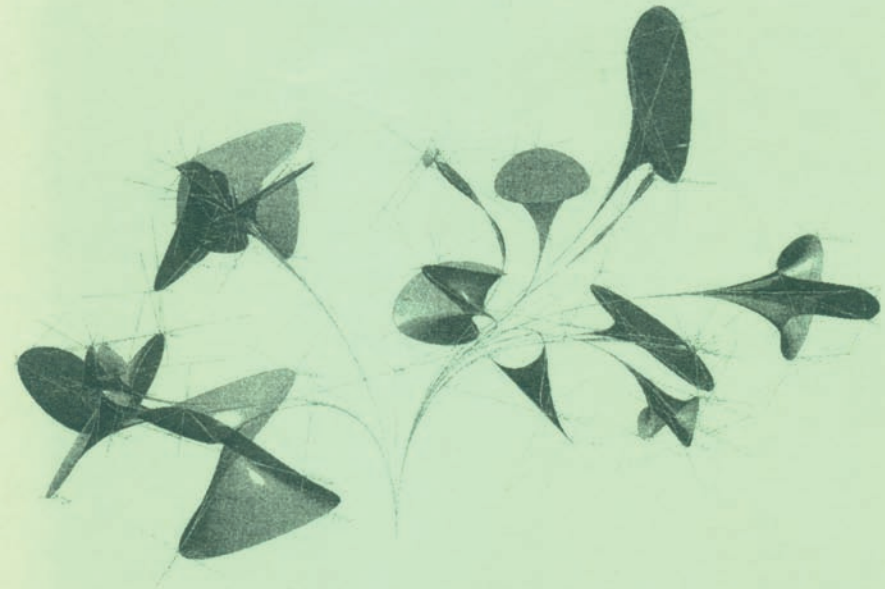
Т. Б. МАРТИНЮК, Н. І. ЗАБОЛОТНА

СИСТЕМОТЕХНІКА ОПТОЕЛЕКТРОННИХ ТА ЛАЗЕРНИХ СИСТЕМ
ЛАБОРАТОРНИЙ ПРАКТИКУМ



Міністерство освіти і науки України
Вінницький національний технічний університет

Т. Б. МАРТИНЮК, Н. І. ЗАБОЛОТНА

СИСТЕМОТЕХНІКА ОПТОЕЛЕКТРОННИХ ТА ЛАЗЕРНИХ СИСТЕМ
ЛАБОРАТОРНИЙ ПРАКТИКУМ

Затверджено Вченою радою Вінницького національного технічного університету як лабораторний практикум для студентів спеціальності "Лазерна та оптоелектронна техніка". Протокол № 10 від 27 квітня 2006р.

Вінниця ВНТУ 2008

УДК 519.17
М 29

Рецензенти:
А. М. Пєтух, доктор технічних наук, професор
Л. І. Тимченко, доктор технічних наук, професор
О. К. Колесницький, кандидат технічних наук, доцент

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

Мартинюк Т. Б.; Заболотна Н. І.
М 29 **Системотехніка оптоелектронних та лазерних систем. Лабораторний практикум.** – Вінниця: ВНТУ, 2008. – 119 с.

У лабораторному практикумі наведено теоретичні відомості з теорії графів, особлива увага приділена основам теорії марковських ланцюгів. Проілюстровано прикладні аспекти теорії графів. Показано та проаналізовано приклади оцінювання параметрів трудомісткості обчислювальних алгоритмів з використанням сітьового підходу теорії марковських ланцюгів та векторно-матричних операцій.

Лабораторний практикум розроблений у відповідності з планом кафедри та програмою дисципліни "Системотехніка оптоелектронних та лазерних систем".

УДК 519.17

Зміст

Лабораторна робота № 7
Пошук мінімального шляху

Перелік скорочень

АЛП – арифметико-логічний пристрій

ЕОМ – електронна обчислювальна машина

ЛАР – лінійне алгебраїчне рівняння

ЛТЧ – лічильник тактів

МВ – матриця вартості

ОП – обчислювальний процес

ОС – обчислювальна система

ПЗП – постійний запам'ятовувальний пристрій

ПП – переповнення

Р – регістр

СМ – суматор

ТА – трудомісткість алгоритму

ТЦ – тіло циклу

ЦОМ – цифрова обчислювальна машина

Вступ

Обов'язковим для дисципліни "Системотехніка оптоелектронних та лазерних систем" є розділ, який присвячений вивченню основ теорії обчислювальних систем, оскільки оптоелектронні системи, що широко використовуються для оброблення зображень, являють собою по суті обчислювальні системи. Це викликає необхідність детального вивчення класу моделей, які породжені специфікою і задачами теорії обчислювальних систем. Серед широковідомих моделей основоположними для даного класу систем є моделі і апарат марковських процесів, а також алгебра матриць. Особливістю використання теорії марковських ланцюгів є можливість подання обчислювального процесу в компактній формі, зручній для аналізу характеристик процесу. Разом з тим, на практиці широко відомі алгоритми розв'язання різних математичних і прикладних задач, наприклад, задач на графах, з використанням матричних моделей, оскільки найважливішою особливістю матриць є природний паралелізм виконання операцій над ними.

В даному лабораторному практикумі головною метою є ознайомлення студентів з основними положеннями теорії марковських ланцюгів та набуття ними навичок оцінювання трудомісткості та дисперсії трудомісткості алгоритмів з використанням марковських моделей обчислювальних процесів. Крім того, студенти ознайомляться з особливостями застосування алгебри матриць для задачі пошуку найкоротшого (мінімального) шляху на графах. Набуті навички оцінювання характеристик обчислювальних алгоритмів не тільки зроблять ширшою сферу застосування даного математичного апарату студентами, але також дозволять їм у подальшому зробити правильний вибір з точки зору ефективності використання отриманих знань при моделюванні обчислювальних процесів.

К.т.н., доц. Заболотна Н. І. підготувала лабораторну роботу №1 (разом з Дроненко О. В.); к.т.н., доц. Мартинюк Т. Б. підготувала лабораторну роботу №2 (разом з Дроненко О. В.), лабораторні роботи №3-6 і лабораторну роботу №7 (разом з Веленчук Л. П.). Варіанти завдань до лабораторних робіт склали Мартинюк Т. Б., Дроненко О. В., Лялюк В. Г. Комп'ютерний набір виконали Дроненко О. В., Дмитрук В. В., Просоловський Р. В., Вільонько Р. М., Персюк С. П., Чорна Л. О.

ЛАБОРАТОРНА РОБОТА №1

БАЗОВІ АРИФМЕТИЧНІ ОПЕРАЦІЇ

Мета роботи – вивчення основних понять теорії алгоритмів, ознайомлення з базовими арифметичними операціями та алгоритмами їх реалізації.

1 ТЕОРЕТИЧНА ЧАСТИНА

Багато явищ в природі описуються одними і тими ж математичними рівняннями. Отже, за допомогою одного фізичного процесу можна моделювати різні процеси, що мають один і той самий математичний опис. В наш час в математиці розроблено велику кількість методів числового розв'язання багатьох видів рівнянь, це дає можливість вирішувати різні задачі за допомогою набору простих арифметичних та логічних операцій [1].

Цифрова обчислювальна машина (ЦОМ) – це комплекс електронного обладнання, що виконує інтерпретацію програм у вигляді фізичних процесів, призначенням яких є реалізація математичних операцій над інформацією, поданою у цифровій формі. Процес розв'язання будь-якої задачі на ЦОМ перш за все має бути виражений алгоритмом.

За означенням академіка А. Н. Колмогорова, алгоритм або алгорифм – це будь-яка система обчислень, які виконуються за строго визначеними правилами, яка після певного числа кроків приводить до розв'язання поставленої задачі [1]. В інженерній практиці найчастіше використовується таке означення: алгоритм – це кінцева сукупність точно сформульованих правил розв'язання будь-якої задачі [1].

За формою подання алгоритми можуть бути словесними та математичними. Приклад словесної форми алгоритму – алгоритм Евкліда для знаходження найбільшого загального дільника двох чисел a і b.

1. Після аналізу двох чисел a і b перейти до наступного пункту.
2. Порівняти ці числа (a дорівнює b, a менше, більше b) і перейти до наступного пункту.
3. Якщо a і b рівні, то зупинити обчислення: кожне з чисел дає шуканий результат. Якщо числа не є рівними, то перейти до наступного пункту.
4. Якщо перше число менше другого, то переставити їх місцями, перейти до наступного пункту.
5. Відняти друге число від першого, аналізуючи два числа: від'ємник і остачу; перейти до пункту 2.

Прикладом математичної форми алгоритму є будь-яка математична формула для знаходження якоїсь величини. Наприклад, значення коренів

рівняння вигляду $ax^2 + bx + c = 0$ можна знайти за формулою $x_{1,2} = (-b \pm \sqrt{b^2 - 4ac})/2a$, яка являє собою алгоритм знаходження цих коренів. Однак для того, щоб реалізувати математичну форму алгоритму, потрібно дати ще ряд словесних вказівок, вказати область застосування алгоритму.

Детермінований алгоритм – алгоритм, що має місце при чіткій і ясній системі правил і вказівок та однозначних дій.

Випадковий алгоритм – алгоритм, який передбачає можливість випадкового вибору тих чи інших правил [2].

Алгоритм повинен забезпечити отримання результату через скінченну кількість кроків для будь-якої задачі певного класу. В іншому випадку задача не має розв'язку.

Чисельний алгоритм – алгоритм, що відповідає розв'язанню поставленої задачі за допомогою арифметичних дій.

Логічний алгоритм – алгоритм, який використовується у випадку, коли при розв'язанні задачі доводиться використовувати деякі логічні дії.

Зовнішньою функцією ЦОМ є виконання арифметичних операцій над числами. У зв'язку з цим однією з основних функціональних частин ЦОМ є арифметико-логічний пристрій (АЛП), який виконує математичні та логічні дії, що є необхідними для оброблення інформації. АЛП виконує такі елементарні дії:

- порівняння;
- додавання;
- віднімання;
- множення;
- ділення;
- набір логічних операцій.

2 СПОСОБИ ВИКОРИСТАННЯ БАЗОВИХ АРИФМЕТИЧНИХ ОПЕРАЦІЙ

2.1 Способи виконання операції додавання

Операція додавання алгебраїчних значень (чисел зі знаками) є основною операцією, яка часто використовується в процесі множення і ділення. Використовується два основних способи виконання операції додавання:

- додавання в оберненому коді;
- додавання у доповняльному коді.

Для контролю за переповненням розрядної сітки використовуються модифікаційні коди, в яких знак подається двома двійковими розрядами: знак плюс – 00 і знак мінус – 11.

Обернений і доповняльний код використовують для кодування від'ємних значень операндів, що вступають в операцію додавання. Обернений код формується шляхом інвертування значень цифрових розрядів. Доповняльний код можна отримати шляхом збільшення оберненого коду на одиницю.

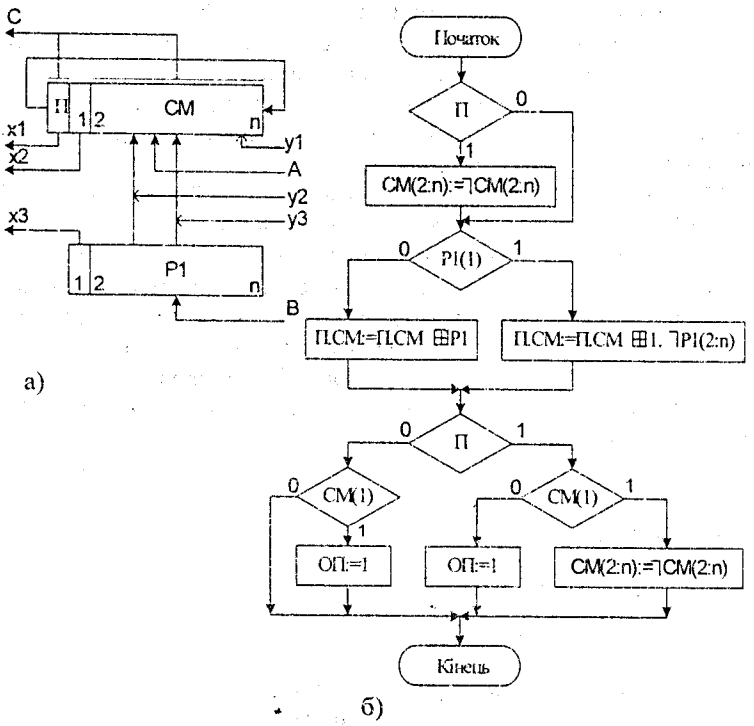


Рисунок 1 – Операційний автомат і мікропрограма додавання в оберненому модифікаційному коді

Операція додавання з використанням оберненого коду реалізується таким алгоритмом.

1. Якщо знак операнда додатний, то він вступає в операцію у прямому модифікаційному коді; якщо знак від'ємний – у модифікаційному оберненому коді.

2. Здійснюється додавання кодів операндів по всіх розрядах, включаючи знакові. Використання оберненого коду передбачає передавання перенесення зі старшого знакового розряду суми як перенесення в молодший розряд суми.

3. Якщо значення знакових розрядів суми дорівнює 00, то сума додатна і подана у прямому коді. Якщо значення знакових розрядів дорівнює 11, то сума від'ємна і подана в оберненому коді. Отримати прямий код результату можна перетворенням оберненого коду в прямий. Якщо значення знакових розрядів дорівнює 01 або 10, то сума переповнює розрядну сітку арифметичного пристрою (містить n+1 розрядів).

При використанні нагромаджувального суматора СМ додавання в оберненому коді реалізується операційним автоматом (рис.1, а), в якому виконується такий набір мікрооперацій:

у1) СМ(2:n):= ¬ СМ(2:n);
у2) П.СМ:=П.СМ ⊞ В(2:n);
у3) П.СМ:=П.СМ ⊞ 11. ¬ В(2:n);
у4) ПП:=1.

Розряд *П* є додатковим знаковим розрядом суматора. Перед початком операції додавання прямі коди n-розрядних доданків *А* та *В* заносяться на суматор *СМ* і регістр *Р1*. Додавання виконується за мікропрограмою, що наведена на рис.1, б.

2.2 Способи виконання операції множення

Множення двійкових чисел зводиться до обчислення добутку модулів співмножників і присвоювання добутку знака плюс, якщо знаки співмножників однакові, та знака мінус – якщо їх знаки є різними. Добуток $C = A \cdot B$ співмножників $A = a_1a_2...a_n$ та $B = b_1b_2...b_n$ обчислюється як сума

$$C = \sum_{i=1}^{n} Ab_i 2^{n-1}, \qquad (1)$$

де Ab_i - частковий добуток C_i, який дорівнює 0 або A, а 2^{n-1} - ваговий коефіцієнт часткового добутку. Добуток двох n-розрядних чисел є $2n$-розрядним числом. При множенні цілих чисел кома розміщується після молодшого розряду добутку, а при множенні двійкових дробів – перед старшим розрядом. Процес множення може здійснюватись двома способами: починаючи від старших розрядів або молодших розрядів множника.

Сума (1), зазвичай, обчислюється за однією з таких формул:

$$C = (...((A \cdot b_1 \cdot 2 + Ab_2) \cdot 2 + Ab_3) \cdot 2 + ... + Ab_{n-1}) \cdot 2 + Ab_n, \qquad (2)$$

$$C = \left(\left(... \left(\left(\frac{1}{2} A \cdot b_n + Ab_{n-1} \right) \cdot \frac{1}{2} + Ab_{n-2} \right) \cdot \frac{1}{2} + ... + Ab_2 \right) \cdot \frac{1}{2} + Ab_n \right) \cdot 2^{2n}. \quad (3)$$

Множення на 2 та $\frac{1}{2}$ реалізується шляхом зсуву множника на один розряд вліво і вправо відповідно. Множення на 2^{2n} відповідає перенесенню коми на $2n$ розрядів вправо і є суто умовною дією. Формула (2) описує процес множення, починаючи зі старших розрядів множника, а формула (3) – з молодших розрядів. Формули (2) та (3) породжують два варіанти схем множення, поданих на рис.2.

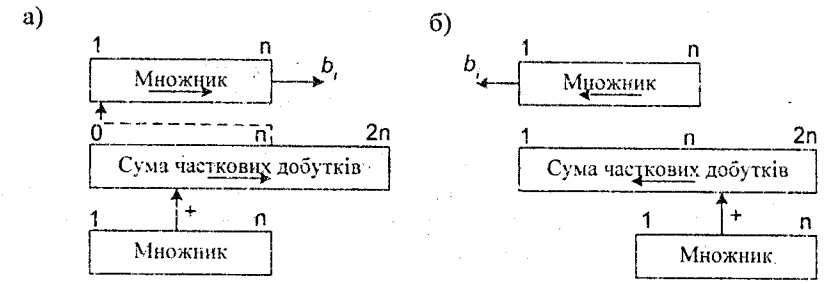


Рисунок 2 – Варіанти виконання операції множення

В першому варіанті (рис.2, а) множення починається з молодших розрядів множника. Наступна цифра множника формується шляхом зсуву множника на один розряд вправо. Якщо цифра множника має значення 1, то сума часткових добутків збільшується на значення множника. При нульовому значенні цифри множника підсумовування не виконується. Після кожного множення на один розряд множника сума часткових добутків зсувається на один розряд вправо. Множник зберігає постійне положення.

В другому варіанті (рис.2, б) множення починається зі старших розрядів множника. Наступна цифра множника формується шляхом зсуву множника на один розряд вліво. Якщо цифра множника має значення 1, то здійснюється додавання суми часткових добутків і значення множника. Після кожного множення на один розряд множника сума часткових добутків зсувається на один розряд вліво.

При множенні, починаючи від молодших розрядів, додавання відбувається лише на $(n+1)$ розрядах суматора і для виконання додавання потрібно менше часу у порівнянні з другим варіантом множення. Тому для множення доцільно використовувати схему рис.2, а.

Алгоритм обчислення добутку шляхом множення, починаючи від молодших розрядів множника здійснюється операційним автоматом (рис.3, а) та мікропрограмою, алгоритм якої поданий на рис.3, б. В операційному автоматі реалізовується такий набір мікрооперацій (додаток А):

y1) $B := CM(n+1).R1(B)$;
y2) $CM := 0$;
y3) $CM := R1(CM)$;
y4) $ЛЧТ := ЛЧТ+1$;
y5) $ЛЧТ := ЛЧТ-1$;
y8) $ЛЧТ := n$.

та набір логічних умов:

x1) $B(n)$;
x2) $ЛЧТ=0$.

Перед початком операції множник $A(n)$ заноситься на регістр A і множник $B(n)$ на регістр B.

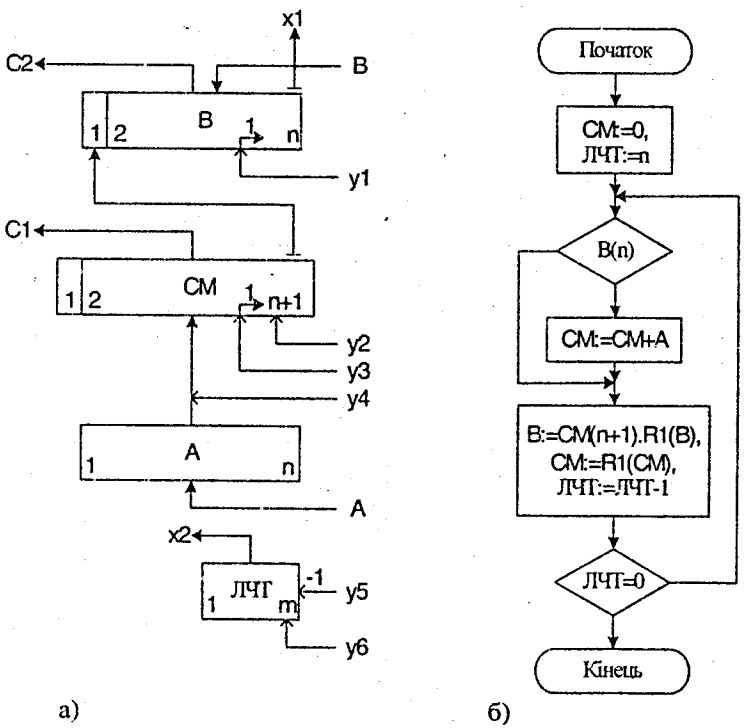


Рисунок 3 – Операційний автомат і мікропрограма множення

Операція множення відноситься до класу "довгих" операцій, що виконуються за велику кількість тактів. З метою прискорення процесу множення використовуються алгоритмічні методи, що базуються на формуванні часткових добутків $C_i = A(b_{i+1} \cdot b_{i+2} ... b_{i+k})$, які отримуються множенням множника на групу, що складається з k сусідніх розрядів множника (k=2, 3,...). При k=2 часткові добутки обчислюються таким чином. Пара розрядів $b_{i+1}b_{i+2}$ множника може мати значення 00, 01, 10, та 11. Першим трьом наборам відповідають часткові добутки 0, A та $2A$, останній з яких являє собою значення A, зсунуте на 1 розряд вліво. Набір 11 можна розглядати у вигляді 11=(100-1) і обробляти його в такому порядку. Значення 100 може бути враховане шляхом корегування множника $b_1b_2...b_i := b_1b_2...b_{i+1}$, що зводиться до збільшення значення множника на одиницю i-го розряду. В такому випадку частковий добуток, що відповідає набору 11, буде дорівнювати ($-A$). Таким чином, в результаті корегування n-розрядний множник може бути перетворений в $(n+1)$-розрядне число, тобто кількість часткових добутків, що формуються і додаються, буде дорівнювати $M \geq (n+1)/2$.

Корегування множника шляхом додавання 1 в розряд, що йде перед парою цифр, що обробляються $b_{i+1}b_{i+2}$, легко реалізується в операційному автоматі із загальними операціями. При використанні автоматів з закріпленими мікрооперацїями виконання корегування потребує введення у регістр множника мікрооперації лічби, тобто призводить до значного збільшення обладнання. Більш економічним є запам'ятовування корегуючого значення d, що дорівнює 0 або 1. При обробці першої пари цифр $b_{n-1}b_n$ маємо d=0. Наступні пари цифр перетворюються з урахуванням раніш сформованого значення d і при цьому обчислюється нове значення d. Порядок формування часткових добутків при вказаному порядку корегування множника наведений у табл.1. Позначимо логічну умову, що породжує значення часткових добутків A, $2A$, $-A$ через f_1, f_2 та f_3 відповідно і цифри множника $b_{i+1}b_{i+2}$ через β_1, β_2. З табл. 1 випливає, що логічні умови є такими функціями змінних d, β_1 та β_2:

$$\left.\begin{aligned} f_1 &= \overline{d}\,\overline{\beta_1}\beta_2 \vee d\overline{\beta_1}\,\overline{\beta_2} = \overline{\beta_1} \cdot \left(\overline{d}\beta_2 \vee d\overline{\beta_2}\right); \\ f_2 &= \overline{d}\beta_1\beta_2 \vee d\beta_1\overline{\beta_2}; \\ f_3 &= \overline{d}\beta_1\beta_2 \vee d\beta_1\beta_2 = \beta_1 \cdot \left(\overline{d}\beta_2 \vee d\overline{\beta_2}\right) \end{aligned}\right\} \qquad (4)$$

При прискореному множенні виникає необхідність у відніманні значення множника від суми часткових добутків. Віднімання доцільно виконувати у доповняльному коді, оскільки використання оберненого коду призводить до збільшення розрядної довжини суматора, яке не є бажаним.

З табл. 1 видно, що значення d, що відмічає необхідність корегування множника, завжди збігається зі знаком суми часткових добутків.

Алгоритм прискореного множення реалізується операційним автоматом (рис.4, а). Передбачено, що n – парне. При непарному n довжина регістра B повинна бути збільшена на один розряд. Операційний автомат реалізує такий набір мікрооперацій (додаток А):

y1) $B := CM(n+2 : n+3).R2(B)$;
y2) $CM := 0$;
y3) $CM := CM(1).CM(1).R2(CM)$;
y4) $CM := CM + A$;
y5) $CM := CM + A.0$;
y6) $CM := CM + 111.\neg A.+1$;
y7) $ЛЧТ := ЛЧТ - 1$;
y8) $ЛЧТ := M$.

Таблиця 1 – Значення часткових добутків при множенні на дві цифри

Корегувальні значення d	Цифри множника β_1, β_2	Значення часткового добутку	Нове значення d
0	00	0	0
	01	A	0
	10	$2A$	0
	11	$-A$	1
1	00	0	0
	01	A	0
	10	$2A$	1
	11	$-A$	1

При виконанні мікрооперації *y1* прямий код двох молодших розрядів добутку вводиться в регістр множника. В мікрооперації зсуву *y3* в розряди суматора *CM*, що звільняються засилається значення знакового розряду, що забезпечує коректність зсуву як прямого, так і доповняльного коду суми. Мікрооперація *y8* встановлює на лічильнику тактів *ЛЧТ* значення параметра $M \geq n/2$, де M – найближче ціле. В операційному автоматі формуються повідомлюючі сигнали *F(3)*, що подають значення логічних умов f_1, f_2 та f_3, що визначаються на основі (4), та сигнали *x1*) *CM*(1); *x2*) *ЛЧТ*=0. Операційний автомат функціонує у відповідності з мікропрограмою рис.4, б. Після обчислення $2n$-розрядного добутку аналізується його знак і, якщо добуток від'ємний, він корегується додаванням множника.

2.3 Способи виконання операції ділення

Ділення двійкових чисел зводиться до обчислення частки модулів діленого та дільника і присвоювання частці знака плюс, якщо знаки

операндів однакові, і знака мінус – в іншому випадку. Ділення дробових та цілих двійкових чисел здійснюється на базі різних алгоритмів.

Ділення чисел з фіксованою комою (правильних двійкових дробів) найбільш часто проводиться таким чином. Для ділення використовуються регістр частки, діленого (остачі) та дільника. Цифри частки визначаються послідовно, починаючи з цифри старшого розряду, і заносяться на регістри частки. Після формування поточної цифри частки здійснюється подвоєння остачі зсувом її на один розряд вліво. Процес ділення триває до формування заданої кількості цифр частки, що забезпечує необхідну точність результату. При діленні n-розрядних чисел з фіксованою комою, зазвичай, формується $(n+1)$ цифр частки і результат округляється до n розрядів за значенням $(n+1)$-го розряду.

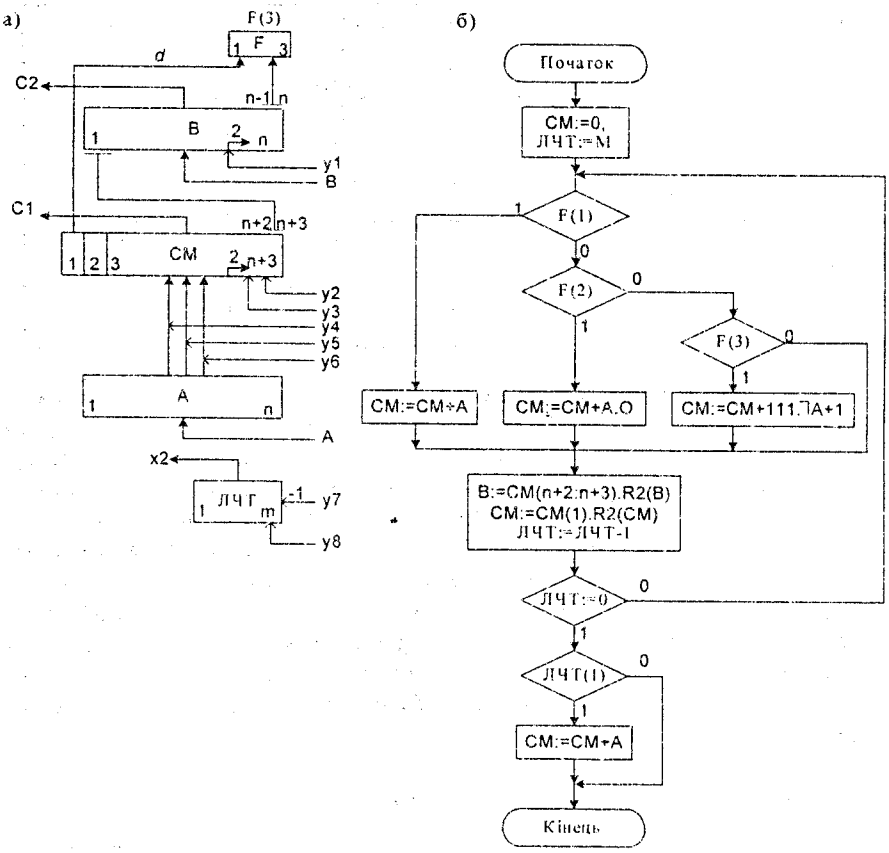


Рисунок 4 – Операційний автомат і мікропрограма прискореного множення

При діленні можливе переповнення розрядної сітки арифметичного пристрою. Переповнення виникає, якщо частка за модулем більше одиниці, тобто якщо дільник менше або дорівнює діленому.

Для ділення чисел з фіксованою комою використовується алгоритм ділення без виправляючих додавань.

1. Від діленого віднімається дільник; якщо результат невід'ємний, то частка переповняє розрядну сітку пристрою.
2. Остача подвоюється зсувом ліворуч.
3. Якщо остача додатна, то дільник віднімається від остачі; якщо остача від'ємна, то дільник додається до остачі.
4. Якщо отримана остача додатна, то цифрі часткового добутку присвоюється значення 1; якщо остача від'ємна, то цифрі часткового добутку присвоюється значення 0.
5. Дії 2, 3, 4 повторюються до формування всіх наступних цифр частки.

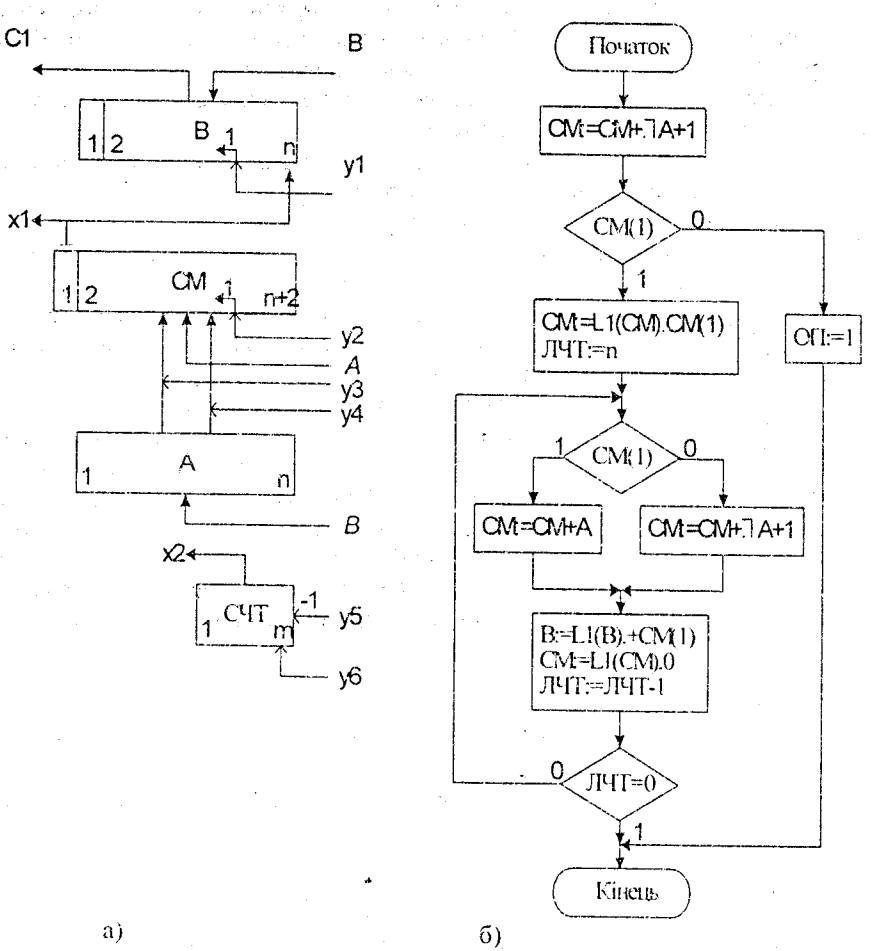
Операційний автомат, що реалізує даний алгоритм ділення, поданий на рис. 5, а. Регістр *B* призначений для зберігання частки, модулі дільника та діленого заносяться на суматор *СМ* та регістр *A*, відповідно. Остача після подвоєння її зсувом ліворуч може містити $(n+1)$ цифровий розряд. Для подання знака остачі використовується знаковий розряд. Таким чином, суматор повинен мати довжину $(n+2)$ двійкових розрядів. Операційний автомат виконує такий набір мікрооперацій (додаток А):

y1) *B*:=L1(*B*).¬ *CM*(1);
y2) *CM*:=L1(*CM*).0;
y3) *CM*:=*CM*+11. ¬ *A*+1;
y4) *CM*:=*CM*+*A*;
y5) *ЛЧТ*:=*ЛЧТ*-1;
y6) *ЛЧТ*:=n.

Для керування порядком виконання операції використовуються такі логічні умови:

x1) *CM*(1);
x2) *ЛЧТ*=0.

В процесі виконання мікрооперації *y1* в молодший розряд регістра *B* вводиться поточна цифра частки, значення якої протилежне знаку остачі. Мікрооопераційя *y2* використовується для подвоєння остачі. Віднімання від остачі дільника замінюється додаванням остачі і доповняльного коду дільника, який формується мікрооперацією *y3*. Передбачається, що на момент початку операції модулі операндів завантажені в суматор *СМ* та регістр *A*. Операція починається з пробного віднімання. Якщо знак остачі додатний, формується ознака переповнення *ПП* і ділення не виконується. В іншому випадку починається послідовне формування *n* цифр частки. Момент закінчення формування частки відмічається переходом лічильника тактів *ЛЧТ* у нульовий стан. Значення частки визначено на шині *C*.



а) б)

Рисунок 5 – Операційний автомат і мікропрограма ділення чисел з фіксованою комою

Для ділення цілих двійкових чисел можна використовувати такий алгоритм.

1. Якщо дільник дорівнює 0, то ділення неможливе.
2. Дільник нормалізується; кількість зсувів, що виконуються при цьому, збільшується на одиницю і визначає кількість p молодших цифр частки, які можливо не дорівнюють нулю; інші $(n\text{-}p)$ старших цифр частки дорівнюють нулю.
3. Послідовно, починаючи зі старших, визначаються значення p цифр частки шляхом виконання кроків 3, 4, 5 раніше наведеного алгоритму ділення.

Даний алгоритм реалізується операційним автоматом (рис.6, а), який функціонує у відповідності з мікропрограмою рис. 6, б. Операційний автомат реалізує такі мікрооперації (додаток А):

y1) *B*:=0;
y2) *B*:=L1(*B*).¬ *CM*(1);
y3) *CM*:=L1(*CM*).0;
y4) *CM*:=*CM*+1.¬ *A*+1;
y5) *CM*:=*CM*+*A*;
y6) *A*:= L1(*A*).1;
y7) *ЛЧТ*:=*ЛЧТ*+1;
y8) *ЛЧТ*:=*ЛЧТ*-1;
y9)*ЛЧТ*:=1.

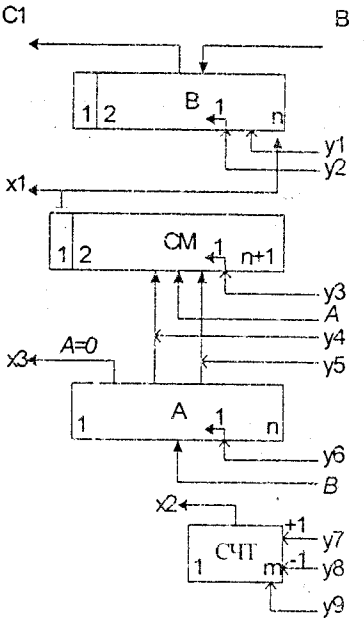
Умовні сигнали подають значення таких логічних умов:

x1) *CM*(1); *x2*)*A* = 0; *x2*) *ЛЧТ*=0.

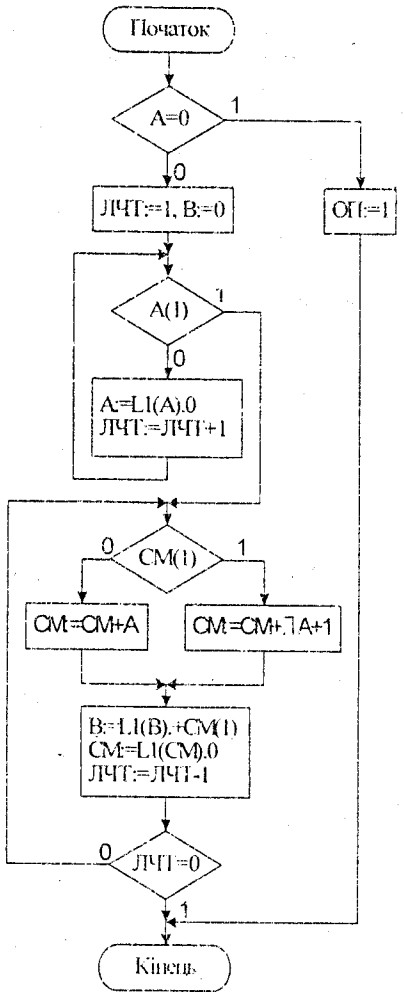
Операція ділення відноситься до найбільш довготривалих арифметичних операцій: для визначення кожної цифри частки виконується додавання і зсув остачі. Час ділення можна зменшити, якщо шляхом аналізу значень остачі і дільника визначати декілька сусідніх нулів або одиниць частки, виконуючи лише додавання чи віднімання.

Контрольні питання

1. Дати означення алгоритму.
2. Вказати форми подання алгоритму, навести приклад.
3. Що таке ЦОМ? Які основні операції АЛП ви знаєте?
4. Написати подання чисел A = - 0,101010 та B = 0,100010 в прямому, оберненому та доповняльному кодах.
5. Які існують методи здійснення операції множення, в чому полягає їх основний принцип?
6. Дати коротку характеристику методу прискореного множення.
7. Які основні блоки використовуються в операційному автоматі, призначеному для ділення чисел?
8. Охарактеризувати мікропрограму ділення цілих чисел.



а)



б)

Рисунок 6 – Операційний автомат і мікропрограма ділення цілих чисел

Зміст завдань

1. Індивідуальні завдання подані у вигляді таблиці (додаток Б), які містять інформацію про розрядність числа (якщо число задається у формі з фіксованою комою) та розрядність мантиси і порядку (якщо число задається у формі з плаваючою комою). Також у варіанті вказується операція і в стовпці приміток можливі коментарі, що стосуються уточнення операції.
2. Необхідно скласти мікропрограму операції і навести відповідний операційний автомат згідно з індивідуальним завданням.

Порядок виконання роботи

1. Ознайомитись з теоретичною частиною роботи.
2. Виконати індивідуальне завдання згідно зі своїм варіантом (додаток Б).
3. Скласти мікропрограму операції і навести відповідний операційний автомат для реалізації заданої двійкової арифметичної операції.

Оформлення звіту

Звіт про виконання даної лабораторної роботи повинен містити.

1. Початкові дані у вигляді таблиці на зразок індивідуальних завдань.
2. Мікропрограму виконання заданої операції.
3. Операційний автомат для реалізації заданої арифметичної операції.
4. Перелік і коментар до задіяних мікрооперацій і логічних умов.
5. Висновки.

ЛАБОРАТОРНА РОБОТА №2

ОСНОВИ ТЕОРІЇ ГРАФІВ АЛГОРИТМІВ

Мета роботи – вивчення основних понять теорії графів алгоритмів, ознайомлення зі складанням графів обчислювальних процесів.

1 ЕЛЕМЕНТИ ТЕОРІЇ ГРАФІВ АЛГОРИТМІВ

До моделей обчислювальних процесів висувають такі вимоги [2, 3].

1. Модель забезпечує можливість визначати порядок породження алгоритмом запитів на кожен з видів обслуговування – обчислення та введення-виведення інформації, що зберігається у кожному з файлів.

2. Модель забезпечує можливість визначати трудомісткість обслуговування запитів – кількість операцій, яку повинен виконати процесор при обслуговуванні запиту на обчислення, та кількість символів інформації, що вводиться та виводиться.

3. Модель повинна відображати обчислювальні процеси як реалізацію випадкового процесу, тобто породжувати запити у випадкові моменти часу та характеризувати трудомісткість запитів випадковими величинами.

4. Обчислювальні процеси, породжувані моделлю, повинні відповідати реальним процесам з точністю до збігу, у будь-якому випадку, математичних сподівань їх однойменних характеристик.

Оператори алгоритму будемо підрозділяти на функціональні, переходу та введення-виведення. Функціональний оператор задає перетворення на множині даних, тобто задає деяку сукупність обчислювальних операцій. Оператор переходу задає порядок обчислення значень предикатів і правило вибору одного з можливих шляхів розвитку обчислювального процесу, який відповідає поточним значенням даних, відношення між якими подаються предикатами. Оператор введення-виведення задає звернення до визначеного файла з метою передавання деякої кількості інформації. Функціональні оператори та оператори переходу задають сукупності обчислювальних операцій над даними та відносяться до одного класу операторів, що називаються основними операторами [3].

Сукупність операторів та зв'язків між ними найбільш наочно подається графом алгоритму, який будується як композиція вершин, що відповідають операторам алгоритму, та дуг, що відображають зв'язки між операторами. Виділяють вершини: початкові, кінцеві, операторні. Початкова вершина не має жодного входу та має тільки один вихід. Така вершина визначає початок алгоритму. Кінцева вершина має не менше одного входу та жодного виходу; вона визначає кінець алгоритму.

Операторна вершина відповідає основному операторові або операторові введення-виведення. Вершина, яка є функціональним оператором або оператором введення-виведення, може мати будь-яку, але не менше одиниці кількість входів та тільки один вихід. Вершина, що є оператором переходу, може мати будь-яку, але не менше одиниці кількість входів та не менше двох виходів. В будь-якій ситуації оператор переходу визначає один і тільки один вихід з вершини, що йому відповідає [4].

Граф алгоритму є коректним, якщо виконуються такі умови:

- існує тільки одна початкова та тільки одна кінцева вершина;
- для кожної вершини, окрім початкової, існує у будь-якому випадку один шлях, що веде у цю вершину з початкової;
- з кожної вершини, окрім кінцевої, існує у будь-якому випадку один шлях, що веде з цієї вершини у кінцеву;
- вихід з будь-якої вершини повинен вести тільки до однієї вершини графу;
- при будь-яких значеннях логічних умов існує шлях з початкової вершини у кінцеву, причому будь-якому фіксованому набору значень умов відповідає тільки один такий шлях.

Приклад графу алгоритму наведений на рис.1.

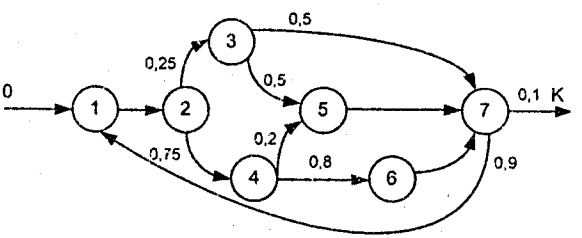


Рисунок 1 – Граф алгоритму

Умовимося вершини графу позначати номерами 0, 1,..., K, причому 0 відповідає початковій вершині графу; K – кінцевій вершині; 1,..., K-1 ідентифікують оператори алгоритму. В програмуванні графи алгоритмів зображають з використанням набору фігур, що позначають тип оператора та називаються схемами алгоритмів (програм) [4].

Граф являє собою структуру алгоритму, визначаючи множину операторів $V = \{V_1, ..., V_{K-1}\}$ та дуг $D = \{(i, j)\}$, $i = \overline{0, K-1}$ та $j = \overline{1, K}$, що пов'язують оператори.

Позначимо основні оператори як $S_O = \left\{ V_{\alpha_1}, ..., V_{\alpha_{m0}} \right\}$, $V_{\alpha_k} \in V$, оператори введення – $S_h = \left\{ V_{\beta_1}, ..., V_{\beta_{mh}} \right\}$, $h = \overline{1,H}$. Переходи між операторами V_i та V_j слід розглядати як випадкові події та характеризувати їх імовірностями p_{ij}, тобто кожна дуга *(i, j)* графу алгоритму має бути відзначена імовірністю переходу p_{ij}, з якою перехід з вершини V_i виконується саме по цій дузі, тобто до вершини V_j. Оскільки обчислювальний процес не може зупинитись у вершині V_j, то з імовірністю 1 відбудеться перехід до будь-якої вершини графу алгоритму. Враховуючи це, імовірності переходів повинні задовольняти умову:

$$\sum_{i=0}^{K-1} p_{ij} = 1, j = \overline{1,K}. \qquad (1)$$

Значення p_{ij} визначаються імовірностями значень предикатів, що залежать від розподілу значень даних, відношення між якими задаються предикатами. Іншими словами, імовірності p_{ij} залежать від імовірностей виконання умови, що перевіряється оператором V_i з метою вибору шляху переходу. Наприклад, нехай оператор 2 (рис. 1) породжує перехід до оператора 3 при від'ємному значенні деякої змінної X та до оператора 4 при додатному значенні X. Якщо відомо, що величина X рівномірно розподілена у діапазоні (- 1, +3), то з імовірністю 0, 25 її знак від'ємний та з імовірністю 0, 75 – додатний. Внаслідок цього перехід до оператора 3 відбувається з імовірністю $p_{2,3}$=0,25 та перехід до оператора 4 – з імовірністю $p_{2,4}$ = 0,75. Нехай далі оператор 7, що замикає цикл, породжує перехід до оператора 1 у дев'яти випадках, а в одному випадку перехід відбувається у кінець алгоритму (цикл, що починається від оператора 1 та закінчується оператором 7, виконується 10 разів). Тоді імовірності переходів $p_{7,1}$=0,9 та $p_{7,K}$=0,1. Якщо за оператором V_i обов'язково виконується оператор V_j,то p_{ij} = 1.

2 МАРКОВСЬКА МОДЕЛЬ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Найпростішу модель можна одержати, якщо прийняти припущення про відсутність післядії в обчислювальному процесі, коли наступний стан обчислювального процесу залежить тільки від поточного стану і не залежить від попередніх станів. В цьому випадку обчислювальний процес стає марковським процесом, що визначається множиною притаманних йому станів $\{S_0, ..., S_{H+1}\}$, матрицею імовірностей переходів вигляду

$$\mathbf{P} = [p_{ij}] = \begin{array}{c|cccc} & S_0 & S_1 & \dots & S_{H+1} \\ S_0 & p_{0,0} & p_{0,1} & \dots & p_{0,H+1} \\ S_1 & p_{1,0} & p_{1,1} & \dots & p_{1,H+1} \\ \dots & \dots & \dots & \dots & \dots \\ S_{H+1} & p_{H+1,1} & p_{H+1,2} & \dots & p_{H+1,H+1} \end{array} \quad (2)$$

та розподілом ймовірностей

$$(a_0, \dots, a_{H+1}) \quad (3)$$

станів $S_0, \dots, S_{H+1}$ у момент часу 0. Елементи p_{ij} матриці **P** визначають імовірності переходу процесу зі станів S_i у стани S_j, тобто імовірності того, що процес, який знаходиться у стані S_i, у наступний момент часу буде знаходиться у стані S_j. Матриця **P** -- стохастична матриця, суми елементів рядків якої дорівнюють одиниці, тобто

$$\sum_{j=1}^{H} p_{ij} = 1, \quad i = \overline{0, H+1}. \quad (4)$$

Імовірності a_i (3), де $i = \overline{0, H+1}$, визначають перший можливий стан S_{i0} процесу [4].

Будемо вважати, що обчислювальний процес розвивається таким чином.

Процес починається зі стану S_0, тобто програма починає виконуватись з етапу обчислення. Етап введення-виведення може бути ініційований тільки процесором, тобто може слідувати тільки за етапом обчислення. Це одночасно означає, що після кожного етапу введення - виведення відбувається етап обчислення. В цьому випадку імовірності початкових станів мають вигляд

$$(a_0, a_1, a_0, \dots, a_{H+1}) = (1, 0, 0, \dots, 0), \quad (5)$$

а матриця імовірностей переходів

$$P = [p_{ij}] = \begin{array}{c|cccccc} & S_0 & S_1 & S_2 & \dots & S_H & S_{H+1} \\ \hline S_0 & 0 & p_{0,1} & p_{0,2} & \dots & p_{0,H} & p_{0,H+1} \\ S_1 & 1 & 0 & 0 & \dots & 0 & 0 \\ S_2 & 1 & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ S_H & 1 & 0 & 0 & \dots & 0 & 0 \\ S_{H+1} & 0 & 0 & 0 & \dots & 0 & 1 \end{array} . \quad (6)$$

Зі стану обчислення S_0 процес з відповідною імовірністю може перейти у стани S_1, ... S_H, у стани, що подають звернення до файлів F_1, ..., F_H, або у поглинаючий стан S_{H+1}. Зі станів S_1, ...S_H процес з імовірністю 1 повертається у стан обчислення S_0. Досягнувши поглинаючого стану S_{H+1}, процес з імовірністю 1 назавжди залишається в ньому. Порядок зміни станів наочно подається графом марковського ланцюга (рис.2). Переходи між станами S_0, ... S_{H+1} подаються на графі дугами, на яких позначені імовірності переходів [4].

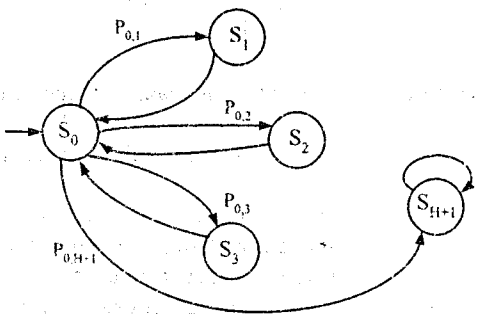


Рисунок 2 – Граф алгоритму з поглинаючим станом

Таким чином, під моделлю обчислювального процесу будемо розуміти марковський ланцюг з (H+2) станами, початковим станом S_0 та матрицею імовірностей переходів (2). Реалізація обчислювального процесу – випадкова послідовність станів $S_0, S_{t1}, S_{t2}, ..., S_{tM}$, порядок зміни яких визначається у стохастичному сенсі матрицею імовірностей переходів [4].

2.1 Приклад розробки графу алгоритму

Задано алгоритм додавання в оберненому коді (рис.3) (варіант №30).

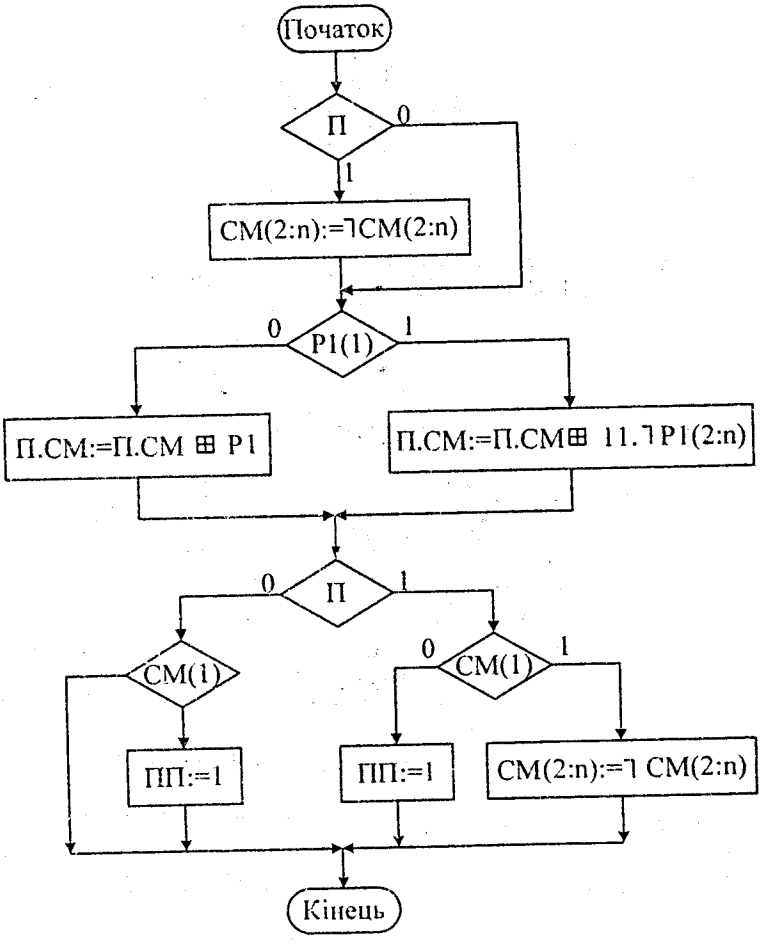


Рисунок 3 – Алгоритм додавання в оберненому коді

Проставимо номери операторних вершин алгоритму від 0 до К, починаючи зверху і зліва направо (рис. 4).

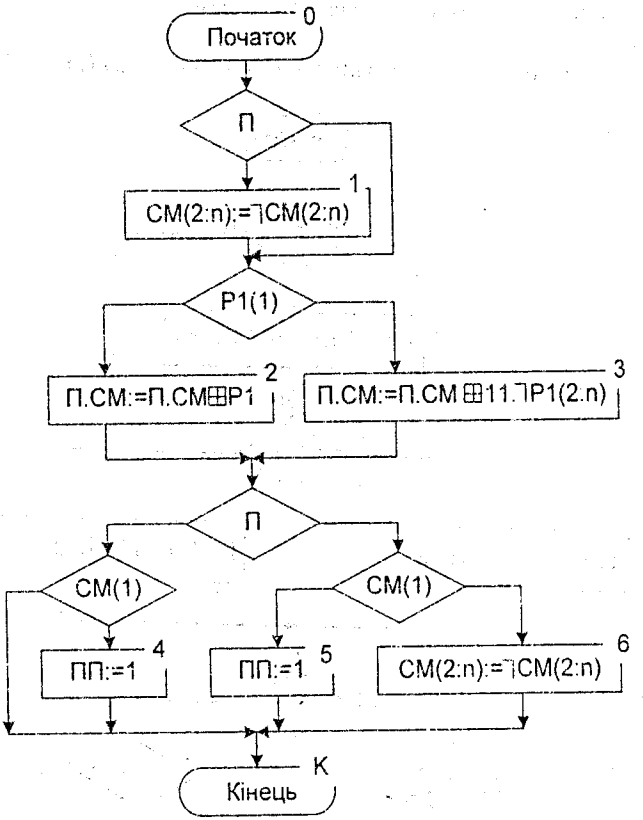


Рисунок 4 – Алгоритм з пронумерованими операторними вершинами

На основі заданого алгоритму розробимо граф (рис. 5).

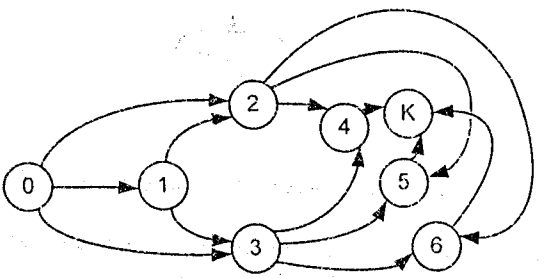


Рисунок 5 – Граф алгоритму

Згідно з індивідуальним завданням проставимо на графі ймовірності переходів (рис. 6).

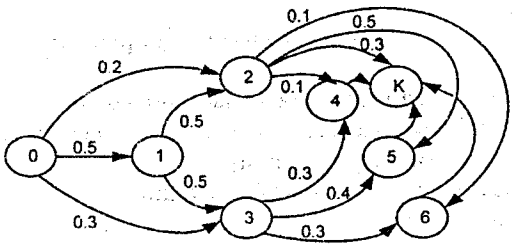


Рисунок 6 – Граф з ймовірностями переходів

Отже, отримано граф, який містить 6 операторних вершин, початкову і кінцеву вершини.

Контрольні питання

1. Які вимоги висувають до моделей обчислювальних процесів?
2. Дайте означення функціонального оператора переходу та оператора введення-виведення.
3. Які оператори складають клас основних операторів?
4. Дайте означення графу алгоритму.
5. Які вершини містить граф? Дайте їх означення.
6. Перерахуйте умови, що визначають коректність графу алгоритму.
7. Напишіть формулу, що визначає імовірності переходів у графі алгоритму.
8. Чим визначається марковська модель обчислювального процесу?
9. Яким чином розвивається обчислювальний процес марковської моделі?
10. Що являє собою матриця імовірностей переходів для марковського процесу?
11. Що таке стохастична матриця?

Зміст завдань

1. Індивідуальні завдання містять операційний автомат та мікропрограму у модифікаційному коді (додаток Б), а також ймовірності переходів вершин графу (додаток В) .

2. Необхідно розробити граф на основі заданих параметрів з врахуванням всіх операторних вершин мікропрограми та проставити ймовірності переходів.

Порядок виконання роботи

1. Ознайомитись з теоретичною частиною роботи.
2. Виконати індивідуальне завдання згідно зі своїм варіантом.
   2.1. Розробити граф алгоритму.
   2.2. Проставити ймовірності переходів.
3. Зробити висновки.

Оформлення звіту

Звіт про виконання даної лабораторної роботи повинен містити.

1. Початкові дані у вигляді таблиці на зразок індивідуальних завдань.
2. Мікропрограму виконання заданої операції.
3. Операційний автомат для реалізації заданої арифметичної операції.
4. Розроблений граф на основі заданих параметрів з проставленими ймовірностями переходів.
5. Висновки.

ЛАБОРАТОРНА РОБОТА №3

ТРУДОМІСТКІСТЬ АЛГОРИТМУ СІТЬОВИЙ ПІДХІД

Мета роботи – вивчення основних понять теорії графів алгоритмів, ознайомлення з сітьовим підходом у визначенні трудомісткості алгоритмів.

1 СЕРЕДНЯ ТРУДОМІСТКІСТЬ АЛГОРИТМІВ

Трудомісткість алгоритму (ТА) - кількість обчислювальної роботи, що необхідна для реалізації алгоритму. ТА інакше називають складністю обчислень [3]. Оцінюється трудомісткість алгоритму кількістю операцій, що виконуються з метою обробки, введення та виведення інформації у процесі розв'язку задачі. Кожній реалізації алгоритму притаманний елемент випадковості, пов'язаний з тим, що початкові дані являють собою у загальному випадку випадкову вибірку з множини початкових даних, до якої застосовується алгоритм. Тому повна характеристика трудомісткості передбачає опис кількості операцій, що виконуються за одну реалізацію алгоритму над випадковими величинами.

1.1 Алгоритми без циклів

Алгоритм будемо подавати у вигляді графу, що складається з K+1 операторних вершин та має єдину кінцеву вершину з номером K; дуги графу відзначені ймовірностями переходів p_{ij}. Кожній вершині графу поставлено у відповідність середнє значення трудомісткості k_j або l_j.

Нехай $n_1, \ldots n_{K-1}$ – середня кількість звернень до операторів $V_1, \ldots, V_{K-1}$ за один прогін алгоритму. В такому випадку характеристики трудомісткості можуть бути обчислені таким чином: середня кількість операцій, що виконуються при одному прогоні алгоритму,

$$\theta = \sum_{V_j \in S_0} n_j k_j , \qquad (1)$$

середня кількість звернень до файла F_h

$$N_h = \sum_{V_j \in S_h} n_j , (h = \overline{1, H}) \qquad (2)$$

середня кількість інформації, що передається при одному зверненні до файла F_h,

$$\theta_h = \left(1/N_h \right) \sum_{V_j \in S_h} n_j l_j, (h = \overline{1, H}). \quad (3)$$

У співвідношеннях (1) – (3) підсумовування виконується по всіх вершинах, що відносяться до класу основних операторів S_0 або класу операторів введення-виведення S_h, що звертаються до файла F_h.

Таким чином, для оцінювання ТА необхідно визначити середню кількість звернень n_1, ..., n_{K-1} до операторів 1,..., K-1, відповідно.

Сітьовий підхід використовується у тих випадках, коли кількість обчислень при розрахунку трудомісткості алгоритмів треба скоротити. Окрім того, цей метод дозволяє визначити середню, мінімальну та максимальну трудомісткість.

Суть сітьового підходу полягає у виділенні шляхів на графі алгоритму, які відповідають мінімальній, середній та максимальній трудомісткості послідовності операторів. Ці шляхи можуть бути виділені тільки на графах, що не містять циклів. З цього приводу методика сітьового підходу починається з аналізу ТА, що не містять цикли. Потім розглядається спосіб виключення циклів з графу алгоритму шляхом заміни їх операторами з еквівалентною трудомісткістю. Цей спосіб дозволяє поширити сітьовий підхід на будь-які алгоритми, у тому числі такі, що містять будь-яку кількість циклів.

З урахуванням виразів (1) – (3) задача оцінювання ТА зводиться до визначення середньої кількості n_1, ..., n_{K-1} звернень до операторів за один прогін алгоритму за умови, що трудомісткість всіх операторів k_i однакова та дорівнює 1.

Розглянемо методику обчислень значень n_1, ..., n_{K-1} для алгоритму, що не містить цикли. Для застосування сітьового підходу до оцінювання трудомісткості цього алгоритму вершини графу мають бути пронумеровані за порядком їх слідування, тобто так, щоб будь-яка вершина мала номер, більший ніж будь-який попередній. Нумерація проводиться таким чином. Початковій вершині присвоюється номер 0. Черговий номер j=1, 2 ... присвоюється вершині, в яку входять дуги від вже пронумерованих вершин з номерами меншими j. При цьому будь-яким двом вершинам повинні відповідати різні номери. Такий порядок нумерації є результативним для будь-якого графу без циклів. Причому кінцева вершина графу буде мати максимальний номер K. Приклад коректної нумерації вершин графу наведений на рис. 1.

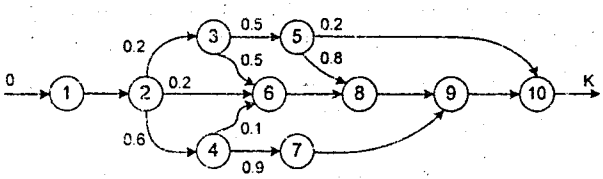


Рисунок 1 – Приклад графу алгоритму без циклів

Оскільки граф не містить циклів, то при прогоні алгоритму вершина 1 буде виконана точно один раз, тобто $n_1=1$. Середня кількість попадань обчислювального процесу у вершину визначається виразом

$$n_j = \sum_{i=1}^{K-1} p_{ij} n_i,\ j = \overline{2,K}, \qquad (4)$$

де p_{ij} – імовірність переходу з вершини i у вершину j.

При встановленому порядку нумерації вершин на момент обчислення n_i значення n_1, ..., n_{i-1} вже визначені. Тому обчислення значення n_j зводиться до підсумовування добутків, причому оскільки $p_{ij}=0$ для всіх $i \geq j$, то підсумовування слід проводити тільки для $i < j$.

1.2 Приклад оцінювання середньої трудомісткості алгоритму без циклів

Визначимо середню кількість звернень n_1, ..., n_K до операторів алгоритму, що зображений на рис. 1.

Застосувавши (4), отримаємо значення n_j, які зведені у табл.1.

Таблиця 1 - Розрахунок середньої трудомісткості алгоритму без циклів

$n_1 = 1$	$n_6 = p_{3,6}n_3 + p_{2,6}n_2 + p_{4,6}n_4 = 0,36$
$n_2 = p_{1,2}n_1 = 1$	$n_7 = p_{4,7}n_4 = 0,54$
$n_3 = p_{2,3}n_2 = 0,2$	$n_8 = p_{5,8}n_5 + p_{6,8}n_6 = 0,44$
$n_4 = p_{2,4}n_2 = 0,6$	$n_9 = p_{8,9}n_8 + p_{7,9}n_7 = 0,98$
$n_5 = p_{3,5}n_3 = 0,1$	$n_{10} = p_{5,10}n_5 + p_{9,10}n_9 = 1$

Отже, ТА (рис.1) складає

$$\theta = \sum_{j=1}^{K} n_j = 6,22 \text{ операцій.}$$

1.3 Алгоритми, що містять цикли

Розглянемо випадок алгоритму, що містить цикли (рис. 2). Безпосереднє застосування описаної методики до таких алгоритмів

неможливе, тому для обчислення значень $n_1, ..., n_{K-1}$ необхідно виключити цикли, замінюючи їх операторами з еквівалентною трудомісткістю.

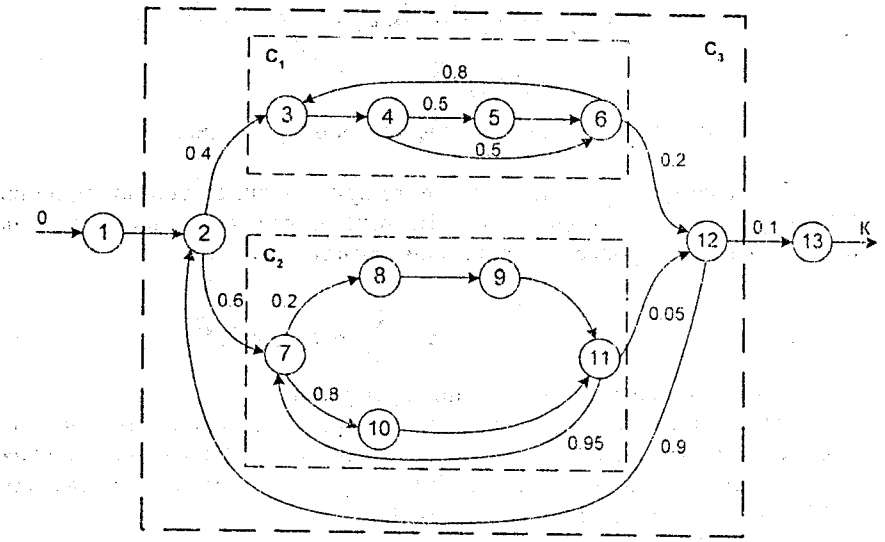


Рисунок 2 – Приклад графу алгоритму, що містить цикли

Для спрощення опису методу приймемо, що алгоритм складається з однотипних операторів, наприклад, тільки основних операторів. Розділимо цикли по рангах. До рангу 1 (r = 1) відносяться цикли, які не містять усередині себе жодного циклу, до рангу 2 (r = 2) – цикли, усередині яких є цикли рангу не вище 1, і т.ін. Наприклад, алгоритм, зображений на рис. 2, містить два цикли C_1 та C_2 рангу 1 та один цикл C_3 рангу 2. Сукупність операторів, що входить у цикл, та дуг, що пов'язують їх, за винятком дуги, що замикає цикл, називають тілом циклу (ТЦ). ТЦ рангу 1 (r = 1) є графом без циклів. Застосовуючи до цього графу вищеописану методику, можна визначити значення n_j для кожного з операторів, приналежних тілу циклу. Отже, ТЦ циклу С дорівнює

$$k_c^r = \sum_{V_j \in C} k_j n_j. \qquad (5)$$

Тут підсумовування проводиться по всіх вершинах V_j, що містяться у циклі С.

Нехай відома середня кількість повторень циклу n_C, що дорівнює кількості виконань тіла циклу при одному прогоні алгоритму. Якщо імовірність переходу по дузі, що замикає цикл, дорівнює p_{kl}, то

$$n_C = 1 + p_{kl} n_C, \tag{6}$$

звідки

$$n_C = 1/(1 - p_{kl}). \tag{7}$$

В цьому випадку середня трудомісткість циклу дорівнює

$$k_C = n_C \sum_{V_j \in C} k_j n_j = n_C \cdot k_c^T, \tag{8}$$

а цикл С можна замінити оператором з трудомісткістю k_C.

Застосовуючи вказану процедуру заміни циклів операторами до всіх циклів рангу 1, потім до циклів рангу 2 і т.д., врешті-решт прийдемо до графу без циклів, трудомісткість якого знаходиться вищеописаним способом.

1.4 Приклад оцінювання середньої трудомісткості алгоритму, що містить цикли

Визначимо трудомісткість алгоритму, заданого графом, зображеним на рис. 2.

Припустимо, що трудомісткість всіх операторів однакова та дорівнює 1, тобто $k_j = 1$. Середня кількість повторень циклів C_1, C_2 та C_3 визначається з імовірностей переходів, вказаних на рис. 2 за формулою (7) такими значеннями: $n_{C1} = 5$, $n_{C2} = 20$ та $n_{C3} = 10$.

Виділимо ТЦ C_1 та C_2 рангу 1. Їх структура зображена на рис. 3,а,б. Застосовуючи вищеописану методику до цих графів, визначаємо середню трудомісткість k_{C1} та k_{C2} виконання ТЦ для цих циклів (табл. 2).

Таблиця 2 – Розрахунок трудомісткості тіл циклів C_1 та C_2

Цикл C_1	*Цикл C_2*
$n_3 = 1$	$n_7 = 1$
$n_4 = p_{3,4} n_3 = 1$	$n_8 = p_{7,8} n_7 = 0{,}2$
$n_5 = p_{4,5} n_4 = 0{,}5$	$n_9 = p_{8,9} n_8 = 0{,}2$
$n_6 = p_{4,6} n_4 + p_{5,6} n_5 = 1$	$n_{10} = p_{7,10} n_7 = 0{,}8$
	$n_{11} = p_{9,11} n_9 + p_{10,11} n_{10} = 1$

В результаті:

$$k_{c1}^{T} = \sum_{j=3}^{6} k_j n_j = 3{,}5; \qquad k_{c2}^{T} = \sum_{j=7}^{11} k_j n_j = 3{,}2.$$

Середня трудомісткість циклів C_1 та C_2 обчислюється множенням отриманих значень ТЦ на середню кількість повторень цих циклів за виразом (8). Отже, $k_{C1} = 17{,}5$ та $k_{C2} = 64$ операцій.

Замінюючи у графі (див. рис. 2) цикли C_1 та C_2 операторами C_1 та C_2 з трудомісткістю k_{C1} та k_{C2}, отримаємо граф, вигляд якого показаний на рис. 3, в. ТЦ C_3 має вигляд, зображений на рис. 3, г.

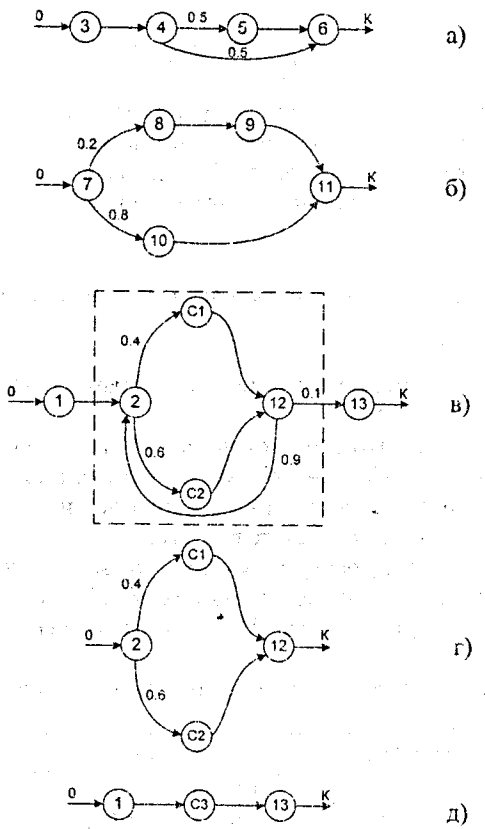


Рисунок 3 – Поетапне перетворення графу алгоритму, що містить цикли

Трудомісткість тіла циклу C_3 визначається таким чином (табл. 3).

Таблиця 3 – Розрахунок трудомісткості тіла циклу C_3

$n_2 = 1$	$n_{C2} = p_{2,C2}n_2 = 0,6$
$n_{C1} = p_{2,C1}n_2 = 0,4$	$n_{12} = p_{C1,12}n_{C1} + p_{C2,12}n_{C2} = 1$

В результаті маємо

$$k_{c3}^{T} = \sum_{V_j \in C3}^{0} k_j n_j = 1 + 17,5 \cdot 0,4 + 64 \cdot 0,6 + 1 = 47,4.$$

З урахуванням кількості n_{C3} повторень циклу C_3 трудомісткість циклу складе k_{C3} = 474 операції. Замінивши цикл C_3 оператором C_3 з трудомісткістю k_{C3}, отримаємо граф, що має вигляд, показаний на рис. 3, д. ТА, що подається цим графом, складає $\theta = k_1 + k_{C3} + k_{13} = 1 + 474 + 1 = 476$ операцій.

2 МІНІМАЛЬНА ТА МАКСИМАЛЬНА ТРУДОМІСТКІСТЬ АЛГОРИТМІВ

2.1 Алгоритми без циклів

Розглянемо методику обчислення мінімальної та максимальної ТА. Спочатку будемо розглядати алгоритми без циклів, вершини графів яких нумерують за порядком їх слідування, як це було описано раніше.

Мінімально можливе та максимально можливе значення трудомісткості на момент закінчення виконання оператора V_j позначимо відповідно через A_j та B_j. Маємо A_0=0 та B_0=0. Тоді для інших вершин з номерами $j = \overline{1, K-1}$ характерні співвідношення

$$A_j = \min_{(i,j) \in D} (A_i) + k_{j\min}, \tag{9}$$

$$B_j = \max_{(i,j) \in D} (B_i) + k_{j\max}, \tag{10}$$

де (i, j) – дуга, що виходить з вершини і та входить у вершину j; D – множина дуг графу програми; мінімальне min (A_i) та максимальне max (B_i) значення визначаються відносно всіх вершин і, з яких виходять дуги, що входять у вершину j; значення k_{jmin} та k_{jmax} характеризують мінімальну та максимальну трудомісткість оператора V_j.

Для кінцевої вершини К графу обчислюються значення

$$A_K = \min_{(i,K) \in D} (A_i), \tag{11}$$

$$B_K = \max_{(i,K)\in D}(B_i), \qquad (12)$$

що характеризують мінімальну та максимальну ТА.

2.2 Приклад оцінювання мінімальної та максимальної трудомісткості алгоритму без циклів

Визначимо мінімальну та максимальну трудомісткість алгоритму, що зображений на рис. 1.

Приймемо, що трудомісткість кожного оператора постійна та дорівнює 1. Нумерація вершин на графі (рис. 1) відповідає розглянутому вище правилу.

Послідовно застосовуючи співвідношення (9), (10), отримаємо такі значення (табл. 4).

Таблиця 4 – Розрахунок мінімальної та максимальної трудомісткості алгоритму без циклів

$A_0 = 0$	$B_0 = 0$
$A_1 = \min(A_0) + 1 = 1$	$B_1 = \max(B_0) + 1 = 1$
$A_2 = \min(A_1) + 1 = 2$	$B_2 = \max(B_1) + 1 = 2$
$A_3 = \min(A_2) + 1 = 3$	$B_3 = \max(B_2) + 1 = 3$
$A_4 = \min(A_2) + 1 = 3$	$B_4 = \max(B_2) + 1 = 3$
$A_5 = \min(A_3) + 1 = 4$	$B_5 = \max(B_3) + 1 = 4$
$A_6 = \min(A_2, A_3, A_4) + 1 = 3$	$B_6 = \max(B_2, B_3, B_4) + 1 = 4$
$A_7 = \min(A_4) + 1 = 4$	$B_7 = \max(B_4) + 1 = 4$
$A_8 = \min(A_5, A_6) + 1 = 4$	$B_8 = \max(B_5, B_6) + 1 = 5$
$A_9 = \min(A_7, A_8) + 1 = 5$	$B_9 = \max(B_7, B_8) + 1 = 6$
$A_{10} = \min(A_5, A_9) + 1 = 5$	$B_{10} = \max(B_5, B_9) + 1 = 7$
$A_K = \min(A_{10}) = 5$	$B_K = \max(B_{10}) = 7$

Таким чином, мінімальна трудомісткість алгоритму A_K дорівнює 5, а максимальна B_K – 7 операціям.

2.3 Алгоритми, що містять цикли

Мінімальна та максимальна трудомісткість алгоритмів, що містять цикли, знаходиться за аналогією з методом визначення середньої трудомісткості алгоритмів з циклами. При цьому виділяють цикли рангу 1. Знаходять мінімальну A та максимальну B трудомісткість тіла циклу. Мінімальна та максимальна трудомісткість циклу визначається значеннями

An_{min} та Bn_{max}, де n_{min} та n_{max} – мінімальна та максимальна кількість повторень циклу. Потім цикл замінюється оператором з трудомісткістю

$$k_{min} = An_{min}, \quad (13)$$

$$k_{max} = Bn_{max}, \quad (14)$$

та знову застосовується процедура виключення циклів. Процес повторюють до тих пір, доки граф алгоритму не буде перетворений до графу без циклів.

2.4 Приклад оцінювання мінімальної та максимальної трудомісткості алгоритму, що містить цикли

Визначимо мінімальну та максимальну трудомісткість алгоритму, граф якого зображений на рис. 2. Трудомісткість всіх операторів постійна та дорівнює 1 ($k_j = 1$). Мінімальна та максимальна кількість повторень циклів C_1 та C_2 подаються такими парами значень: $n_1 = (3;7)$; $n_2 = (15; 25)$. Кількість повторень циклу C_3 постійна та дорівнює 10.

Виділяємо тіла циклів C_1 та C_2, структура яких подана графами на рис. 3а,б. Мінімальна та максимальна трудомісткість цих частин алгоритму визначається так:

а) для графу на рис. 3, а (табл. 5);

б) для графу на рис. 3, б (табл. 6).

Трудомісткість циклів C_1 та C_2 подається мінімальними та максимальними значеннями (табл. 7).

Замінивши у графі на рис. 2 цикли C_1 та C_2 еквівалентними їм операторами C_1 та C_2, отримаємо граф (рис. 3, в). Тіло циклу C_3 має структуру, зображену на рис. 3, г. Мінімальна та максимальна трудомісткість тіла циклу подана у табл. 8.

Таблиця 5 – Розрахунок трудомісткості алгоритму для графу (рис. 3, а)

$A_0 = 0$	$B_0 = 0$
$A_3 = \min(A_0) + 1 = 1$	$B_3 = \max(B_0) + 1 = 1$
$A_4 = \min(A_3) + 1 = 2$	$B_4 = \max(B_3) + 1 = 2$
$A_5 = \min(A_4) + 1 = 3$	$B_5 = \max(B_4) + 1 = 3$
$A_6 = \min(A_4, A_5) + 1 = 3$	$B_6 = \max(B_4, B_5) + 1 = 4$
$A_{C1} = 3$	$B_{C1} = 4$

Таблиця 6 – Розрахунок трудомісткості алгоритму для графу (рис.3,б)

$A_0 = 0$	$B_0 = 0$
$A_7 = \min(A_0) + 1 = 1$	$B_7 = \max(B_0) + 1 = 1$
$A_8 = \min(A_7) + 1 = 2$	$B_8 = \max(B_7) + 1 = 2$
$A_9 = \min(A_8) + 1 = 3$	$B_9 = \max(B_8) + 1 = 3$
$A_{10} = \min(A_7) + 1 = 2$	$B_{10} = \max(B_7) + 1 = 2$
$A_{11} = \min(A_9, A_{10}) + 1 = 3$	$B_{11} = (B_9, B_{10}) + 1 = 4$
$A_{C2} = 3$	$B_{C2} = 4$

Таблиця 7 – Розрахунок мінімальної та максимальної трудомісткості циклів C_1 та C_2

$k_{C1\,min} = A_{C1}n_{1\,min} = 9;$	$k_{C2\,min} = A_{C2}n_{2\,min} = 45;$
$k_{C1\,max} = B_{C1}n_{1max} = 28;$	$K_{C2\,max} = B_{C2}n_{2max} = 100.$

Таблиця 8 – Розрахунок мінімальної та максимальної трудомісткості циклу C_3

$A_0 = 0$	$B_0 = 0$
$A_2 = \min(A0) + 1 = 1$	$B_2 = \max(B_0) + 1 = 1$
$A_{C1} = \min(A2) + k_{C1min} = 10$	$B_{C1} = \max(B_2) + k_{C1max} = 29$
$A_{C2} = \min(A2) + k_{C2min} = 46$	$B_{C2} = \max(B_2) + k_{C2max} = 101$
$A_{12} = \min(A_{C1}, A_{C2}) + 1 = 11$	$B_{12} = \max(B_{C1}, B_{C2}) + 1 = 102$
$A_{C3} = 11$	$B_{C3} = 102$

Замінивши цикл C_3 еквівалентним оператором C_3, одержуємо граф (рис. 3, д), мінімальна та максимальна трудомісткість якого складає

$A_K = k_{1min} + k_{C3\,min} + k_{13\,min} = 112$ операцій,
$B_K = k_{1max} + k_{C3\,max} + k_{13\,max} = 1022$операцій.

Середня трудомісткість цього алгоритму, підрахована раніше (п.1.4), складала 476 операцій.

Контрольні питання

1. Як визначається середня кількість операцій при одному прогоні алгоритму?
2. Як визначається середня кількість інформації, яка передається при одному зверненні до файла?
3. Напишіть формулу для визначення середньої кількості звернень n_i до i-ї вершини графу алгоритму.
4. Дайте означення рангів 1, 2, ... циклів алгоритму.
5. Дайте означення тіла циклу.
6. Як визначається середня кількість повторень циклів?
7. Напишіть формулу для визначення середньої трудомісткості циклу.
8. Напишіть формулу для визначення мінімально можливого та максимально можливого значення трудомісткості виконання j-го оператора.
9. Назвіть вимоги, які ставляться до обчислювальних процесів.
10. Дайте означення трудомісткості алгоритму.
11. Поясніть методику визначення трудомісткості алгоритму, що містить цикли.

Зміст завдань

1. Лабораторна робота складається з двох частин: перша присвячена оцінюванню середньої трудомісткості алгоритмів, друга – оцінюванню мінімальної та максимальної трудомісткості алгоритмів.
2. Індивідуальні завдання містять граф алгоритму та його параметри.
3. При виконанні першої частини роботи необхідно розрахувати середню трудомісткість заданого алгоритму вручну, а також скласти програму розрахунку та порівняти результати обох варіантів.
4. При виконанні другої частини роботи необхідно розрахувати мінімальну та максимальну трудомісткість заданого алгоритму вручну, а також скласти програму розрахунку та порівняти результати обох варіантів.

Порядок виконання роботи

1. Ознайомитись з теоретичною частиною роботи.
2. Виконати індивідуальне завдання згідно зі своїм варіантом першої частини лабораторної роботи.
   2.1. Визначити цикли рангу (r = 1) у графі алгоритму.

2.2. Визначити середню кількість повторень циклів n_{Ci} за формулою (7).

2.3. Визначити середню трудомісткість k^T_{Ci} виконання тіл циклів C_i за формулою (5) та оформити послідовність її одержання у вигляді відповідної таблиці (див. табл. 2, 3).

2.4. Визначити середню трудомісткість k_{Ci} циклів за формулою (8).

2.5. Замінити у графі цикли C_i операторами C_i з трудомісткістю k_{Ci}.

2.6. Визначити цикли рангу r = 2, 3, ... у графі алгоритму.

2.7. Виконати послідовність дій пп. 2.2 - 2.5 для циклів рангу r = 2, 3, ..

2.8. Визначити трудомісткість алгоритму за формулою (1).

2.9. Скласти програму розрахунку середньої трудомісткості алгоритму для свого варіанта.

3. Виконати індивідуальне завдання згідно зі своїм варіантом другої частини лабораторної роботи.

3.1. Визначити цикли рангу 1 (r = 1) у графі алгоритму.

3.2. Визначити мінімальну та максимальну трудомісткість тіл циклів за формулами (9), (10) та оформити послідовність їх одержання у вигляді відповідних таблиць (див. табл. 5, 6).

3.3. Визначити мінімальну та максимальну трудомісткість циклів C_i за формулами (13) та (14).

3.4. Замінити у графі цикли C_i еквівалентними їм операторами C_i.

3.5. Визначити цикли рангу r = 2, 3, ... у графі алгоритму.

3.6. Виконати послідовність дій пп. 3.2 - 3.4 для циклів рангів r = 2, 3, ..

3.7. Визначити мінімальну та максимальну трудомісткість алгоритму за формулами (11), (12).

3.8. Скласти програму розрахунку максимальної та мінімальної трудомісткості алгоритму для свого варіанта.

Оформлення звіту

1. Звіт про виконання першої частини даної лабораторної роботи повинен містити:

1.1. Початкові дані у вигляді графу алгоритму та його параметрів згідно з варіантом завдання.

1.2. Послідовність виділення циклів, починаючи з рангу 1 (r = 1), та перетворення початкового графу алгоритму.

1.3. Послідовність розрахунку середньої трудомісткості k^T_{Ci} циклів, починаючи з рангу 1 (r = 1), у вигляді відповідних таблиць.

1.4. Формули, за якими виконується розрахунок середньої трудомісткості та її складових.
1.5. Результати проміжних розрахунків та кінцевий результат.
1.6. Програму розрахунку середньої трудомісткості алгоритму для свого варіанта завдання.
1.7. Результат виконання даної програми.
1.8. Висновки.

2. Звіт про виконання другої частини даної лабораторної роботи повинен містити:
2.1. Початкові параметри графу алгоритму.
2.2. Формули, за якими виконується розрахунок мінімальної та максимальної трудомісткості алгоритму та їх складових.
2.3. Послідовність розрахунку мінімальної та максимальної трудомісткості циклів, починаючи з рангу 1 (r = 1), у вигляді відповідних таблиць.
2.4. Результати проміжних розрахунків та кінцевий результат.
2.5. Програму розрахунку мінімальної та максимальної трудомісткості алгоритму для свого варіанта завдання.
2.6. Результат виконання даної програми.
2.7. Висновки.

ЛАБОРАТОРНА РОБОТА №4

ТРУДОМІСТКІСТЬ АЛГОРИТМІВ МЕТОД МАРКОВСЬКИХ ЛАНЦЮГІВ

Мета роботи – вивчення методу оцінювання трудомісткості алгоритмів за допомогою марковської моделі обчислювальних процесів.

1 ТЕОРЕТИЧНА ЧАСТИНА

Значення ймовірностей $[p_{0,1}, \dots, p_{0,H+1}]$ зумовлюють хід обчислювального процесу (ОП) й залежать від параметрів трудомісткості алгоритму. Ці значення обчислюються таким чином. ТА визначає, зокрема, середнє число $N_1, \dots, N_H$ звернень до файлів $F_1, \dots, F_H$. Отже, середня кількість переходів зі стану S_0 у стани $S_1, \dots, S_H$ дорівнює $(N_1+\dots+N_H)$. Один раз процес переходить зі стану S_0 у поглинаючий стан S_{H+1} (рис. 1). Таким чином, ОП повинен виходити зі стану S_0 у середньому

$$N = \sum_{h=1}^{H} N_h + 1 \qquad (1)$$

разів, тобто в процесі реалізації алгоритму етапи обчислення зустрічаються в середньому N раз. Значення $p_{0,h}$ визначає частку переходів у стан S_h відносно всіх можливих переходів зі стану S_0 у стани $S_1, \dots, S_{H+1}$.

Ця частка дорівнює в середньому N_h/N, де N_h – середня кількість переходів у стан S_h. Отже,

$$p_{0,h} = N_h / N, \quad h = \overline{1, H}; \quad p_{0,H+1} = 1/H. \qquad (2)$$

Кількість роботи, що виконується на кожному з етапів, характеризується параметрами $\theta_1, \dots, \theta_H$ алгоритму. Значення θ визначає середню кількість процесорних операцій, що виконуються за одну реалізацію алгоритму, а значення $\theta_1, \dots, \theta_H$ – середню трудомісткість етапів, що відповідають станам $S_1^*, \dots, S_H$. Середня трудомісткість етапу обчислення дорівнює

$$\theta_0 = \theta / N, \qquad (3)$$

де N – середня кількість етапів обчислення, що визначається виразом (1).

Трудомісткість кожного етапу будемо розглядати як випадкову величину θ_h з математичним очікуванням $\overline{\theta_h}$, $h = \overline{0, H}$ закон розподілу випадкових величин $\theta_0, \dots, \theta_H$ будемо розглядати окремо.

Діаграма ОП, породжуваного марковською моделлю, зображена на рис. 2. Значення θ_i^j визначає трудомісткість відповідного етапу та являє собою j-е значення випадкової величини θ_i, математичне сподівання якої $\overline{\theta_i}$. В даному випадку стан S_3 є поглинаючим: досягнувши його, процес припиняється.

Побудуємо марковську модель ОП, породжуваного алгоритмом, трудомісткість якого характеризується такими параметрами: θ=100 млн операцій; N_1=19 звернень до файлу F_1 ; N_2 =180 звернень до файлу F_2; θ_1=2000 байт; θ_2 =500 байт за звернення до файлу.

Згідно з виразом (1) середня кількість етапів обчислення при одному прогоні алгоритму $N=N_1+N_2+1=200$. Імовірності переходу процесу зі стану обчислення S_0 у стани S_1, S_2 і S_3 визначаються співвідношеннями (2) та дорівнюють відповідно:

$p_{0,1}=N_1/N=19/200=0,095$; $p_{0,2}=N_2/N=180/200=0,9$; $p_{0,3}=N_3/N=1/N=0,005$.

Підставляючи ці значення у матрицю (6) з лабораторної роботи № 2, одержуємо матрицю імовірностей переходів:

$$P=[p_{ij}]=\begin{array}{c|cccc} & S_0 & S_1 & S_2 & S_3 \\ S_0 & 0 & 0,095 & 0.9 & 0,05 \\ S_1 & 1 & 0 & 0 & 0 \\ S_2 & 1 & 0 & 0 & 0 \\ S_3 & 0 & 0 & 0 & 0 \end{array}$$

З матриці видно, що з імовірністю 0,095 за етапом обчислення відбудеться звернення до файлу F_1, з імовірністю 0,9 – звернення до файлу F_2 і з імовірністю 0,005 ОП припиниться.

Середня трудомісткість етапу обчислення відповідно до виразу (3) становить $\theta_0=\theta/N=0,5$ млн. операцій.

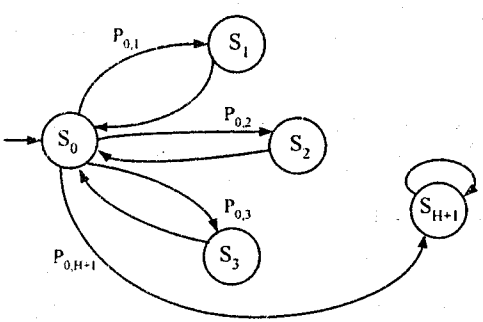


Рисунок 1 – Граф алгоритму з поглинаючим станом

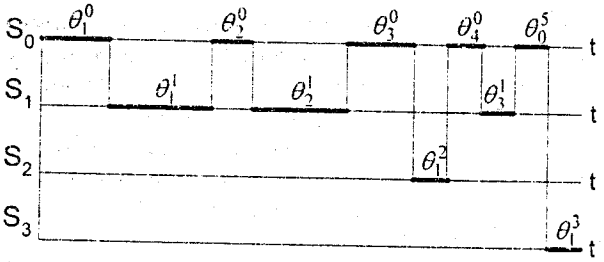


Рисунок 2 – Діаграма обчислювального процесу, породжуваного марковською моделлю

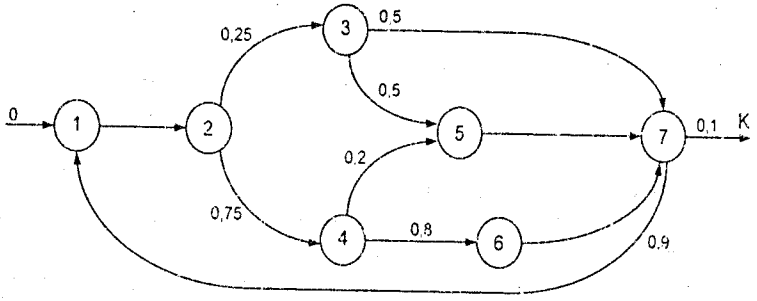


Рисунок 3 – Граф алгоритму

2 ОЦІНЮВАННЯ ТРУДОМІСТКОСТІ АЛГОРИТМІВ МЕТОДОМ МАРКОВСЬКИХ ЛАНЦЮГІВ

На рис. 3 поданий граф алгоритму, де показано множину операторів $V = \{ V_1, \dots, V_{K-1} \}$ і дуг $D = \{(i,j)\}$, $i = \overline{0, K-1}$. $j = \overline{1, K}$, що з'єднують оператори.

Для оцінювання ТА необхідно:

1. Розбити множину операторів на класи: основних операторів $S_0 = \{V_{\alpha_1}, \dots, V_{\alpha_{m0}}\}$, $V_{\alpha k} \in V$, операторів введення-виведення $S_h = \{V_{\beta_1}, \dots, V_{\beta_{mh}}\}$, $h = \overline{0, H}$, (кожен з цих операторів задає звернення до файла F_h).

2. Для кожного основного оператора V_α необхідно визначити середню кількість операцій k_α – складових оператора, і для кожного

оператора введення-виведення V_β – середню кількість інформації I_β, що передається при виконанні оператора.

3. Переходи між операторами V_i і V_j варто розглядати як випадкові події й характеризувати їх ймовірностями p_{ij}, тобто кожна дуга (i,j) графу алгоритму повинна бути відзначена ймовірністю переходу p_{ij}, з якої перехід з вершини V_i виконується саме по цій дузі, тобто до вершини V_j. Імовірності переходів повинні відповідати умові:

$$\sum_{i=0}^{K-1} p_{i,j} = 1, \quad \overline{j = 1, K}. \tag{4}$$

Припустимо, що ймовірності переходів p_{kj} постійні і після виконання оператора V_k (k=1, ... , K–1) перехід до наступного оператора визначається тільки розподілом імовірностей p_{ik}, тобто не залежить від ходу обчислювального процесу в минулому – до переходу до оператора V_k. При вказаних допущеннях процес виконання алгоритму є марковським процесом з K станів S_1, ...,S_K, які відповідають перебуванню процесу у вершинах V_1, ...,V_K графу алгоритму. Стани S_1, ...,S_{K-1} – незворотні (процес після деякої кількості кроків обов'язково залишає їх), стан S_K – поглинальний (досягши його, процес припиняється).

Початковим є стан S_i, визначений дугою (0, i), що виходить із початкової вершини графу. Для спрощення позначень домовимося, що i=1, тобто початковий стан – це стан S_1. Порядок зміни станів визначається графом алгоритму, дуги якого відзначені ймовірностями переходів p_{ij}. Відзначений граф алгоритму можна розглядати як граф марковського ланцюга.

Середня кількість n_1,...,n-1 перебувань марковського процесу в незворотних станах S_1, ...,S_{K-1} визначається коренями системи лінійних алгебраїчних (ЛАР) рівнянь вигляду

$$n_j = \delta_{ij} + \sum_{i=1}^{K-1} p_{ij} n_i \quad \overline{j = 2, K} \tag{5}$$

де δ_{ij} – символ Кронекера (δ_{ij} =1 при i = j і δ_{ij} = 0 при i ≠ j).

Рівняння (5) складаються, виходячи з таких міркувань. Значення n_j буде дорівнювати одиниці, якщо процес починається від стану з номером j=1, що позначено символом δ_{jj}. В інших випадках процес попадає в стан S_j тільки з інших станів i=1,...,K–1 з ймовірностями p_{ij}. Якщо процес перебував у i-му стані n_i разів і $p_{ij} \neq 0$, то процес з цього стану попадає у стан j в середньому $p_{ij} \cdot n_i$ разів. Підсумовуванням значень $p_{ij} \cdot n_j$ за всіма і знаходиться кількість попадань процесу в стан j з усіх інших станів i.

Значення n_1,..., n-1 визначаються рішенням системи ЛАР (5), канонічний запис якого має вигляд

$$\begin{aligned}
&(p_{1,1} - 1)\, n_1 + p_{2,1}\, n_2 + \ldots + p_{K-1,1}\, n_{K-1} = -1;\\
&p_{1,2}\, n_1 + (p_{2,2} - 1)\, n_2 + \ldots + p_{K-1,2}\, n_{K-1} = 0;\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&p_{1,K-1}\, n_1 + p_{2,K-1}\, n_2 + \ldots + (p_{K-1,K-1} - 1)\, n_{K-1} = 0.
\end{aligned} \qquad (6)$$

У результаті можна визначити ТА у вигляді середньої кількості операцій при одному прогоні алгоритму в такий спосіб:

$$\theta = \sum_{V_j \in S_0} n_j k_j, \qquad (7)$$

де k_j – кількість операцій, породжуваних кожним оператором V_j.

Розглянутий спосіб визначення ТА є універсальним, який дозволяє одержувати оцінки для алгоритмів з будь-якою структурою, простий з точки зору програмування, але вимагає розв'язання системи ЛАР, зазвичай, високого порядку.

2.1 Приклад оцінювання трудомісткості алгоритму

Нехай потрібно визначити трудомісткість алгоритму, заданого графом (рис. 3).

Для спрощення приймемо, що всі оператори алгоритму – основні (оператори введення-виведення відсутні) і кількість операцій k_j, породжуваних кожним оператором, постійна і дорівнює 1.

На основі графу (рис. 3) і формули (6) будуємо систему з семи ЛАР:

$$\begin{aligned}
&-n_1 && && && +0{,}9\, n_7 && = -1;\\
&n_1 && - n_2 && && && = 0;\\
&&0{,}25\, n_2 && - n_3 && && = 0;\\
&&0{,}75\, n_2 && && - n_4 && = 0;\\
&&&0{,}5\, n_3 + 0{,}2\, n_4 && - n_5 && = 0;\\
&&&&0{,}8\, n_4 && - n_6 && = 0;\\
&&&0{,}5\, n_3 && + n_5 + n_6 && - n_7 && = 0.
\end{aligned}$$

Розв'язуючи систему ЛАР, визначаємо середню кількість попадань обчислювального процесу в стані $S_1, \ldots, S_7$: $n_1 = 10$; $n_2 = 10$; $n_3 = 2{,}5$; $n_4 = 7{,}5$; $n_5 = 2{,}75$; $n_6 = 6{,}00$; $n_7 = 10$.

Підставляючи отримані значення у вираз (7), одержуємо оцінку ТА:

$$\theta = \sum_{j=1}^{7} n_j k_j = 10 + 10 + \ldots + 6{,}0 + 10 = 48{,}79 \text{ операцій.}$$

Контрольні питання

1. Дайте означення поняття „трудомісткість алгоритму".

2. Чим визначається марковська модель обчислювального процесу?
3. Що являє собою матриця ймовірностей переходів для марковського процесу?
4. Як складається граф алгоритму для оцінювання трудомісткості?
5. Яка умова повинна виконуватися для ймовірностей переходів графу алгоритму?
6. Дайте означення ергодичного процесу.
7. Як визначається середня трудомісткість етапу обчислення?
8. Що таке незворотний стан марковського процесу?

Зміст завдань

1. Індивідуальні завдання містять граф алгоритму (додаток Г) і його параметри (додаток Д).
2. Необхідно скласти систему лінійних алгебраїчних рівнянь.
3. Скласти програму розрахунку коренів системи лінійних алгебраїчних рівнянь.
4. Оцінити середню кількість операцій при одному прогоні алгоритму.

Порядок виконання роботи

1. Ознайомитися з теоретичною частиною роботи.
2. Виконати індивідуальне завдання відповідно до свого варіанта.
   2.1. Скласти систему лінійних алгебраїчних рівнянь вигляду (6).
   2.2. Скласти програму розв'язку системи лінійних алгебраїчних рівнянь.
   2.3. Визначити корені системи лінійних алгебраїчних рівнянь.
   2.4. Визначити трудомісткість алгоритму за формулою (7).

Оформлення звіту

Звіт про виконання даної лабораторної роботи повинен містити.

1. Вихідні дані у вигляді графу алгоритму і його параметрів відповідно до варіанта завдання.
2. Формули, за якими здійснюється розрахунок коренів системи лінійних алгебраїчних рівнянь і трудомісткості алгоритму.
3. Програму розрахунку коренів системи лінійних алгебраїчних рівнянь.
4. Результати виконання цієї програми.
5. Кінцевий результат розрахунків.
6. Висновки.

ЛАБОРАТОРНА РОБОТА №5

ДИСПЕРСІЯ ТРУДОМІСТКОСТІ АЛГОРИТМІВ

Мета роботи – вивчення основних понять теорії імовірностей, ознайомлення з основними характеристиками випадкових величин та методами їх визначення за допомогою векторно-матричних операцій.

1 ТЕОРЕТИЧНА ЧАСТИНА

Для забезпечення необхідної точності розв'язку деяких задач аналізу обчислювальних систем (ОС) необхідні відомості про дисперсію трудомісткості алгоритму. За означенням дисперсією (розсіюванням) дискретної випадкової величини називають математичне сподівання квадрата відхилення випадкової величини від її математичного сподівання[2]

$$D(x) = M\left[x - M(x)\right]^2, \qquad (1)$$

де x – дискретна випадкова величина; M(x) – математичне сподівання величини x.

Дисперсія характеризує ступінь розкиду значень випадкової величини відносно свого середнього значення. Отже, чим більша дисперсія, тим більший розкид значень. Дисперсія має розмірність, що дорівнює квадрату розмірності випадкової величини. Коли бажано, щоб оцінка розсіювання мала розмірність випадкової величини, обчислюють середнє квадратичне відхилення, а не дисперсію..

Середнім квадратичним відхиленням випадкової величини x називають квадратний корінь з дисперсії, тобто:

$$\sigma(x) = \sqrt{D(x)}. \qquad (2)$$

Процедура визначення дисперсії трудомісткості складна. В даній роботі розглядається формальна сторона цього питання [3].

Для обчислення дисперсії алгоритм подають як марковський ланцюг з матрицею імовірностей переходів [3] вигляду

$$\mathbf{P} = \left[p_{ij}\right] = \begin{array}{c} \\ 1 \\ 2 \\ \dots \\ K \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \dots & K \end{array} \\ \begin{vmatrix} p_{1,1} & p_{1,2} \cdots p_{0,K} \\ p_{2,1} & p_{2,2} \cdots p_{2,K} \\ \cdots & \cdots \cdots \cdots \\ p_{K,1} & p_{K,2} \cdots p_{k,K} \end{vmatrix} \end{array}$$

Тут K – кількість операторів алгоритму, а елементи визначають імовірності переходу зі станів $i = \overline{1,K}$ у стани $j = \overline{1,K}$. Наприклад, алгоритм (рис.1) відповідає матриці імовірностей переходів:

$$\mathbf{P} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0.75 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0.2 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0.9 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

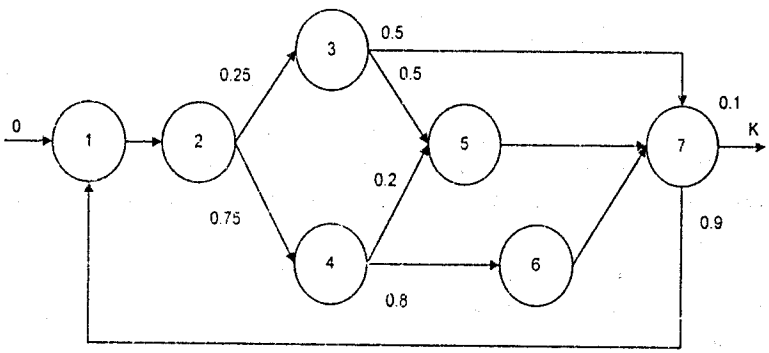


Рисунок 1 – Граф алгоритму

Трудомісткість операторів подамо вектором-стовпцем:

$$\mathbf{L} = \begin{bmatrix} l_1 \\ l_2 \\ .. \\ l_K \end{bmatrix}.$$

Для визначення дисперсії необхідно обчислити квадратну матрицю

$$N = [n_{ij}] = (I - P)^{-1}, \qquad (3)$$

де $I = \begin{bmatrix} 1 & 0 & 0 & .. & 0 \\ 0 & 1 & 0 & .. & 0 \\ 0 & 0 & 1 & .. & 0 \\ .. & .. & .. & .. & .. \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} 1 \\ \\ \\ \\ K \end{matrix}$ - одинична матриця;

(I – P) -- поелементна різниця матриць I та P, що являє собою матрицю K-го порядку:

$$(\mathbf{I} - \mathbf{P}) = \begin{bmatrix} (1 - p_{11}) & - p_{12} & .. & - p_{1K} \\ - p_{21} & (1 - p_{22}) & .. & - p_{2K} \\ .. & .. & .. & .. \\ - p_{K1} & - p_{K2} & .. & (1 - p_{KK}) \end{bmatrix}.$$

Оберненням матриці (I - P) формується матриця N, елементи n_{ij} якої мають зміст середньої кількості попадань процесу у стани з номерами $j = \overline{1,K}$, якщо процес починається зі станів з номерами $i = \overline{1,K}$.

Таким чином, якщо алгоритм починає виконуватись від оператора з номером i=1, то кількість звернень до операторів $j = \overline{1,K}$ задається першим рядком $n_{11},...,n_{1K}$ матриці N. Помноживши матрицю N на вектор-стовпець L, отримаємо вектор-стовпець

$$\boldsymbol{\theta} = N \cdot L = \begin{bmatrix} \theta_1 \\ \theta_2 \\ .. \\ \theta_K \end{bmatrix}, \qquad (4)$$

елементи якого визначають середню трудомісткість алгоритму, ініційованого від операторів $V_1,...,V_K$, відповідно.

Значення дисперсії можна визначити таким матричним виразом:

$$\mathbf{D}=\mathbf{N}(2\mathbf{L}_{dq}\mathbf{P}\boldsymbol{\theta}+\mathbf{L}_{sq})-\boldsymbol{\theta}_{sq}. \qquad (5)$$

Тут $\mathbf{L}_{dq}$ – діагональна матриця вигляду

$$\mathbf{L}_{dq}=\begin{bmatrix} I_1 & 0 & 0 & .. & 0 \\ 0 & I_2 & 0 & .. & 0 \\ 0 & 0 & I_3 & .. & 0 \\ .. & .. & .. & .. & .. \\ 0 & 0 & 0 & 0 & I_K \end{bmatrix},$$

елементи якої одержують з елементів вектора $\mathbf{L}$; $\mathbf{L}_{sq}$, $\boldsymbol{\theta}_{sq}$ – вектори-стовпці вигляду

$$\mathbf{L}_{sq}=\begin{bmatrix} l_1^2 \\ l_2^2 \\ .. \\ l_K^2 \end{bmatrix}, \qquad (6)$$

$$\boldsymbol{\theta}_{sq}=\begin{bmatrix} \theta_1^2 \\ \theta_2^2 \\ .. \\ \theta_K^2 \end{bmatrix}, \qquad (7)$$

елементи яких визначаються шляхом піднесення до квадрата елементів векторів $\mathbf{L}$ та $\boldsymbol{\theta}$.

Елементи вектора-стовпця

$$\mathbf{D}=\begin{bmatrix} D_1 \\ D_2 \\ .. \\ D_K \end{bmatrix}$$

визначають дисперсію часу виконання програми, що починається від операторів $V_1,...,V_K$, відповідно. Якщо алгоритм виконується завжди з першого оператора, то дисперсія задається першим елементом D_1 вектора $\mathbf{D}$.

1.1 Приклади оцінювання дисперсії трудомісткості алгоритму

Для першого варіанта оцінювання приймемо, що трудомісткість операторів складає $l_1 = 3, l_2 = 1, l_3 = 2$ операцій.

Згідно з графом (рис. 2) будуємо матрицю імовірностей переходів:

$$P = \begin{bmatrix} 0 & 2/3 & 0 \\ 1/3 & 0 & 2/3 \\ 0 & 1/3 & 0 \end{bmatrix}.$$

Обчислюємо матрицю N:

$$N = [n_{ij}] = (I - P)^{-1} = \begin{bmatrix} 1 & -2/3 & 0 \\ -1/3 & 1 & -2/3 \\ 0 & -1/3 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1.4 & 1.2 & 0.8 \\ 0.6 & 1.8 & 1.2 \\ 0.2 & 0.6 & 1.4 \end{bmatrix}.$$

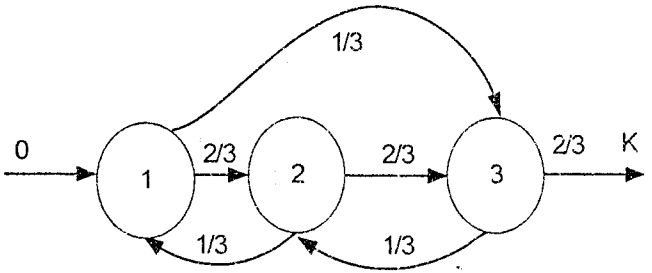


Рисунок 2 – Граф алгоритму

Визначаємо вектор середньої трудомісткості алгоритму:

$$\mathbf{0} = N \cdot L = \begin{bmatrix} 1.4 & 1.2 & 0.8 \\ 0.6 & 1.8 & 1.2 \\ 0.2 & 0.6 & 1.4 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 7 \\ 6 \\ 4 \end{bmatrix}.$$

Перший елемент $\theta_1 = 7$ вектора визначає середню трудомісткість алгоритму, ініційованого від оператора 1, елемент $\theta_2 = 6$ – трудомісткість алгоритму, ініційованого від оператора 2, і т.ін.

Застосовуючи вираз (5), знаходимо

$$\boldsymbol{D}=\begin{bmatrix}1.4 & 1.2 & 0.8\\ 0.6 & 1.8 & 1.2\\ 0.2 & 0.6 & 1.4\end{bmatrix}\cdot\left\{\left[2\cdot\begin{bmatrix}3 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 2\end{bmatrix}\cdot\begin{bmatrix}0 & 2/3 & 0\\ 1/3 & 0 & 2/3\\ 0 & 1/3 & 0\end{bmatrix}\cdot\begin{bmatrix}7\\ 6\\ 4\end{bmatrix}+\begin{bmatrix}9\\ 1\\ 4\end{bmatrix}\right\}-\begin{bmatrix}49\\ 36\\ 16\end{bmatrix}=\begin{bmatrix}20\\ 18\\ 14\end{bmatrix}$$

Середньоквадратичне відхилення трудомісткості алгоритму, ініційованого від першого оператора, становить $\sigma=\sqrt{D_1}=\sqrt{20}\approx 4.5$.

Для другого варіанта нехай потрібно обчислити дисперсію трудомісткості алгоритму (рис.2), трудомісткість операторів якого складає $l_1=1, l_2=1, l_3=1$ операцій.

Згідно з графом (рис. 2) будуємо матрицю імовірностей переходів:

$$\boldsymbol{P}=\begin{bmatrix}0 & 2/3 & 0\\ 1/3 & 0 & 2/3\\ 0 & 1/3 & 0\end{bmatrix}.$$

Обчислюємо матрицю N:

$$\boldsymbol{N}=\left[n_{ij}\right]=(\boldsymbol{I}-\boldsymbol{P})^{-1}=\begin{bmatrix}1 & -2/3 & 0\\ -1/3 & 1 & -2/3\\ 0 & -1/3 & 1\end{bmatrix}^{-1}=\begin{bmatrix}1.4 & 1.2 & 0.8\\ 0.6 & 1.8 & 1.2\\ 0.2 & 0.6 & 1.4\end{bmatrix}.$$

Визначаємо вектор середньої трудомісткості алгоритму:

$$\boldsymbol{\theta}=\boldsymbol{N}\cdot\boldsymbol{L}=\begin{bmatrix}1.4 & 1.2 & 0.8\\ 0.6 & 1.8 & 1.2\\ 0.2 & 0.6 & 1.4\end{bmatrix}\cdot\begin{bmatrix}1\\ 1\\ 1\end{bmatrix}=\begin{bmatrix}3.4\\ 3.6\\ 2.2\end{bmatrix}.$$

Перший елемент $\theta_1=3.4$ вектора визначає середню трудомісткість алгоритму, ініційованого від оператора 1, елемент $\theta_2=3.6$ – трудомісткість алгоритму, ініційованого від оператора 2, і т.ін.

Застосовуючи вираз (5), знаходимо

$$\boldsymbol{D}=\begin{bmatrix}1.4 & 1.2 & 0.8\\ 0.6 & 1.8 & 1.2\\ 0.2 & 0.6 & 1.4\end{bmatrix}\cdot\left\{\left[2\cdot\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}0 & 2/3 & 0\\ 1/3 & 0 & 2/3\\ 0 & 1/3 & 0\end{bmatrix}\cdot\begin{bmatrix}3.4\\ 3.6\\ 2.2\end{bmatrix}+\begin{bmatrix}1\\ 1\\ 1\end{bmatrix}\right\}-\begin{bmatrix}11.56\\ 12.96\\ 4.84\end{bmatrix}=\begin{bmatrix}11.04\\ 13.74\\ 6.36\end{bmatrix}.$$

Середньоквадратичне відхилення трудомісткості алгоритму, ініційованого від першого оператора, становить $\sigma = \sqrt{D_1} = \sqrt{11.04} \approx 3.32$.

Контрольні питання

1. Дайте означення дисперсії дискретної випадкової величини.
2. Дайте означення середнього квадратичного відхилення випадкової величини.
3. Що таке матриця імовірностей переходів?
4. Як виконується множення матриці на вектор? Наведіть приклад.
5. Що означають елементи вектора?
6. Як визначити середнє квадратичне відхилення трудомісткості алгоритму, ініційованого, наприклад, від 2-го оператора?
7. Дайте визначення математичного сподівання випадкової величини.
8. Що таке закон розподілу випадкової величини?
9. Коли в результаті векторного перемноження отримують скаляр (число), а коли – вектор?
10. Що таке діагональна матриця?
11. Особливості діагональної матриці (її обернена матриця) при матрично-матричному перемноженні.

Зміст завдань

1. Індивідуальні завдання містять граф алгоритму (додаток Г) та його параметри (додаток Д).
2. Необхідно визначити дисперсію трудомісткості алгоритму з ініціюванням алгоритму від операторів V_i.
3. Необхідно розрахувати середнє квадратичне відхилення трудомісткості алгоритму з ініціюванням його від всіх операторів.
4. Скласти програму розрахунку дисперсії та середнього квадратичного відхилення трудомісткості алгоритму.

Порядок виконання роботи

1. Ознайомитись з теоретичною частиною роботи.
2. Виконати індивідуальне завдання згідно зі своїм варіантом.
   2.1. Скласти матрицю імовірностей переходів графу алгоритму.
   2.2. Обчислити матрицю **N** за формулою (3).
   2.3. Обчислити вектор середньої трудомісткості алгоритму за формулою (4).
   2.4. Обчислити вектори за формулами (6) та (7), відповідно.
   2.5. Обчислити вектор дисперсії трудомісткості **D** за формулою (5).
   2.6. Обчислити середнє квадратичне відхилення за формулою (2).

2.7. Звести проміжні та остаточні результати у таблицю.
2.8. Скласти програму розрахунку дисперсії та середнього квадратичного відхилення трудомісткості алгоритму.

Оформлення звіту

Звіт про виконання даної лабораторної роботи повинен містити.

1. Початкові дані у вигляді графу алгоритму та його параметрів згідно з варіантом завдання.
2. Формули, за якими виконується розрахунок дисперсії та середнього квадратичного відхилення трудомісткості алгоритму.
3. Результати проміжних розрахунків та кінцевий результат.
4. Програму розрахунку дисперсії та середнього квадратичного відхилення трудомісткості алгоритму.
5. Результати виконання даної програми.
6. Висновки.

ЛАБОРАТОРНА РОБОТА №6

ЕФЕКТИВНІСТЬ ПАРАЛЕЛЬНОЇ ВИБІРКИ МІКРОКОМАНД

Мета роботи – вивчення методу оцінювання ефективності способу паралельної вибірки мікрокоманд з використанням марковської моделі.

1 ТЕОРЕТИЧНА ЧАСТИНА

Швидкодія керуючого автомата характеризується часом, що витрачається на формування одного набору керуючих сигналів. Ця величина складається з трьох складових: 1) часу формування адреси наступної мікрокоманди; 2) часу звернення до постійного запам'ятовувального пристрою (ПЗП); 3) часу дешифрування мікрокоманди.

Основна частка часу припадає на читання мікрокоманд з ПЗП, тому значне збільшення швидкодії керуючого автомата може бути досягнуте двома способами: 1) за рахунок зменшення часу звернення до ПЗП; 2) за рахунок скорочення кількості звернень до ПЗП в процесі функціонування керуючого автомата [5].

Розглянемо більш детально 2-й спосіб. Довжину слова ПЗП можна призначити рівній сумарній довжині K мікрокоманд. В такому випадку за одне звернення до ПЗП вибирається слово, що містить K мікрокоманд. Ці мікрокоманди будуть оброблятися послідовно в порядку, що диктує мікропрограма. Очевидно, що з K вибраних мікрокоманд реалізується в середньому $1 < N < K$ мікрокоманд, в результаті чого витрати часу на вибірку з ПЗП однієї мікрокоманди зменшуються в N раз, тобто ефективна швидкодія ПЗП збільшується в N раз.

Керуючі автомати, одне слово яких містить декілька мікрокоманд, називаються автоматами з паралельною вибіркою мікрокоманд. Паралельна вибірка використовується виняткова з метою збільшення швидкодії.

1.1 Приклад оцінювання ефективності паралельної вибірки мікрокоманд

Оцінимо ефективність способу паралельної вибірки мікрокоманд.

Нехай слово ПЗП містить K мікрокоманд. Будемо говорити, що у разі звернення до мікрокоманди $1,2,\ldots,K$ процес виконання мікропрограми знаходиться у стані $1,2,\ldots,K$ відповідно і при виконанні будь-якої іншої мікрокоманди, що не належить даному слову, знаходиться у стані 0. Приймемо, що процес зі стану 0 переходить в будь-який стан $j = 1,2,\ldots,K$ з однаковою імовірністю $p_{0j} = 1/K$. Після переходу в стан j процес

переходить або в стан $(j+1)$, зумовлений природним порядком слідування мікрокоманд, або в деякий стан, що задається мікрокомандами умовного або безумовного переходу. Процес розвивається в межах станів 1,2,..., K до тих пір, поки деяка мікрокоманда не передасть керування мікрокоманді, що належить іншому слову ПЗП, тобто до тих пір, поки процес не перейде у стан 0. Імовірнісною моделлю означеного процесу є марковський ланцюг з $(K+1)$ станами [6], в кожному з яких процес перебуває одиницю часу і переходить зі стану i у стан j з імовірністю p_{ij} причому

$$\sum_{i=0}^{K} p_{i,j} = 1 \quad \left(j = \overline{0,K}\right) \tag{1}$$

Граф, що відповідає розглянутому марковському ланцюгу, зображений на рис. 1.

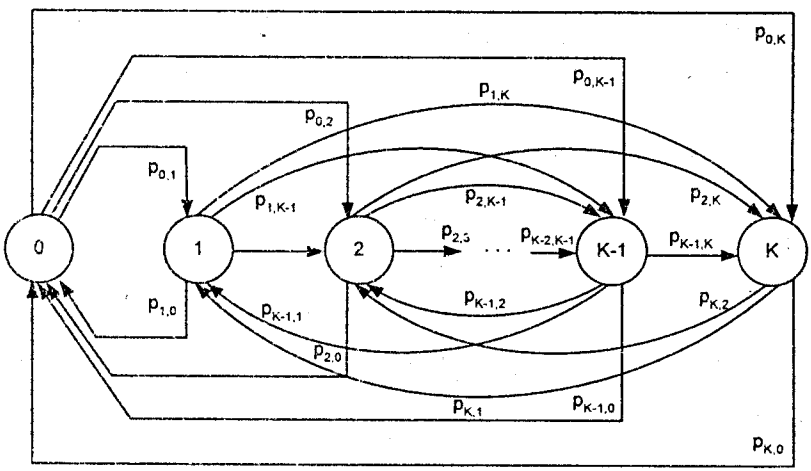


Рисунок 1 – Граф марковського ланцюга

Щоб оцінити ефективність паралельної вибірки мікрокоманд, необхідно визначити час перебування процесу в станах 1,2,..., K за умови, що процес попадає в ці стани, виходячи зі стану 0 тільки один раз. Позначимо через $n_0, n_1, ..., n_K$ кількість перебувань процесу в станах 1,2,..., K, причому $n_0 = 1$. Тоді сумарний час перебування процесу в станах 1,2,..., K складе

$$N = \sum_{j=1}^{K} n_j \tag{2}$$

і кількість звернень до ПЗП, що приходиться в середньому на одну мікрокоманду, буде дорівнювати

$$k = l / N. \tag{3}$$

Для марковського ланцюга (рис.1) справедлива система співвідношень

$$n_j = \sum_{\substack{i=0 \\ (i \neq j)}}^{K} p_{i,j} n_i \quad \left(j = \overline{0,K}\right), \tag{4}$$

яка встановлює, що кількість перебувань процесу в j-му стані n_j дорівнює сумарній кількості перебувань у станах $\left(i = \overline{0,K}\right)$, помноженому на ймовірність переходів зі станів i у стани j. Таким чином, при заданих ймовірностях переходів p_{ij} визначення значень $n_1, n_2, \ldots, n_K$ зводиться до рішення системи лінійних алгебраїчних рівнянь вигляду (4) K-го порядку.

Імовірності переходів p_{ij} визначаються таким чином. Нехай мікрокоманди, що являють собою функціональні оператори, виконуються з імовірністю q, тоді мікрокоманди, що реалізують оператори переходу, виконуються з імовірністю

$$\bar{q} = 1 - q. \tag{5}$$

Отже, мікрокоманди породжують природний порядок прямування з імовірністю q і цей порядок порушується з імовірністю $\bar{q}$. Якщо слідом за мікрокомандою з адресою α виконується мікрокоманда з адресою β, то говорять, що різниця

$$l = |\alpha - \beta| \tag{6}$$

визначає довжину переходу, причому $l \geq 1$. Зазвичай, можна прийняти, що довжини переходів розподілені за геометричним законом, отже, імовірність переходу з довжиною l дорівнює

$$\sigma_l = \rho(1-\rho)^{l-1} \quad (l = 1,2,\ldots), \tag{7}$$

причому

$$\rho = 1 / L, \tag{8}$$

де L – середня довжина переходу. Позначимо імовірність переходу вперед у бік мікрокоманд з більшими адресами через r, а імовірність переходу назад через

$$\bar{r} = 1 - r. \tag{9}$$

Розглянутому марковському ланцюгу відповідає матриця **P** перехідних імовірностей вигляду (10).

$$P = [p_{ij}] = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ \dots \\ K-1 \\ K \end{array} \begin{array}{c} \begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & \dots & K-1 & K \end{array} \\ \begin{bmatrix} 0 & \pi_0 & \pi_0 & \pi_0 & \pi_0 & \dots & \pi_0 & \pi_0 \\ P_1 & 0 & a & b_2 & b_3 & \dots & b_{K-2} & b_{K-1} \\ P_2 & c_1 & 0 & a & b_2 & \dots & b_{K-3} & b_{K-2} \\ P_3 & c_2 & c_1 & 0 & a & \dots & b_{K-4} & b_{K-3} \\ P_4 & c_3 & c_2 & c_1 & 0 & \dots & b_{K-5} & b_{K-4} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ P_{K-1} & c_{K-2} & c_{K-3} & c_{K-4} & c_{K-5} & \dots & 0 & a \\ P_K & c_{K-1} & c_{K-2} & c_{K-3} & c_{K-4} & \dots & c_1 & 0 \end{bmatrix} \end{array}. \quad (10)$$

Тут π_0 – імовірність звернення до i-ї мікрокоманди вигляду

$$\pi_0 = p_{0,i} = 1/K \ \left(i = \overline{1,K}\right), \quad (11)$$

a – імовірність переходу вперед на одну мікрокоманду, що виконується в природному порядку або шляхом передавання керування наступній мікрокоманді:

$$a = p_{i,i+1} = q + \bar{q}r\sigma_1 \ \left(i = \overline{1,K-1}\right), \quad (12)$$

b_l – імовірність переходу вперед на $l = \overline{2,K-i}$ мікрокоманд:

$$b_l = p_{i,i+l} = \bar{q}r\sigma_l \ \left(i = \overline{1,K-2}\right), \quad (13)$$

c_l – імовірність переходу назад на $l = \overline{1,i-1}$ мікрокоманд:

$$c_l = p_{i,i-l} = \bar{q}\bar{r}\sigma_l \ \left(i = \overline{2,K}\right), \quad (14)$$

p_i – імовірність того, що мікрокоманда i передасть керування мікрокоманді, яка не міститься у вибраному слові:

$$P_i = p_{i,0} = \bar{q}\left(r\sum_{l=K-i}^{\infty}\sigma_l + \bar{r}\sum_{l=i}^{\infty}\sigma_l\right) = \bar{q}\left[r(1-\rho)^{K-i} + \bar{r}(1-\rho)^{i-1}\right] \ \left(i = \overline{1,K-1}\right), \quad (15)$$

причому

$$P_K' = P_K + q\,. \quad (16)$$

Використовуючи визначені значення перехідних імовірностей, систему (4) можна записати таким чином:

$$\sum_{j=1}^{i-2} b_{i-j}n_j + an_{i-1} - n_i + \sum_{j=i+1}^{K} c_{j-i}n_j = -\pi_0 \ \left(i = \overline{1,K}\right). \quad (17)$$

У рівняннях (17) перша сума не існує (відсутня) при $i = 1$ та $i = 2$, другий доданок не існує при $i = 1$ і остання сума – при $i = K$. Корені системи ЛАР (17) визначають значення $n_1, n_2, ..., n_K$, підстановка яких у вираз (2) дозволяє визначити кількість мікрокоманд N, що виконуються за одне звернення до ПЗП.

Графіки на рис. 2 використовуються для визначення кількості K мікрокоманд, що вибираються паралельно, яка необхідна для забезпечення потрібної швидкодії керуючого автомата. Якщо τ – граничнo припустимий час вибірки однієї мікрокоманди, а T – тривалість циклу звернення до ПЗП, то

$$k = \tau / T \qquad (18)$$

і величина K вибирається з графіків рис.2 як аргумент функції $k=\varphi(q,L,K)$.

На рис. 2 подані значення k, які визначають кількість звернень до ПЗП, що приходиться на одну виконану мікрокоманду, причому кривим 1, 2, 3, 4 відповідають значення L = 3, 5, 7, 11. Значення k залежать від кількості K мікрокоманд, що вибираються паралельно, і характеристик q і L мікропрограми. Графіки $k=\varphi(q,L,K)$ побудовані з припущенням про однакову імовірність переходів вперед і назад (r=0.5). Експеримент дозволяє зробити висновок про слабкий вплив значення r на величину k: при $0.25 \le r \le 0.75$ відхилення k від значень, що визначаються рис. 2, не перевищує 6%.

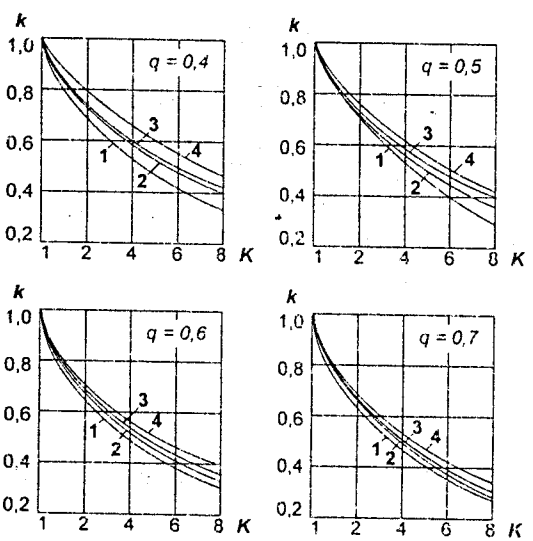


Рисунок 2 – Залежність $k=\varphi(q,L,K)$

Контрольні питання

1. Назвіть три складові часу, який необхідний для формування одного набору керуючих сигналів.
2. Назвіть два способи, що знижують витрати часу на читання мікрокоманд з ПЗП.
3. Поясніть, що таке автомат з паралельною вибіркою мікрокоманд.
4. Який вигляд має граф марковського ланцюга, що описує переходи між мікрокомандами?
5. Як визначається сумарний час перебування процесу в станах 1,..., K?
6. Як визначається кількість перебувань процесу в i-му стані n_i?
7. Напишіть формулу, що визначає імовірність переходу довжиною l.
8. Напишіть формули, що визначають імовірність переходу: а) вперед на одну мікрокоманду; б) вперед на $l = 2,3,...,K - i$ мікрокоманд; в) назад на $l = 1,2,...,i - 1$ мікрокоманд.
9. Як визначається імовірність переходу за умови, що мікрокоманда i передає керування мікрокоманді, що відсутня в даному слові?
10. Напишіть систему рівнянь для марковського ланцюга з використанням перехідних імовірностей.

Зміст завдань

1. Індивідуальні завдання містять значення параметрів q, r, L, k марковського ланцюга (додаток Е).
2. Необхідно визначити кількість мікрокоманд, що вибираються паралельно.
3. Скласти матрицю перехідних імовірностей для марковського ланцюга.
4. Скласти систему ЛАР.
5. Скласти програму розрахунку коренів системи ЛАР.
6. Оцінити сумарний час перебування у станах 1,..., K.

Порядок виконання роботи

1. Ознайомитися з теоретичною частиною роботи.
2. Визначити індивідуальне завдання згідно зі своїм варіантом.
   2.1. Визначити величину K мікрокоманд, що вибираються паралельно, використовуючи дані додатку Е і графіки на рис. 2.
   2.2. Скласти матрицю вигляду (9), заздалегідь розрахувавши значення її елементів за формулами (10) – (15).

2.3. Скласти систему ЛАР вигляду (16).
2.4. Скласти програму розв'язання системи ЛАР.
2.5. Визначити корені системи лінійних алгебраїчних рівнянь.
2.6. Визначити сумарний час перебування у станах 1 ,..., K за формулою (2).
2.7. Розрахувати реальну кількість звернень k' до ПЗП на одну команду за формулою (3).
2.8. Знайти абсолютну та відносну величину відхилення величини k' від k за виразами

$$\Delta = |k - k'|,$$

$$\delta = \frac{\Delta}{k} \cdot 100\%.$$

Оформлення звіту

Звіт про виконання даної лабораторної роботи повинен містити:

1. Вхідні дані у вигляді графу марковського ланцюга, що описує переходи між мікрокомандами, і вхідні параметри процесу згідно з варіантом завдання.
2. Формули, за якими виконується розрахунок матриці перехідних ймовірностей.
3. Систему ЛАР.
4. Програму розрахунку коренів системи ЛАР.
5. Результати виконання цієї програми.
6. Кінцевий результат розрахунків.
7. Висновки.

ЛАБОРАТОРНА РОБОТА №7

ПОШУК МІНІМАЛЬНОГО ШЛЯХУ

Мета роботи – ознайомитись з матричними моделями розв'язання прикладних задач на графах.

1 ТЕОРЕТИЧНА ЧАСТИНА

Графи, завдяки своїй наочності і наявності багатьох корисних властивостей, досить популярні при розв'язанні задач у різних наукових і практичних областях. Визначимо графи як об'єкти, які можна зобразити у вигляді пронумерованих точок (вершин), з'єднаних лініями (ребрами). Якщо граф **G** (рис. 1) задається множиною **X** точок чи вершин $x_1, x_2, ..., x_n$ і множиною **A** ліній або ребер $a_1, a_2, ..., a_m$, що з'єднують всі або частину цих точок, то вважають, що граф повністю задається парою **(X, A)** [7].

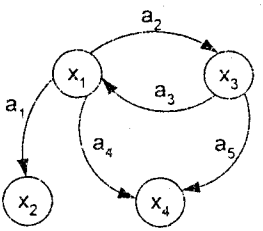


Рисунок 1 – Орієнтований граф

Якщо ребра з множини **A** орієнтовані, що зображається стрілкою, то вони називаються дугами (дуга від вершини x_i до вершини x_j позначається $a=(x_i, x_j)$) і граф називається *орієнтованим графом (орграфом)* (рис.1). Якщо ребра не мають орієнтації, то граф називається *неорієнтованим графом (неографом)*.

Ребра $a=(x_i, x_j)$, $x_i \neq x_j$, які мають спільну кінцеву вершину, називаються суміжними. Дві вершини називаються *суміжними*, якщо існує хоча б одне ребро, яке з'єднує ці вершини.

Ребро $a=(x_i, x_j)$ *інцидентне* вершинам x_i , x_j і, навпаки, вершини x_i, x_j інциденти до ребра a. Кількість ребер, інцидентних вершині x_i, має назву *степінь вершини*. Графи, для яких зберігається відношення інцидентності, називаються *ізоморфними графами*.

Якщо ребрам графу **G** ставляться у відповідність числа, тобто ребру (x_i, x_j) ставиться у відповідність деяке число c_{ij}, яке називається вагою ребра, то граф **G** називається *графом зі зваженими ребрами*. Іноді вага приписується до вершин, тоді граф називається *графом зі зваженими вершинами*. Якщо ваги приписані і до ребер і до вершин, то граф називається *зваженим графом*.

Граф **G = (X, A)**, в якого існує хоча б одна пара вершин, що з'єднуються m ребрами ($m>1$), називається *мультиграфом*. Ребра, що зв'язують одну і ту саму пару вершин, називаються кратними ребрами.

Граф **G = (X, A)** називається *повним графом*, якщо для будь-якої пари вершин x_i та x_j в множині **X** існує принаймні одне ребро, що зв'язує їх.

Граф **G = (X, A)** називається *двочастковим графом (біграфом)*, якщо множину його вершин **X** можна розбити на такі дві підмножини $\mathbf{X}^a$ та $\mathbf{X}^b$ так, що кожне ребро має один кінець в підмножині $\mathbf{X}^a$, а другий – в підмножині $\mathbf{X}^b$ (рис. 2).

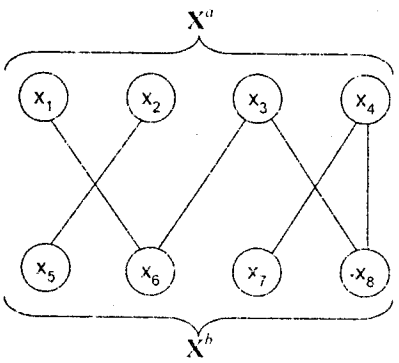


Рисунок 2 – Двочастковий граф

Граф називається *планарним графом*, якщо його можна зобразити на площині таким чином, що деякі два його ребра не перетинаються.

Плоский граф – граф, що зображається на площині без перетину ребер.

Просторовий граф – граф, що зображається в тривимірному просторі.

Підграфом графу **G = (X, A)** називається граф **G' = (X', A')**, для якого $\mathbf{A'} \subset \mathbf{A}$ та $\mathbf{X'} \subset \mathbf{X}$.

Суграфом графу **G** = (**X**, **A**) називається граф **G'** = (**X**, **A'**), для якого **A'** ⊂ **A**.

Нехай **G** = (**X**, **A**) – неорієнтований граф без петель (ребер (x_i, x_j)) і кратних ребер. *Шляхом s* в графі **G** називається послідовність ребер в якій пари сусідніх ребер a_i, a_{i-1} (i = 1, 2 , ..., n -1) – суміжні (n – довжина маршруту). Шлях *s*, в якому немає ребер, які б повторювались, називається *ланцюгом*. Якщо в деякому ланцюгу збігаються початкова та кінцева вершини, то такий ланцюг називається *циклом c*. Цикл буде простим, якщо у ньому немає вершин, що повторюються, і складним – у протилежному випадку. Цикл c_E, до складу якого входять всі вершини графу, називається *ейлеревим*. Цикл $c_Г$, який проходить через всі вершини графу **G** по одному разу називається *гамільтоновим*.

Зв'язний граф – граф, в якому будь-які дві вершини можна з'єднати ланцюгом. Зв'язний граф без циклів називається *деревом* **T**. Множина дерев графу **G** називається *лісом*. У дереві будь-які дві вершини зв'язані єдиним ланцюгом.

Для алгебраїчного подання графів та їх обробки на ЕОМ використовуються різні матричні еквіваленти.

Матрицею суміжності $\mathbf{A} = \|a_{ij}\|_{n \times n}$ (n – порядок матриці,nxn – розмірність матриці) орграфу **G** називається матриця, в якій: $a_{ij} = m$, якщо в **G** є m ребер (x_i, x_j); $a_{ij} = 0$, якщо в **G** немає ребра (x_i, x_j); n – кількість вершин.

Матриця суміжності для графу, поданого на рис.1, має вигляд

$$\mathbf{A} = \begin{array}{c|cccc} & x_1 & x_2 & x_3 & x_4 \\ x_1 & 0 & 1 & 1 & 1 \\ x_2 & 0 & 0 & 0 & 0 \\ x_3 & 1 & 0 & 0 & 1 \\ x_4 & 0 & 0 & 0 & 0 \end{array}. \qquad (1)$$

Для неографів елемент a_{ij} дорівнює кількості кратних ребер між вершинами x_i та x_j, а матриця **A** – симетрична. Матрицею *інциденцій* $\mathbf{B} = \|b_{ij}\|_{n \times m}$ (nxm – розмірність матриці) для орграфу називається матриця в якій: b_{ij} =1, якщо x_i є початковою вершиною дуги a_i , b_{ij} =-1, якщо x_j є кінцевою вершиною дуги a_j; b_{ij} =0 якщо x_j не є вершиною дуги a_j або a_j є петлею; n – кількість вершин; m – кількість дуг в графі **G**.

Матриця інциденцій для графу, показаного на рис.1, має вигляд

$$\mathbf{B}=\begin{array}{c|ccccc} & a_1 & a_2 & a_3 & a_4 & a_5 \\ x_1 & 1 & 1 & -1 & 1 & 0 \\ x_2 & -1 & 0 & 0 & 0 & 0 \\ x_3 & 0 & -1 & 1 & 0 & 1 \\ x_4 & 0 & 0 & 0 & -1 & -1 \end{array}. \qquad (2)$$

Для неографу b_{ij} =1, якщо вершина x_i є інцидентною до ребра a_j, в протилежному випадку b_{ij} =0.

Відстанню $d(x_i, x_j)$ між вершинами $x_i, x_j \in \mathbf{X}$ графу $\mathbf{G} = (\mathbf{X}, \mathbf{A})$ називається довжина найкоротшого ланцюга, що з'єднує ці вершини. Під довжиною ланцюга розуміємо кількість ребер, що в нього входять. Функцію відстані графу **G** зручно задавати матрицею відстаней $\mathbf{D}=\|d_{ij}\|_{n\times n}$, де

$$d_{ij} = \begin{cases} 0, & якщо\ x_i = x_j; \\ d(x_i, x_j), & якщо\ x_i \neq x_j. \end{cases} \qquad (3)$$

Для графу, зображеного на рис.1, матриця **D** має вигляд

$$\mathbf{D}=\begin{array}{c|cccc} & x_1 & x_2 & x_3 & x_4 \\ x_1 & 0 & 1 & 1 & 1 \\ x_2 & 0 & 0 & 2 & 2 \\ x_3 & 1 & 2 & 2 & 1 \\ x_4 & 1 & 2 & 1 & 0 \end{array} \qquad (4)$$

Для графу, що розглядається в системі координат XY, функція відстані між вершинами x_i, x_j може бути визначена такими способами [7].

1. В евклідовій метриці – як відстань між двома точками площини $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.
2. В ортогональній (лінійній) метриці (Манхеттенів простір) $d_{ij} = |x_i - x_j| + |y_i - y_j|$.
3. У нелінійній метриці $d_{ij} = (x_i - x_j)^k + (y_i - y_j)^k$, де $k = 2, 3, \ldots$.

2 ЗАДАЧА ПОШУКУ МІНІМАЛЬНОГО ШЛЯХУ НА ГРАФАХ

Графи можна інтерпретувати, наприклад, як схеми районів міста, в яких зображені вулиці з двостороннім рухом (неорієнтований граф) і з однобічним рухом (орієнтований граф). Ця інтерпретація широко використовується в практиці оптимізації транспортних потоків [8].

Розглянемо орієнтований граф, зображений на рис. 3. Отже, рис. 3 із відповідною матрицею зв'язності:

$$B = \begin{vmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix} \quad (5)$$

показує нам, що з вершини 1 безпосередньо доступні вершини 2, 3, 4, 6, тобто, існує ребро, що з'єднує вершину 1 з цими вершинами. У такому випадку говорять, що існує ***шлях довжини один*** з вершини 1 у вершину 2, з вершини 1 у вершину 3 і так далі. Легко помітити, що з вершини 1 у вершину 2 можна перейти по двох ребрах через вершину 3, а у вершину 6 – по трьох ребрах через вершини 3 і 4. Тобто з вершини 1 у вершину 2 існує ***шлях довжини два***, а у вершину 6 – ***шлях довжини три.*** Отже, вершина *i* доступна з вершини j, якщо існує принаймні один шлях з вершини *i* у вершину j. Матриця зв'язності відображає доступність вершин за допомогою шляхів довжини один. Поставимо перед собою задачу – на основі матриці зв'язності одержати матриці доступності вершин за допомогою шляхів довжини два, три і так далі.

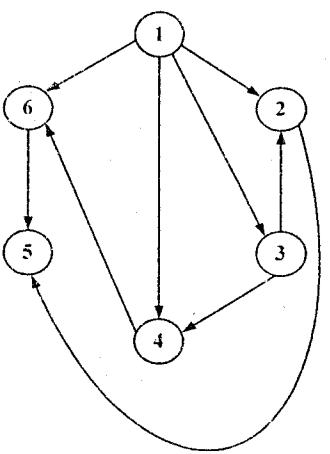


Рисунок 3 – Орієнтований граф

Для розв'язання цієї задачі скористаємося тим, що елементи матриць приймають тільки два значення 0 і 1. Отже можна використовувати замість арифметичних операцій додавання і множення елементів матриць логічні операції диз'юнкції V і кон'юнкції Λ. Тоді елементи суми двох матриць обчислюються за формулою

$$c_{ij} = a_{ij} \vee b_{ij} \tag{6}$$

а елементи добутку – за формулою

$$c_{ik} = \bigvee_{j=1}^{n} a_{ij} \,\&\, b_{jk} . \tag{7}$$

Розглянемо матрицю $\boldsymbol{B}$ зв'язності вигляду (5) для орієнтованого графу (рис. 3). Обчислимо матрицю $\boldsymbol{B^2 = B \times B}$. Вона має вигляд

$$\boldsymbol{B}^2 = \begin{vmatrix} 0 & 1 & 0 & i & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix} . \tag{8}$$

Ця матриця відображає доступність вершин за допомогою шляхів довжини два. Дійсно, на орієнтованому графі (рис.3) існують шляхи довжини два з вершини 1 у вершини 2,4,5 і 6, з вершини 2 у вершини 5 і 6, з вершини 4 у вершину 5. Обчислимо тепер матрицю $\boldsymbol{B^3 = B^2 \times B}$. Її значення

$$\boldsymbol{B}^3 = \begin{vmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix} \tag{9}$$

свідчать, що на нашому графі (рис.3) існують шляхи довжини три з вершини 1 у вершини 5 і 6, з вершини 3 у вершину 5. Можна перевірити,

що всі елементи матриці $B^4=B^3 x\ B$ дорівнюють нулю. Це свідчить про відсутність на графі шляхів довжини чотири і більше.

Зіставимо кожному ребру орієнтованого графу невід'ємне число, що називають вартістю. Позначимо c_{ij} вартість ребра, що виходить з вершини i у вершину j. Приклад такого графу наведений на рис. 4.

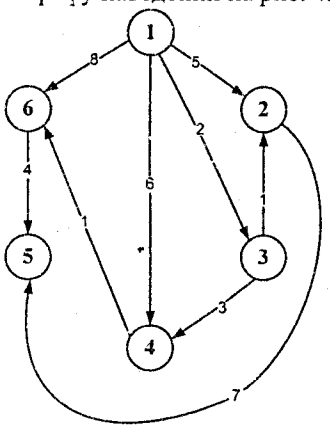


Рисунок 4 - Орієнтований граф із заданими вартостями ребер

Поставимо перед собою задачу: для кожної пари вершин знайти шлях, що складається з ребер, сумарна вартість яких буде мінімальною. Дійсно, на рис.4 ми бачимо, що з вершини *1* у вершину 6 ведуть три шляхи: шлях довжини один, вартість якого 8, шлях довжини два, через вершину *4*, вартість якого 7, і шлях довжини три через вершини *3* і *4*, вартість якого 6. Останній шлях і буде розв'язанням задачі для вершин 1 і 6.

2.1 Приклад розв'язання задачі пошуку мінімального шляху

Для розв'язання задачі пошуку шляху мінімальної вартості, названої також задачею пошуку мінімального шляху, подамо граф за допомогою матриці вартості (МВ), яку позначимо $D=\|d_{ij}\|$. Якщо немає ребра з вершини i у вершину j, то будемо вважати, що відповідна вартість $d_{ij}=\infty$, тобто нескінченно велика. Тоді графу, наведеному на рис. 4, відповідає МВ **D** вигляду

$$\mathbf{D} = \begin{vmatrix} 0 & 5 & 2 & 6 & \infty & 8 \\ \infty & 0 & \infty & \infty & 7 & \infty \\ \infty & 1 & 0 & 3 & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & 1 \\ \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & 4 & 0 \end{vmatrix}. \qquad (10)$$

Визначимо операцію множення матриць вартості. Якщо **A** і **B** – дві матриці, то елементи їх добутку – матриці **C**, обчислюються за формулою:

$$c_{ik} = \min\left(a_{i1} + b_{1k}, \ldots, a_{in} + b_{nk}\right). \qquad (11)$$

Обчислимо матрицю

$$\mathbf{D^2} = \begin{vmatrix} 0 & 3 & 2 & 5 & 12 & 7 \\ \infty & 0 & \infty & \infty & 7 & \infty \\ \infty & 1 & 0 & 3 & 8 & 4 \\ \infty & \infty & \infty & 0 & 5 & 1 \\ \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & 4 & 0 \end{vmatrix} \qquad (12)$$

і матрицю

$$\mathbf{D^3} = \begin{vmatrix} 0 & 3 & 2 & 5 & 10 & 6 \\ \infty & 0 & \infty & \infty & 7 & \infty \\ \infty & 1 & 0 & 3 & 8 & 4 \\ \infty & \infty & \infty & 0 & 5 & 1 \\ \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & 4 & 0 \end{vmatrix}. \qquad (13)$$

Елементи цих матриць відображають вартості шляхів різної довжини. Оскільки на графі (рис.4) немає шляхів довжиною більшою ніж три, подальші ступені МВ **D** нічим не будуть відрізнятися від матриці $\mathbf{D^3}$.

Контрольні питання

1. Дайте означення орієнтованому і неорієнтованому графу?
2. Що таке матриця суміжності?
3. Що таке матриця інциденцій?

4. Що означає шлях довжини 1(2,3..)?
5. Коли вершина *i* доступна з вершини *j*?
6. Який граф називають повним, планарним, просторовим, зв'язним?
7. Чи дозволяється використати замість арифметичних операцій додавання і множення елементів матриць логічні операції диз'юнкції та кон'юнкції?
8. Чим орграф відрізняється від неографу?
9. Що таке матриця вартості?
10. Що таке відстань між вершинами графу?
11. У чому суть задачі пошуку мінімального шляху на графах?
12. Наведіть особливості евклідової, дінійної та нелінійної метрик?

Зміст завдань

Необхідно розрахувати вартості доступності вершин шляхів різної довжини згідно зі своїм варіантом (додаток Ж).

Порядок виконання роботи

1. Ознайомитися з теоретичною частиною роботи.
2. Виконати індивідуальне завдання згідно зі своїм варіантом.
   2.1. Подати свій граф за допомогою матриці вартості.
   2.2. Визначити матриці, що відображають вартості шляхів різної довжини.

Оформлення звіту

Звіт про виконання даної лабораторної роботи повинен містити.
1. Початкові параметри графу алгоритму згідно з варіантом завдання.
2. Матрицю вартості за даним графом.
3. Формули розрахунку доступності вершин вартості шляхів різної довжини.
4. Результати розрахунку доступності вершин вартості шляхів різної довжини.
5. Висновки.

Додаток А

Параметри базових арифметичних операцій

Мікрооперації

Мікрооперація встановлення використовується для присвоювання слову константи і записується в такому вигляді:
<слово>:=<константа>.
Наприклад: *ПП:=1, ЛЧТ:*=1.

Мікрооперація інвертування використовується для заміни значення слова на обернене і записується у вигляді:
<слово>:= ┐ <слово>,
де ліворуч і праворуч вказується одне і те ж слово.
Наприклад: *СМ:=* ┐ *СМ.*

Мікрооперація передачі використовується для присвоювання одному слову значення, складеного з декількох слів, полів або їх інверсій. Наприклад: *С:*=11. ┐ *М* – мікрооперація присвоювання слову *С* оберненого значення слова *М*, яке з боку старших розрядів доповнене кодом 11.

Мікрооперація зсуву використовується для зміни положення розрядів слова відповідно до початкового положення. Наприклад, якщо *А*(32) та *В*(32) – два 32-розрядні слова, то мікрооперації передачі *А*:=*В*(1:30) та *А*:=*В*(3:32) можуть розглядатися як передачі з зсувом на 2 розряди відповідно праворуч (у бік молодших розрядів) та ліворуч (у бік старших розрядів). При цьому в розряди слова *А*, що звільнилися вводяться нулі. Мікрооперації зсуву можуть бути пов'язані з одним словом. В цьому випадку мікрооперації переміщують значення між розрядами слова. Наприклад: мікрооперації *А*:=*А*(1:30) та *А*:=*А*(3:32).00 описують зсув слова А на 2 розряди відповідно праворуч і ліворуч.

Мікрооперація додавання визначає суму або різницю значень. Наприклад: *СМ*:=*СМ*+1, ┐ *А*+1, *П СМ:=П.СМ* ⊞ *11,* ┐ *В(2:n).*

Додаток Б

Таблиця Б. 1 – Варіанти операцій та їх параметри

Варіант	Форма подання числа	Розряд операнда (мантиси)	Розряд порядку	Операція	Примітки
1	2	3	4	5	6
1	з фіксованою комою	8	-	додавання	доповняльний код
2	з плаваючою комою	16	4	додавання	обернений код
3	з фіксованою комою	8	-	додавання	обернений код
4	з плаваючою комою	16	4	додавання	доповняльний код
5	з фіксованою комою	8	-	множення	-
6	з плаваючою комою	16	4	множення	-
7	з фіксованою комою	8	-	прискорене множення	-
8	з плаваючою комою	16	4	прискорене множення	-
9	з фіксованою комою	8	-	ділення	-
10	-	8	-	ділення	цілі числа
11	з фіксованою комою	16	-	додавання	доповняльний код
12	з плаваючою комою	8	4	додавання	обернений код
13	з фіксованою комою	16	-	додавання	обернений код
14	з плаваючою комою	8	4	додавання	доповняльний код
15	з фіксованою комою	16	-	множення	-
16	з плаваючою комою	8	4	множення	-
17	з фіксованою комою	16	-	прискорене множення	-
18	з плаваючою комою	8	4	прискорене множення	

Продовження таблиці Б. 1

1	2	3	4	5	6
19	з фіксованою комою	16	-	ділення	-
20	-	16	-	ділення	цілі числа
21	з фіксованою комою	4	-	додавання	доповняльний код
22	з плаваючою комою	4	8	додавання	обернений код
23	з фіксованою комою	4		додавання	обернений код
24	з плаваючою комою	4	8	додавання	доповняльний код
25	з фіксованою комою	4	-	множення	-
26	з плаваючою комою	4	8	множення	-
27	з фіксованою комою	4	-	прискорене множення	-
28	з плаваючою комою	4	8	прискорене множення	-
29	з фіксованою комою	4	-	ділення	-
30	-	4	-	ділення	цілі числа

Додаток В

Таблиця В. 1 – Ймовірності переходів вершин графу

Операція	Вер-Шина	Варіант														
		1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
Додавання в оберненому коді	1	0,1	0,5	0,5	0,5	0,1	0,1	0,1	0,2	0,1	0,5	0,4	0,2	0,5	0,3	0,9
		0,9	0,5	0,5	0,5	0,9	0,9	0,9	0,8	0,9	0,5	0,6	0,8	0,5	0,7	0,1
	2	0,2	0,4	0,6	0,3	0,1	0,8	0,1	0,3	0,3	0,4	0,3	0,8	0,5	0,5	0,6
		0,8	0,6	0,4	0,7	0,9	0,2	0,9	0,7	0,7	0,6	0,7	0,2	0,5	0,5	0,4
	3	0,3	0,3	0,3	0,5	0,3	0,9	0,3	0,5	0,6	0,2	0,2	0,6	0,4	0,5	0,5
		0,7	0,7	0,7	0,5	0,7	0,1	0,7	0,5	0,4	0,8	0,8	0,4	0,6	0,5	0,5
	4	0,4	0,2	0,2	0,5	0,5	0,6	0,5	0,5	0,5	0,2	0,1	0,9	0,3	0,4	0,1
		0,6	0,8	0,8	0,5	0,5	0,4	0,5	0,5	0,5	0,8	0,9	0,1	0,7	0,6	0,9
	5	0,5	0,1	0,1	0,4	0,5	0,5	0,5	0,4	0,2	0,4	0,5	0,5	0,2	0,2	0,1
		0,5	0,9	0,9	0,6	0,5	0,5	0,5	0,6	0,8	0,6	0,5	0,5	0,8	0,8	0,9
Множення	1	0,3	0,1	0,8	0,3	0,4	0,1	0,4	0,3	0,8	0,3	0,4	0,1	0,3	0,5	0,6
		0,7	0,9	0,2	0,7	0,6	0,9	0,6	0,7	0,2	0,7	0,6	0,9	0,7	0,5	0,4
	2	0,5	0,3	0,9	0,2	0,2	0,1	0,2	0,2	0,6	0,2	0,2	0,1	0,5	0,5	0,5
		0,5	0,7	0,1	0,8	0,8	0,9	0,8	0,8	0,4	0,8	0,8	0,9	0,5	0,5	0,5
Прискорене множення	1	0,5	0,5	0,6	0,1	0,2	0,1	0,2	0,1	0,9	0,1	0,2	0,1	0,1	0,5	0,5
		0,5	0,5	0,4	0,9	0,8	0,9	0,8	0,9	0,1	0,9	0,8	0,9	0,9	0,5	0,5
	2	0,4	0,5	0,5	0,1	0,3	0,3	0,1	0,5	0,5	0,5	0,1	0,1	0,2	0,4	0,6
		0,6	0,5	0,5	0,9	0,7	0,7	0,9	0,5	0,5	0,5	0,9	0,9	0,8	0,6	0,4
	3	0,3	0,4	0,1	0,3	0,5	0,6	0,2	0,4	0,6	0,3	0,1	0,8	0,3	0,3	0,3
		0,7	0,6	0,9	0,7	0,5	0,4	0,8	0,6	0,4	0,7	0,9	0,2	0,7	0,7	0,7

Продовження таблиці В. 1

Операція	Вершина	Варіант														
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Додавання в оберненому коді	1	0,2	0,2	0,1	0,2	0,2	0,6	0,2	0,2	0,1	0,9	0,2	0,2	0,1	0,5	0,3
		0,8	0,8	0,9	0,8	0,8	0,4	0,8	0,8	0,9	0,1	0,8	0,8	0,9	0,5	0,7
	2	0,1	0,2	0,1	0,2	0,1	0,9	0,1	0,2	0,1	0,6	0,1	0,2	0,1	0,5	0,5
		0,9	0,8	0,9	0,8	0,9	0,1	0,9	0,8	0,9	0,4	0,9	0,8	0,9	0,5	0,5
	3	0,1	0,3	0,3	0,1	0,5	0,5	0,5	0,1	0,1	0,5	0,1	0,3	0,3	0,4	0,2
		0,9	0,7	0,7	0,9	0,5	0,5	0,5	0,9	0,9	0,5	0,9	0,7	0,7	0,6	0,8
	4	0,3	0,5	0,6	0,2	0,4	0,6	0,3	0,1	0,8	0,1	0,3	0,5	0,6	0,5	0,1
		0,7	0,5	0,4	0,8	0,6	0,4	0,7	0,9	0,2	0,9	0,7	0,5	0,4	0,5	0,9
	5	0,5	0,5	0,5	0,3	0,3	0,3	0,5	0,3	0,9	0,1	0,5	0,5	0,5	0,3	0,1
		0,5	0,5	0,5	0,7	0,7	0,7	0,5	0,7	0,1	0,9	0,5	0,5	0,5	0,7	0,9
Множення	1	0,2	0,2	0,6	0,3	0,1	0,8	0,3	0,4	0,1	0,3	0,5	0,3	0,9	0,3	0,5
		0,8	0,8	0,4	0,7	0,9	0,2	0,7	0,6	0,9	0,7	0,5	0,7	0,1	0,7	0,5
	2	0,2	0,1	0,9	0,5	0,3	0,9	0,2	0,2	0,1	0,2	0,5	0,5	0,6	0,5	0,5
		0,8	0,9	0,1	0,5	0,7	0,1	0,8	0,8	0,9	0,8	0,5	0,5	0,4	0,5	0,5
Прискорене множення	1	0,5	0,1	0,1	0,1	0,2	0,1	0,5	0,4	0,2	0,3	0,5	0,6	0,2	0,2	0,6
		0,5	0,9	0,9	0,9	0,8	0,9	0,5	0,6	0,8	0,7	0,5	0,4	0,8	0,8	0,4
	2	0,3	0,1	0,8	0,1	0,3	0,3	0,4	0,3	0,8	0,5	0,5	0,5	0,2	0,1	0,9
		0,7	0,9	0,2	0,9	0,7	0,7	0,6	0,7	0,2	0,5	0,5	0,5	0,8	0,9	0,1
	3	0,5	0,3	0,9	0,3	0,5	0,6	0,2	0,2	0,6	0,5	0,4	0,2	0,4	0,5	0,5
		0,5	0,7	0,1	0,7	0,5	0,4	0,8	0,8	0,4	0,5	0,6	0,8	0,6	0,5	0,5

Таблиця В.2

Операція	Вершина	Варіант														
		1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
Прискорене множення	4	0,2	0,2	0,1	0,5	0,5	0,5	0,3	0,3	0,3	0,5	0,3	0,9	0,4	0,2	0,2
		0,8	0,8	0,9	0,5	0,5	0,5	0,7	0,7	0,7	0,5	0,7	0,1	0,6	0,8	0,8
	5	0,1	0,2	0,1	0,5	0,4	0,2	0,4	0,2	0,2	0,5	0,5	0,6	0,2	0,2	0,1
		0,9	0,8	0,9	0,5	0,6	0,8	0,6	0,8	0,8	0,5	0,5	0,4	0,8	0,8	0,9
Ділення цілих чисел	2	0,1	0,3	0,3	0,4	0,3	0,8	0,5	0,1	0,1	0,4	0,5	0,5	0,5	0,1	0,1
		0,9	0,7	0,7	0,6	0,7	0,2	0,5	0,9	0,9	0,6	0,5	0,5	0,5	0,9	0,9
	3	0,3	0,5	0,6	0,2	0,2	0,6	0,3	0,1	0,8	0,3	0,4	0,1	0,3	0,1	0,8
		0,7	0,5	0,4	0,8	0,8	0,4	0,7	0,9	0,2	0,7	0,6	0,9	0,7	0,9	0,2
	4	0,5	0,5	0,5	0,2	0,1	0,9	0,5	0,3	0,9	0,2	0,2	0,1	0,5	0,3	0,9
		0,5	0,5	0,5	0,8	0,9	0,1	0,5	0,7	0,1	0,8	0,8	0,9	0,5	0,7	0,1
Ділення з фіксованою комою	1	0,5	0,4	0,2	0,4	0,5	0,5	0,5	0,5	0,6	0,1	0,2	0,1	0,5	0,5	0,6
		0,5	0,6	0,8	0,6	0,5	0,5	0,5	0,5	0,4	0,9	0,8	0,9	0,5	0,5	0,4
	2	0,4	0,3	0,8	0,3	0,4	0,1	0,4	0,5	0,5	0,1	0,3	0,3	0,4	0,2	0,2
		0,6	0,7	0,2	0,7	0,6	0,9	0,6	0,5	0,5	0,9	0,7	0,7	0,6	0,8	0,8
	3	0,2	0,2	0,6	0,2	0,2	0,1	0,3	0,4	0,1	0,3	0,5	0,6	0,5	0,1	0,1
		0,8	0,8	0,4	0,8	0,8	0,9	0,7	0,6	0,9	0,7	0,5	0,4	0,5	0,9	0,9
	2	0,2	0,1	0,9	0,1	0,2	0,1	0,2	0,2	0,1	0,5	0,5	0,5	0,3	0,1	0,8
		0,8	0,9	0,1	0,9	0,8	0,9	0,8	0,8	0,9	0,5	0,5	0,5	0,7	0,9	0,2

Продовження таблиці В. 2

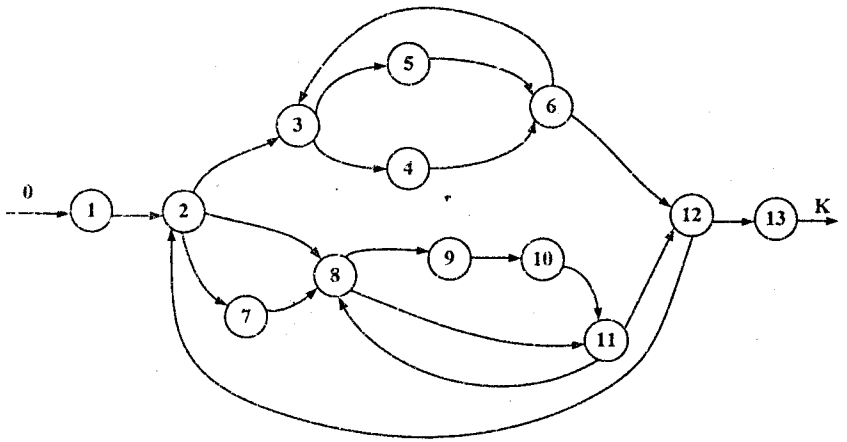
Операція	Вершина	Варіант														
		16	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*
Прискорене множення	4	0,5	0,5	0,6	0,5	0,5	0,5	0,2	0,1	0,9	0,4	0,3	0,8	0,3	0,4	0,1
		0,5	0,5	0,4	0,5	0,5	0,5	0,8	0,9	0,1	0,6	0,7	0,2	0,7	0,6	0,9
	5	0,5	0,5	0,5	0,3	0,3	0,3	0,5	0,3	0,9	0,2	0,2	0,6	0,2	0,2	0,1
		0,5	0,5	0,5	0,7	0,7	0,7	0,5	0,7	0,1	0,8	0,8	0,4	0,8	0,8	0,9
Ділення цілих чисел	2	0,4	0,5	0,5	0,5	0,4	0,2	0,4	0,5	0,5	0,3	0,3	0,5	0,3	0,9	0,3
		0,6	0,5	0,5	0,5	0,6	0,8	0,6	0,5	0,5	0,7	0,7	0,5	0,7	0,1	0,7
	3	0,3	0,4	0,1	0,4	0,3	0,8	0,3	0,4	0,1	0,2	0,2	0,5	0,5	0,6	0,5
		0,7	0,6	0,9	0,6	0,7	0,2	0,7	0,6	0,9	0,8	0,8	0,5	0,5	0,4	0,5
	4	0,2	0,2	0,1	0,2	0,2	0,6	0,2	0,2	0,1	0,2	0,1	0,5	0,5	0,5	0,3
		0,8	0,8	0,9	0,8	0,8	0,4	0,8	0,8	0,9	0,8	0,9	0,5	0,5	0,5	0,7
	1	0,1	0,2	0,1	0,2	0,1	0,9	0,1	0,2	0,1	0,1	0,1	0,4	0,5	0,5	0,5
		0,9	0,8	0,9	0,8	0,9	0,1	0,9	0,8	0,9	0,9	0,9	0,6	0,5	0,5	0,5
Ділення з фіксованою комою	2	0,5	0,5	0,6	0,5	0,5	0,5	0,2	0,1	0,9	0,1	0,8	0,3	0,4	0,1	0,4
		0,5	0,5	0,4	0,5	0,5	0,5	0,8	0,9	0,1	0,9	0,2	0,7	0,6	0,9	0,6
	3	0,4	0,5	0,5	0,5	0,4	0,2	0,4	0,5	0,5	0,3	0,9	0,2	0,2	0,1	0,2
		0,6	0,5	0,5	0,5	0,6	0,8	0,6	0,5	0,5	0,7	0,1	0,8	0,8	0,9	0,8
	2	0,3	0,4	0,1	0,4	0,3	0,8	0,3	0,4	0,1	0,5	0,6	0,1	0,2	0,1	0,2
		0,7	0,6	0,9	0,6	0,7	0,2	0,7	0,6	0,9	0,5	0,4	0,9	0,8	0,9	0,8

Примітка. Ймовірності проставляти з 0-ї до К-ї вершини графу, починаючи з найменшої.

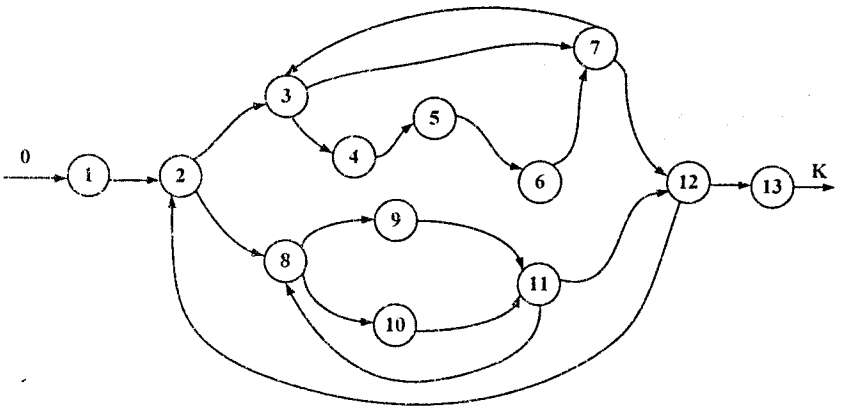
Додаток Г

Варіанти графічних завдань

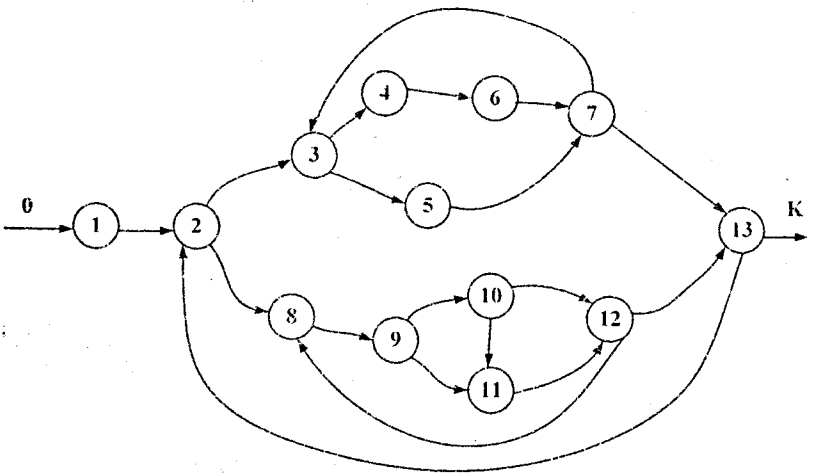
Варіант 1



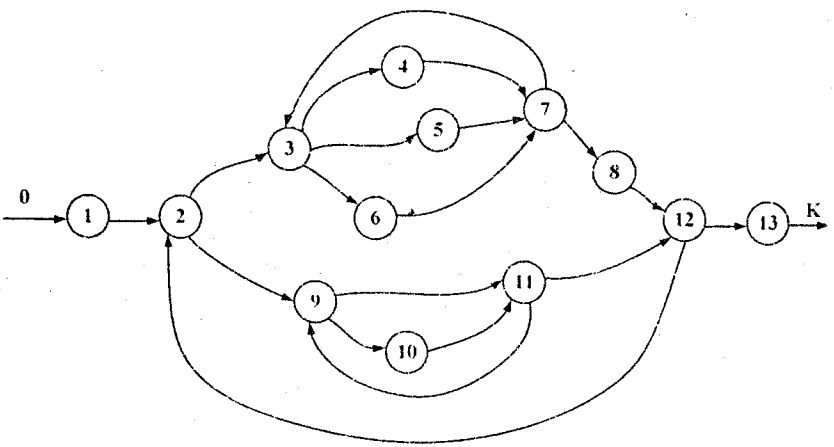
Варіант 2



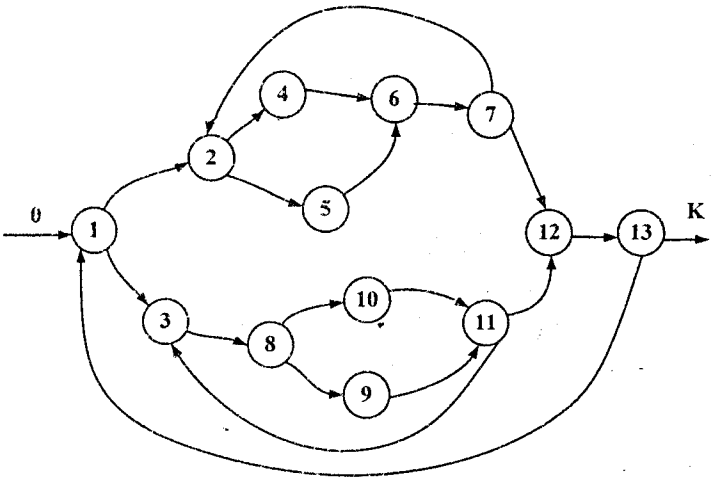
Варіант 3



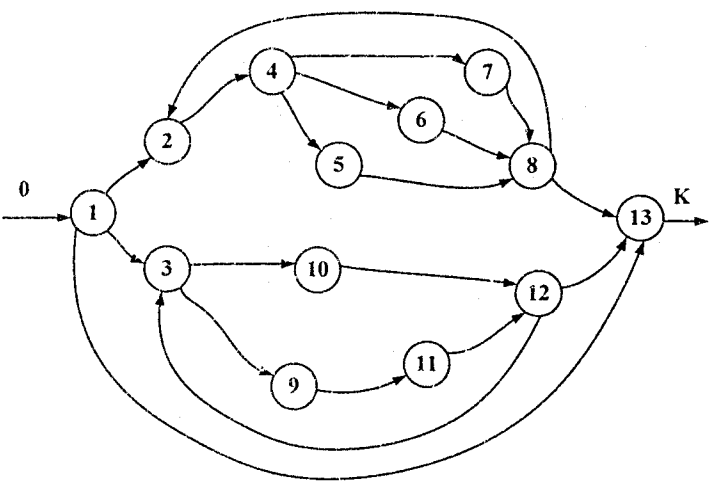
Варіант 4



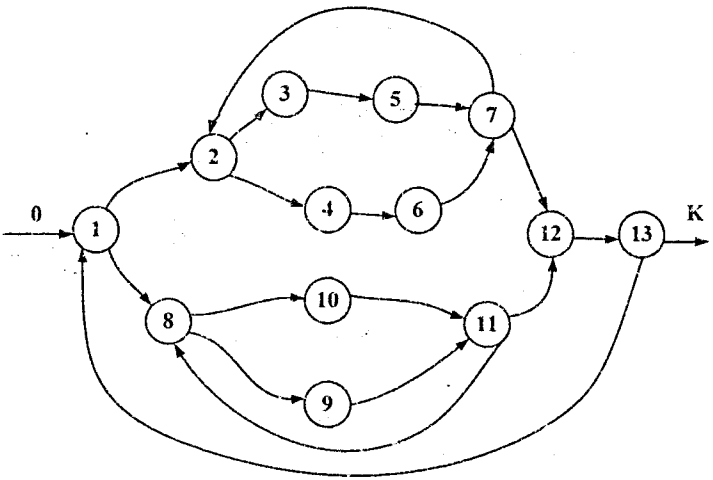
Варіант 5



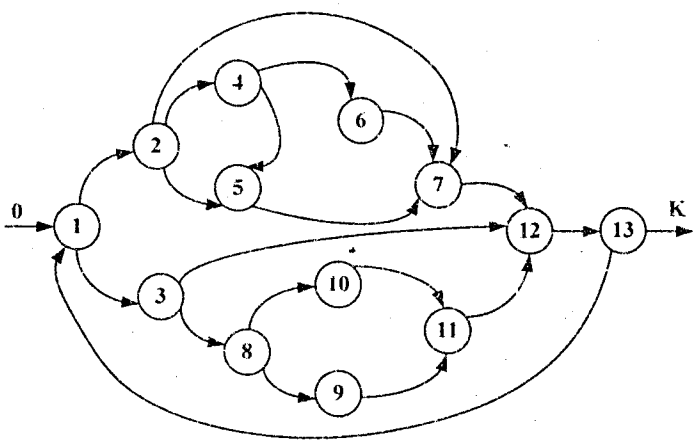
Варіант 6



Варіант 7



Варіант 8



Додаток Д

Матриці імовірностей переходів до варіантів графічних завдань

Варіант 1.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.2	0	0	0	0.3	0.5	0	0	0	0	0
3	0	0	0	0	0.3	0.7	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0.4	0	0	0	0	0	0	0	0	0.6	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0.5	0	0.5	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0.7	0	0	0	0.3	0
12	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0.9
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.4	0	0	0	0.3	0.3	0	0	0	0	0
3	0	0	0	0	0.6	0.4	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0.2	0	0	0	0	0	0	0	0	0.8	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0.2	0	0.8	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0.5	0	0	0	0.5	0
12	0	0	0.3	0	0	0	0	0	0	0	0	0	0	0.7
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.5	0	0	0	0.4	0.1	0	0	0	0	0
3	0	0	0	0	0.2	0.8	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0.7	0	0	0	0	0	0	0	0	0.3	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0.3	0	0.7	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0.2	0	0	0	0.8	0
12	0	0	0.4	0	0	0	0	0	0	0	0	0	0	0.6
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.3	0	0	0	0.5	0.2	0	0	0	0	0
3	0	0	0	0	0.5	0.5	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0.1	0	0	0	0	0	0	0	0	0.9	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0.8	0	0.2	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0.2	0	0	0	0.6	0
12	0	0	0.2	0	0	0	0	0	0	0	0	0	0	0.8
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Варіант 2.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.2	0	0	0	0	0.8	0	0	0	0	0
3	0	0	0	0	0.3	0	0	0.7	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0.5	0	0	0	0	0	0	0	0	0.5	0
8	0	0	0	0	0	0	0	0	0	0.6	0.4	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0.7	0	0	0	0.3	0
12	0	0	0.8	0	0	0	0	0	0	0	0	0	0	0.2
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.3	0	0	0	0	0.7	0	0	0	0	0
3	0	0	0	0	0.4	0	0	0.6	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0.6	0	0	0	0	0	0	0	0	0.4	0
8	0	0	0	0	0	0	0	0	0	0.7	0.3	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0.8	0	0	0	0.2	0
12	0	0	0.9	0	0	0	0	0	0	0	0	0	0	0.1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.4	0	0	0	0	0.6	0	0	0	0	0
3	0	0	0	0	0.5	0	0	0.5	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0.7	0	0	0	0	0	0	0	0	0.3	0
8	0	0	0	0	0	0	0	0	0	0.8	0.2	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0.9	0	0	0	0.1	0
12	0	0	0.3	0	0	0	0	0	0	0	0	0	0	0.7
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.5	0	0	0	0	0.5	0	0	0	0	0
3	0	0	0	0	0.6	0	0	0.4	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0.2	0	0	0	0	0	0	0	0	0.8	0
8	0	0	0	0	0	0	0	0	0	0.1	0.9	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0.5	0	0	0	0.5	0
12	0	0	0.4	0	0	0	0	0	0	0	0	0	0	0.6
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Варіант 3.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.1	0	0	0	0	0.9	0	0	0	0	0
3	0	0	0	0	0.2	0.8	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0.3	0	0	0	0	0	0	0	0	0	0.7
8	0	0	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0.4	0.6	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0.6	0	0	0	0	0.4
13	0	0	0.9	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.2	0	0	0	0	0.8	0	0	0	0	0
3	0	0	0	0	0.3	0.7	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0.4	0	0	0	0	0	0	0	0	0	0.6
8	0	0	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0.6	0.4	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0.7	0	0	0	0	0.3
13	0	0	0.8	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.3	0	0	0	0	0.7	0	0	0	0	0
3	0	0	0	0	0.4	0.6	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0.5
8	0	0	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0.6	0.4	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0.3	0.7	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0.2	0	0	0	0	0.8
13	0	0	0.7	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.4	0	0	0	0	0.6	0	0	0	0	0
3	0	0	0	0	0.5	0.5	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0.2	0	0	0	0	0	0	0	0	0	0.8
8	0	0	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0.3	0.7	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0.2	0.8	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0.9
13	0	0	0.6	0	0	0	0	0	0	0	0	0	0	0

Варіант 4.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.1	0	0	0	0	0	0.9	0	0	0	0
3	0	0	0	0	0.2	0.4	0.4	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0.2	0	0	0	0	0.8	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	0.3	0.7	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0.4	0	0	0.6	0
12	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0.5
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.2	0	0	0	0	0	0.9	0	0	0	0
3	0	0	0	0	0.3	0.4	0.3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0.3	0	0	0	0	0.7	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	0.4	0.6	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0.5	0	0	0.5	0
12	0	0	0.4	0	0	0	0	0	0	0	0	0	0	0.6
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.3	0	0	0	0	0	0.9	0	0	0	0
3	0	0	0	0	0.4	0.5	0.1	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0.4	0	0	0	0	0.6	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0.3	0	0	0.7	0
12	0	0	0.2	0	0	0	0	0	0	0	0	0	0	0.8
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.4	0	0	0	0	0	0.6	0	0	0	0
3	0	0	0	0	0.5	0.1	0.4	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	0.2	0.8	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0.1	0	0	0.9	0
12	0	0	0.3	0	0	0	0	0	0	0	0	0	0	0.7
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Варіант 5.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.2	0.8	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0.3	0.7	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0.4	0	0	0	0	0	0	0	0	0	0.6	0
8	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0.6	0	0	0	0	0	0	0	0	0.4	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.3	0.7	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0.4	0.6	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0.5	0	0	0	0	0	0	0	0	0	0.5	0
8	0	0	0	0	0	0	0	0	0	0.6	0.4	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0.7	0	0	0	0	0	0	0	0	0.3	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.4	0.6	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0.5	0.5	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0.6	0	0	0	0	0	0	0	0	0	0.4	0
8	0	0	0	0	0	0	0	0	0	0.7	0.3	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0.8	0	0	0	0	0	0	0	0	0.2	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0.9	0	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.5	0.5	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0.6	0.4	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0.7	0	0	0	0	0	0	0	0	0	0.3	0
8	0	0	0	0	0	0	0	0	0	0.8	0.2	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0.8	0	0	0	0	0	0	0	0	0.1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0

Варіант 6.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.1	0.9	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0.2	0.8	0	0	0
4	0	0	0	0	0	0.3	0.3	0.4	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0.5
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0.4	0	0	0	0	0	0	0	0	0	0.6
13	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.2	0.8	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0.3	0.7	0	0	0
4	0	0	0	0	0	0.4	0.4	0.2	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0.8	0	0	0	0	0	0	0	0	0	0	0.2
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0.5
13	0	0.6	0	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.3	0.7	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0.4	0.6	0	0	0
4	0	0	0	0	0	0.5	0.3	0.2	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0.6	0	0	0	0	0	0	0	0	0	0	0.4
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0.7	0	0	0	0	0	0	0	0	0	0.3
13	0	0.8	0	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.4	0.6	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0	0
4	0	0	0	0	0	0.6	0.2	0.2	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0.7	0	0	0	0	0	0	0	0	0	0	0.3
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0.8	0	0	0	0	0	0	0	0	0	0.2
13	0	0.9	0	0	0	0	0	0	0	0	0	0	0	0

Варіант 7.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.6	0	0	0	0	0	0.4	0	0	0	0	0
2	0	0	0	0.4	0.6	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0.8	0	0	0	0	0	0	0	0	0	0.2	0
8	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0.7	0	0	0	0.3	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0.3	0	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.3	0	0	0	0	0	0.7	0	0	0	0	0
2	0	0	0	0.8	0.2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0.5	0	0	0	0	0	0	0	0	0	0.5	0
8	0	0	0	0	0	0	0	0	0	0.1	0.9	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0.4	0	0	0	0.6	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.8	0	0	0	0	0	0.2	0	0	0	0	0
2	0	0	0	0.6	0.4	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0.3	0	0	0	0	0	0	0	0	0	0.7	0
8	0	0	0	0	0	0	0	0	0	0.1	0.9	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0.3	0	0	0	0.7	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0.4	0	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.5	0	0	0	0	0	0.5	0	0	0	0	0
2	0	0	0	0.1	0.9	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0.6	0	0	0	0	0	0	0	0	0	0.4	0
8	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0.9	0	0	0	0.1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0.8	0	0	0	0	0	0	0	0	0	0	0	0

Варіант 8.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.1	0.9	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0.2	0.8	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0.7	0	0	0	0.3	0
4	0	0	0	0	0	0.6	0.4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.5	0.5	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0.6	0.4	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0.7	0	0	0	0.3	0
4	0	0	0	0	0	0.2	0.8	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	0	0	0.1	0.9	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0.3	0	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.2	0.8	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0.3	0.7	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0.5	0	0	0	0.5	0
4	0	0	0	0	0	0.6	0.4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	0	0	0.1	0.9	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.3	0.7	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0.5	0.5	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0.2	0	0	0	0.8	0
4	0	0	0	0	0	0.1	0.9	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	0	0	0.6	0.4	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0.6	0	0	0	0	0	0	0	0	0	0	0	0

Додаток Е

Таблиця Е. 1 – Параметри графу марковського ланцюга

Варіанти завдань	Імовірність переходу вперед, r	Імовірність природнього прямування мікрокоманд, q	Середня довжина переходу, L	Кількість звернень до ПЗП на одну мікрокоманду, k
Варіант 1: А	0,5	0,4	3	0,4
Б	0,5	0,5	5	0,6
В	0,5	0,6	7	0,8
Г	0,5	0,7	11	0,4
Варіант 2: А	0,5	0,4	5	0,4
Б	0,5	0,5	7	0,6
В	0,5	0,6	11	0,8
Г	0,5	0,7	3	0,6
Варіант 3: А	0,5	0,4	7	0,4
Б	0,5	0,5	11	0,6
В	0,5	0,6	3	0,8
Г	0,5	0,7	5	0,4
Варіант 4: А	0,5	0,4	11	0,4
Б	0,5	0,5	3	0,6
В	0,5	0,6	5	0,8
Г	0,5	0,7	7	0,6
Варіант 5: А	0,5	0,4	3	0,6
Б	0,5	0,5	5	0,4
В	0,5	0,6	7	0,4
Г	0,5	0,7	11	0,8
Варіант 6: А	0,5	0,4	5	0,8
Б	0,5	0,5	7	0,4
В	0,5	0,6	11	0,6
Г	0,5	0,7	3	0,4
Варіант 7: А	0,5	0,4	7	0,6
Б	0,5	0,5	11	0,4
В	0,5	0,8	3	0,5
Г	0,5	0,7	5	0,7
Варіант 8: А	0,5	0,4	11	0,5
Б	0,5	0,6	3	0,6
В	0,5	0,7	5	0,4
Г	0,5	0,8	7	0,7

Додаток Ж

Матриці вартості

Варіант 1.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	2	0	0	0	3	5	0	0	0	0	0
3	0	0	0	0	3	7	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	4	0	0	0	0	0	0	0	0	6	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	5	0	5	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	7	0	0	0	3	0
12	0	0	1	0	0	0	0	0	0	0	0	0	0	9
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	4	0	0	0	3	3	0	0	0	0	0
3	0	0	0	0	6	4	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	2	0	0	0	0	0	0	0	0	8	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	2	0	8	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	5	0	0	0	5	0
12	0	0	3	0	0	0	0	0	0	0	0	0	0	7
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	5	0	0	0	4	1	0	0	0	0	0
3	0	0	0	0	2	8	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	7	0	0	0	0	0	0	0	0	3	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	3	0	7	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	2	0	0	0	8	0
12	0	0	4	0	0	0	0	0	0	0	0	0	0	6
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	3	0	0	0	5	2	0	0	0	0	0
3	0	0	0	0	5	5	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	1	0	0	0	0	0	0	0	0	9	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	8	0	2	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	2	0	0	0	6	0
12	0	0	2	0	0	0	0	0	0	0	0	0	0	8
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Варіант 2.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	2	0	0	0	0	8	0	0	0	0	0
3	0	0	0	0	3	0	0	7	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	5	0	0	0	0	0	0	0	0	5	0
8	0	0	0	0	0	0	0	0	0	6	4	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	7	0	0	0	3	0
12	0	0	8	0	0	0	0	0	0	0	0	0	0	2
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	3	0	0	0	0	7	0	0	0	0	0
3	0	0	0	0	4	0	0	6	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	6	0	0	0	0	0	0	0	0	4	0
8	0	0	0	0	0	0	0	0	0	7	3	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	8	0	0	0	2	0
12	0	0	9	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	4	0	0	0	0	6	0	0	0	0	0
3	0	0	0	0	5	0	0	5	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	7	0	0	0	0	0	0	0	0	3	0
8	0	0	0	0	0	0	0	0	0	8	2	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	9	0	0	0	1	0
12	0	0	3	0	0	0	0	0	0	0	0	0	0	7
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	5	0	0	0	0	5	0	0	0	0	0
3	0	0	0	0	6	0	0	4	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	2	0	0	0	0	0	0	0	0	8	0
8	0	0	0	0	0	0	0	0	0	1	9	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	5	0	0	0	5	0
12	0	0	4	0	0	0	0	0	0	0	0	0	0	6
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Варіант 3.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	9	0	0	0	0	0
3	0	0	0	0	2	8	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	3	0	0	0	0	0	0	0	0	0	7
8	0	0	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	4	6	0	0
10	0	0	0	0	0	0	0	0	0	0	0	5	5	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	6	0	0	0	0	4
13	0	0	9	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	2	0	0	0	0	8	0	0	0	0	0
3	0	0	0	0	3	7	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	4	0	0	0	0	0	0	0	0	0	6
8	0	0	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	5	5	0	0
10	0	0	0	0	0	0	0	0	0	0	0	6	4	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	7	0	0	0	0	3
13	0	0	8	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	3	0	0	0	0	7	0	0	0	0	0
3	0	0	0	0	4	6	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	5	0	0	0	0	0	0	0	0	0	5
8	0	0	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	6	4	0	0
10	0	0	0	0	0	0	0	0	0	0	0	3	7	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	2	0	0	0	0	8
13	0	0	7	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	4	0	0	0	0	6	0	0	0	0	0
3	0	0	0	0	5	5	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	2	0	0	0	0	0	0	0	0	0	8
8	0	0	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	3	7	0	0
10	0	0	0	0	0	0	0	0	0	0	0	2	8	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	1	0	0	0	0	9
13	0	0	6	0	0	0	0	0	0	0	0	0	0	0

Варіант 4.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0	9	0	0	0	0
3	0	0	0	0	2	4	4	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	2	0	0	0	0	8	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	3	7	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	4	0	0	6	0
12	0	0	5	0	0	0	0	0	0	0	0	0	0	5
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	2	0	0	0	0	0	9	0	0	0	0
3	0	0	0	0	3	4	3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	3	0	0	0	0	7	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	4	6	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	5	0	0	5	0
12	0	0	4	0	0	0	0	0	0	0	0	0	0	6
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	3	0	0	0	0	0	9	0	0	0	0
3	0	0	0	0	4	5	1	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	4	0	0	0	0	6	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	5	5	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	3	0	0	7	0
12	0	0	2	0	0	0	0	0	0	0	0	0	0	8
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	4	0	0	0	0	0	6	0	0	0	0
3	0	0	0	0	5	1	4	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	5	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	2	8	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	1	0	0	9	0
12	0	0	3	0	0	0	0	0	0	0	0	0	0	7
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Варіант 5.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	2	8	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	3	7	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	4	0	0	0	0	0	0	0	0	0	6	0
8	0	0	0	0	0	0	0	0	0	5	5	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	6	0	0	0	0	0	0	0	0	4	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	7	0	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	7	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	4	6	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	5	0	0	0	0	0	0	0	0	0	5	0
8	0	0	0	0	0	0	0	0	0	6	4	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	7	0	0	0	0	0	0	0	0	3	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	7	0	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	4	6	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	5	5	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	6	0	0	0	0	0	0	0	0	0	4	0
8	0	0	0	0	0	0	0	0	0	7	3	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	8	0	0	0	0	0	0	0	0	2	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	9	0	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	6	4	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	7	0	0	0	0	0	0	0	0	0	3	0
8	0	0	0	0	0	0	0	0	0	8	2	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	8	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	2	0	0	0	0	0	0	0	0	0	0	0	0

Варіант 6.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	9	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	2	8	0	0	0
4	0	0	0	0	0	3	3	4	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	5	0	0	0	0	0	0	0	0	0	0	5
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	4	0	0	0	0	0	0	0	0	0	6
13	0	5	0	0	0	0	0	0	0	0	0	0	0	0

Б.

P=

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	2	8	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	3	7	0	0	0	0
4	0	0	0	0	0	4	4	2	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	8	0	0	0	0	0	0	0	0	0	0	2
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	5	0	0	0	0	0	0	0	0	0	5
13	0	6	0	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	7	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	4	6	0	0	0
4	0	0	0	0	0	5	3	2	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	6	0	0	0	0	0	0	0	0	0	0	4
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	7	0	0	0	0	0	0	0	0	0	3
13	0	8	0	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	4	6	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	5	5	0	0	0
4	0	0	0	0	0	6	2	2	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	7	0	0	0	0	0	0	0	0	0	0	3
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	8	0	0	0	0	0	0	0	0	0	2
13	0	9	0	0	0	0	0	0	0	0	0	0	0	0

Варіант 7.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	6	0	0	0	0	0	4	0	0	0	0	0
2	0	0	0	4	6	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	8	0	0	0	0	0	0	0	0	0	2	0
8	0	0	0	0	0	0	0	0	0	5	5	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	7	0	0	0	3	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	0		0	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	0	0	0	0	0	7	0	0	0	0	0
2	0	0	0	8	2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	5	0	0	0	0	0	0	0	0	0	5	0
8	0	0	0	0	0	0	0	0	0	1	9	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	4	0	0	0	6	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	5	0	0	0	0	0	0	0	0	0	0	0	0

В.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	8	0	0	0	0	0	2	0	0	0	0	0
2	0	0	0	6	4	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	3	0	0	0	0	0	0	0	0	0	7	0
8	0	0	0	0	0	0	0	0	0	1	9	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	3	0	0	0	7	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	4	0	0	0	0	0	0	0	0	0	0	0	0

Г.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	0	0	0	0	0	5	0	0	0	0	0
2	0	0	0	1	9	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	6	0	0	0	0	0	0	0	0	0	4	0
8	0	0	0	0	0	0	0	0	0	5	5	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	9	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	8	0	0	0	0	0	0	0	0	0	0	0	0

Варіант 8.

А.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	2	8	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	3	7	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	5	0	0	0	5	0
4	0	0	0	0	0	6	4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	0	0	1	9	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	7	0	0	0	0	0	0	0	0	0	0	0	0

Б.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	6	4	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	7	0	0	0	3	0
4	0	0	0	0	0	2	8	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	0	0	1	9	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	5	0	0	0	0	0	0	0	0	0	0	0	0

B.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	2	8	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	3	7	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	5	0	0	0	5	0
4	0	0	0	0	0	6	4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	0	0	1	9	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	5	0	0	0	0	0	0	0	0	0	0	0	0

Γ.

P =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	7	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	5	5	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	2	0	0	0	8	0
4	0	0	0	0	0	1	9	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	0	0	6	4	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	6	0	0	0	0	0	0	0	0	0	0	0	0

Література

1. Майоров С. А., Новиков Г. И. Структуры электронных вычислительных машин.-Л.: Машиностроение, 1979. – 384 с .
2. Гмурман В. Е. Теория вероятностей и математическая статистика: Учебное пособие для вузов. – М.: Высшая школа, 1972. – 368 с.
3. Основы теории вычислительных систем: Учебное пособие для вузов/Под ред. С. А. Майорова. – Высшая школа, 1978. – 408 с.
4. Кемени Дж., Снелл Дж, Кнепп А. Счетные цепи Маркова: Пер. с англ. – М.: Наука, 1987. – 413 с.
5. Майоров С. А., Новиков Г. И. Принципы организации цифровых машин. – Л.: Машиностроение, 1974. – 432 с.
6. Ларионов А. М., Майоров С. А., Новиков Г. И. Вычислительные комп-лексы, системы и сети: Учебник для вузов. – Л.: Энергоатомиздат, 1987. – 288 с.
7. Норенков И. П., Маничев В. Б., Системы автоматизированного проектирования электронной и вычислительной аппаратуры: Учеб. пособие для вузов. – М.: Высшая школа, 1983. – 272 с.
8. www.smolensk.ru. Мунерман В. И. Параллельная обработка данных. Методы и средства.
9. Феррари Д. Оценка производительности вычислительных систем: Пер. с англ. – М.: Мир, 1981. – 576 с.

Навчальне видання

Тетяна Борисівна Мартинюк

Наталія Іванівна Заболотна

СИСТЕМОТЕХНІКА ОПТОЕЛЕКТРОННИХ ТА ЛАЗЕРНИХ СИСТЕМ ЛАБОРАТОРНИЙ ПРАКТИКУМ

Оригінал-макет підготовлено Т. Б. Мартинюк

Редактор О. Д. Скалоцька

Науково-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021 м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку 28.01.2009 р.
Формат 29.7×42 ¼
Друк різографічний
Тираж 85 прим.
Зам. № 2009-022

Гарнітура Times New Roman
Папір офсетний
Ум. друк. арк. 7.5

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