РОЗДІЛ 2. Теорія чисел і криптографія

Тема 6. Застосування конгруенцій. Класична криптографія

План лекції

- **Застосування конгруенцій**
  - **Геш-функції**
  - **Генерування псевдовипадкових чисел**
  - **Контрольні розряди**
- **Класична криптографія**
  - **Класифікація шифросистем**
  - **Шифри перестановки**

Застосування конгруенцій

У цій і наступній лекціях ми розглянемо важливі застосування модулярної арифметики, такі як призначення комп'ютерної пам'яті для розміщення файлів, генерування псевдовипадкових чисел, знаходження контрольних розрядів, шифрування повідомлень.

Геш-функції

Це додаткова інформація

Геш-функція (або ж *хеш-функція*) – функція, що перетворює вхідні дані будь-якого (як правило великого) розміру в дані фіксованого розміру. *Гешування* (або ж хешування, англ. *hashing*) — перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини. Такі перетворення також називають *геш-функціями* або *функціями* **з***гортки*, а їх результати називають гешем, *геш-кодом*, *геш-сумою*, або *дайджестом повідомлення* (англ. *messages digest*).

Геш-функція використовується зокрема у структурах даних – геш-таблицях, широко використовується у програмному забезпеченні для швидкого пошуку даних. Геш-функції використовуються для оптимізації таблиць та баз даних за рахунок того, що в однакових записів однакові значення геш-функції. Такий підхід пошуку дублікатів ефективний у файлах великого розміру. Прикладом цього буде знаходження подібних ділянок у послідовностях ДНК.

Геш-таблиця – це структура даних, що дозволяє зберігати пари виду (ключ, геш-код) і підтримує операції пошуку, вставки та видалення елементів. Завданням геш-таблиць є прискорення пошуку, наприклад, у випадку записів до текстових полів в базі даних може розраховуватися їх геш-код і дані можуть поміщатися у розділ, що відповідає цьому геш-коду. Тоді при пошуку даних треба буде спочатку обчислити геш-код тексту і відразу стане відомо, в якому розділі їх треба шукати, тобто, шукати треба буде не по всій базі, а тільки по одному її розділу (це сильно прискорює пошук).

Побутовим аналогом гешування в даному випадку може служити розміщення слів у словнику за алфавітом. Перша літера слова є його геш-кодом, і при пошуку ми переглядаємо не весь словник, а тільки потрібну літеру.

Словник: Memory location = адреса пам’яті, комірка пам’яті, адреса комірки пам’яті.

Розглянемо таку ситуацію. Центральний комп'ютер у страховій компанії веде облік для кожного зі своїх клієнтів. Як потрібно призначити адреси пам'яті із записами облікової інформації клієнтів так, щоб ці записи можна було швидко отримати? Вирішення цієї проблеми полягає у використанні відповідним чином підібраної геш-функції (функції гешування). Облікову інформацію про клієнта однозначно ідентифікує ключ. Наприклад, у США як ключ часто використовують номер соціального забезпечення клієнта. Геш-функція h присвоює адресу пам'яті $h(k)$ до запису, що має k у якості свого ключа. На практиці використовують багато різних геш-функцій. Однією з найбільш поширених є функція

$$h(k) = k \bmod m,$$

де m – кількість доступних адрес пам'яті.

Геш-функції мають бути такими, що їх легко обчислити, щоб файли можна було швидко знайти. Функція $h(k) = k \bmod m$ задовольняє цю вимогу: щоб знайти $h(k)$, нам потрібно тільки обчислити остачу від ділення k на m. Крім того, геш-функція має бути сюр'єктивною, щоб всі адреси пам'яті були доступні. Функція $h(k) = k \bmod m$ задовольняє цю властивість також.

Приклад. Знайти адреси пам'яті, присвоєні геш-функцією $h(k) = k \bmod 111$ для записів з обліковою інформацією клієнтів, у яких номери соціального забезпечення 064212848 і 037149212.

Запис з обліковою інформацією клієнта з номером соціального забезпечення 064212848 дістає адресу пам'яті 14, бо $h(064212848) = 064212848 \bmod 111 = 14$, а для клієнта з номером 037149212 адреса пам'яті для запису з обліковою інформацією є 65, бо

$$h(037149212) = 037149212 \bmod 111 = 65.$$

Оскільки геш-функція не є взаємно-однозначною (тому що є більше можливих ключів ніж областей пам'яті), більш ніж один файл може бути призначений на якусь область пам'яті. Коли це трапляється, ми говоримо, що відбулася **колізія**. Один із способів розв'язання колізії полягає в присвоєнні першої вільної адреси, що слідує за зайнятою адресою пам'яті, визначеної геш-функцією.

Приклад. Після надання адрес записам у попередньому прикладі, присвоїти адресу пам'яті для клієнта з номером соціального забезпечення 107405723. Перш за все зауважимо, що геш-функція $h(k) = k \bmod m$ відображає номер 107405723 в адресу 14, бо

$$h(107405723) = 107405723 \bmod 111 = 14.$$

Проте ця адреса пам'яті зайнята клієнтом з номером соціального забезпечення 064212848. Оскільки адреса пам'яті 15, наступна за адресою 14, вільна, то присвоюємо цю адресу пам'яті клієнту з номером 107405723.

В останньому прикладі ми використали *лінійну пробну функцію* $h(k,i) = h(k) + i \bmod m$ для знаходження першої вільної адреси пам'яті, де i пробігає від 0 до $m-1$. Зазначимо, що існує багато інших способів розв'язання колізій.

Генерування псевдовипадкових чисел

Вибір випадкових чисел часто необхідний у комп'ютерному моделюванні. Нині винайдено різні методи, які випадково вибирають числа. Оскільки числа, генеровані систематичними методами, насправді не є реально випадковими, їх називають *псевдовипадковими числами*.

У більшості випадків для генерування псевдовипадкових чисел використовують **метод лінійних конгруенцій**. У цьому методі вибирають чотири цілих числа: **модуль** m, **коефіцієнт** a, **крок (приріст, інкремент)** c і **початкове значення** x_0, де $2 \le a < m$, $0 \le c < m$ і $0 \le x_0 < m$. Послідовність псевдовипадкових чисел $\{x_n\}$, $0 \le x_n < m$ для всіх n, генерують послідовним обчислення рекурсивно визначеної функції

$$x_{n+1} = (ax_n + c) \bmod m.$$

Багато комп'ютерних експериментів потребують генерування псевдовипадкових чисел між 0 та 1. Для генерування таких чисел потрібно поділити числа, генеровані лінійним конгруентним методом, на модуль; ми матимемо числа x_n / m.

Приклад. Знайдемо послідовність псевдовипадкових чисел, генерованих лінійним конгруентним методом, з модулем $m = 9$, коефіцієнтом $a = 7$, кроком $c = 4$ і початковим значенням $x_0 = 3$. Обчислимо члени цієї послідовності, послідовно використовуючи рекурсивно визначену функцію $x_{n+1} = (7x_n + 4) \mathbf{mod} 9$, починаючи з $x_0 = 3$:

$$x_1 = (7x_0 + 4) \mathbf{mod} 9 = (7 \cdot 3 + 4) \mathbf{mod} 9 = 25 \mathbf{mod} 9 = 7;$$
$$x_2 = (7x_1 + 4) \mathbf{mod} 9 = (7 \cdot 7 + 4) \mathbf{mod} 9 = 53 \mathbf{mod} 9 = 8;$$
$$x_3 = (7x_2 + 4) \mathbf{mod} 9 = (7 \cdot 8 + 4) \mathbf{mod} 9 = 60 \mathbf{mod} 9 = 6;$$
$$x_4 = (7x_3 + 4) \mathbf{mod} 9 = (7 \cdot 6 + 4) \mathbf{mod} 9 = 46 \mathbf{mod} 9 = 1;$$
$$x_5 = (7x_4 + 4) \mathbf{mod} 9 = (7 \cdot 1 + 4) \mathbf{mod} 9 = 11 \mathbf{mod} 9 = 2;$$
$$x_6 = (7x_5 + 4) \mathbf{mod} 9 = (7 \cdot 2 + 4) \mathbf{mod} 9 = 18 \mathbf{mod} 9 = 0;$$
$$x_7 = (7x_6 + 4) \mathbf{mod} 9 = (7 \cdot 0 + 4) \mathbf{mod} 9 = 4 \mathbf{mod} 9 = 4;$$
$$x_8 = (7x_7 + 4) \mathbf{mod} 9 = (7 \cdot 4 + 4) \mathbf{mod} 9 = 32 \mathbf{mod} 9 = 5;$$
$$x_9 = (7x_8 + 4) \mathbf{mod} 9 = (7 \cdot 5 + 4) \mathbf{mod} 9 = 39 \mathbf{mod} 9 = 3.$$

Тому що $x_9 = x_0$, а кожний член послідовності визначається тільки попереднім членом, то потрібну послідовність знайдено:

3, 7, 8, 6, 1, 2, 0, 4, 5, 3, 7, 8, 6, 1, 2, 0, 4, 5, 3, …

Ця послідовність перед повторенням містить дев’ять різних членів.

Більшість комп'ютерів використовують лінійний конгруентний метод для генерування псевдовипадкових чисел. Часто використовують генератор зі значенням кроку $c = 0$; такий генератор називають *строгим мультиплікативним генератором*. Наприклад, широко застосовують строгий мультиплікативний генератор з модулем $m = 2^{31} - 1$ і коефіцієнтом $a = 7^5 = 16807$. Доведено, що з такими значеннями $2^{31} - 2$ числа генеруються перед початком повторення послідовності. Псевдовипадкові числа, одержані описаним способом, застосовують при розв'язуванні різних задач. Нажаль, можна показати, що вони не задовольняють деякі важливі статистичні властивості, яким задовольняють справжні випадкові числа. Тому вони не завжди можуть бути застосовані. У таких випадках використовують інші методи для генерування псевдовипадкових чисел.

Контрольні розряди

Конгруенції використовують для перевірки помилок у цифрових рядках. Загальна техніка для визначення помилок у таких рядках полягає в додаванні контрольного розряду в кінці рядка. Цей контрольний розряд обчислюють, використовуючи конкретну функцію. Тоді щоб визначити, чи цифровий рядок коректний, треба перевірити значення контрольного розряду. Ми почнемо з аплікації цієї ідеї для перевірки коректності бітового рядка.

Приклад. Контрольний розряд. Цифрова інформація подається у вигляді бітових рядків, поділених на блоки заданого розміру. Перед тим, як кожний блок буде збережено або передано, додатковий біт, який називають *контрольним бітом*, приєднують у кінець кожного блоку. Контрольний біт x_{n+1} для бітового рядка $x_1 x_2 \ldots x_n$ визначають так

$$x_{n+1} = x_1 + x_2 + \ldots + x_n \bmod 2.$$

Звідси випливає, що біт x_{n+1} дорівнює 0, якщо в блоці з n бітів парна кількість 1, і дорівнює 1, коли в блоці з n бітів непарна кількість 1. При перевірці рядка, який містить контрольний біт, помилка фіксується, коли контрольний біт неправильний. Проте, якщо контрольний біт правильний, помилка все ж таки може бути. Річ у тім, що контрольний біт виявляє непарну кількість помилок у попередніх n бітах, але не може виявити парну кількість помилок.

Нехай ми отримали такі бітові рядки: 01100101 та 11010110, кожний з яких закінчується контрольним бітом. Чи можемо ми сприйняти ці рядки як правильні? Для відповіді на це питання перевіримо контрольні біти цих рядків. Контрольний біт першого рядка дорівнює 1. Оскільки $0+1+1+0+0+1+0 \equiv 1 (\mathrm{mod}\, 2)$, то контрольний біт коректний. Контрольний біт другого рядка дорівнює 0. Обчислюємо $1+1+0+1+0+1+1 \equiv 1 (\mathrm{mod}\, 2)$, отже, контрольний біт некоректний. Робимо висновок, що перший рядок, можливо, переданий коректно, а другий рядок переданий некоректно. Ми приймаємо перший рядок як коректний (незважаючи на те, що він може містити парну кількість помилок), а другий рядок відкидаємо.

Приклад. UPC. Товари для продажу ідентифікуються їхніми кодами UPC (Universal Product Code). Поширена форма UPC має 12 десяткових розрядів. Перший визначає категорію товару, наступні п'ять – виробника, ще п'ять наступних – індивідуальний продукт, і останній розряд – контрольний. Цей контрольний розряд визначають такою конгруенцією

$$3x_1 + x_2 + 3x_3 + x_4 + 3x_5 + x_6 + 3x_7 + x_8 + 3x_9 + x_{10} + 3x_{11} + x_{12} \equiv 0 \pmod{10}.$$

Розглянемо два питання.

1. Нехай перші 11 розрядів UPC такі: 79357343104. Знайти контрольний розряд.

2. Задано код 041331021641. Чи цей код є правильним UPC?

Дамо відповіді.

1. Запишемо розряди коду 79357343104 у конгруенцію для визначення контрольного розряду UPC. Це дасть

$$3 \cdot 7 + 9 + 3 \cdot 3 + 5 + 3 \cdot 7 + 3 + 3 \cdot 4 + 3 + 3 \cdot 1 + 0 + 3 \cdot 4 + x_{12} \equiv 0 \pmod{10},$$

звідки $98 + x_{12} \equiv 0 \pmod{10}$. Звідси випливає, що $x_{12} \equiv 2 \pmod{10}$. Отже, контрольний розряд дорівнює 2.

2. Для перевірки чи код 041331021641 правильний, запишемо його розряди в конгруенцію, яку він має задовольняти. Тоді

$$3 \cdot 0 + 4 + 3 \cdot 1 + 3 + 3 \cdot 3 + 1 + 3 \cdot 0 + 2 + 3 \cdot 1 + 6 + 3 \cdot 4 + 1 \equiv 4 \not\equiv 0 \pmod{10}.$$

Отже, 041331021641 – некоректний UPC.

Приклад. ISBN. Нині всі книги мають код **International Standard Book Number (ISBN-10)**, це десятибітовий код $x_1x_2\ldots x_{10}$. Код ISBN-10 містить блоки ідентифікації мови, видавця, номер книги присвоєний її видавничою компанією, і, нарешті, контрольний розряд, котрий являє собою десяткову цифру або букву X (у разі значення 10). Контрольний розряд розраховують так:

$x_{10} \equiv \sum_{i=1}^{9} ix_i \pmod{11}$, або в еквівалентному варіанті $\sum_{i=1}^{10} ix_i \equiv 0 \pmod{11}$.

Знайдемо тепер відповіді на питання щодо коду ISBN-10.

1. Перші дев'ять розрядів коду ISBN-10 підручника Ю. Нікольський, В. Пасічник, Ю. Щербина «Дискретна математика», видавнича група BHV, Київ, 2007, такі: 966-552-201-. Знайти контрольний розряд.

2. Чи код 084-930-149-X є коректним кодом ISBN-10?

Дамо відповіді.

1. Контрольний розряд визначається конгруенцією $\sum_{i=1}^{10} ix_i \equiv 0 \pmod{11}$. Записавши цю конгруенцію для перших дев'яти розрядів 966552201, дістанемо

$$x_{10} \equiv 1\cdot 9 + 2\cdot 6 + 3\cdot 6 + 4\cdot 5 + 5\cdot 5 + 6\cdot 2 + 7\cdot 2 + 8\cdot 0 + 9\cdot 1 \pmod{11},$$

звідки $x_{10} \equiv 119 \equiv 9 \pmod{11}$. Отже, $x_{10} = 9$, тобто код ISBN-10 для згаданого підручника є 966-552-201-9.

2. Для вияснення, чи є 084-930-149-X коректним, перевіримо, чи $\sum_{i=1}^{10} ix_i \equiv 0 \pmod{11}$. Обчислимо

$$1\cdot 0 + 2\cdot 8 + 3\cdot 4 + 4\cdot 9 + 5\cdot 3 + 6\cdot 0 + 7\cdot 1 + 8\cdot 4 + 9\cdot 9 + 10\cdot 10 = 299 \equiv 2 \not\equiv 0 \pmod{11}.$$

Отже, 084-930-149-X – некоректний код.

Нещодавно запроваджено тринадцяти бітний код ISBN-13, це зумовлено необхідністю ідентифікації більшої кількості книг. Контрольний розряд a_{13} для цього коду з початковими розрядами $a_1 a_2 \ldots a_{12}$ у цьому коді визначається конгруенцією

$$(a_1 + a_3 + \ldots + a_{13}) + 3(a_2 + a_4 + \ldots + a_{12}) \equiv 0 (\mathrm{mod} 10).$$

Проілюструємо обчислення контрольного розряду. Перші дев'ять розрядів коду ISBN-13 підручника Ю. Нікольський, В. Пасічник, Ю. Щербина «Дискретна математика», вид-во «Магнолія–2006», Львів, 20016, такі: 978-966-2025-76-. Знайдемо контрольний розряд. Цей розряд визначається конгруенцією $(a_1 + a_3 + \ldots + a_{13}) + 3(a_2 + a_4 + \ldots + a_{12}) \equiv 0 (\mathrm{mod} 10)$. У нашому випадку матимемо $(9 + 8 + 6 + 2 + 2 + 7 + a_{13}) + 3(7 + 9 + 6 + 0 + 5 + 6) \equiv 0 (\mathrm{mod} 10)$, звідки $a_{13} + 133 \equiv 0 (\mathrm{mod} 10)$.

Отже, $a_{13} = 7$, код ISBN-13 для цієї книги є 978-966-2025-76-7.

Класична криптографія

Криптографія – наука про побудову відображень інформації, які використовують з метою її захисту (такі відображення називають *криптографічними*). Іншими словами, криптографія вивчає способи шифрування інформації з метою забезпечення її секретності.

КЛАСИФІКАЦІЯ ШИФРІВ

Шифри класифікують за різними ознаками: за видами інформації, що захищається (текст, розмова, відеоінформація), за криптографічною стійкістю, за принципами забезпечення захисту інформації (симетричні, асиметричні, гібридні), за конструкційними принципами (блокові та потокові) тощо. **Симетричні** та **асиметричні** шифри розглянемо пізніше. **Блокові шифри.** Часто алгоритм шифрування буває призначений для перетворення послідовностей лише фіксованої довжини l. Коли ж потрібно застосувати його до більшого тексту, цей текст розбивають на блоки – групи по l символів, і кожен блок перетворюють окремо. Такі шифри називають *блоковими з періодом l*. Якщо загальна кількість символів у тексті не ділиться націло на l, то остання група символів доповнюється до повного блоку довільним наперед обумовленим способом.

При побудові відображень шифру використовують з математичної точки зору два види відображень: перестановка елементів відкритого тексту та заміна елементів відкритого тексту на елементи якоїсь множини. Відповідно множина шифрів ділиться на три види: **шифри перестановки**, **шифри заміни** та **композиційні шифри**; останні використовують комбінацію перестановок і замін. Нижче наведено таблицю класифікації.

КЛАСИФІКАЦІЯ ШИФРІВ

За принципами забезпечення захисту інформації		
Симетричні	***Асиметричні***	***Гібридні***

За конструкційними принципами	
Блокові	***Потокові***

За видами відображень шифру		
Шифри перестановки	***Шифри заміни***	***Композиційні шифри***

ШИФРИ ПЕРЕСТАНОВКИ

Шифри перестановки зберігають всі букви відкритого тексту, але розміщують їх у криптотексті в іншому порядку.

Як типовий приклад шифру перестановки наведемо *матричний шифр обходу*. Повідомлення записується рядками у вигляді прямокутної матриці. Криптотекст формується зчитуванням букв із матриці у зміненому порядку, а саме, стовпчиками. При цьому, послідовність, у якій зчитуються стовпчики, визначається ключем. Було поширеним задання ключа у вигляді ключового слова, яке легко запам'ятовувалось. Порядок зчитування стовпчиків збігався з алфавітним порядком букв ключового слова.

Приклад.

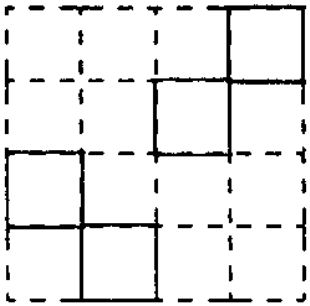
```
GARDEN      Повідомлення:
416235        DON'T PUT IT OFF TILL TOMORROW.
------
DONTPU      Ключове слово:
TITOFF        GARDEN
TILLTO      Криптотекст:
MORROW        OIIOTOLRPFTODTTMUFOWNTLR
```

Матричний шифр обходу

Як ще один приклад шифру перестановки наведемо доволі відомий у популярній математиці *шифр Кардано*. Це блоковий шифр з періодом $l = k^2$, де k парне число. Ключем є вирізаний з паперу в клітинку квадрат $k \times k$, що складається з k^2 клітинок, четверту частину яких, тобто $k^2/4$, прорізують. На рисунку подано приклад для $k = 4$, причому квадрат накреслено пунктиром, а контури чотирьох прорізаних клітинок виділено суцільною лінією.

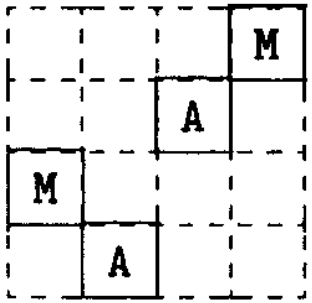
Нехай потрібно зашифрувати блок повідомлення, у якому k^2 букв. Криптотекст записують на клітчастому папері у квадраті $k \times k$. Процес складається з чотирьох кроків. На першому кроці на аркуш, на якому буде писатись криптотекст, накладають ключ і вписують перші $k^2/4$ букв повідомлення у прорізані клітинки, починаючи з верхнього рядка.

На другому кроці ключ повертають на 90° за годинниковою стрілкою відносно центру квадрату і у прорізані клітинки записують наступні $k^2/4$ букв повідомлення. Подібно виконують третій і четвертий кроки – щоразу ключ повертають на 90° і в нові позиції прорізаних клітинок записують чергові $k^2/4$ букв повідомлення. Ключ має бути виготовлений у такий спосіб, щоб при повороті прорізані в ключі клітинки попадали на вільні клітинки аркуша, і в жодному разі не накладалися на клітинки, вже заповнені на попередніх кроках. У результаті після четвертого кроку всі k^2 букв блоку повідомлення виявляються розміщеними в деякому порядку у квадраті $k \times k$. Прочитавши їх рядками, отримаємо криптотекст. На рисунку показано процес перетворення повідомлення мама мила раму рано у криптотекст рмрмиаааммлнуаоа.

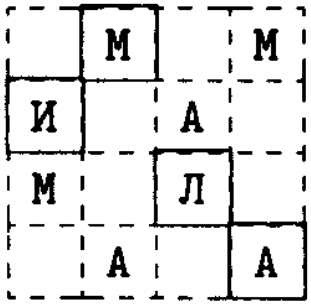


Ключ

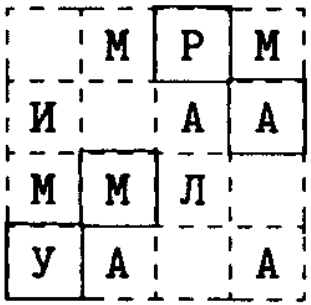
Повідомлення



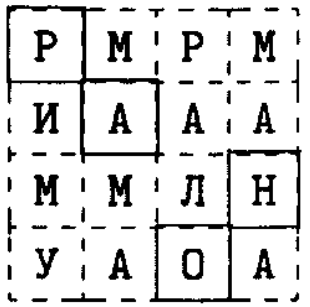
1-й крок



2-й крок



3-й крок



4-й крок:
криптотекст

Шифр Кардано

Шифр Кардано є шифром перестановки спеціального виду, у якому правило переставлення букв у блоці зручне для реалізації на папері за допомогою ножиців та олівця. Загальний шифр перестановки з періодом l переставляє l букв у довільному порядку, який визначається ключем. Ключ зручно задавати табличкою $\begin{pmatrix} 1 & 2 & \dots & l \\ i_1 & i_2 & \dots & i_l \end{pmatrix}$, яка показує, що перша буква блоку відкритого тексту займає позицію i_1 у відповідному блоці криптотексту, друга буква переміщується на позицію i_2, і так далі. Наприклад, при $l = 4$ шифр перестановки з ключем $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix}$ перетворює відкритий текст мамамиларамурано у криптотекст амамалимумаронар. А зображений на малюнку ключ для шифру Кардано можна задати табличкою

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 4 & 7 & 9 & 14 & 2 & 5 & 11 & 16 & 3 & 8 & 10 & 13 & 1 & 6 & 12 & 15 \end{pmatrix}.$$

Розглянемо ще один приклад для $l = 4$. Застосуємо перестановку $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}$

Зашифруємо повідомлення «Я ПРИЇДУ ЗАВТРА». Розбиваємо на блоки по чотири букви, останній блок доповнюємо випадковими буквами:

ЯПРИЇДУЗАВТРАДЛГ.

Застосуємо для кожного блоку перестановку σ й отримаємо:

ПИЯРДЗЇУВРАТДГАЛ.

Для дешифрування використаємо обернену перестановку: $\sigma^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \end{pmatrix}$.

На завершення зробимо ще одне просте, але корисне зауваження. Якщо ми маємо взірець відкритого тексту і криптотексту, то в цьому випадку **легко визначити, що використовується саме шифр перестановки.** Справді, досить пересвідчитися, що кожна буква зустрічається в повідомленні та криптотексті з однаковою частотою.

Додаток 1

Українська абетка

А	Б	В	Г	Ґ	Д	Е	Є	Ж	З	И
0	1	2	3	4	5	6	7	8	9	10
І	Ї	Й	К	Л	М	Н	О	П	Р	С
11	12	13	14	15	16	17	18	19	20	21
Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я
22	23	24	25	26	27	28	29	30	31	32

Додаток 2

Латинська абетка

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25