a$ (в адитивному записі) і $(a \cdot b)^{-1} = b^{-1} \cdot a^{-1}$

(у мультиплікативному записі).

Кільце - це множина, наділена двома законами композиції $A \bigcup B \in K$ і $A \setminus B \in K$: щодо першого (адитивного) воно утворює абелеву групу, а другий закон (мультиплікативний) є асоціативним, а також дистрибутивним щодо першого закону. Тілом називають кільце з одиницею, у якому кожний відмінний від нуля елемент є симетричним щодо другого (мультиплікативного) закону. Поле - це комутативне тіло.

Вивчення алгебраїчних систем дозволяє виявити загальні властивості операцій на множинах об'єктів різної природи. Ці властивості використовуються при рішеннях багатьох наукових і технічних задач. З наведених алгебраїчних систем найбільш широкими поняттями є моноїд і група, а найбільш вузькими - тіло й поле. Останні обслуговують в основному числові множини, у той час як більш широкі поняття поширюються й на більш далекі від чисел сукупності об'єктів.

5.2.2. Вільне кільце

Вільна група $T(\Omega, X, Eq)$ називається ***вільним кільцем,*** якщо Eq визначає $T(\Omega, X, Eq)$ як:

1) вільну абелеву групу відносно додавання;

2) вільний групоїд відносно множення;

3) таку, в якій виконуються закони дистрибутивності, тобто для будь-яких x, x', x'' із $T(\Omega, X, Eq)$

$$x(x'+x'') = (xx') + (xx''), \tag{5.2}$$

$$(x+x')x'' = (xx'') + (x'x''). \tag{5.3}$$

Інакше кажучи, Ω— це чотири операції: бінарні операції додавання і множення, унарна операція знаходження оберненого до операції додавання і нульарна операція, яка фіксує нульовий елемент абелевої групи кільця. Цей елемент називається ***нульовим*** елементом кільця.

5.2.3. Векторні простори

Вільна група $T(\Omega, X, Eq)$ є векторним простором над деяким полем P, якщо Ω містить бінарну операцію додавання елементів, нескінченне число унарних операцій множення елементів па елементи з поля P (для кожного a з P своя операція) і нульарну операцію, яка фіксує нульовий елемент. Множина Eq — це всі співвідношення, що визначають $T(\Omega, X, Eq)$ як вільну абелеву групу, тобто для довільних x, x', x'' із $T(\Omega, X, Eq)$ маємо

$$x+(x'+x'') = (x+x')+x'', \quad x+0 = x,$$

$$x+x' = x'+x, \qquad x+(-x) = 0,$$

а також для довільних x, x' із $T(\Omega, X, Eq)$ і a, b із P виконуються рівності

$$a\cdot(x+x') = a\cdot x + a\cdot x';$$

$$(a+b)\cdot x = a\cdot x + b\cdot x$$

$$a\cdot(b\cdot x) = (a\cdot b)\cdot x.$$

Елементами $T(\Omega, X, Eq)$ є елементи, що називаються ***векторами.*** Символам x_i із X ставиться у відповідність вектор

$$0\cdot x_1 + 0\cdot x_2 + \ldots + 1\cdot x_i + \ldots + 0\cdot x_n + \ldots,$$

а роль нульового вектора простору відіграє елемент

$$0\cdot x_1 + 0\cdot x_2 + \ldots + 0\cdot x_i + \ldots + 0\cdot x_n + \ldots = 0$$

де $i = 1, 2, \ldots$

5.2.4. Структури

Алгебра $G = (A, \Omega) \in K(\Omega, Eq)$ називається ***структурою,*** якщо Ω складається з двох бінарних операцій $\vee$ (верхня грань) і $\wedge$ (нижня грань), $a \forall a,\ b,\ c \in A$ справедливі такі співвідношення (закони):

1) $a \vee a = a \wedge a = a$ — ідемпотентність;
2) $a \vee b = b \vee a,\ a \wedge b = b \wedge a$ — комутативність;
3) $a \vee (b \vee c) = (a \wedge b) \vee c$,
   $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ — асоціативність;
4) $a \vee (a \wedge b) = a,\ a \wedge (a \vee b) = a$ — поглинання.

5.3. Приклади алгебраїчних систем

Алгебраїчний підхід до постановки і розв'язання задач втілився у побудову алгоритмічних мов. У сучасних мовах програмування поняттю алгебри відповідає поняття *типу* або *класу*. Найкраще такий підхід був запроваджений Н. Віртом при створенні мови Паскаль, яка у подальшому була взята за основу стандарту ANSI щодо побудови алгоритмічних мов.

Тип змінної у мовах програмування – це множина значень, які може приймати змінна, і відповідний набір операцій над цими значеннями, що цілком відповідає поняттю алгебри.

У базовому варіанті мови Паскаль були реалізовані основні типи (алгебри): цілих чисел, дійсних чисел, логіки (булева), множин, термів (рядків символів). Крім того, структурний підхід до програмування мовою Паскаль відповідає поняттям і правилам алгебри алгоритмів. Але означення множин значень і означення операцій над ними були ще відокремлені. З подальшим розвитком об'єктного підходу до програмування і введенням поняття *класу* поєднання множини значень з набором операцій і правил їх виконання стало обов'язковим.

5.3.1. Лінійна алгебра

Лінійна алгебра — один із найбільших і важливих розділів сучасної алгебри, що має численні застосування в усіх областях математики і кібернетики. На сучасному етапі розвитку можна вважати, що лінійна

алгебра— це та область алгебри, що вивчає властивості векторних просторів.

Найстарішою проблемою лінійної алгебри є задача знаходження розв'язку систем лінійних рівнянь і вивчення властивостей таких розв'язків.

Інша важлива тема лінійної алгебри — вивчення лінійних перетворень вигляду:

$$\begin{aligned} y_1 &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ y_2 &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ y_n &= a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n \end{aligned} \qquad (5.4)$$

Ця тема виникла спочатку в аналітичній геометрії у зв'язку з перетвореннями координат: саме за зазначеною схемою перетворюються декартові координати точки при переході від однієї системи координат до іншої. У кібернетиці такі перетворення є моделлю лінійних систем у просторі станів.

Алгебра матриць — розділ лінійної алгебри, в якому вивчаються матриці й операції над ними. Матриці — це прямокутні або квадратні таблиці вигляду

$$A = \begin{pmatrix} a_{11}a_{12}\cdots a_{1n} \\ a_{21}a_{22}\cdots a_{2n} \\ \cdots\cdots\cdots\cdots \\ a_{n1}a_{n2}\cdots a_{nn} \end{pmatrix}, \qquad (5.5)$$

де a_{ik} — елементи будь-якої множини S; говорять, що A — матриця над S.

Послідовності $(a_{i1}, a_{i2}, \ldots, a_{in})$ $(i = 1, 2, \ldots, m)$ — рядки, а послідовності $(a_{1k}, a_{2k}, \ldots, a_{mk})$ $(i = 1, 2, \ldots, n)$ – стовпці матриці A. Послідовність $(a_{11}, a_{22}, \ldots)$ називається ***діагоналлю*** матриці A. Матриця розміру $m \times n$ (коротко $A[m, n]$-матриця) — це матриця з m рядками й n стовпцями; при $m = n$ її називають ***квадратною матрицею*** порядку n. .

Для матриць визначені операції додавання, віднімання, множення (в тому числі множення на скаляр), які виконуються за правилами:

- додавання $C[m,n] = A[m,n] + B[m,n]$, якщо $\forall i, j \to c_{ij} = a_{ij} + b_{ij}$;
- віднімання $C[m,n] = A[m,n] - B[m,n]$, якщо $\forall i, j \to c_{ij} = a_{ij} - b_{ij}$;

- множення $C[m,n]=A[m,k]\bullet B[k,n]$, якщо $\forall i,j \to c_{ij}=\sum_{l=1}^{k} a_{il}b_{lj}$;
- множення на скаляр $C[m,n]=k\cdot A[m,n]$, якщо $\forall i,j \to c_{ij}=k\cdot a_{ij}$,

а також визначені поняття одиничної і оберненої матриць.

Операція віднімання є похідною від множення і додавання. Операція додавання є комутативною, а операція множення – некомутативною.

5.3.2. Алгебра множин

Алгебра множин — розділ теорії множин, що вивчає операції над підмножинами (частинами) заданої множини й поводження цих операцій при відображеннях множин. Алгебра множин застосовується в теоретичній кібернетиці й у техніці. Ідеї алгебри множин висловив Дж. Буль в 1847 р. Одночасно він дав перше формулювання сучасної (символічної або математичної) логіки. Основи алгебри множин розглянуто у розділі 1.

5.3.3. Алгебра логіки

Алгебра логіки почала формуватися в 19 ст. у працях англійського математика Дж. Буля. Алгебру логіки створили для розв'язання традиційних логічних задач алгебраїчними методами.

Основою алгебри логіки є множина, що складається з двох елементів: істинного (позначається T – True або 1) та хибного (позначається F – False або 0) висловлювань.

Вживані у звичайній мові логічні зв'язування «і», «або», «якщо.., то», «еквівалентно», частка «не» і т.д. дозволяють із вже заданих висловлень будувати нові, більш «складні» висловлювання. Так, з висловлювань «x>2», «x≤3» за допомогою зв'язування «і» можна одержати висловлення «x>2» і «x≤3», за допомогою зв'язування «або» — висловлення «x>2» або «x≤3» тощо. Зв'язки «і», «чи», «якщо...; то», «еквівалентно» позначаються відповідно знаками & *(кон'юнкція)*, ∨ *(диз'юнкція)*, → *(імплікація)*, ~ (еквівалентність); для заперечення вводиться знак $\bar{}$ (риска зверху).

Істинність або хибність одержуваних таким чином висловлень залежить від істинності або хибності вихідних висловлень і відповідного трактування зв'язувань як операцій над висловленнями.

Введені операції дозволяють кожній формулі при заданих значеннях вхідних у неї висловлень приписати одне із двох значень – «0» або «1». При цьому формули a і b називають еквівалентними (позначення: $a=b$), якщо вони реалізують рівні функції. Для задавання функцій алгебри логіки іноді використовують таблиці, що містять всі набори значень змінних і значення функцій на цих наборах. Це так званий табличний спосіб задання функцій. Самі ж таблиці називають ***таблицями істинности***. Так, наприклад, зведена таблиця 5.2, що задає функції $\bar{x}$, $x\&y$, $x\vee y$, $x\to y$ та $x\sim y$, має вигляд

Таблиця 5.2

Основні функції алгебри логіки

x	y	$\bar{x}$	$x\&y$	$x\vee y$	$x\to y$	$x\sim y$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Алгебра $G=(A, \Omega)\in K(\Omega, Eq)$ називається ***булевою алгеброю***, якщо Ω складається з:

- двох бінарних операцій — $\vee$ *(або)* і $\&$ *(і)*;
- однієї унарної операції — $\neg$ *(заперечення)*;
- двох нульарних операцій — 0 *(нуль)* і 1 *(одиниця)*,

і для довільних $a, b, c\in A$ виконується така сукупність співвідношень Eq:

1) $a\vee b=b\vee a$, $a\&b=b\&a$ — комутативність;

2) $a\vee(b\vee c)=(a\vee b)\vee c$,

$a\&(b\&c)=(a\&b)\&c$ — асоціативність;

3) $a\vee(b\&c)=(a\vee b)\&(a\vee c)$,

$a\&(b\vee c)=(a\&b)\vee(a\&c)$ — дистрибутивність;

4) $a\vee 0=a$, $a\vee\neg a=1$, $a\&1=a$, $a\&\neg a=0$ — закони для нуля, одиниці і заперечення.

5.3.4. Мова (алгебра) термів

Для задання функцій і операцій, а також для вивчення їх властивостей користуються формальною мовою — ***мовою термів***. Для визначення

деякої формальної мови необхідно задати її алфавіт і правила, за якими будуються слова із символів цього алфавіту.

Довільна сукупність попарно різних символів $X = \{x_1, x_2, ..., x_n, ...\}$ називається ***алфавітом.*** Символи алфавіту часто ще називають ***буквами.*** Найпростіший приклад мови — це мова слів в алфавіті X. Із скінченної послідовності $p = x_{i1}...x_{ik}$ букв складається слово в алфавіті X. При цьому число букв x_{ij} називається довжиною $l(p)$ слова p. Крім слів, довжина яких виражається цілим додатним числом, розглядається слово нульової довжини, яке за означенням не має жодного символу і називається ***пустим словом.*** Для позначення цього слова вводиться спеціальний символ $e \in X$.

Очевидно, що коли p, q — деякі слова в алфавіті X, тобто послідовності $x_{i1}...x_{ik}$ та $x_{j1}...x_{jk}$, відповідно, то послідовність $pq = x_{i1}...x_{ik}x_{j1}...x_{jk}$ одержана в результаті приписування слова q безпосередньо після останнього символу слова p, теж буде, очевидно, словом в алфавіті X. Це випливає безпосередньо з означення слова. Отже, таке сполучення слів в алфавіті X можна розглядати як операцію на множині слів алфавіту X. Ця операція має назву ***конкатенація*** або ***добуток слів.***

Термами називаються слова, побудовані за такими правилами:

1) всі символи з T_0 — терми;

2) якщо $t_1,...,t_n$ — терми, то слово $f^n(t_1,...,t_n)$ — терм $(f^n \in F, n \geq 1)$.

Множину термів $T(\Omega, X)$ можна розглядати як універсальну Ω-***алгебру термів.***

Контрольні питання і завдання для самостійної роботи

1. Дайте означення алгебри, формальної системи; склад формальної системи.
2. Охарактеризуйте універсальну та абсолютно вільну алгебру.
3. Дайте означення σ-алгебри та її властивостей.
4. Визначте основні типи алгебраїчних систем та їх властивості.
5. Наведіть приклади алгебраїчних систем та їх застосування.
6. Сформулюйте властивості операцій алгебри цілих чисел. До якого типу алгебр вона відноситься?

1. Які алгебри реалізовані у мові програмування С?

Рекомендована література

1. Барвайс Дж. Справочная книга по математической логике. - М.: Наука, 1982. – 376 с.
2. Шенфилд Дж. Математическая логика. - М.: Наука, 1975. – 528 с.
3. Дубовой В.М. Моделювання систем контролю та керування. – Вінниця: ВНТУ, 2005. – 175 с.
4. Дубовой В.М., Кветний Р.Н. Програмування комп'ютеризованих систем управління та автоматики. – Вінниця: ВДТУ, 1997.
5. Лавров И.А., Максимова Л.Л. Задачи по теории множеств, математической логике и теории алгоритмов. - М.: Наука, 1984.–224 с.
6. Бондарев В.М. и др. Основы программирования. – Харьков: Фолио, 1997.
7. Капітонова Ю.В. та ін. Основи дискретної математики. – К.: Наукова думка, 2002.
8. Корн Г., Корн Т. Справочник по математике. – М.: Наука, 1974.
9. Коршунов Ю.М. Математические основы кибернетики. – М.: Энергия, 1972. – 376 с.
10. Ланкастер П. Теория матриц. - М.: Наука, 1978.
11. Рейуорд-Смит В.Дж. Теория формальных языков. – М.: Радио и связь, 1988.
12. Энциклопедия кибернетики. – К.: Главная редакция УСЭ, 1974.
13. Мальцев А. И. Основы линейной алгебры. - М., 1970.
14. Артин Э. Геометрическая алгебра. - М., 1989.
15. Гантмахер Ф. Р. Теория матриц. - М., 1967.
16. Беллман Р. Введение в теорию матриц. - М., 1969.
17. Александров П. С. Введение в теорию множеств и теорию функций. - М.- Л., 1948.
18. Столл Р.Р. Множества. Логика. Аксиоматические теории. - М., 1968.
19. Новиков П.С. Элементы математической логики. - М., 1959.
20. Гильберт Д., Аккерман В. Основы теоретической логики. - М., 1947.
21. Яблонский С.В., Гаврилов Г.П., Кудрявцев В.Б. Функции алгебры логики и классы Поста. - М., 1966.

6. ОСНОВИ ТЕОРІЇ АЛГОРИТМІВ

Бурхливий розвиток комп'ютерної техніки, проникнення її в усі сфери сучасної цивілізації привернули особливу увагу до тих розділів математики, які є теоретичною основою методів комп'ютеризації і програмування. Такою основою є в першу чергу теорія алгоритмів.

Умовно можна виділити три основні етапи розвитку теорії алгоритмів. На першому етапі відбувалося накопичення фактів людиною, що засвоювала знання про природу і стихійно формувала та закріплювала власні алгоритми поведінки.

Другий етап становлення алгоритмічного апарату пов'язаний з розвитком математичних наук. Для нього є характерним чітке задання математичних алгоритмів розв'язання різних прикладних задач і опис значних алгоритмічних проблем, що не піддавались розв'язанню традиційними методами, наприклад трисекція кута.

Третій період бере відлік з початку XX ст. Було побудовано формальну класичну теорію алгоритмів, яка уточнювала можливості теоретичного обчислення для практичного застосування в кібернетиці й програмуванні.

Теорію алгоритмів можна поділити на класичну і прикладну. У класичній теорії алгоритмів існує безліч формальних методів опису алгоритмів. Серед них можна виділити найзначніші: арифметичне числення предикатів Геделя, машини Поста і Тьюрінга, автомати Маркова, схеми Янова, блок-схеми.

Даний розділ підготовлений з використанням матеріалів [1, 5, 6, 11, 12].

6.1. Означення і властивості алгоритму

6.1.1. Інтуїтивне поняття алгоритму

Поняття алгоритму є одним з базових понять математики (таких, як число, множина, точка тощо), про які існує лише інтуїтивне уявлення без строгого математичного означення, оскільки будь-яке означення само ґрунтується на базових поняттях. Спроби дати означення таким базовим поняттям приводить лише до заміни одних інтуїтивних понять іншими.

Інтуїтивне поняття алгоритму містить у собі кілька загальних рис.

1. ***Алгоритм*** — це процес послідовної побудови величин, який проходить у дискретному часі таким чином, що в початковий момент задається початкова скінченна множина величин, а в кожний наступний система величин одержується за цілком визначеним законом (програмою) із системи величин, які були в попередній момент часу (*дискретність алгоритму*).

2. Система величин, що одержується в деякий відмінний від початкового момент часу, однозначно визначається системою величин, одержаних у попередні моменти часу (*детермінованість алгоритму*).

3. Закон одержання подальших систем величин з попередніх повинен бути простим і локальним (*елементарність кроків алгоритму*).

4. Якщо спосіб одержання наступної величини з будь-якої заданої величини не дає результату, то слід вказати, що вважається результатом алгоритму (*результативність алгоритму*).

5. Початкова система величин може вибиратись із деякої потенціально нескінченної множини (*масовість алгоритму*).

Користуючись інтуїтивним поняттям алгоритму, можна описувати процес розв'язання тієї чи іншої задачі, але з його допомогою не можна впевнитися в тому, що описаний процес являє собою алгоритм. Дійсно, одна справа — довести існування алгоритму, а зовсім інша — довести його відсутність. Для цього потрібно знати точно, що таке алгоритм. Розв'язок цієї задачі одержано в середині 30-х років XX ст. у двох формах. Перша ґрунтувалась на понятті рекурсивної функції, а друга — на точно окресленому класі процесів. Обидві форми одержали назву ***алгоритмічні системи.***

Теорія алгоритмів вивчає основні закономірності роботи алгоритмів, способи доведення їх правильності, способи визначення чи є у алгоритмічної задачі розв'язок. З іншого боку, теорія алгоритмів займається пошуком алгоритмічно нерозв'язних проблем.

Існує декілька напрямків у теорії алгоритмів, які відрізняються так само, як відрізняються між собою різні алгебри, тобто переліком об'єктів, з якими працюють алгоритми, та операцій, які можуть виконуватися над цими об'єктами.

Основні напрямки теорії алгоритмів:

- алгебра рекурсивних функцій;

- теорія автоматів.

Останнім часом цікаві результати в теорії алгоритмів отримали за допомогою алгоритмічної алгебри Глушкова. В теорії алгоритмів доведений ізоморфізм цих теорій, тобто, якщо результат отримано за допомогою, наприклад, рекурсивних функцій, то аналогічний результат обов'язково може бути отримано за допомогою теорії автоматів, і навпаки. Цей ізоморфізм сформульований англійським вченим Черчем

Алгоритм стосовно обчислювальної машини – точне розпорядження, тобто набір операцій і правил їх чергування, за допомогою якого, починаючи з деяких вихідних даних, можна розв'язати будь-яку задачу фіксованого типу.

Види алгоритмів як логіко-математичних засобів відбивають зазначені компоненти людської діяльності і тенденції, а самі алгоритми в залежності від мети, початкових умов задачі, шляхів її розв'язання, визначення дій виконавця підрозділяються на декілька типів (рис.6.1).

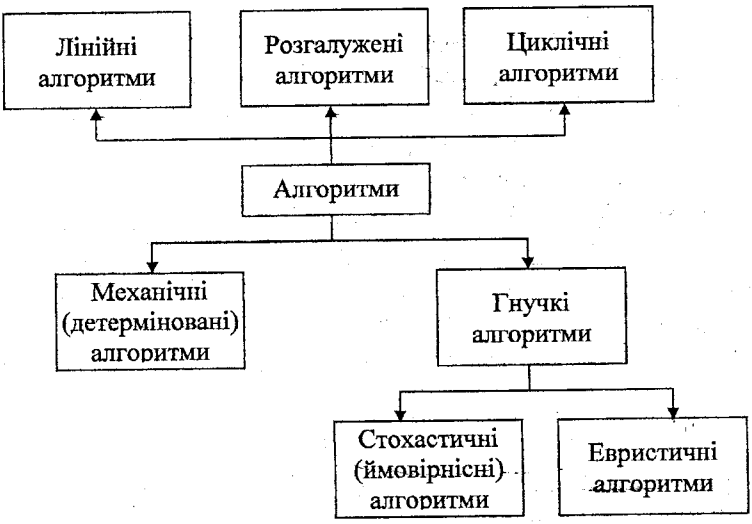


Рис. 6.1. Загальна класифікація алгоритмів в залежності від шляху розв'язування задачі та визначення дії виконавця

▪ Механічні алгоритми, чи інакше детерміновані, жорсткі (наприклад, алгоритм роботи машини, двигуна і т.п.). Механічний

алгоритм задає певні дії, позначаючи їх у єдиній і достовірній послідовності, забезпечуючи тим самим однозначний необхідний чи шуканий результат, якщо виконуються ті умови процесу, задачі, для яких розроблений алгоритм;

- Гнучкі алгоритми, наприклад стохастичні (ймовірнісні) й евристичні. Ймовірнісний (стохастичний) алгоритм дає програму розв'язування задачі декількома шляхами чи способами, що приводять до ймовірного досягнення результату. Евристичний алгоритм (від грецького слова "евристика") – це такий алгоритм, у якому досягнення кінцевого результату програми дій однозначно не визначено, так само як не позначена вся послідовність дій, не виявлені всі дії виконавця. До евристичних алгоритмів відносять, наприклад, інструкції і розпорядження. У цих алгоритмах використовуються універсальні логічні процедури і способи прийняття рішень, засновані на аналогіях, асоціаціях і минулому досвіді розв'язання схожих задач.

За структурою алгоритми поділяються на лінійні, розгалужені і циклічні.

- Лінійний алгоритм — набір команд (вказівок), виконуваних послідовно в часі один за одним.
- Розгалужений алгоритм - алгоритм, що містить хоча б одну умову, у результаті перевірки якого ЕОМ забезпечує перехід на один із двох можливих кроків.
- Циклічний алгоритм – алгоритм, що передбачає багаторазове повторення однієї і тієї ж дії (одних і тих же операцій) над новими вихідними даними. До циклічних алгоритмів зводиться більшість методів обчислень, перебору варіантів.

Найбільш поширені такі *форми зображення алгоритмів*:

— формальна.

— словесна (запис природною мовою). Даний спосіб одержав значно менше поширення через його багатослівність і відсутність наочності. Словесний спосіб не має широкого поширення через такі причини: такі описи строго не формалізуються; страждають багатослівністю записів; допускають неоднозначність тлумачення окремих розпоряджень.

— графічна (зображення з графічних символів). А цей спосіб виявився дуже зручним засобом зображення алгоритмів і одержав широке

поширення в науковій і навчальній літературі. Структурна схема алгоритму – графічне зображення алгоритму у вигляді схеми зв'язаних між собою за допомогою стрілок (ліній переходу) блоків – графічних символів, кожний з яких відповідає одному кроку алгоритму. Усередині блока дається опис відповідної дії. Принцип програмування «зверху вниз» вимагає, щоб блок-схема поетапно конкретизувалася і кожен блок «розписувався» до елементарних операцій. Але такий підхід можна здійснити при розв'язуванні нескладних задач. При розв'язуванні серйозної задачі блок-схема "розповзеться" до такої величини, що її неможливо буде охопити одним поглядом. Блок-схеми алгоритмів зручно використовувати для пояснення роботи вже готового алгоритму, при цьому як блоки беруться дійсно блоки алгоритму, робота яких не вимагає пояснень. Блок-схема алгоритму повинна служити для спрощення зображення алгоритму, а не для ускладнення.

— псевдокоди (напівформалізовані описи алгоритмів умовною алгоритмічною мовою, що включають у себе як елементи мови програмування, так і фрази природної мови, загальноприйняті математичні позначення й ін.). З одного боку, він близький до звичайної природної мови, тому алгоритми можуть на ньому записуватися і читатися як звичайний текст. З іншого боку, у псевдокоді використовуються деякі формальні конструкції і математична символіка, що наближає запис алгоритму до загальноприйнятого математичного запису. У псевдокоді не прийняті строгі синтаксичні правила для запису команд, властиві формальним мовам, що полегшує запис алгоритму на стадії його проектування і дає можливість використовувати більш широкий набір команд, розрахований на абстрактного виконавця. Однак у псевдокоді звичайно існують деякі конструкції, властиві формальним мовам, що полегшує перехід від запису на псевдокоді до запису алгоритму формальною мовою. Зокрема, у псевдокоді, так само, як і у формальних мовах, є службові слова, зміст яких визначений раз і назавжди. Вони виділяються в друкованому тексті жирним шрифтом, а в рукописному тексті підкреслюються. Єдиного чи формального визначення псевдокоду не існує, тому можливі різні псевдокоди, що відрізняються набором службових слів і основних (базових) конструкцій.

— програмна (тексти мовами програмування). На практиці як виконавців алгоритмів використовують спеціальні автомати — комп'ютери. Тому алгоритм, призначений для виконання на комп'ютері, повинен бути записаний «зрозумілою» йому мовою. І тут на перший план висувається необхідність точного запису команд, що не залишає місця для довільного тлумачення їхнім виконавцем.

Більш строгим стає поняття алгоритму при розгляданні нормального алгоритму Маркова. Виявляється, будь-який логічний алгоритм можна досить простими методами звести до чисельного. Тому теорію чисельних алгоритмів можна вважати універсальним апаратом для дослідження всіх алгоритмічних проблем.

Покажемо, як будь-яку алгоритмічну проблему можна звести до обчислення значень деякої цілочисельної функції при цілочисельних значеннях аргументів.

Позначимо всі умови задачі, що перероблюються даним алгоритмом А, у вигляді послідовності із цілими невід'ємними індексами-номерами:

$$A_0, A_1, ..., A_n, ...$$

Розв'язок може з'являтися за нумерованою послідовністю

$$B_0, B_1, ..., B_m, ...$$

Після введення нумерації будемо оперувати не із самими записами умов і розв'язань, а з їхніми номерами. Тепер можна подати алгоритм, що переробляє номер запису умов у номер запису розв'язку. Цей алгоритм здійснює обчислення значень числової функції

$$m = \varphi(n),$$

тобто він є чисельним алгоритмом.

Якщо існує алгоритм, що розв'язує вихідну задачу, то існує алгоритм, що обчислює значення відповідної функції, Справді, для знаходження значення $\varphi(n)$ при $n=n^*$ можна вибрати запис умови для n^*, далі за допомогою наявного алгоритму знайти запис розв'язування й за ним визначити відповідний номер m^*. Таким чином,

$$\varphi(n^*) = m^*$$

Справедливе й зворотне твердження: якщо існує алгоритм обчислення функції $\varphi(n)$, то існує й алгоритм розв'язування вихідної

задачі.

Враховуючи, що будь-яке число можна розкласти на прості множники єдиним образом, можна стверджувати, що кожному числу однозначно відповідає набір $a_0, a_1, ..., a_m$ і, навпаки, кожному набору $a_0, a_1, ..., a_m$ однозначно відповідає число n.

Інакше кажучи, не тільки арифметичні алгоритми зводяться до обчислення значень цілочисельних функцій. Будь-який нормальний алгоритм Маркова із застосуванням методу Гегеля також можна звести до обчислення значень цілочисельних функцій. Так що алгоритм обчислення значень цілочисельних функцій можна вважати універсальною формою алгоритму.

Варто помітити, що при розв'язуванні алгоритмічних задач множина вихідних даних передбачається зчисленною, хоча може бути й нескінченною.

6.1.2. Загальні властивості алгоритмів

Наведене вище означення алгоритму не можна вважати строгим – не цілком ясно, що таке «точний припис» чи «послідовність дій, що забезпечує одержання необхідного результату». Тому звичайно формулюють кілька загальних властивостей алгоритмів, що дозволяють відрізняти алгоритми від інших інструкцій (табл. 6.1).

Алгоритм може бути призначений для виконання його людиною чи автоматичним пристроєм. Створення алгоритму, нехай навіть найпростішого, - процес творчий. Реалізацію вже наявного алгоритму можна доручити суб'єкту чи об'єкту, що не зобов'язаний вникати в суть справи, а можливо, і не здатний його зрозуміти. Такий суб'єкт чи об'єкт прийнято називати формальним виконавцем. Людина теж може виступати в ролі формального виконавця, але в першу чергу формальними виконавцями є різні автоматичні пристрої, і комп'ютер у тому числі. Ті дії, що може робити виконавець, називаються його припустимими діями. Сукупність припустимих дій утворить систему команд виконавця. Алгоритм повинен містити тільки ті дії, що припустимі для даного виконавця.

Таблиця 6.1

Властивості алгоритму

Властивість	Опис властивості алгоритму
Дискретність (преривність, роздільність)	Алгоритм повинен зображати процес розв'язування задачі як послідовне виконання простих кроків. Кожна дія, яка передбачена алгоритмом, виконується тільки після того, як закінчилось виконання попередньої дії.
Визначеність	Кожне правило алгоритму повинно бути чітким та однозначним. Завдяки цій властивості виконання алгоритму носить механічний характер і не потребує ніяких додаткових вказівок і відомостей про задачу.
Результативність (кінцевість)	Алгоритм повинен приводити до розв'язання задачі за кінцеву кількість кроків. Алгоритм є скінченним об'єктом, що є необхідною умовою його механічної реалізованості;
Масовість	Алгоритм розв'язування задачі розробляється в загальному вигляді, тобто він може застосовуватись для класу задач, які відрізняються тільки вихідними даними,
Правильність	Застосування алгоритму до правильних вхідних даних повинно приводити до отримання необхідних результатів. Доведення правильності алгоритму – один з найбільш складних етапів його створення. Частіше за все правильність перевіряється на наборі тестів, які підібрані таким чином, щоб захопити всі допустимі вхідні та вихідні дані.
Ефективність	Алгоритм повинен забезпечити розв'язання задачі за мінімальний проміжок часу з мінімальними затратами пам'яті. Для оцінки алгоритмів існує багато критеріїв. Частіше за все оцінка алгоритму складається з оцінки часових затрат на розв'язання задачі в залежності від „розміру" вхідних даних. Використовується також термін „часова здатність" і „трудомісткість алгоритму". Фактично ця оцінка зводиться до оцінки кількості основних операцій, які виконує алгоритм, оскільки кожна конкретна операція виконується за кінцевий завчасно відомий час.

6.2. Алгоритми і цифрові автомати

6.2.1. Автоматний спосіб задання алгоритму

Основою обчислювальної частини комп'ютера є перетворювачі, що сприймають деякі вхідні дані (послідовності нулів та одиниць) і перетворюють їх на вихідні дані (послідовності). Такі перетворювачі називають логічними схемами.

Логічні схеми можна розділити на два класи. Перший з них – комбінаційні схеми. Значення вихідних змінних комбінаційної схеми визначаються лише значеннями вхідних змінних і не залежать від поточного стану схеми (від попередніх даних). Будь-яка комбінаційна схема реалізує ту чи іншу булеву функцію від своїх вхідних змінних.

На відміну від комбінаційної схеми скінченний автомат є перетворювачем, вихід якого залежить не тільки від вхідних і вихідних сигналів, але й від поточного стану автомата, причому кількість вхідних і вихідних змінних, кількість можливих значень цих змінних, а також кількість можливих станів автомата скінченна. Поточний стан автомата зберігається в його пам'яті.

Автомати можна розглядати як машини, що мають пам'ять у вигляді стрічки та пристрій для зчитування інформації зі стрічки. Стрічка розділена на квадрати, в яких розміщуються певні символи. Автомат, проглядаючи ці квадрати по черзі, виконує окремі обчислення і розв'язує задачі.

Скінченні автомати мають досить обмежені обчислювальні можливості. Більше можливостей мають автомати з нескінченною пам'яттю магазинного типу, але й вони є недостатньо універсальними. Найбільш універсальною моделлю комп'ютерних обчислень є машина Тьюрінга.

6.2.2. Машини Тьюрінга та Поста

Машина Тьюрінга (МТ) — математичне поняття, введене для формального уточнення інтуїтивного поняття алгоритму. Названа на честь англійського математика Алана Тьюрінга, який і запропонував це поняття в 1936. Аналогічну конструкцію машини згодом і незалежно від Тьюрінга ввів американський математик Еміль Пост.

У кожної машини Тьюрінга є стрічка, потенційно нескінченна в обидві сторони. Є скінченна множина символів стрічки S_0, …, S_n, що називається ***алфавітом машини***. У кожний момент часу кожна комірка може бути зайнята не більше ніж одним символом. Машина має деяку скінченну множину внутрішніх станів q_0, q_1, …, q_n. У кожний даний момент часу машина знаходиться в тільки одному із цих станів.

Нарешті, є головка, яка у кожний даний момент часу знаходиться на одній із комірок стрічки. Машина діє не безупинно а лише в дискретні моменти часу. Якщо в якийсь момент t головка сприймає комірку (тобто знаходиться на комірці), що містить символ S_i, і машина знаходиться у внутрішньому стані q_j, то дія машини визначена.

Описуючи різноманітні алгоритми для машин Тьюрінга і стверджуючи реалізованість всіх можливих композицій алгоритмів, Тьюрінг переконливо показав розмаїтість можливостей запропонованої ним конструкції, що дозволило йому виступити з такою тезою: "*Всякий алгоритм може бути реалізований відповідною машиною Тьюрінга*".

Це основна гіпотеза теорії алгоритмів у формі Тьюрінга. Одночасно ця теза є формальним визначенням алгоритму. Завдяки їй можна доводити існування або неіснування алгоритмів, створюючи відповідні машини Тюрінга або доводячи неможливість їхньої побудови. Завдяки цьому з'являється загальний підхід до пошуку алгоритмічних розв'язків.

Довести тезу Тьюрінга не можна, тому що в його формулюванні не визначене поняття "всякий алгоритм", тобто ліва частина тотожності. Його можна тільки обґрунтувати, подаючи різноманітні відомі алгоритми у вигляді машинної програми Тьюрінга.

Машина Поста (МП) - абстрактна обчислювальна машина, запропонована Емілем Леоном Постом, що відрізняється від машини Тьюрінга більшою простотою.

МП складається з каретки (або головки) і розбитої на секції нескінченної в обидва боки стрічки. Кожна секція стрічки може бути або порожня - 0, або позначена міткою 1. За один крок каретка може зсунутися на одну позицію вліво або вправо, поставити або знищити символ у тому місці, де вона стоїть. Робота МП визначається програмою, що складається з кінцевого числа рядків. Усього команд шість:

N. → J зсув вправо

N. ← J зсув вліво

N. 1 J запис мітки

N. 0 J видалення мітки

N. ? J1, J0 умовний перехід по мітці

N. Stop зупинка

де N. - номер рядка, J – рядок, на який переходить керування далі.

Для роботи машини потрібно задати програму і її початковий стан (тобто стан стрічки й позицію каретки). Після запуску можливі варіанти:

- робота може закінчитися нездійсненною командою (стирання неіснуючої мітки або запис у позначене поле);
- робота може закінчитися командою Stop;
- робота ніколи не закінчиться.

Машини Тьюрінга і Поста еквівалентні з точки зору відображення поняття "алгоритм".

6.3. Проблема алгоритмічної розв'язності

Однією з головних проблем, які підштовхнули розвиток теорії алгоритмів, є з'ясовування, чи мають розв'язки задачі, для яких ще не розроблено відповідного алгоритму. Відповідь на це питання часто потребує складного математичного аналізу, оскільки звичайно розробити алгоритм розв'язання задачі (якщо він існує!) значно легше, ніж довести, що це неможливо. Це ілюструє навіть древня китайська мудрість: "Важко знайти чорного кота у темній кімнаті, особливо, якщо його там немає!"

6.3.1.Алгоритмічно нерозв'язні масові проблеми

Найпростішим прикладом алгоритмічно нерозв'язної масової проблеми є так звана проблема застосовності алгоритму (названа також проблемою зупинки). Вона полягає в тому, що потрібно знайти загальний метод, який дозволяв би для довільної машини Тьюрінга (заданої за допомогою своєї програми) і довільного початкового стану стрічки цієї машини визначити, чи завершиться робота машини за кінцеве число кроків, або ж буде тривати необмежено довго ("зависання" програми). Тьюрінг довів в 1936 році, що загальний алгоритм для розв'язання проблеми зависання для будь-яких можливих вхідних даних не може існувати. Ми можемо сказати, що проблема зависання нерозв'язна на машині Тьюрінга.

Алан Тьюрінг висловив припущення (відоме як "Теза Черча – Тьюрінга"), що будь-який алгоритм в інтуїтивному розумінні цього слова може бути поданий еквівалентною машиною Тьюрінга. Уточнення уявлення про обчислюваність на основі поняття машини Тьюрінга (і інших еквівалентних їй понять) відкрило можливості для строгого доведення алгоритмічної нерозв'язності різних масових проблем (тобто, проблем про знаходження єдиного методу розв'язання деякого класу завдань, умови яких можуть варіюватися у відомих межах).

6.3.2. Обчислювані і частково рекурсивні функції. Теза Черча

Нехай N — множина натуральних чисел і $F = \{f_1, f_2, \ldots, f_n, \ldots\}$ — сукупність n-арних функцій на множині N.

Далі часто будемо використовувати унарні функції o, s і n-арну функцію $I_m^n(x_1, \ldots, x_n)$, які визначаються таким чином:

$$o(x) = 0, \tag{6.1}$$

$$s(x) = x + 1, \tag{6.2}$$

$$I_m^n(x_1, \ldots, x_n) = x_m. \tag{6.3}$$

Ці функції називатимемо ***найпростішими функціями.*** До найпростіших відносять також і n-арну функцію $o^n(x_1, \ldots, x_n)$, яка дорівнює нулю для всіх $x_1, x_2, \ldots, x_n \in N$.

Розглянемо $n+1$ функцію із F — функцію f арності n і функції $f_1, \ldots, f_n$ однієї і тієї ж арності m.

Говорять, що m-арну часткову функцію $g(x_1, \ldots, x_m)$ одержано внаслідок ***операції суперпозиції*** або ***підстановки*** (S^{n+1}) із функцій $f^n, f_1^m, \ldots, f_n^m$, якщо для будь-яких $x_1, \ldots, x_m \in N$

$$g^m(x_1, \ldots, x_m) = f^n(f_1^m(x_1, \ldots, x_m), \ldots, f_n^m(x_1, \ldots, x_m)),$$

деякі з x_i можуть входити в f фіктивно.

Нехай задано довільні часткові функції: n-арна функція g і $n+2$-арна функція A. Говорять, що $n+1$-арна функція f одержана ***операцією примітивної рекурсії*** з функцій $g, h(R(g, h))$, якщо для всіх $x_1, \ldots, x_n \in N$ маємо

$$f(x_1,\ldots,x_n,0)=g(x_1,\ldots,x_n),$$
$$f(x_1,\ldots,x_n,y+1)=h(x_1,\ldots,x_n,y,f(x_1,\ldots,x_n,y)).$$

Нехай Z — деяка система часткових функцій. Функція f називається ***примітивно рекурсивною*** відносно системи функцій Z, якщо f можна одержати з функцій системи Z і найпростіших функцій o, s, I за допомогою скінченного числа операцій підстановки і примітивної рекурсії. Примітивно рекурсивна функція f відносно пустої системи функцій Z називається ***просто примітивно рекурсивною функцією.***

Якщо деяким елементам множини X поставлені у відповідність однозначно визначені елементи множини Y, то кажуть, що задана ***часткова функція*** з X в Y. Сукупність тих елементів множини X, у яких є відповідні елементи в Y, називається ***областю визначення функції.*** Якщо область визначення функції з X в Y збігається з множиною X, то функція називається ***всюди визначеною.***

Часткова функція f називається ***частково рекурсивною*** відносно системи часткових функцій Z, якщо f може бути одержана з функцій системи Z і найпростіших функцій o, s, I_m^n за допомогою застосування скінченого числа операцій підстановки, примітивної рекурсії і мінімізації.

Часткова рекурсивна функція f відносно пустої системи функцій Z називається ***просто частково рекурсивною функцією.***

За означенням універсальної алгебри пара:

$$G=(F,\{R,M,S^2,\ldots\}), \tag{6.4}$$

де F — множина всіх часткових функцій на N з будь-яким числом аргументів, є частковою алгеброю. Сукупність всіх частково рекурсивних функцій відносно якої-небудь системи функцій Z є підалгеброю алгебри G, породженою множиною функцій $Z\cup\{o,s,I_m^n\}, m,n=1,2,\ldots$. Те саме можна сказати і про частково рекурсивні функції.

Алгебра $G_{пр}=(F,\{R,S^2,S^3,\ldots\})$ є частковою алгеброю. Сукупність всіх часткових примітивно рекурсивних функцій відносно будь-якої системи функцій $Z=F$ є підалгеброю алгебри $G_{пр}$, породженої множиною $Z\cup\{o,s,I_m^n\}, m,n=1,2,\ldots$

Важливість розглянутих класів функцій полягає у їх зв'язку з класом обчислюваних функцій. Це формулюється у вигляді таких тез.

Теза Черча. Клас алгоритмічних або машинно-обчислюваних часткових числових функцій збігається з класом всіх частково рекурсивних функцій.

Більш загальною є теза Тьюрінга.

Теза Тьюрінга. Клас функцій, які алгоритмічно обчислюються відносно системи функцій Z, збігається з класом частково рекурсивних функцій відносно системи Z.

Оскільки поняття обчислюваної функції не має точного означення, то довести ці тези неможливо. Але завдяки їм стало можливим надати необхідну точність формулюванням алгоритмічних проблем.

6.4. Поняття про алгоритмічну алгебру

Алгебри, задані на кількох різних носіях, називають ***багатоосновними*** або ***багатосортними.***

Важливим прикладом багатоосновної алгебри є алгебра алгоритмів, яку запропонував В.М. Глушков у 1965 р..

Алгебра алгоритмів – система, яка складається з двох алгебр u та v, які називаються відповідно алгеброю операторів та алгеброю умов. Елементи алгебри u — це часткові перетворення (оператори) деякої абстрактної множини B, а елементи алгебри v — часткові предикати (умови), визначені на множині B. Алгебру алгоритмів використовують для опису перетворень, виконуваних дискретними перетворювачами. У цьому випадку множина B називається інформаційною множиною.

Основна операція алгебри u — це звичайна операція множення (суперпозиції) операторів. Крім цієї операції, для кожної умови β з v в алгебрі u визначаються ще дві операції, названі β-диз'юнкцією і β-ітерацією операторів. Результатом β-диз'юнкції $(P \vee Q)$ двох операторів P і Q є оператор R такий, що для будь-якого стану $b \in B$, $bR = bP$, якщо умова β істинна на стані b, $bR=bQ$, якщо $\beta(b)$ хибна й, нарешті, оператор R вважається невизначеним на стані b, якщо $\beta(b)$ не визначено. Результатом β-ітерації $\{P\}_\beta$ оператора P є оператор Q такий, що для будь-якого $b \in B$

має місце $bQ = bP^n$, де n — найменше із чисел m=0, 1, ... таких, що $\beta(bP^m)$ істинно $(bP^0 = b$ для будь-якого оператора $P)$.

На множині v умов визначені звичайні булеві операції $\wedge$, $\vee$, $\neg$. Наприклад, диз'юнкція $\alpha \vee \beta$ двох умов є новою умовою γ. Ця умова приймає значення «1» на тих елементах множини B, на яких одна з умов α або β приймає значення «1». Значення «0» вона приймає на тих елементах, на яких α та β рівні «0», і невизначена, якщо одна з умов α та β невизначена, а інша дорівнює «0». Крім цих операцій визначається операція $P \cdot \alpha$ множення оператора на умову. Результатом виконання цієї операції є умова β, значення якої дорівнює значенню умови α після виконання оператора P.

Правила виконання операцій у цьому випадку задаються такими співвідношеннями:

$$1 \vee \alpha = \alpha \vee 1 = 1,$$
$$0 \vee \alpha = \alpha \vee 0 = \alpha,$$
$$\alpha \vee \alpha = \alpha,$$
$$1 \wedge \alpha = \alpha \wedge 1 = \alpha,$$
$$0 \wedge \alpha = \alpha \wedge 0 = 0,$$
$$\alpha \wedge \alpha = \alpha,$$
$$\neg\alpha = \alpha.$$

Нехай $u \in$ УМ, $P \in$ ОП — довільні умова і оператор, $b \in B$. Тоді Pu є умовою u', такою, що $u'(b) = 1 \Leftrightarrow u(P(b)) = 1, u'(b) = 0 \Leftrightarrow u(P(b)) = 0$ і $u'(b) = н$, якщо $P(b)$ або $u(P(b))$ невизначене.

Якщо в алгебрі операторів і умов визначити базисні системи твірних операторів, то елементи кожної з цих алгебр можна задавати виразами, які складаються з твірних і символів операцій системи Z=(ОП, УМ). Такі вирази називаються ***регулярними,*** *а* сама система Z=(ОП, УМ) — ***алгеброю алгоритмів.***

Зокрема, алгебра алгоритмів дозволяє подавати програми у вигляді виразів алгебри алгоритмів і проводити над програмами формальні перетворення. Ці перетворення грунтуються насамперед на ***тотожних*** і ***квазітотожних*** співвідношеннях алгебри алгоритмів і деяких інших співвідношеннях.

Тотожності алгебри алгоритмів:

1. (P v Q)= (Q v P);

2. $P(Q \vee R) = (PQ \vee PR)$

3. $(QP \vee RP) = (Q \vee R)P$;

4. $((P \vee Q) \vee R) = (P \vee (Q \vee R))$;

5. $\{e\} = e$, $\{\{P\}\} = \{P\}$.

Квазітотожності алгебри алгоритмів:

6. $Pu = u \Rightarrow (PQ \vee PR) = P(Q \vee R)$

7. $Pu = u$, $PQ = QP \Rightarrow P\{Q\} = \{Q\}P$;

8. $Pu = u$, $PQ = QP$, $P^2 = P \Rightarrow \{PQ\} = \{Q\,P\} = (e \vee P)\{Q\}$.

6.5. Базові алгоритми

Переважна більшість задач і алгоритмів може бути зведена до використання комбінації невеликої кількості типових базових алгоритмів. До таких слід віднести:

- послідовне виконання;
- вибір варіанта дій за умовою;
- циклічне виконання;
- рекурсія;
- ітерація;
- вибір одного значення з множини.

Для реалізації цих прийомів у мовах програмування передбачені спеціальні оператори та засоби.

Лінійні алгоритми (послідовне виконання) використовуються, наприклад, для обчислення значення функції з необмеженою областю визначення та для інших задач із заздалегідь визначеною послідовністю розв'язання.

Алгоритми з розгалуженням (вибір варіанта дій за умовою) використовуються, наприклад, для обчислення значення функції з обмеженою областю визначення та для інших задач, при вирішенні яких виникає необхідність перевірки умов, а також у випадках вибору варіанта з декількох можливих.

Циклічні алгоритми використовуються при необхідності багаторазового виконання одних і тих же дій.

Рекурсія – це фундаментальне поняття, яке означає покрокове отримання результату, причому на наступному кроці використовується результат, отриманий на попередньому кроці.

Рекурсія може бути отримана трьома способами:

1) циклічний виклик оператора $x:=f(x)$, де змінній x у лівій частині присвоюється нове значення, отримане за допомогою деякої функції f від старого значення.

2) визначення рекурсивної функції, якщо така можливість передбачена мовою програмування.

У вузькому програмістському розумінні рекурсія – це виклик у процедурі або функції її самої. Головний момент у побудові рекурсивної програми – визначення умови припинення рекурсії. Структура рекурсивної функції має вигляд

```
Function ім'я(опис);
 Begin
   if <умова виходу з рекурсії>
      then <вихід>
      else < ... виклик функції "ім'я" ... >
 End;
```

Ітерація – це процесс поступового наближення до правильного розв'язку. Використовується найчастіше для розв'язання рівнянь і систем рівнянь.

Вибір необхідного даного, яке задовольняє певну умову, з набору даних залежить від способу організації цього набору, наприклад, *вибір найменшого даного* з масиву, *вибір з файла* рядків за заданою ознакою.

6.6. Змістовні логічні схеми алгоритмів

Деякі теоретичні проблеми (наприклад, проблема алгоритмічної розв'язності) і потреби практики (наприклад, необхідність формулювання принципів роботи пристроїв, що роблять автоматизовану обробку інформації) потребують побудови строгого визначення алгоритму. Різні

варіанти розв'язування проблеми привели до побудови так званих абстрактних алгоритмічних систем (їх називають також алгоритмічними моделями). Раніше були розглянуті лише деякі з них; їх перелік може бути проілюстрований схемою на рис. 6.2.

Для формалізованого опису алгоритмів роботи таких систем контролю та керування може використовується ***апарат змістовних логічних схем алгоритмів*** (рис.6.2). Він дозволяє розробляти алгоритми роботи систем, може суттєво зменшити об'єм текстового матеріалу і покласти початок розробці програмного забезпечення систем.

Логічні схеми алгоритмів (ЛСА), які використовують для вирішення цієї проблеми, виділяються серед способів формального опису роботи автоматичних пристроїв компактністю запису, можливістю подання алгоритму з наперед установленим ступенем деталізації, відомим зв'язком з логічними схемами програм, можливістю мінімізації обсягу деяких видів алгоритмів.

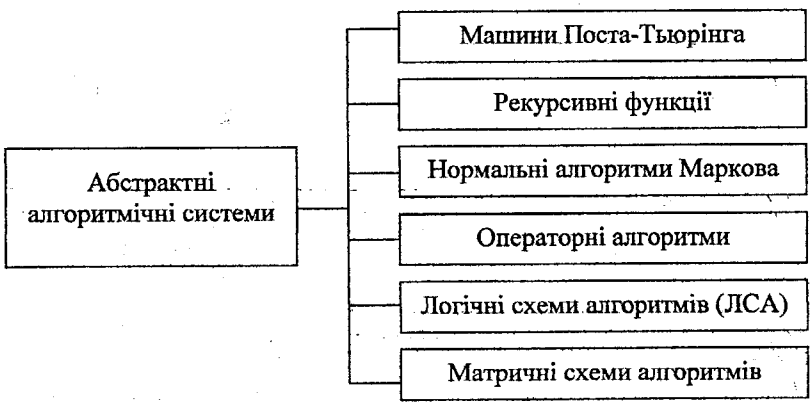


Рис. 6.2. Клас абстрактних алгоритмічних систем (алгоритмічні моделі)

Крім множин функціональних $\{A_i\}$ і логічних $\{w_i\}$ операторів, які використовуються в алгоритмічній алгебрі, в ЛСА об'єднуються оператори, які визначають обмін інформаційними і службовими сигналами між функціональними блоками системи, а також перетворення цих сигналів. У складних алгоритмах виділяються групи пов'язаних між собою операторів. Окрім літерних позначень у ЛСА використовують символи,

якими позначається порядок виконання операторів або структура апаратної частини системи. Основні елементи алгоритмічних моделей наведені в таблиці 6.2.

При розробці ЛСА передбачається, що вони повинні:

- описувати функціонування як апаратної, так і програмно-керованої частин системи;
- якщо можна наочно відображати зміст перетворень, що описуються;
- описувати не тільки інформаційні перетворення, але і службові операції;
- слугувати основою для складання програм;
- описувати функціонування систем з різною деталізацією.

До основних недоліків ЛСА можна віднести необхідність складання і постійного використання списків операторів $\{A_i\}$ і $\{w_i\}$ з розшифровкою їх змісту, відсутність ефективних методів мінімізації записів. Ці недоліки слабо проявляються при відносно простих і загострюються при складних ЛСА.

Формальна алгоритмічна модель систем контролю та керування

У формалізованому вигляді подамо означення алгоритмічної моделі за допомогою нотації Бекуса – Наура (наведені лише основні означення):

<алгоритмічна модель>::= *<ідентифікатор>* ! A(B) < *блок* > A(E)

< *блок* > ::= *<ідентифікатор>* ! < *пусто* > ! < *елемент* > ![< *блок* >< *блок* >]

< *елемент* >::= < *перетворення* > ! < *умова* > ! < *знак* >

< *перетворення* > ::=E*<номер>*(< *операція* >;< *вхідні дані*>;*<результатиц* >)

< *умова* >::= w *<номер>* (*<логічний вираз>* ; *<вхідні дані>*) *<знак переходу* >

< *знак переходу* > ::= |_< *номер* > ! _|< *номер* > ! >>

< *знак* >::= < *знак переходу* > ! << ! ||

< *операція* >::= < *пусто* > ! < *тип операції* >! < *зміст операції* >

< *тип операції* > ::= < I > ! < S > ! < C > ! <T> ! ...

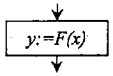
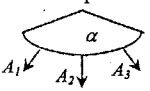
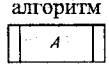
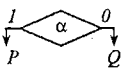
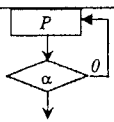
< C > ::= + ! – ! * ! div ! log ...

< *вхідні дані* >::= < *пусто* > ! (*<ім'я>*; *<тип>*!*<значення вхідного даного* >)

<результати >::= < *пусто* > ! (*<ім'я>*; *<тип>*)

Таблиця 6.2

Порівняння елементів алгоритмічних моделей

Елементи ЛСА		Операції системи алгоритмічних алгебр		Типові елементи схем програм	
Зміст	Позначення	Зміст	Позначення	Зміст	Позначення
Функціональний оператор	A_i	оператор	$\hat{u}$	Функція	$F(x)$
Логічний оператор	w_i	умова	$\hat{g}$	Логічний вираз (функція)	L
Перетворення сигналів	x/y	присвоювання		Процес визначення значення 	$y := F(x)$
Виконання будь-якого перетворення	$A_1 \mid A_2 \mid A_3$		$((Al_{\alpha_1} \vee A2_{\alpha 2}) \vee \vee A3) = N$	Вибір 	case x of 1: A_1 2: A_2 3: A_3
Об'єднання перетворення	[.], {.}	Множення (композиція)	$A \times B$ чи AB	Складний оператор	Begin End
Виконання алгоритму програмним шляхом				Окремо визначений алгоритм 	Procedure A
Перенос виконання алгоритму вперед		α-диз'юнкція $(\alpha \in \hat{g})$	$(P_\alpha \vee Q) = N$	Умовний перехід 	if α then P else Q
Перенос виконання алгоритму назад		α-ітерація $(\alpha \in \hat{g})$	$\{P_\alpha\}$		repeat P until α

Примітки: 1. Наведені лише основні означення.

2. Службові символи нотації: <, > – межі синтаксичної конструкції; ::= – "це"; ! – "або"; … - "інші".

3. |_< *номер* > - перехід вперед; _|< *номер* > - перехід назад.

4. < I > - інформаційні операції; < S > - вимірювальні операції; < C > - обчислювальні операції; <T>- операція передачі.

Перетворення алгоритмічної моделі формалізовані у вигляді алгебраїчної системи:

$$AS = (AM, OP), \tag{6.5}$$

де AM – множина алгоритмічних моделей;

OP – множина операцій над ними.

Множина операцій OP складається з двох елементів:

- ***paste(B,n1,n2)*** - вставка блока ***B*** в алгоритмічну модель між елементами з номерами ***n1*** і ***n2***;
- ***cut(n1,n2)*** - вирізання блока з алгоритмічної моделі між елементами з номерами ***n1*** і ***n2***,

а також з понять одиничної операції **1**, яка не змінює моделі, і зворотної операції op^{-1}, яка задовольняє умову

$$op^{-1}(op\,(AM)) = AM. \tag{6.6}$$

Еквівалентними перетвореннями алгоритмічної моделі будемо називати таку послідовність операцій над моделлю, яка не змінює змісту результатів роботи системи (хоча може змінювати якісні показники як результатів, так і самої системи).

Еквівалентні перетворення здійснюються на основі *властивостей алгебри AS*:

1) $paste(B,n1,n2)\ cut(n1,n2) \equiv 1$

2) $cut(n1,n2)\ paste(B,n1,n2) \equiv 1$

3) $paste(B1,n1,n2)\ paste(B2,n3,n4) \equiv paste(B2,n3,n4)\ paste(B1,n1,n2)$ якщо $(n1,n2) \cap (n3,n4) = \varnothing$

4) $cut(n1,n2)\ cut(n3,n4) \equiv cut(n3,n4)\ cut(n1,n2)$ якщо $(n1,n2) \cap (n3,n4) = \varnothing$

5) $En1(op,X,Y)\ En2(op^{-1},Y,X) \equiv 1$

При розв'язанні задачі оптимального розподілу операцій на апаратні і програмні здійснюються еквівалентні перетворення моделі, які полягають у заміні частини апаратних операцій на програмні та додаванні інтерфейсних перетворень або навпаки.

Заміна апаратних перетворень на програмні і навпаки здійснюється на підставі тверджень:

$$1.E_0(op_H, X_H, Y_H) = E_1(I_1, X_H, X_S)\, E_2(op_S, X_S, Y_S)\, E_3(I_2, Y_S, Y_H)$$

$$2.E_0(op_S, X_S, Y_S) = E_1(I_2, X_S, X_H)\, E_2(op_H, X_H, Y_H)\, E_3(I_1, Y_H, Y_S),$$

де H – позначення апаратно-орієнтованих операцій і форми даних;

S – позначення програмно-орієнтованих операцій і форми даних;

I_1 – апаратно-програмне інтерфейсне перетворення;

I_2 – програмно-апаратне інтерфейсне перетворення.

Отже, заміна апаратних перетворень [*n1 – n2*] на програмні здійснюється за допомогою операцій

$$cut(n1,n2)\ paste(E_1(I_1, X_H, X_S)E_2(op_S, X_S, Y_S)\ E_3(I_2, Y_S, Y_H), prev(n1), next(n2)) \qquad (6.7)$$

і аналогічно при зворотному перетворенні.

При розпаралелюванні виконання операцій до алгоритмічної моделі додаються операції синхронізації і збирання даних, що паралельно надходять.

Заміна послідовних операцій на паралельні здійснюється на підставі твердження

$$3.\ E_1(op_1, X1, Y)\, E_2(op_2, X2, Y) = \| [E_3(I_3, (X1,X2), X1)\, E_1(op_1, X1, Y1)]\, [E_3(I_3, (X1,X2), X2)\, E_2(op_2, X2, Y2)]\, E_4(I_4, (Y1,Y2), Y),$$

де I_3 – операція розподілення вхідних даних;

I_4 – операція об'єднання результатів.

Методика оптимізації СК на основі алгоритмічної моделі передбачає:

1) аналіз залежностей у послідовностях перетворень і виділення незалежних послідовностей;
2) розпаралелювання перетворень на основі твердження 3;
3) апаратна підтримка паралельності на основі операції (6.7);
4) оцінка витрат та вибір найкращого варіанта на основі критеріїв оптимальності.

Під критерієм оптимальності будемо розуміти вартісну функцію

$$g([P_{11}, P_{12}], [P_{21}, P_{22}]),$$

де P_{11} – множина послідовних операцій;

P_{12} – множина паралельних операцій;

P_{21} – множина апаратно реалізованих операцій;

P_{22} – множина програмно реалізованих операцій.

Вартісна функція повинна відповідати умовам:

- вартісна функція паралельно з'єднаних підсистем

$$g_{пар} = \sum_i g_{P_{12}i} + c_t \max_i (\sum_{P_{11}} T_i), \tag{6.8}$$

де $g_{P_{12}i}$ - вартість апаратних засобів i-ї підсистеми,

T_i - час виконання операції i-ю підсистемою, яка здійснює послідовність операцій P_{11},

c_t - вартість витрат часу;

- вартісна функція послідовно з'єднаних підсистем

$$g_{пос} = \sum_i g_{P_{12}i} + c_t \sum_i T_i . \tag{6.9}$$

Вираз вартісної функції отримується з алгоритмічної моделі СК.

5) зміна структури системи у відповідності до обраного варіанта.

Процедура розпаралелювання використовує ГПС – граф перетворення сигналів і ГЗ – граф залежностей. Для формалізації процедури в алгебрі AS визначаються відношення залежності ⊗:

$E_1(op_1, X1, Y1) \otimes E_2(op_2, X2, Y2)$ якщо $(X1 \otimes X2) \cup (X1 \otimes Y2) \cup (X2 \otimes Y1)$

Для відношень залежності виконується властивість транзитивності.

Аналіз залежностей може бути здійснений за допомогою динамічної структури типу двовимірний зв'язаний список, який формується за правилами:

1) елементи в алгоритмічній моделі переглядаються зліва направо;
2) якщо вхідними даними елемента алгоритмічної моделі є результати попереднього перетворення або його номер є адресою переходу умови, то елемент додається до послідовного списку, який містить попереднє перетворення або умову (рис.6.3,а);
3) якщо вхідними даними елемента алгоритмічної моделі є лише вхідні дані алгоритму, то елемент починає новий послідовний список (рис.6.3,б);
4) якщо вхідними даними елемента алгоритмічної моделі є результати перетворень або адреси переходів з декількох послідовних списків, то цей елемент є об'єднувальним для відповідних послідовних ланцюгів (рис.6.3,в).

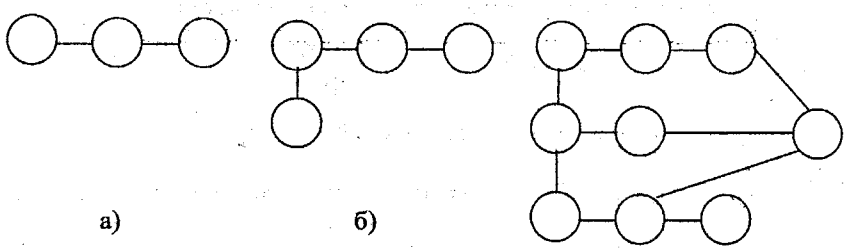


Рис.6.3. Граф перетворень і залежностей

Отриманий таким способом граф перетворень і залежностей відтворює максимально можливе розпаралелювання.

6.7. Алгоритми в умовах невизначеності

Класичне уявлення про алгоритм: це точний рецепт одержання результату, який можна реалізувати, використовуючи заздалегідь обговорений набір найпростіших дій.

Поки алгоритми використовувалися в основах математики, нічого іншого й не вимагалося. Маючи строге визначення алгоритму, математики одержали можливість відповідати на запитання: чи існує для даного завдання алгоритм його розв'язку? Майже у всіх практично цікавих випадках на це питання вдається відповісти.

Однак з розвитком галузей застосування алгоритмів з'явилися задачі, які виходять за рамки класичних постановок. Серед таких задач виділимо ті, де для розв'язання необхідно здійснити таку велику кількість дискретних кроків, що навіть сучасним комп'ютерам для цього необхідний значний час, який виходить за межі допустимого. Існують також задачі, в яких початкові дані точно не визначені, отже невизначеним буде і результат.

6.7.1. Стохастичні алгоритми

Досить ефективними є алгоритми, що відносяться до класу стохастичних алгоритмів, зокрема генетичні алгоритми.

Стохастичними методами прийнято називати методи, прямо або побічно основані на використанні генераторів псевдовипадкових послідовностей (ПВП).

Результат роботи ймовірнісного алгоритму точно не визначений. Можна лише говорити про ймовірності того або іншого результату. Алгоритм вважається нормальним, якщо ймовірність помилки не занадто велика. Такий розпливчастий опис, звичайно, має потребу в уточненні. Із практичної точки зору цікаві тільки ті алгоритми, які помиляються дуже рідко.

Методи, які використовують елементи випадковості, стали з'являтися відносно недавно. В основі першого з таких методів лежить випадковий пошук в просторі задачі зі збереженням найкращого отриманого результату.

Функціонування даних алгоритмів здійснюється таким чином. Випадковий пошук забезпечує добір вихідних точок для проведення локальної оптимізації. Отже, рішення покращуються не тільки шляхом локальної оптимізації, але й шляхом випадкового пошуку кращого з локально оптимальних рішень. Вихідну точку для реалізації локально-стохастичного пошуку доцільно підбирати за допомогою вагового алгоритму, який забезпечує "адаптацію" пошуку до параметрів задачі.

Застосування такого методу не гарантує отримання оптимального рішення. Крім того, результат роботи методу не може бути кращим, ніж при інших методах пошуку, оскільки в обох випадках розглядаються одні й ті ж дискретні точки простору пошуку задачі.

6.7.2.Нечіткі алгоритми

Нечіткий алгоритм визначається впорядкованою множиною нечітких інструкцій (нечітких висловлень), які містять поняття, що формалізуються нечіткими множинами.

Під нечіткими інструкціями розуміються інструкції, що містять нечітке поняття, наприклад, "пройти близько 100 метрів", а під машинними - інструкції, які не містять ніяких нечітких понять: "пройти 100 метрів". Тут і далі чіткі інструкції ми будемо називати машинними, щоб підкреслити можливість моделювання нечітких алгоритмів на ЕОМ, що сприймають тільки читання інструкцій.

Інструкції в нечітких алгоритмах можна розділити на три класи:

1. Назначальні речення, наприклад, x = великий.

2. Нечіткі висловлення типу "якщо A, тоді B". У таких пропозиціях або перша умова, або друга, або обидві можуть бути символами нечітких множин.

3. Безумовні активні речення, наприклад, набагато зменшити x, друк x, стоп.

Деякі із цих інструкцій є нечіткими, інші - ні.

Комбінувати назначальні і нечіткі висловлення необхідно відповідно до композиційного правила висновку. Наприклад, якщо в деякий момент виконання алгоритму ми зустрічаємо інструкції

1) x - дуже малий,

2) якщо x - малий, тоді y - великий, інакше y - невеликий,

Наведемо точне визначення нечіткого алгоритму, ввівши ряд визначень та позначень.

Замість інтервалу [0,1] , загальновизнаної множини значень функції належності, розглядається непуста ***множина W*** з відношенням часткового порядку $>$ та операціями $\otimes,\oplus$, які задовольняють властивість комутативності, асоціативності та дистрибутивності, а також вміщають нульовий та одиничний елементи: $a \oplus 0 = a$, $a \otimes 1 = a$, $a \oplus b > a$.

Розглядаються інструкції такого вигляду:

start: go to L	(інструкція початку)
L: do F; go to L1	(інструкція операції)
L: if P then go to (L_1, L_2, …, L_n)	(інструкція умови)
L: halt	(інструкція закінчення)

де $L_n \in L$ (множина символів міток інструкцій); $F \in F$ (множина символів операторів чи функцій); $P \in P$ (множина символів n-значних предикатів чи умов).

Введення поняття ***інструкції*** дозволяє визначити поняття програми. Під програмою розуміється кінцева множина інструкцій π, яка містить точно одну інструкцію початку, і ніякі інструкції з π не мають однакових міток.

Визначається поняття ***W-машини.*** W-машина є функція M, визначена на множину символів$\{I\} \cup \{F\} \cup \{P\} \cup \{O\}$, для яких існує множина входів X, множина станів пам'яті M і множина виходів Y, а також виконані умови:

1) $M(I):X\times M\to W$ (функція входів)

2) $\forall F\in \mathrm{F}\quad M(F):M\times M\to W$ (функція операцій)

3) $\forall P\in \mathrm{P}\quad n>0\quad M(P):M\times\{1,\ldots,n\}\to W$ (функція умов)

4) $M(O):X\times Y\to W$ (функція виходів)

Символи O та I позначають вхід та вихід. І, нарешті, програма π разом з W-машиною, яка допускає π (тобто, машина визначена на всіх операціях F і умовах P, які містяться в інструкціях операцій і в інструкціях умов програми π), називається ***нечіткою програмою***. Відповідно, послідовність інструкцій, які складають нечітку програму, визначає ***нечіткий алгоритм***.

Очевидною *областю впровадження* алгоритмів нечіткої логіки є всілякі експертні системи, у тому числі:

- нелінійний контроль за процесами (виробництво);
- системи, що самонавчаються (або класифікатори), дослідження ризикових і критичних ситуацій ;
- розпізнавання образів;
- фінансовий аналіз (ринки цінних паперів) ;
- дослідження даних (корпоративні сховища);
- удосконалювання стратегій керування й координації дій, наприклад, складне промислове виробництво.

Основні задачі, які розв'язують за допомогою нечітких алгоритмів.

- Алгоритми визначення складного нечіткого поняття A через більш прості поняття, які легко описати нечіткими множинами; результатом застосування таких алгоритмів до деякого елемента u області міркувань U буде ступінь приналежності u поняттю A (ступінь, з яким елемент u може характеризуватися поняттям A);
- Алгоритми породження, у результаті виконання яких породжується один з елементів нечіткої множини, що описує поняття, яке цікавить нас (наприклад, алгоритм породження зразків почерку, рецептів готування їжі, твору, музики, речень у природній мові);
- Алгоритми опису відношень між нечіткими змінними, наприклад, у вигляді послідовності нечітких інструкцій; такі алгоритми дозволяють приблизно описувати поводження систем, вхідні й вихідні сигнали яких є нечіткими підмножинами;

•Алгоритми прийняття рішення, що дозволяють приблизно описувати стратегію або найважливіше правило, наприклад, алгоритм проїзду перехрестя, що містить послідовність дій, які необхідно виконати, при цьому описи цих дій складаються з нечітких понять типу: нормальна швидкість, кілька секунд, повільно наближатися.

У багатьох випадках нечіткий алгоритм зручно подавати у вигляді орієнтованого графа. Кожній дузі ставлять у відповідність інструкцію умови або інструкцію операції. Вхідні, вихідні, внутрішні змінні в нечіткому алгоритмі подаються нечіткими множинами. Виконання алгоритму еквівалентне пошуку в графі шляхів, що зв'язують позначені вершини: початкові й кінцеві.

Контрольні питання і завдання для самостійної роботи

1. Визначте загальні риси інтуїтивного поняття алгоритму.
2. Охарактеризуйте основні напрямки теорії алгоритмів.
3. Наведіть найбільш поширені форми зображення алгоритмів.
4. Визначте загальні властивості алгоритмів.
5. Опишіть принцип роботи машини Тьюрінга та Поста.
6. Дайте означення багатоосновної алгебри, алгебри алгоритмів Глушкова.
7. Наведіть тотожності алгебри алгоритмів.
8. Визначте базові алгоритми та охарактеризуйте їх.
9. Яка функція називається примітивно рекурсивною, частково рекурсивною?
10. Сформулюйте тезу Черча, тезу Тьюрінга.

Рекомендована література

1. Аверкин А.Н. и др. Нечеткие множества в моделях управления и искусственного интеллекта / Под ред. Поспелова Д.А. – М.: Наука, 1986. –312 с.

2. Гриншпан Л.А. Методы анализа стохастических сетевых моделей вычислительных систем. - Минск: Наука и техника, 1988.
3. Дубовой В.М. Моделювання систем контролю та керування. – Вінниця: ВНТУ, 2005. – 175 с.
4. Лавров И.А., Максимова Л.Л. Задачи по теории множеств, математической логике и теории алгоритмов. - М.: Наука, 1984.–224 с.
5. Капітонова Ю.В. та ін. Основи дискретної математики. – К.: «Наукова думка», 2002.
6. Криницкий Н.А. Алгоритмы вокруг нас. – М.: Наука, 1984. – 224 с.
7. Новиков Ф.А. Дискретная математика для программистов. - СПб.: Питер, 2001.
8. Питерсон Дж. Теория сетей Петри и моделирование систем. - М.: Мир, 1984.
9. Рейуорд-Смит В.Дж. Теория формальных языков. – М.: Радио и связь, 1988.
10. Манин Ю.И. Доказуемое и недоказуемое. – М.: Советское радио, 1979. – 168 с.
11. Энциклопедия кибернетики. – К.: Главная редакция УСЭ, 1974.
12. Дубовой В.М., Никитенко О.Д. Формалізація перетворень алгоритмічних моделей систем керування в умовах невизначеності. / Вісник Хмельницького національного університету. – 2006. – №4, Ч1., Т1 (68). - С.54-57
13. Глушков В.М. Теория автоматов и формальные преобразования микропрограмм. – М.: «Кибернетика», 1965, № 5.
14. Назаров А.В., Лоскутов А.И. "Нейросетевые алгоритмы прогнозирования и оптимизации систем". - СПб.: Наука и техника, 2003. - 384 с.
15. Корнеев В.В., Гареев А.Ф., Васютин С.В., Райх В.В. "Базы данных. Интеллектуальная обработка информации". - М.:"Нолидж",2000.-352 с.
16. Свами М., Тхуласираман К. "Графы, сети и алгоритмы". - М.: Мир, 1984. - 454 с.

ПІСЛЯМОВА

Підсумовуючи наведені в цьому посібнику відомості, слід зазначити, що знання з математики, якими бажано володіти фахівцю з системної інженерії у галузі управління і автоматики, не вичерпуються загальним курсом і спеціальними розділами математики, наведеними у цьому посібнику. Розвиток науки і техніки, хоча й звільняє від рутинної роботи, ставить перед нами все складніші інтелектуальні проблеми. Для їх розв'язання створюються нові математичні методи, постійно розвивається математичний апарат.

Автори сподіваються, що посібник допоможе студентам глибше зрозуміти математичні основи відповідних спеціальностей і зорієнтуватися у нових математичних ідеях і інструментах.

Рекурсивний перебір варіантів

```
Const
 N=3;
Type
 IntSet=set of 1..N;
Var
 A:array [1..N] of integer;

Procedure Per(S:IntSet; K:integer);
 Var
  i:integer;
 Begin
  if S=[] then
   begin
    for i:=1 to N do write(A[i],' ');
    writeln
   end
  else
   begin
    for i:=1 to N do
     if i in S then
      begin
       A[K]:=i;
       Per(S-[i],K+1)
      end
   end
 End;

Begin
 Per([1..N],1)
End.
```

Генерація перестановок з мінімальною кількістю транспозицій

```
{$M 16000, 0, 600000}
```

```
const
 N=4;
var
 P:array [1..N] of integer;
 i:integer;

Function B(m,i:integer):integer;
 Begin
  if (m mod 2 = 0) and (m>2)
   then if i<(m-1) then B:=i
                   else B:=m-2
   else B:=m-1
 End;

Procedure Permin(m:integer);
 Var
  buf:integer;
  i:integer;
 Begin
  if m=1 then
    begin
     for i:=1 to n do write(P[i],' ');
     writeln
    end
   else
    for i:=1 to m do
     begin
      Permin(m-1);
      if i<m then
        begin
         buf:=P[B(m,i)];
         P[B(m,i)]:=P[m];
         P[m]:=buf
        end
     end
 End;
```

```
BEGIN
 for i:=1 to n do P[i]:=i;
 Permin(n);
 readln
END.
```

Генерація перестановок з транспозицією сусідніх елементів

```
const
 N=5;
var
 P:array [1..N] of integer;
 i:integer;

Procedure Sosed(m:integer);
 Var
  i,j,x,k,buf:integer;
  C:array [1..N] of integer;
  PR:array [1..N] of boolean;
 Begin
  for i:=1 to m do begin C[i]:=1; PR[i]:=true end;
  C[m]:=0;
  for i:=1 to m do write(P[i],' '); writeln;
  i:=1;
  while i<m do
   begin
    i:=1;
    x:=0;
    while C[i]=m-i+1 do
     begin
      PR[i]:=not PR[i];
      C[i]:=1;
      if PR[i] then x:=x+1;
      i:=i+1;
     end;
    if i<n then
```

```
    begin
     if PR[i] then k:=C[i]+x
           else k:=m-i+1-C[i]+x;
     buf:=P[k];
     P[k]:=P[k+1];
     P[k+1]:=buf;
     for j:=1 to m do write(P[j],' '); writeln;
     C[i]:=C[i]+1;
    end
  end
 End;

BEGIN
 for i:=1 to n do P[i]:=i;
 Sosed(N);
 readln
END.
```

Генерація перестановок антилексикографічним методом

```
{$M 16000, 0, 600000}
const
 N=4;
var
 P:array [1..N] of integer;
 i:integer;
Procedure Reverse(m:integer);
 Var
  buf:integer;
  i,j:integer;
 Begin
  i:=1; j:=m;
  while i<j do
   begin
    buf:=P[i]; P[i]:=P[j]; P[j]:=buf;
    inc(i); dec(j)
   end
```

```
End;

Procedure Antilex(m:integer);
 Var
  buf:integer;
  i:integer;
 Begin
  if m=1 then
   begin
    for i:=1 to n do write(P[i],' ');
    writeln
   end
  else
   for i:=1 to m do
    begin
     Antilex(m-1);
     if i<m then
      begin
       buf:=P[i];
       P[i]:=P[m];
       P[m]:=buf;
       Reverse(m-1);
      end
    end
 End;

BEGIN
 for i:=1 to n do P[i]:=i;
 Antilex(n);
 readln
END.
```

Алгоритм пошуку в глибину

У наступному прикладі показаний результат пошуку шляху від вершини 1 до вершини 5 (рис. 1).

```
Const
```

```
N=5;
MS:array[1..N,1..N]of byte=((0,1,1,0,0),
                            (1,0,1,1,1),
                            (1,1,0,0,0),
                            (0,1,0,0,1),
                            (0,1,0,1,0));
Var
 V0,VN,V:byte;
 Stek:array[1..N]of byte;
 Top:byte;
 Found:boolean;
 SetV:set of 1..N;
Begin
 SetV:=[1..N];
 writeln('Введите начало и конец пути:');
 readln(V0,VN);
 if V0=VN then Found:=true else Found:=false;
 Top:=1;
 Stek[Top]:=V0;
 SetV:=SetV-[V0];
 while (not Found) and ((Top>0) and (SetV<>[])) do
  begin
   V:=1;
   while ((not (V in SetV)) or (MS[Stek[Top],V]=0)) and (V<=N) do inc(V);
   if V<=N then begin
                  inc(Top);
                  Stek[Top]:=V;
                  if V=VN then Found:=true;
                  SetV:=SetV-[V]
                end
           else dec(Top)
  end;
 for V:=1 to Top do write(Stek[V],' ');
 writeln
End.
```

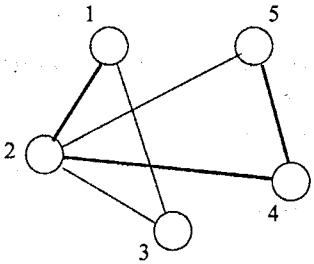


Рис. Д1. Приклад графа

Алгоритм пошуку в ширину

У наступному прикладі показаний шлях та зміст черги на шляху від 3 до 5 вершини (рис. 2).

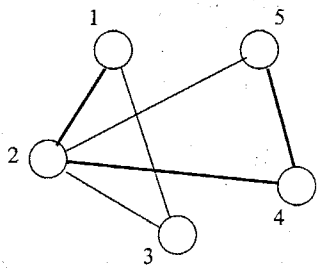


Рис. Д2. Приклад графа

```
Const
 N=5;
 MS:array[1..N,1..N]of byte=((0,1,1,0,0),
                             (1,0,1,1,1),
                             (1,1,0,0,0),
                             (0,1,0,0,1),
                             (0,1,0,1,0));
Var
 V0,VN,V:byte;
 Q:array[1..N*(N-1),1..2]of byte;
 BQ,EQ:byte;
 Found:boolean;
 SetV:set of 1..N;
 Way:array[1..N] of byte;
```

3	1	2	4	5		
0	1	1	3	3		
			↑		↑	
			BQ		EQ	

Рис. Д3. Послідовність вершин у черзі

```
Begin
SetV:=[1..N];
writeln('Введите начало и конец пути:');
readln(V0,VN);
if V0=VN then Found:=true else Found:=false;
BQ:=1;
EQ:=1;
Q[EQ,1]:=V0; Q[EQ,2]:=0;
SetV:=SetV-[V0];
while (not Found) and (SetV<>[]) do
 begin
  for V:=1 to N do
   if (V in SetV) and (MS[Q[BQ,1],V]=1) and (not Found)
       then begin
            inc(EQ);
            Q[EQ,1]:=V; Q[EQ,2]:=BQ;
            if V=VN then Found:=true;
            SetV:=SetV-[V]
           end;
```

```
  inc(BQ)
 end;
 V:=N;
 while EQ>0 do
 begin
  Way[V]:=Q[EQ,1];
  dec(V);
  EQ:=Q[EQ,2]
 end;
inc(V);
while V<=N do begin write(Way[V],' '); inc(V) end;
writeln
End.
```

Обхід графа по мінімальному шляху

Рекурсивний метод

```
Const
 N=5;
 Dist:array[1..N,1..N] of integer =((1,3,5,4,6),
                                    (3,2,5,4,7),
                                    (4,8,6,9,7),
                                    (3,4,2,7,6),
                                    (6,9,3,2,5));
Type
 IntSet = set of 1..N;
Var
 A,Amin:array [1..N] of integer;
 Lmin,j:integer;

Procedure Commi(S:IntSet; K:integer);
 Var
 L,i:integer;
Begin
 if S=[] then
  begin
   L:=0;
```

```
    for i:=1 to N-1 do L:=L+Dist[A[i],A[i+1]];
    if L<Lmin then
      begin
       Lmin:=L;
       Amin:=A
      end
   end
  else
   for i:=1 to N do
    if i in S then
     begin
      A[K]:=i;
      Commi(S-[i],K+1)
     end
End;
BEGIN
 Lmin:=Maxint;
 Commi([1..N],1);
 for j:=1 to N do writeln(Amin[j])
END.
```

Метод гілок та границь

Обхід графа по мінімальному шляху

```
Const
 N=5;
 Dist:array[1..N,1..N] of integer =((1,3,5,4,6),
                                    (3,2,5,4,7),
                                    (4,8,6,9,7),
                                    (3,4,2,7,6),
                                    (6,9,3,2,5));
Type
 IntSet = set of 1..N;
Var
 A,Amin:array [1..N] of integer;
 L,Lmin,j:integer;
```

```
Procedure Commi(S:IntSet; K:integer);
 Var
  i:integer;
 Begin
  if S=[] then
   begin
    if L<Lmin then
     begin
      Lmin:=L;
      Amin:=A
     end
   end
  else
   for i:=1 to N do
    if i in S then
     begin
      A[K]:=i;
      if K>1 then L:=L+Dist[A[K-1],A[K]];
      if L<Lmin then Commi(S-[i],K+1);
      if K>1 then L:=L-Dist[A[K-1],A[K]];
     end
End;
BEGIN
 Lmin:=Maxint;
 L:=0;
 Commi([1..N],1);
 for j:=1 to N do writeln(Amin[j])
END.
```

Алгоритм Дейкстри

У наступній програмі a – множина розглянутих вершин, b – масив відстаней від початкової вершини до кожної з решти вершин графа, c – масив вершин, через які проходять найкоротші шляхи.

```
Const
 N=8;
 M=999;
```

```
D:array [1..N,1..N] of real=((0, 23, 12, M, M, M, M, M),
                    (23, 0, 25, M, 22, M, M, 35),
                    (12, 25, 0, 18, M, M, M, M),
                    (M, M, 18, 0, M, 20, M, M),
                    (M, 22, M, M, 0, 23, 14, M),
                    (M, M, M, 20, 23, 0, 24, M),
                    (M, M, M, M, 14, 24, 0, 16),
                    (M, 35, M, M, M, M, 16, 0));
Var
 a:set of 1..N;
 c,P:array[1..N]of integer;
 b:array[1..N]of real;
 V0,j,k,all,VN:integer;
 bmin:real;

Begin
 write('Начальная вершина: '); readln(V0);
 write('Конечная вершина: ');readln(VN);
 writeln;
for j:=1 to N do
 begin
  c[j]:=V0;
  b[j]:=D[V0,j]
 end;
a:=[V0]; c[V0]:=0;
while a<>[1..N] do
begin
 bmin:=M;
 for k:=1 to N do
  if (not (k in a)) and (b[k]<bmin) then
   begin
    bmin:=b[k];
    j:=k
   end;
 a:=a+[j];
 for k:=1 to N do
```

```
  if b[k]>(b[j]+D[j,k]) then
   begin
    b[k]:=b[j]+D[j,k];
    c[k]:=j
   end;
 end;
 j:=1;
 P[j]:=VN;
 while c[P[j]]<>0 do
  begin
   inc(j);
   P[j]:=c[P[j-1]];
  end;
 for k:=j downto 2 do write(P[k],'-->'); writeln(P[1]);
 writeln('Lmin=',b[VN]:6:1);
 readln
End.
```

Послідовність кроків алгоритму для випадку: початкова вершина – 3, кінцева вершина – 8 наведена у табл. 1.

Алгоритм Пріма-Краскала з використанням множини

```
Const
 N=6;  B=1000;
 D:array [1..N,1..N] of real=((0, 1, 3, B, B, B),
                              (1, 0, B, B, B, B),
                              (1, B, 0, 1, 2, B),
                              (B, 1, 1, 0, B, 1),
                              (B, B, 5, B, 0, B),
                              (B, B, B, 1, 1, 0));
Var
 COL:set of 1..N;
 i,j,i0,j0,k:integer;
 L,Dmin:real;
 REB:array[1..N-1,1..2]of integer;
```

Таблиця Д1

Послідовність кроків алгоритму

1	a	0	0	1	0	0	0	0	0
	b	12	25	0	18	999	999	999	999
	c	3	3	0	3	3	3	3	3
2	a	1	0	1	0	0	0	0	0
	b	12	25	0	18	999	999	999	999
	c	3	3	0	3	3	3	3	3
3	a	1	0	1	1	0	0	0	0
	b	12	25	0	18	999	38	999	999
	c	3	3	0	3	3	4	3	3
4	a	1	1	1	1	0	0	0	0
	b	12	25	0	18	47	38	999	60
	c	3	3	0	3	2	4	3	2
5	a	1	1	1	1	0	1	0	0
	b	12	25	0	18	47	38	62	60
	c	3	3	0	3	2	4	6	2
6	a	1	1	1	1	1	1	0	0
	b	12	25	0	18	47	38	61	60
	c	3	3	0	3	2	4	5	2
7	a	1	1	1	1	1	1	0	1
	b	12	25	0	18	47	38	61	60
	c	3	3	0	3	2	4	5	2
8	a	1	1	1	1	1	1	1	1
	b	12	25	0	18	47	38	61	60
	c	3	3	0	3	2	4	5	2
Мінімальний шлях 3-->2-->8, Lmin= 60.0									

```
Begin
 COL:=[1..N];
  Dmin:=B;
  for i0:=1 to N-1 do
   for j0:=i0+1 to N do
    if (D[i0,j0]<Dmin) then
     begin
```

```
      Dmin:=D[i0,j0];
      i:=i0; j:=j0;
     end;
  REB[1,1]:=i; REB[1,2]:=j;
  COL:=COL-[i,j];
  L:=Dmin;
  k:=2;
 while k<=N-1 do
  begin
  Dmin:=B;
  for i0:=1 to N-1 do
   for j0:=i0+1 to N do
    if (D[i0,j0]<Dmin) and ((i0 in COL) xor (j0 in COL)) then
     begin
      Dmin:=D[i0,j0];
      i:=i0; j:=j0;
     end;
  L:=L+Dmin;
  REB[k,1]:=i; REB[k,2]:=j;
  if i in COL then COL:=COL-[i] else COL:=COL-[j];
  inc(k);
 end;
for k:=1 to N-1 do writeln(REB[k,1],' - ',REB[k,2]);
 writeln('Lmin=',L:9:1)
End.
```

Алгоритм Пріма-Краскала з використанням фарбування вершин

```
Const
 N=6;
 B=1000;
 D:array [1..N,1..N] of real=((0, 1, 3, B, B, B),
                              (1, 0, B, B, B, B),
                              (1, B, 0, 1, 2, B),
                              (B, 1, 1, 0, B, 1),
                               (B, B, 5, B, 0, B),
                               (B, B, B, 1, 1, 0));
```

```
Var
 COL:array[1..N]of integer;
 i,j,i0,j0,k:integer;
 L,Dmin:real;
 REB:array[1..N-1,1..2]of integer;
Begin
for i:=1 to N do COL[i]:=i;
k:=1;
L:=0;
while k<=N-1 do
 begin
  Dmin:=B;
  for i0:=1 to N-1 do
   for j0:=i0+1 to N do
    if (D[i0,j0]<Dmin) and (COL[i0]<>COL[j0]) then
     begin
      Dmin:=D[i0,j0];
      i:=i0; j:=j0;
     end;
  L:=L+Dmin;
  REB[k,1]:=i; REB[k,2]:=j;
  for i0:=1 to N do
   if COL[i0]=COL[j] then COL[i0]:=COL[i];
  inc(k);
 end;
for k:=1 to N-1 do writeln(REB[k,1],' - ',REB[k,2]);
 writeln('Lmin=',L:9:1)
End.
```

Алгоритм Форда-Фолкерсона

```
Const
 N=5;
 PS:array[1..N,1..N]of real= ((0,15,3,0,2),      { Матриця пропускних
здатностей }
                              (5,0,4,7,3),
                              (3,4,0,2,0),
```

```
                    (0,7,0,0,8),
                    (0,3,0,8,0));
Var
 V0,VN,V:byte;
 Q:array[1..N*(N-1),1..2]of byte; { Черга для пошуку збільшуючого ланцюга
}
 BQ,EQ:byte;
 Found:boolean;
 SetV:set of 1..N;
 Way:array[1..N] of byte;  { Збільшуючий ланцюг }
 Potok:array[1..N,1..N] of real;      { Шукана матриця потоків}
 i,j : byte;
 dPmin:real;

Procedure MinIzm;  { Знаходження величини збільшення потоку }
 var
  dP:real;
 begin
  V:=N;
  if Potok[Way[V-1],Way[V]]>=0
     then dPmin:=PS[Way[V-1],Way[V]]-Potok[Way[V-1],Way[V]]
     else dPmin:=PS[Way[V],Way[V-1]]-Potok[Way[V],Way[V-1]];
  while Way[V-1]<>V0 do
   begin
    dec(V);
    if Potok[Way[V-1],Way[V]]>=0
       then dP:=PS[Way[V-1],Way[V]]-Potok[Way[V-1],Way[V]]
       else dP:=PS[Way[V],Way[V-1]]+Potok[Way[V-1],Way[V]];
    if dP<dPmin then dPmin:=dP
   end
 end;
Procedure CorPot; { Зміна потоку у збільшуючому ланцюгу }
 begin
  V:=N;
  while Way[V]<>V0 do
   begin
```

```
    if Potok[Way[V-1],Way[V]]>=0
       then Potok[Way[V-1],Way[V]]:=Potok[Way[V-1],Way[V]]+dPmin
       else Potok[Way[V-1],Way[V]]:=Potok[Way[V-1],Way[V]]-dPmin;
    Potok[Way[V],Way[V-1]]:=-Potok[Way[V-1],Way[V]];
    dec(V);
   end
 end;
Procedure Poisk;  { Пошук збільшуючого ланцюга }
Begin
 SetV:=[1..N];
 Found:=false;
 while (not Found) and (SetV<>[]) and (BQ<=EQ) do
  begin
  inc(BQ);
  while Q[BQ,1]=VN do inc(BQ);
  if BQ<=EQ then
   begin
    j:=BQ;
    while j>0 do
     begin
      SetV:=SetV-[Q[j,1]];
      j:=Q[j,2]
     end;
    for V:=1 to N do
     if (V in SetV) and (PS[Q[BQ,1],V]<>0) and (not Found)
        then begin
             inc(EQ);
             Q[EQ,1]:=V; Q[EQ,2]:=BQ;
             if V=VN then Found:=true;
            end;
   end;
  i:=N;
  j:=EQ;
  while j>0 do
   begin
    Way[i]:=Q[j,1];
```

```
    dec(i);
    j:=Q[j,2]
    end;
  end
End;
BEGIN
 writeln('Введіть витік і стік потоку:');
 readln(V0,VN);
 BQ:=0;   EQ:=1;
 Q[EQ,1]:=V0; Q[EQ,2]:=0;
 for i:=1 to N do
   for j:=1 to N do
     Potok[i,j]:=0; {Тривіальний розв'язок}
 while BQ<=EQ do
   begin
    Poisk;
    MinIzm;
    CorPot;
   end
END.
```

Навчальне видання

Володимир Михайлович Дубовой
Олена Дмитрівна Никитенко

Спеціальні розділи математики

Навчальний посібник

Оригінал-макет підготовлено авторами

Редактор *В.О. Дружиніна*
Коректор З.В. Поліщук

Науково-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку 28.12.07 р. Гарнітура Times New Roman
Формат 29,7х42 1/4 Папір офсетний
Друк різографічний Ум. друк. арк. 10.4
Тираж 75 прим.
Зам. № 2007-185

Віддруковано в комп'ютерному інформаційно-видавничому центрі
Вінницького національного технічного університету
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ