Одеський національний університет імені І. І. Мечникова

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра комп'ютерної алгебри та дискретної математики

(повна назва кафедри)

К в а л і ф і к а ц і й н а р о б о т а

на здобуття ступеня вищої освіти «бакалавр»

«Псевдовипадкові числа (побудова та застосування в криптографії)»

(тема кваліфікаційної роботи українською мовою)

«Pseudorandom numbers (construction and cryptographic use)»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання
спеціальності 123 «Комп'ютерна інженерія»
(код, назва спеціальності)
Палієнко Денис Віталійович
(прізвище, ім'я, по-батькові здобувача)

Керівник д-р ф.-м. н. проф. Варбанець П.Д.
(науковий ступінь, вчене звання,прізвище, ініціали) (підпис)

Рецензент к. ф.-м. н. доц. Бєлозьоров Г.С.
(науковий ступінь, вчене звання,прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№ 12 від 09.06. 2023 р.

Завідувач(ка) кафедри

_____ Павло ВАРБАНЕЦЬ
(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № _____
протокол № __від ____.____.20___ р.

Оцінка_____/____/_____
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

_____ Алла КОБОЗЄВА
(підпис) (ім'я, прізвище)

Одеса 2023

АНОТАЦІЯ

Ця робота присвячена дослідженню псевдовипадкових чисел (ПВЧ) та їх використання в криптографії. Псевдовипадкові числа відіграють важливу роль у багатьох криптографічних протоколах, де вони використовуються для створення ключів, маскування даних та забезпечення конфіденційності та цілісності інформації.

У роботі розглядаються основні поняття та принципи псевдовипадкових чисел. Описуються різні методи генерації ПВЧ, зокрема лінійний конгруентний метод, метод середніх квадратів. Досліджуються переваги та недоліки кожного методу.

Крім того, робота розглядає основні види генераторів псевдовипадкових чисел (ГПВЧ), такі як апаратні, табличні та алгоритмічні, та їх структуру. А також було розглянуто їх основні переваги та недоліки.

Особливу увагу було приділено аналізу інверсного конгруентного методу, на базі якого і був спроектований та побудований інверсний генератор псевдовипадкових чисел.

ABSTRACT

This work is dedicated to the study of pseudorandom numbers (PRNs) and their application in cryptography. Pseudorandom numbers play a crucial role in many cryptographic protocols where they are used for key generation, data masking, and ensuring the confidentiality and integrity of information.

The paper explores the fundamental concepts and principles of pseudorandom numbers. Various methods of PRN generation are described, including the linear congruential method, the method of the middle squares. The advantages and disadvantages of each method are investigated.

Additionally, the paper examines the main types of pseudorandom number generators (PRNGs), such as hardware-based, table-based, and algorithmic generators, along with their structures. Their key advantages and disadvantages are also discussed.

Special attention is given to the analysis of the inverse congruential method, upon which the inverse pseudorandom number generator was designed and constructed.

ЗМІСТ

Стор.

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

Скорочення :

ПЧ - послідовність чисел

ГПВЧ – генератор псевдовипадкових чисел

ГВЧ – генератор випадкових чисел

ГПВБ – генератор псевдовипадкових бітів

ГВБ – генератор випадкових бітів

ЛКМ – лінійно конгруентний метод

ЛКП – лінійно конгруентна послідовність

ДЕ – джерела ентропії

Терміни:

Кореляція – це будь-який взаємозв’язок між двома випадковими змінними або двовимірними даними.

Конгруенція – відношення еквівалентності на алгебраїчній структурі, що зберігається за основних операцій

Рекурсія – метод визначення класу чи об’єкта через попереднє задання одного чи декількох (зазвичай простих) його базових випадків чи методів, а потім завданням на їхній основі правила побудови класу, який визначається.

ВСТУП

За останні роки розвиток обчислювальної техніки досяг меж, які здавалися неможливими ще якесь десятиріччя тому. Повсякденне її використання стимулюється розширенням сфери можливих застосувань, а масовість реалізацій призводить до доступності з точки зору цінового фактору.

Досить часто в нашому житті виникають ситуації, коли необхідно одержувати випадкові або псевдовипадкові числові послідовності. Перерахуємо деякі сфери їх застосування:

1) Соціологічні та наукові дослідження. Підготовка випадкових вибірок під час збору даних, опитування думок чи дослідженні фізичних явищ із випадковим вибором результатів експериментів.

2) Моделювання. У комп'ютерному моделюванні фізичних явищ. Крім того, математичне моделювання використовує випадкові числа як один із інструментів чисельного аналізу.

3) Криптографія та інформаційна безпека. Випадкові числа можуть використовуватись у тестуванні коректності чи ефективності алгоритмів та програм. Багато алгоритмів використовують генерацію псевдовипадкових чисел для вирішення прикладних завдань (наприклад, криптографічні алгоритми шифрування, генерація унікальних ідентифікаторів та ін.).

4) Ухвалення рішень в автоматизованих експертних системах. Використання випадкових чисел є складовою частиною стратегій прийняття рішень. Наприклад, для неупередженості вибору екзаменаційного квитка студентом на іспиті. Випадковість також використовується в теорії матричних ігор.

5) Оптимізація функціональних залежностей. Деякі математичні методи оптимізації використовують статистичні методи пошуку екстремумів функцій.

6) Розваги та ігри. Випадковість у іграх має значну роль. У комп'ютерних чи настільних іграх випадковість допомагає урізноманітнити ігровий процес.

У глибокому ознайомленні з природою випадковості та випадкових чисел виникають питання про те, що таке "справжнє" випадкове число, яка повинна бути послідовність випадкових чисел, яким може бути розподіл випадкових чисел на певному інтервалі (наприклад, в часовому). У цій книзі наведені встановлені формальні визначення та популярні алгоритми, які містять відповіді на ці питання. Потреба використовувати випадкові числа у науковій роботі виникла давно.

Спочатку для цієї цілі використовувалася урна з кулями, з якої наугад витягували кулю з цифрою. Пізніше були побудовані механічні генератори випадкових чисел. З появою електронних схем з'явилися й електронні генератори випадкових чисел. Один з перших таких генераторів, запропонований А.М. Тьюрінгом, використовував резисторний генератор шуму для отримання 20 випадкових бітів, які вводилися до суматора. Проте генератори випадкових чисел не завжди давали "добрі" результати (поняття "добрих" випадкових чисел буде розглянуто далі). Крім того, апаратні генератори випадкових чисел часто зазнають збоїв.

Але таблиці "добрих" випадкових чисел, які були попередньо обчислені, були дуже незручними використовувати через обмеженість комп'ютерної пам'яті. У 90-ті роки XX століття зростання можливостей комп'ютера, збільшення щільності запису на магнітних та оптичних носіях дозволили створити достатньо великі таблиці випадково згенерованих байтів. Так, наприклад, Джордж Марсалья створив великий каталог таблиць випадкових чисел обсягом 650 мегабайт, який був поміщений на оптичний диск.

Проте ні табличний метод, ні апаратне генерування випадкових чисел не могли задовольнити потребу в надійних, швидких і ефективних генераторах випадкових чисел через вроджені недоліки цих методів. Тому

вже на початку комп'ютерної ери увага математиків була звернута на алгоритмічні способи отримання випадкових чисел. Так, ще в 1946 році Джон фон Нейман запропонував арифметичний спосіб створення числа, "схожого на випадкове". Ідея методу полягає в отриманні наступного випадкового числа з попереднього шляхом піднесення його до квадрату і виділення середніх цифр. Наприклад, якщо попереднє число було 259, то піднісши його до квадрату, ми отримаємо 67081, а середні цифри - 708 - будуть наступним випадковим числом. Очевидно, що отримана в результаті таких операцій послідовність не є випадковою, оскільки повністю визначається формулою та початковим числом. Проте у багатьох застосуваннях така псевдовипадкова послідовність є достатньою, оскільки виглядає "досить" випадковою. У літературі числа, згенеровані арифметичним способом, називають **псевдовипадковими**, а метод їх генерації - **генератором псевдовипадкових чисел**. Послідовності псевдовипадкових чисел мають ряд суттєвих недоліків. Один з головних недоліків - циклічність послідовності. Тобто генератор псевдовипадкових чисел може видати, наприклад, 6100 чисел, відмінних одне від одного, але потім, починаючи з 6101-го числа, буде циклічно відтворювати ці 6100 попередніх чисел. Іншим недоліком, властивим всім алгоритмічним методам генерації чисел, є очевидна залежність наступних чисел від попередніх. Це може бути непомітно для людського сприйняття, але в подальшому негативно вплине на момент застосування таких чисел при вирішенні прикладної задачі. Алгоритмічні методи генерації можуть лише частково "маскувати" таку залежність, але не можуть позбавитися від неї. Проблема створення швидкого, ефективного генератора псевдовипадкових чисел гостро стоїть і зараз. Створити генератор, позбавлений традиційних недоліків, виявилося дуже складною задачею.

Успіхи, досягнуті в останні роки в області електроніки, обумовили широке впровадження мікроелектроніки в засоби електричних вимірів.

Заміна електронних ламп транзисторами й особливо інтегральними мікросхемами стимулювала розробників приладів до пошуків і впровадження нових методів, реалізація яких у минулі роки була неможливою через велику складність, а отже, низьку надійність приладів і високу вартість.

Мета даної кваліфікаційної роботи є розробка електронно-лічильного генератора випадкових чисел.

РОЗДІЛ 1
СТРУКТУРА ГЕНЕРАТОРА ПСЕВДОВИПАДКОВИХ ЧИСЕЛ

1.1 Основні поняття

Спочатку наведемо визначення основних понять теорії генерації випадкових чисел.

Визначення 1.1 Випадкове число – число, що представляє собою реалізацію випадкової величини.

Визначення 1.2 Детермінований алгоритм – алгоритм, який повертає ті самі вихідні значення при тих самих вхідних значеннях.

Визначення 1.3 Псевдовипадкове число – число, отримане детермінованим алгоритмом, що використовується як випадкове число.

Визначення 1.4 Фізичне випадкове число (справді випадкове) - випадкове число, отримане на основі деякого фізичного явища.

Як правило, генерація випадкового числа складається з двох етапів:

1) генерація нормалізованого випадкового числа (тобто рівномірно розподіленого від 0 до 1);

2) перетворення нормалізованих випадкових чисел r_i у випадкові числа x_i, які розподілені за заданим законом розподілу чи необхідному інтервалі.

Визначення 1.5 Генератор випадкових бітів (ГВБ) - це пристрій або алгоритм, що видає послідовність статистично незалежних і неперекошених бітів (тобто, які підпорядковуються закону розподілу).

Зауваження: Генератор випадкових бітів може бути використаний для генерації рівномірно розподілених випадкових чисел. Наприклад, випадкове ціле число в інтервалі $[0; n]$ може бути отримано зі згенерованої

послідовності випадкових бітів довжини $\lfloor \lg n \rfloor + 1$ шляхом конвертації її у відповідну систему числення.

Якщо отримане в результаті ціле число перевищує n, то його можна відкинути і згенерувати ще одну послідовність бітів. Тому далі ми використовуватимемо термін «генератор випадкових чисел» нарівні з терміном «генератор випадкових бітів».

Визначення 1.6 Генератором псевдовипадкових бітів (детермінованим ГПВБ) називатимемо детермінований алгоритм, який отримує на вході двійкову послідовність довжини k і видає на виході двійкову послідовність довжиною $l \gg k$ l (l значно більше k), яка «виглядає випадковою».

Вхідне значення ГПВБ називається початковим вектором (також називають ініціалізаційним вектором та позначають IV), а вихід називається псевдовипадковою послідовністю бітів. Пояснимо поняття «виглядає випадковим». Зрозуміло, що послідовність, що згенерована детермінованим алгоритмом, не є випадковою. Однак мета алгоритму в тому, щоб взяти деяку маленьку послідовність істинно випадкових чисел і використовувати її для генерації довгої послідовності, яка не відрізняється від випадкової послідовності чисел тієї ж довжини.

Переконатися у тому, що послідовність чисел випадкова (або не випадкова) можна за допомогою статистичних тестів, що виявляють специфічні особливості випадкових послідовностей, або аналітико-обчислювальними методами.

Визначення 1.7 Кажуть, що ГПВБ проходить всі поліноміальні за часом ймовірнісні тести на статистичну випадковість, якщо не існує поліноміального за часом ймовірнісного алгоритму, який міг би коректно відрізнити вихідну послідовність генератора від істинно випадкової послідовності тієї ж довжини з ймовірністю більш ніж $1/2$.

Визначення 1.8 Говорять, що ГПВБ успішно проходить тест на наступний біт, якщо не існує поліноміального за часом алгоритму, який може

за допомогою вхідних l бітів послідовності s передбачити (l + 1)-й біт s з ймовірністю, що перевищує $1/2$. Більш формально, ГПВБ проходить тест на наступний біт, якщо для будь-якого $i \in \mathbb{N}$ та будь-якого ймовірнісного поліноміального за часом алгоритму $A:\{0,1\}^i \to \{0,1\}$, виконується наступна нерівність $\left| P(A(s_1^{i-1}) = s_i) - \frac{1}{2} \right| < 0(v(n))$, де $0(v(n))$ - позначення функції, що спадає швидше за обернений поліном степені n. Незважаючи на те, що означення 1.7 накладає більш жорсткі умови на ГПСЧ, ніж означення 1.8, можна довести, що ці означення еквівалентні.

Твердження 1.1 (Універсальність тесту на наступний біт) ГПВБ проходить тест на наступний біт тоді і лише тоді, коли він проходить всі поліноміальні статистичні тести. Доведення цього твердження було отримано Ендрю Яо у 1982 році.

1.2 Види генераторів випадкових чисел

Генератори випадкових чисел за способом отримання чисел поділяються на:

- апаратні;
- табличні;
- алгоритмічні.

Табличні генератори використовують заздалегідь підготовлені таблиці, що містять перевірені некорельовані числа, як джерело випадкових чисел і не є суворими генераторами у суворому розумінні цього терміну. Недоліки такого методу очевидні: використання зовнішнього ресурсу для зберігання чисел, обмеженість послідовності, передбачуваність значень.

Апаратні (істинні) генератори випадкових послідовностей повинні мати джерело ентропії. Розробка генераторів, які використовують джерела ентропії та генерують некорельовані і статистично незалежні числа, є досить складною задачею. Крім того, для багатьох криптографічних застосувань

такий ГПВЧ не повинен бути предметом вивчення та впливу протилежної сторони.

Алгоритмічний генератор є комбінацією фізичного генератора та визначеного алгоритму. Такий генератор використовує обмежений набір даних, отриманих з виходу фізичного генератора, для створення довгої послідовності чисел шляхом перетворень вихідних чисел. Цей вид генераторів викликає найбільший інтерес через його очевидні переваги над генераторами випадкових чисел інших типів.

1.3 Апаратні генератори випадкових чисел

Визначення 1.9 Апаратний генератор випадкових чисел - це пристрій, який використовує виміри параметрів деяких фізичних процесів для створення випадкових чисел. Зазвичай апаратний генератор випадкових чисел складається з джерела ентропії та пристрою, що перетворює значення, отримані з джерела ентропії, у потрібний формат.

Джерелами ентропії (ДЕ) можуть бути:

1) підкидання монети;
2) часові затримки між моментами випромінювання частинок під час радіоактивного розпаду;
3) теплові шуми під час роботи напівпровідникового діода або резистора;
4) частотні відхилення вільно працюючого генератора частот;
5) фотоефект - випромінювання електронів речовиною під впливом світла;
6) звук з мікрофона або відео з підключеної камери;
7) стан деяких блоків пам'яті комп'ютера.

Однією з найпоширеніших технологій, що використовуються в апаратних ГВЧ, є тепловий шум. Вони вимірюють електричний шум, який генерується напівпровідниковими компонентами, і перетворюють його на

випадкові числа. Квантові генератори випадкових чисел базуються на явищах квантової механіки, наприклад, на фотоефекті або квантових флуктуаціях. Вони створюють випадкові числа на основі випадкових квантових процесів.

Апаратні ГВЧ зазвичай забезпечують високий рівень випадковості та непередбачуваності в згенерованих числах, що робить їх корисними в широкому спектрі застосувань. Вони використовуються в криптографії, комп'ютерній безпеці, моделюванні випадкових подій, наукових дослідженнях та інших областях, де потрібна «високоякісна» випадковість.

РОЗДІЛ 2
АЛГОРИТМИ ГЕНЕРАЦІЇ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ

Розробка алгоритмічних генераторів може бути ще складнішою задачею порівняно зі створенням апаратних генераторів випадкових чисел. Нагадаю, що в програмних генераторах використовуються початкові вектори - деякі істинно випадкові числа для генерації всіх інших чисел послідовності. Таким чином, з деякого джерела ентропії беруться дані, які конвертуються в початкове істинно випадкове число. Далі це число використовується в алгоритмі для генерації інших членів послідовності. З-за високої вартості апаратних генераторів випадкових чисел у більшості випадків в якості джерела ентропії використовуються ресурси обчислювальної машини, на якій виконується програма генерації ПВЧ. При відсутності апаратного генератора випадкових чисел в якості джерела ентропії можуть використовуватися:

1) стан системного годинника;
2) часові затримки між натисканням клавіш на клавіатурі або рухами мишки;
3) вміст буферів введення/виводу;
4) значення, отримані під час роботи системи (час завантаження системи, мережева активність і т.д.).

Генератори ПВЧ, що базуються на перетвореннях системного часу, мають свої недоліки: алгоритм, що використовує ГПВЧ, може бути прив'язаним до системного часу, тому числа, згенеровані в ці моменти, можуть бути менш випадковими. Наприклад, програма повинна видаляти

серію випадкових чисел в початку кожної секунди. ГПВЧ, що використовують події, що генеруються користувачем (затримки між натисканням клавіш, координати руху миші), при детальному аналізі виявляються нерівномірно розподіленими, тому деякі числа менш випадкові. Хороший програмний генератор випадкових бітів повинен використовувати якомога більше різних джерел ентропії. Це зменшить можливість злоумисника проаналізувати алгоритм створення випадкових бітів, а також підвищить надійність генератора у випадках, коли одне або кілька джерел ентропії можуть вийти з ладу. Кожне з джерел ентропії має бути, наскільки це можливо, незалежним. Отримані випадкові послідовності "перемішуються" за допомогою певної спеціальної функції. Один з способів такого "перемішування" - це застосування до послідовності функції хешування. Розглянемо способи отримання випадкових чисел на інтервалі [0,1]. Знаючи, що точність подання дійсних чисел у комп'ютері обмежена певними значеннями, ми будемо використовувати той факт, що будь-яке число заданої точності з інтервалу [0,1] може бути однозначно перетворене в число з інтервалу [0, m]. І навпаки, число з інтервалу [0, m] поділом на m перетворюється до інтервалу [0,1]. Загальна формула для алгоритмічного ГПВЧ виглядає так: $x_{i+1} = f(x_{i-k+1}, x_{i-k+2}, ..., x_i)$, де f – деяка функція перетворення останніх членів послідовності чисел в нове значення. Послідовності з таких чисел обов'язково утворюють цикли. Повторювані цикли називають періодом. Іншими словами, такі послідовності завжди мають період. Період послідовності ПВЧ - одна з ключових її властивостей, яка оцінюється в першу чергу. Перевагами алгоритмічних генераторів є швидкодія та компактність реалізації. Основним недоліком є низька якість "випадковості", - такі послідовності, як правило, не пройшли більшість поліноміальних тестів на випадковість.

2.1 Метод середніх квадратів

Один із перших алгоритмічних методів отримання рівномірно розподілених псевдовипадкових чисел отримав назву "метод середини квадрату". Він був запропонований Джоном фон Нейманом і полягає в наступному:

1) вибрати початкове випадкове число X_0 із n-розрядним представленням (можливо, отримане від зовнішнього джерела ентропії);

2) піднести це число X_i до квадрату, в результаті чого отримаємо число з 2n-розрядним представленням Y_i;

3) наступне число X_{i+1} отримуємо, складаючи його n-розрядне представлення, вибираючи середні n розрядів з числа Y_i.

Наприклад, якщо початкове число має такий вигляд: X_0 = 3485, то Y_1 = 3485^2 = 12145225, X_1 = 1452, а X_2 = 1083, и т.д.

Як початкове число для цього алгоритму часто обирають раціональне число в десятковому записі. Наприклад, X_0 = 0,3485, X_1 = 0,1452, X_2=0,1083 і так далі.

Очевидно, що цей метод можна реалізувати за допомогою операцій цілочисельного ділення та отримання остачі від ділення попереднього члена послідовності.

Недоліком цього методу є наявність кореляції між числами послідовності, а в деяких випадках може відсутність випадковості взагалі. Наприклад, якщо X_0 = 0,4500, X_1 = 0,2500, X_2 = 0,2500, X_3 = 0,2500 і так далі. Крім того, цей метод має невеликий період і зараз цікавий переважно з історичної точки зору.

2.2 Лінійний конгруентний метод

Вибір початкових параметрів (ключових чисел) є основою лінійного конгруентного методу (ЛКМ), який є простим і популярним методом

генерації псевдовипадкових чисел. Д.Г. Лехмером було запропоновано цей метод у 1949 році. ЛКМ базується на наступних чотирьох ключових числах:

1) $m > 0$, модуль;
2) $0 \le a \le m$, множник;
3) $0 \le c \le m$, приріст (інкремент);
4) $0 \le X_0 \le m$, початкове значення.

Визначення 2.1 Послідовність ПВЧ, отримуване за формулою:

$$X_{n+1} = (aX_n + c) \bmod m, n \ge 1, \qquad (2.1)$$

називається *лінійною конгруєнтною послідовністю* (ЛКП). Ключем для неї слугує X_0.

У даній формулі дуже важливим є успішний вибір параметрів. Наприклад, для $X_0 = 7$, $a = 8$, $c = 9$, $m = 10$ отримуємо послідовність

7, 5, 9, 1, 7, 5, 9, 1,…

Це означає, що послідовність зовсім не виглядає "випадковою". На цьому прикладі ілюструється, що конгруентна послідовність завжди утворює цикл. Більше того, надалі буде показано, що ця властивість є у всіх послідовностей, що має вид $X_{n+1} = f(X_n)$. На малюнках 2.1-2.2 показані графіки ЛКП довжиною 500 чисел. Як видно з малюнків, навіть невелика зміна параметра функції призводить до появи короткого періоду.

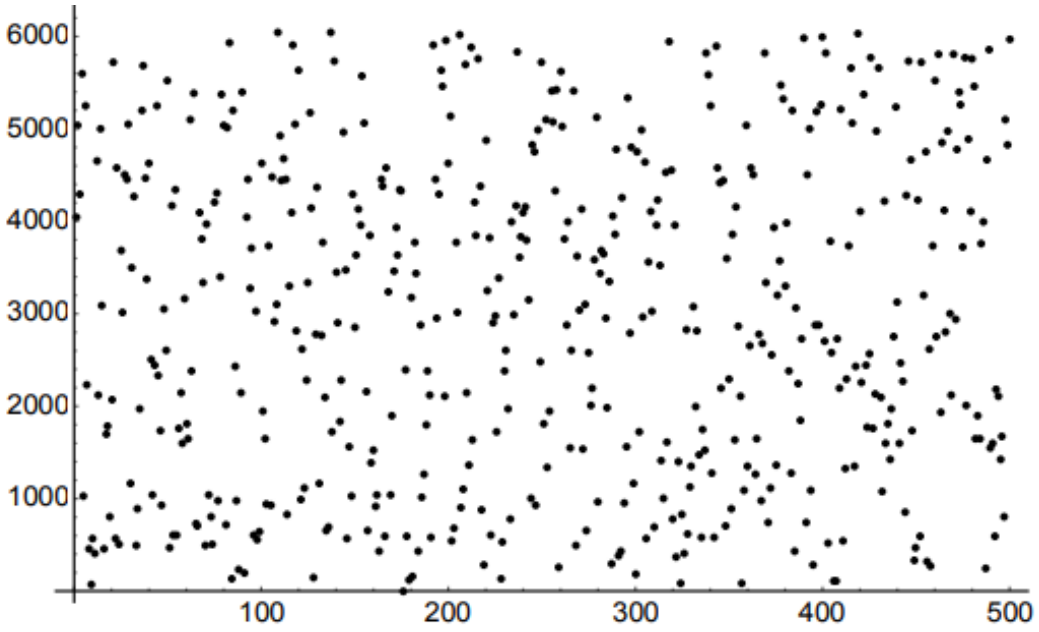

Рисунок 2.1 - Графік ЛКП для X_0 = 7, a = 106, c = 1283, m = 6075.

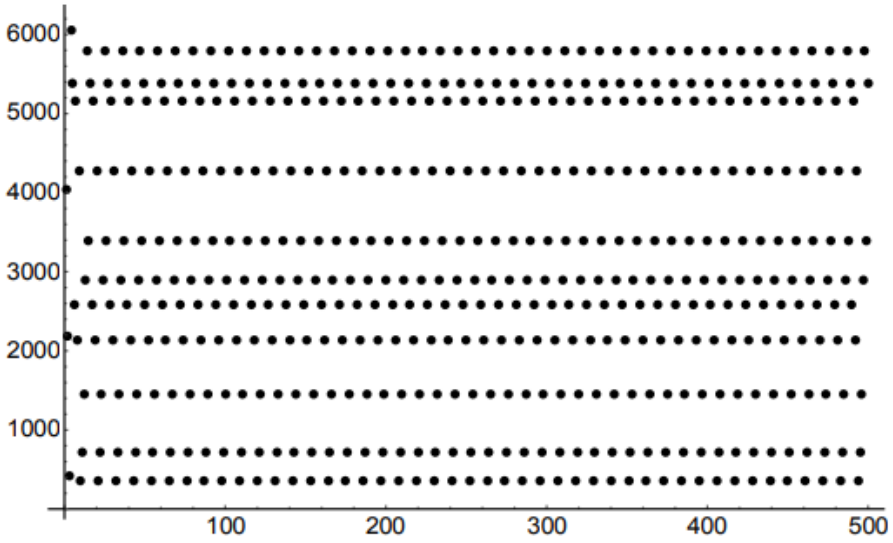

Рисунок 2.2 - Графік ЛКП для X_0 = 7, a = 105, c = 1283, m = 6075.

Крім того, у ЛКП є ще один явний недолік - наявність "решітчатої" структури послідовності чисел. Цей ефект виявляється у впорядкуванні чисел на деяких нахилених прямих, якщо зображати числа на площині. Типовий приклад "решітчатої" структури послідовності чисел (ПЧ)

зображений на малюнку 2.3. Цей недолік обумовлений властивостями операцій у скінченному полі, що використовуються в формулі 2.1.

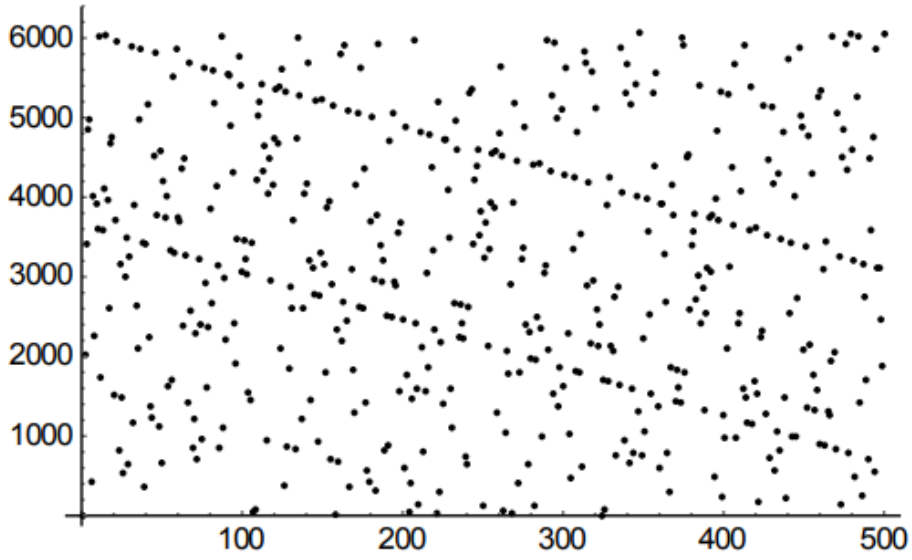

Рисунок 2.3 - Графік ЛКП для параметрів: X_0 =7,a=106,c= 1284,m=6075.

Існує узагальнення формули ЛКМ:

$$X_{n+1} = (a^k X_n + \frac{(a^k - 1)}{(a-1)} c) \bmod m, k \geq 0, n \geq 0. \qquad (2.2)$$

Розглянемо задачу вибору правильного значення m. По-перше, m повинно бути достатньо великим. Наприклад, при m = 2, послідовність в найкращому випадку буде мати вигляд 0, 1, 0, 1, ...

По-друге, розумно вибирати значення m, рівне 2^q, кількість бітів у машинному слові, оскільки це дозволяє уникнути використання операції модуля в формулі 2.2. Однак значення m повинно задовольняти додатковим умовам, які будуть описані нижче.

Множник будемо шукати, враховуючи умови максимальної довжини періоду. Варто враховувати, що вимога до довжини періоду не є єдиною. Наприклад, при a=c=1 послідовність матиме вигляд:

$$X_{n+1} = (X_n + 1) \bmod m.$$

Вона має максимальний період, але не є випадковою.

Теорема 2.1 Лінійна конгруентна послідовність, визначена числами m, a, c та X_0, має період довжиною m тоді і лише тоді, коли:

1) числа c та m взаємно прості;

2) a−1 є кратним p для деякого простого p, що являється дільником m;

3) a −1 є кратним 4, якщо m є кратним 4.

Розглянемо деякі додаткові умови для ЛКМ, які не включені до теореми 2.1, оскільки ця теорема встановлює лише необхідні вимоги для максимальності періоду, але нічого не каже про якість отриманої ЛКП.

Визначення 2.2 Потенціал лінійної конгруентної послідовності з максималь ним періодом визначається як найменше ціле число s, таке, що $(a-1)s=0$ по модулю m.

Нехай X_0=0. Це припущення допустиме, оскільки за умовами теореми 2.1 число 0 зустрінеться у послідовності принаймні один раз протягом періоду. З таким припущенням формула 2.2 спрощується до $X_n=\frac{(a^n-1)}{(a-1)}c \bmod m$ і, якщо розкласти вираз a^n-1 за біноміальною формулою, отримаємо:

$$X_n=c(n+\binom{n}{2}(a-1)+...+\binom{n}{s}(a-1)^{s-1}) \bmod m. \qquad (2.3)$$

За умови кратності чисел $(a-1)^s$, $(a-1)^{s+1}$ значенню m їх можна виключити з формули.

Подальший аналіз зводиться до розгляду властивостей різниці $X_{n+1}-X_n$ для різних значень a і потенціалу.

Наприклад, якщо a=1, потенціал s=1, то $X_n=cn \bmod m$, і послідовність зовсім не виглядає випадковою. Якщо потенціал s=2, то $X_n=cn+c(a-1)\binom{n}{2}$, и знову послідовність не виглядає випадковою. Дійсно, в цьому випадку $X_{n+1}-X_n=c+c(a-1)n$, що є простою лінійною залежністю від n між послідовно згенерованими числами та надає явну "решітчасту" структуру ЛКП.

Якщо потенціал дорівнює 3, то послідовність стає більш-менш подібною до випадкової, але все ще існує висока ступінь залежності

X_n, X_{n+1}, X_{n+2}. Тести показують, що послідовності з потенціалом 3 також не позбавлені проблеми "решітчастості". Було повідомлено, що прийнятні результати були отримані у деяких випадках при потенціалі, рівному 4, але це було спростовано іншими дослідниками. Можна припустити, що послідовності з потенціалом 5 і більше мають достатньо хороші випадкові властивості.

Зауваження. Зазначимо, що високий потенціал є необхідною, але не достатньою умовою випадковості. Поняття потенціалу використовується для виключення недостатніх генераторів, але не для безумовного прийняття генераторів з високим потенціалом. Далі ми ознайомимося з іншими тестами для визнання ПВЧ близькими до випадкових.

Нижче наведена таблиця з деякими константами для лінійних конгруентних генераторів.

Таблиця 2.1 Параметри ЛКП для формули 2.1.

(a,c,m)	(a,c,m)	(a,c,m)
(106,1283,6075)	(625,6571,31104)	(1277,24749,117128)
(211,1663,7875)	(1541,2957,14000)	(2041,25673,121500)
(421,1663,7875)	(1741,2731,12960)	(2311,25367,120050)
(430,2531,11979)	(1291,4621,21870)	(1597,51749,244944)
(936,1399,6655)	(205,29573,139968)	(2661,36979,175000)
(1366,1283,6075)	(421, 17117,81000)	(4081,25673,121500)
(171,11213,53125)	(1255,6173,29282)	(3661,30809,145800)
(859,2531,11979)	(281,28411,134456)	(3613,45289,214326)
(419,6173,29282)	(1093,18257,86436)	(1366,150889,714025)
(967,3041,14406)	(421,54773,259200)	(8121,28411,134456)
(141,28411,134456)	(1021,24631,116640)	(4561,51349,243000)

При реалізації ЛКМ на комп'ютері важливо пам'ятати про можливість переповнення стандартного машинного слова, яке використовується для збереження значень обчислень за формулами 2.1, 2.2.

Лінійний конгруентний метод можна узагальнити до поліноміального конгруентного методу (ПКМ). Наприклад, існують наступні різновиди ПКМ.

Квадратичний конгруентний генератор:

$X_n = (aX_{n-1}^2 + bX_{n-1} + c) \bmod m$.

Кубічний конгруентний генератор:

$X_n = (aX_{n-1}^3 + bX_{n-1}^2 + cX_{n-1} + d) \bmod m$.

Поліноміальний генератор степені r:

$X_n = \sum_{i=0}^{r} a_i X_{n-1}^i \bmod m$, r≥1.

Так, лінійні конгруентні генератори мають перевагу в простоті їх реалізації та швидкості. Однак їх неможливо використовувати в криптографії, оскільки їх легко "підібрати". Це означає, що, маючи послідовність чисел, отриманих таким генератором, опонент може відновити параметри генератора з мінімальними зусиллями. Перші це продемонстрував Джим Рідс, а потім Джоан Бояр. Тим не менш, ЛКМ є корисними для не-криптографічних застосувань. Наприклад, в моделюванні або в ігрових додатках.

2.3 Адитивний ГПВЧ

Ідею рекурсивного обчислення значення можна узагальнити до формули, яка використовує два попередні значення послідовності. Наприклад, ми можемо рекурсивно обчислити значення X_{n+1} как лінійну комбінацію значення X_n та X_{n-1}. Тоді максимальна довжина послідовності у найкращому випадку буде дорівнювати m^2, так як послідовність не буде повторюватися, поки не буде отримана рівність $(X_{n+l}, X_{n+l+1}) = (X_n, X_{n+1})$. Простейша послідовність, у якій X_{n+1} залежить від більш ніж одного попереднього значення, - це послідовність Фібоначчі

$X_{n+1} = (X_n + X_{n-1}) \bmod m$.

Цей генератор був розглянутий наприкінці 1950-х років і, зазвичай, має період, що більший за m. Однак числа, отримані за допомогою рекурентного співвідношення Фібоначчі, недостатньо випадкові. Але можна зробити ще один крок узагальнення і використовувати формулу

$$X_{n+1} = (X_{n-k} + X_{n-j}) \bmod m, \quad j > k \geq 1.$$

Наприклад, у 1958 році Дж. Ж. Мітчел та Д.Ф. Мур запропонували послідовність, визначену наступним чином:

$$X_n = (X_{n-24} + X_{n-55}) \bmod m, n \geq 55, \tag{2.4}$$

де m – парне число, а $X_0,\ldots,X_{54}$ – довільні цілі числа.

Числа k та j зазвичай називаються затримкою, а послідовність 2.4 - послідовністю Фібоначчі з затримкою. Для ПВЧ Фібоначчі з затримкою справедлива наступна теорема.

Теорема 2.2 Якщо многочлен $x^k + x^j + 1$ являється примітивним многочленом над полем GF(2), то послідовність, формована адитивним ГПВЧ Фібоначчі з затримкою, має максимальний період, рівний $2^{\log_2 m - 1}(2^{\max\{k,j\}} - 1)$.

Зокрема, для формули 2.4 справедлива оцінка довжини періоду, яка становить $2^{31}(2^{55}-1)$, при $m=2^{32}$.

Наведемо інші варіанти запізнювань (k,j) для адитивного ГПВЧ [1, 18]: (9,49), (19,58), (18,65), (25,73), (38,89), (2,93), (21,94), (11,95), (37, 100), (33,118), (10,111), (37,124), (29,132), (52,145), (57,134), (83, 258) (107,378), (273, 607), (1029, 2281), (576, 3217), (4187, 9689), (7083, 19937), (9739, 23209). При великих значеннях запізнювання цей метод стає неефективним через високі вимоги до пам'яті.

Також були запропоновані модифікації такого генератора, в яких замість додавання використовувалося множення. Ці генератори добре себе зарекомендували і з 1958 року застосовувалися в різних прикладних задачах. Зокрема, одна з варіацій такого ГПВЧ використовується в пакеті Matlab. Проте в 1990-і роки вони провалили спектральний тест.

Зауваження. В хорошому джерелі випадкових чисел нерівності $X_{n-1} < X_n < X_{n+1}$ буде зустрічатися приблизно один раз з шести, оскільки кожне з шести можливих співвідношень порядку X_{n-1}, X_n, X_{n+1} буде мати однакову ймовірність. Можна показати, що таке відношення ніколи не виникне, якщо використовувати послідовність Фібоначчі: $X_{n+1} = (X_n + X_{n-1}) \bmod m$.

Даний адитивний генератор має узагальнення: можна обчислювати наступне значення послідовності як лінійну комбінацію попередніх k елементів цієї послідовності. Коли m=p - просте число, то згідно з теорією скінченних полів можна знайти множники a1,a2,...,ak, такі, що послідовність, визначена формулою

$$X_n = (a_1 X_{n-1} + ... + a_{k-1} X_{n-k+1} + a_k) \bmod p, \qquad (2.5)$$

буде мати максимально можливий період, рівний $p^k - 1$. Тобто, справедлива наступна теорема.

Теорема 2.3 Якщо константи $a_1, a_2, \ldots, a_k$ такі, що многочлен $x^k - a_1 x^{k-1} - ... - a_k$ являється примітивним над полем GF (p) і хоча б один з елементів $X_0, X_1, ..., X_k$ не дорівнює нулю, то період генератора дорівнює $p^k - 1$.

Однак, вибір "добрих" коефіцієнтів a_i – нетривіальне завдання, яке вимагає від дослідника аналітичних обчислень. Справа в тому, що константи a1, a2, ..., ak мають відповідні властивості, тільки коли поліном

$$\Phi(x) = x^k - a_1 x^{k-1} - ... - a_k$$

є первісним поліномом за модулем p, що виконується тоді й тільки тоді, коли корінь цього полінома є первісним елементом поля з p^k елементами. Перевірка цього може вимагати факторизації великого числа на прості множники, тому, як правило, для перевірки "випадковості" ГПВЧ, заснованого на формулі 2.5, використовують критерії зі стандартного набору перевірки якості ГПВЧ.

На основі адитивних ГПВЧ створено кілька алгоритмів шифрування.

РОЗДІЛ 3
ІНВЕРСНІ КОНГРУЕНЦІАЛЬНІ ГЕНЕРАТОРИ ПОСЛІДОВНОСТІ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ

Послідовність дійсний чисел відрізку $[0,1)$ ми будемо називати псевдовипадковою, якщо вона має статистичні властивості, які притаманні випадковим числам, тобто рівномірно розподілені і незалежні. Таким чином, послідовність псевдовипадкових чисел повина бути приблизно рівномірно розподілена на $[0,1)$, а її послідовні числа некорельовані. За міру рівномірного розподілу точок $x_0, x_1, \dots, x_{N-1}$ на $[0,1)$ приймають функцію розкиду (дискрепансія), яка визначається як

$$D_N := \sup_{[a,b)\subset[0,1)} \left| \frac{A_N(a,b)}{N} - (b-a) \right|,$$

де $A_N(a,b)$ є кількість точок серед $x_0, x_1, \dots, x_{N-1}$, які потрапили на відрізок $[a,b)$, а sup береться по всім відрізкам $[a,b)$, які входять в $[0,1)$.

Зрозуміло, що D_N завжди не перевищує 1. Тому, якщо $D_N = o(1)$ при $N \to \infty$, то вважають, що послідовність проходить тест на рівномірність.

Друга фундаментальна статистична властивість псевдовипадкових чисел є незалежність. Добре відомо, що, якщо $X_1, \dots, X_s$ рівномірно розподілені в $[0,1)$, то $X_1, \dots, X_s$ будуть незалежними тоді і тільки тоді, коли S -мірний вектор $(X_1, \dots, X_s)$ рівномірно розподілений в $[0,1)^s$. Цей факт дозволяє встановити тест на незалежність послідовності $x_0, \dots, x_{N-1}$ з $[0,1)$: ми формуємо точки з $[0,1)^s$ типу

$$X_n^{(s)} = (x_n, x_{n+1}, \dots, x_{n+s-1}), n = 0,1, \dots \text{(це"перетинні"точки)}$$

або

$$X_n^{(s)} = (x_n s, x_{ns+1}, \dots, x_{ns+s-1}), n = 0,1, \dots \text{(це"неперетинні"точки).}$$

Якщо точки $X_n^{(s)}$ рівномірно розподілені в $[0,1)^s$, $s = 2,3, \dots$, то вважають, що послідовність $x_0, x_1, \dots, x_{N-1}$ проходить серіальний тест на псевдовипадковість. Таким чином, ми бачимо, що s -мірна функція розкиду $D_N^{(s)}$, де

$$D_N^{(s)} := \sup_{\Delta\subset[0,1)^s} \left| \frac{A_N(\Delta)}{N} - |\Delta| \right|,$$

Δ - s -мірний паралелепіпед з $[0,1)^s$, $|\Delta|$ - об'єм Δ, а $A_N(\Delta)$ кількість точок $X_n^{(s)}$, $n = 0,1, \dots, N$, які попали в Δ, є мірою непередбаченності послідовності $\{X_n\}$.

При використанні псевдовипадкових чисел в наближеному аналізі, головну увагу приділяють рівномірності розподілу послідовності $x_0, x_1, \dots, x_{N-1}$, але для криптографічних застосувань цього недостатньо. Тут важливо, щоб послідовність мала властивість незавбаченності, тобто маючи

деякий відрізок послідовності, не можливо передбачити наступний елемент цієї послідовності. Ця незавбаченність забезпечується властивістю незалежності членів послідовності.

Для побудови послідовності псевдовипадкових чисел використовуються генератори псевдовипадкових чисел. Якісний генератор псевдовипадкових чисел повинен задовільняти наступним умовам:

1) послідовність псевдовипадкових чисел за своїми статистичними властивостями не повина відрізнятись від істино випадкової послідовності;

2) незавбаченність елементів послідовності;

3) великий період сформованої послідовності, якщо цей період існує;

4) ефективна програмна та апаратна реалізація.

Багатьом з цих умов відповідають конгруентні генератори, які породжують послідовність чисел y_0, y_1, ..., яка визначається рекурсією

$$y_{n+1} \equiv f(y_n)(modM), n = 0,1, \ldots$$

де фіксується ініціальне значення $y_0 \in [0, M-1]$, модуль M і цілозначна функція $f(u), u \in Z$.

За допомогою нормалізації $x_n = \frac{y_n}{M}$ ми отримуємо послідовність чисел з $[0,1)$.

Уперше Д.Г. Лемер в 1951р. побудував генератор псевдовипадкових чисел

$$y_{n+1} \equiv ay_n + b(modM), n = 0,1,2, \ldots, \quad (3.1)$$

де M - велике натуральне число (модуль генератора), y_0, a, b - фіксовані невід'ємні цілі числа, які менш за M.

Цей генератор зветься лінійним конгруентним генератором псевдовипадкових чисел. М. Сато дав повний опис найменшого періоду послідовностей псевдовипадкових чисел, породжуваних лінійним конгруентним генератором. Цей період залежить від a, b, y_0, M. Але на жаль, у $80^{\text{х}}$ роках минулого століття стало зрозуміло, що лінійний генератор (3.1)

не задовільняє умові незавбаченності. Також Марсал'я було доведено, що двомірні точки (x_n, x_{n+1}), координати яких задовільняють конгруенції (3.1), розташовані на паралельних лініях двомірного квадрату $[0,1)^2$, а тому можуть бути пердбачені.

Почався пошук нелінійних генераторів. У 1989р. Й. Ейченауер, Й. Лєхн, а згодом і Г. Нідеррайтер запропонували інверсний конгруентний генератор. Нехай $p > 2$ просте число, $a, b \in Z,\ 0 \le a, b < p,\ 0 \le x_0 < p$. Розглянемо рекурсію

$$y_{n+1} \equiv a\overline{y}_n + b(modp), n = 0,1,2, \ldots, \qquad (3.2)$$

де $\overline{y}_n = 0$, якщо $y_n = 0$, і $y_n \cdot \overline{y}_n \equiv 1(modp)$, якщо $(y_n, p) = 1$ (тобто, для $(y_n, p) = 1$ ми визначаємо $\overline{y_n}$ як мультиплікативне обернене для y_n). (Г. Нідеррайтер замість конгруенції (3.2) розглядав рівняння $y_{n+1} = a\overline{y}_n + b$ над скінченним полем F_q).

Було доведено, що для певних значень параметрів a, b, y_0 генератор (3.2) породжує послідовність псевдовипадкових чисел, яка задовільняє усім вимогам до таких послідовностей. В роботах Ейченауера, Лехна, Чоу був дан опис періодів послідовності $\{y_n\}$, яка породжена інверсним конгруентним генератором (3.2) з простим модулем p. Наведемо теорему Чоу:

Теорема 3.1 Нехай $p > 2$просте число, $y_0, a, b \in F_p$. Нехай $X(y_0, a, b)$ відповідна послідовність, яка породжена рекурсією (2), $L(y_0, a, b)$ - період $X(y_0, a, b)$, $f(y) = y^2 - by - a$. Через $\partial(m_f)$ позначимо порядок поліному $m_f(y) = y^2 - (b^2a^{-1} + 2)y + 1$ для $a \not\equiv 0$. Тоді

1) якщо $b^2 + 4a = 0$ і $f(y) = (y - \alpha)^2$ для деякого $\alpha \in F_p$, то $L(y_0, a, b) = p - 1$ для кожного $y_0 \neq \alpha$.
2) якщо $f(y) = (y - \alpha)(y - \beta)$ для деяких $\alpha, \beta \in F_{p^2}$, $\alpha \neq \beta$, то

(a) $L\left(\frac{b}{2}; a, b\right) = \partial(m_f) - 1$ для парного $\partial(m_f)$;

(b) $L\left(\frac{b}{2}; a, b\right) = \partial(m_f)$ для непарного $\partial(m_f)$;

(c) $L(y_0; a, b) = \partial(m_f)$ у випадку $f(y_0) \neq 0$, $y_0 \neq \frac{b}{2}$ і поліном $M_f(y) = y^2 - (2 + (b^2 + 4a)f^{-1}(y_0) + 1)y + 1$ ділить $y^{\partial(m_f)} - 1$;

(d) $L(y_0; a, b) = \partial(m_f)$ у випадку $f(y_0) \neq 0$, $y_0 \neq \frac{b}{2}$ і поліном $M_f(y)$ не ділить $y^{\partial(m_f)} - 1$

3) $L(y_0; a, b) \leq 2$, інакше.

Цілком зрозуміло, що досить великий період послідовності $\{y_n\}$, породженої рекурсією (3.2), можливий лише для великих значень p. Тому Ейченауер-Герман, Ейченауер-Герман і Топузоглу, Хубер, Нідеррайтер і Шпарлінський побудували узагальнення генератора (3.2) за допомогою рекурсії

$$y_{n+1} \equiv ay_n^{-1} + b(mod p^m), p > 2 - \text{просте}, m \geq 2 - \text{натуральне} \quad (3.3)$$

(тут $y_n \cdot y_n^{-1} \equiv 1(mod p^m)$).

Але зрозуміло, що ця рекурсія перестає працювати, якщо на деякому кроці з'явиться y_n таке, що $(y_n, p) > 1$.

Чоу вказав умови існування генератора (3.3) і дав опис періодів відповідних послідовностей псевдовипадкових чисел. Наведемо його результати:

Теорема 3.2 Нехай p - просте, m, a, b, y_0 - цілі, $m \geq 2$; $f(y) = y^2 - by - a$. Тоді рекурсія (3.2) визначає нескінчену послідовність $X(y_0; a, b)$ за модулем p^m в тому і тільки в тому випадку, коли параметри рекурсії y_0, a, b задовільняють одній з наступних умов:

1) $a \equiv 0(\text{mod}p)$ і $(by_0, p) = 1$;
2) $(ay_0, p) = 1$ і $b \equiv 0(\text{mod}p)$;
3) $(aby_0, p) = 1$, $b \equiv 2y_0(\text{mod}p)$ і $a \equiv -y_0^2(\text{mod}p)$;
4) $(aby_0(b^2 + 4a), p) = 1$ і $y_0^2 - y_0 b - a \equiv 0(\text{mod}p)$;

5) $p \neq 2, (ab(b^2+4a), p) = 1, y_0 \equiv \frac{b}{2} (mod p)$ і $o(m_f)$ поліному $m_f(y) = y^2 + (b^2a^{-1} + 2)y + 1$ над F_p є непарне число;

6) $(aby_0(b^2+4a)f(y_0), p) = 1, y_0 \not\equiv \frac{b}{2} (mod p)$ для $p \neq 2$, і $o(M_f)$ поліному $y^2 - (2 + (b^2+4a)f^{-1}(y_0))y + 1$ над F_p не ділить $o(m_f)$.

Теорема 3.3 Нехай p - просте, m, a, b, x_0 - цілі, $m \geq 2$. Припустимо, що послідовність $X(y_0; a, b)$ за модулем p^m визначена рекурсією (3.3). Нехай $f(y) = y^2 - by - a$. Тоді

1) $L(y_0; a, b) = 2p^{m-\max[\nu_p(b)\nu_p(f(y_0))+1]}$, якщо $m > \nu_p(b)$ і $k > \nu_p(f(y_0)) + 1$;

2) $L(y_0; a, b) = 2p^{m-\nu_p(b)}$, якщо $(ay_0(a - y_0^2), p) = 1, b \equiv 0 (mod p)$;

3) $L(y_0; a, b) = p^{m-\nu_p(f(y_0))}$, якщо $m > \nu_p(f(y_0)) \geq 2, (ab, p) = 1, b \equiv 2y_0 (mod p), a \equiv -y_0^2 (mod p), f(y_0) \neq 0$;

4) $L(y_0; a, b) = p^{m-1}$, якщо $p \geq 5, (ab, p) = 1, b \equiv 2y_0 (mod p), a \equiv -y_0^2 (mod p), f(y_0) \neq 0$ і $\nu_p(f(y_0)) = 1$;

5) нехай $p = 3, (ab, p) = 1, b \equiv -y_0 (mod p), a \equiv -y_0^2 (mod p), f(y_0) \neq 0$. Ми маємо $L(y_0; a, b) = 3^{m-s+1}$, де $s = \nu_p(a + b^2)$ і $m \geq s + 1$;

6) нехай $(aby_0(b^2+4a), p) = 1, f(x_0) \neq 0$;

(a) $L(y_0; a, b) = \nu_p(ay_0^2)p^{m-\nu_p(y_0)-1}$, якщо $m > \nu_p(f(y_0)) \geq 2$;

(b) $L(y_0; a, b) = \mu$, де $\mu = \nu_p(ay_0^2)$, якщо $y_\mu \equiv y_0 (mod p^m)$ і $m > \nu_p(f(y_0)) = 1$;

(c) $L(y_0; a, b) = \mu p^{m-t+2}$, де $t = \nu_p(y_\mu - y_0)$, якщо $m > \nu_p(f(y_0)) = 1$ і $3 \leq t \leq m$;

7) нехай $(aby_0(b^2+4a)f(y_0), p) = 1, p > 2, y_0 \equiv \frac{b}{2} (mod p), \lambda = o(m_f)$ - непарне або $o(M_f)$ не ділить λ. Тоді

(a) $L(y_0; a, b) = \lambda$, якщо $y_\lambda \equiv y_0 (mod p^m)$,

(b) $L(y_0; a, b) = \lambda p^{m-t+2}$, де $t = \nu_p(y_\lambda - y_0)$ і $2 \le t \le m$.

Нідеррайтер і Шпарлінський знайшли нетривіальні оцінки дискрепансії D_N для відповідних послідовностей псевдовипадкових чисел, що породжені інверсними конгруентними генераторами (3.2) і (3.3).

В першій главі дисертаційної роботи ми будуємо нові генератори псевдовипадкових чисел, які узагальнюють генератор (3.3).

А саме, нехай $p > 2$ - просте число, $m \ge 3$ - натуральне, $a, b, c \in Z_{p^m}$, $(a, p) = 1$, $b \equiv c \equiv 0 (\mathrm{mod} p)$, $(\omega_0, p) = 1$, $\nu_p(b) = \nu \le \nu_p(c) = \mu$, $bc \equiv 0 (\mathrm{mod} p^m)$.

Ми розглядаємо два генератори:

$$\omega_{n+1} \equiv a\omega_n^{-1} + b + c(n+1)\omega_0 (\mathrm{mod} p^m), n = 0,1,2, \dots \quad (3.4)$$

$$\omega_{n+1} \equiv a\omega_n^{-1} + b + c\omega_n (\mathrm{mod} p^m), n = 0,1,2, \dots \quad (3.5)$$

Генератор типу (5) у випадку $p = 2$ досліджувався Ейченауером, Лєхном, Топузоглу, Ейченауером і Нідеррайтером, Ейченауером і Гросом, Като, Ву і Янагіхара. Але випадок $p > 2$ суттєво відрізняється від випадку $p = 2$, а дослідження тригонометричних сум на відповідних послідовностях псевдовипадкових чисел мають різні підходи.

Генератор типу (3.4) був введений і досліджувався С. П. Варбанцем, зокрема в його кандидатській дисертації, і отримав називу генератору зі змінним зсувом.

Головна різниця нашого методу дослідження генераторів (3.3),(3.4) і (3.5) від методів робіт Ейченауера, Нідеррайтера, Като та інших полягає у тому, що ми будуємо, а потім і суттєво використовуємо, два зображення ω_n як поліномів від ω_0 і ω_0^{-1} (з коефіцієнтами, що залежать від k) або як поліномів від k (з коефіцієнтами, які явно залежать від ω_0 та ω_0^{-1}).

Генератори (3.3) та (3.4) породжують послідовності псевдовипадкових чисел за рахунок росту не тільни простого числа, але й показника степені m. В роботах Ейченауера і Емеріха, Сал’є і Зинов'єва, Левіна, Нідеррайтера і Вінтерхофа розглядаються складені конгруентні генератори, які також

дозволяють збільшити період послідовності псевдовипадкових чисел без використання великих простих чисел.

В наступних статтях та обзорах можна ознайомитись з дослідженнями рівномірності розподілу псевдовипадкових чисел, їх узагальненями і практичним застосуванням в наближеному аналізі та криптографії:

[1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12],[13],[14],[15],[16],[17].

РОЗДІЛ 4
ІНВЕРСНИЙ КОНГРУЕНЦІАЛЬНИЙ ГЕНЕРАТОР ПСЕВДОВИПАДКОВИХ ЧИСЕЛ ЗА МОДУЛЕМ СТЕПЕНІ ПРОСТОГО ЧИСЛА

Нехай $p \geq 3$ просте і $m \geq 2$ ціле, та нехай R_m (відповідно, R_m^*) означає групу класів лишків (відповідно, групу зведених класів лишків) за модулем p^m. Для $a \in R_m^*,\ b, c \in R_m,\ b \equiv c \equiv 0(modp)$, ми розглядаємо відображення $\Psi_r, r = 0,1,2, \dots, R_m^* \to R_m$ виду

$$\Psi_{r+1}(\omega) = \frac{a}{\Psi_r(\omega)} + b + c\omega, \Psi_0(\omega) = \omega \tag{4.1}$$

(тут і надалі ми, задля зручності, будемо писати $\frac{u}{v}$ для виразу uv^{-1} в мультиплікативній абелевій групі R_m^*).

Умова $b \equiv c \equiv 0(modp)$ гарантує, що $(\Psi_r(\omega), p) = 1$, якщо $(a, p) = (\omega, p) = 1$.

Рекурсія (4.1) узагальнює інверсний конгруенціальний генератор за модулем степені простого.

4.1 Допоміжні результати

Для $k \in Z,\ k \geq 0$, ми розглядаємо відображення Ψ_k, що отримується із рекурсії (4.1).

Лема 4.1 Для кожного цілого k, $k \geq 0$, маємо

$$\Psi_k(\omega) = \frac{A_0^{(k)} + A_1^{(k)}\omega + \dots + A_{k+1}^{(k)}\omega^{k+1}}{B_0^{(k)} + B_1^{(k)}\omega + \dots + B_k^{(k)}\omega^k}, \tag{4.2}$$

окрім того,

$$\begin{gathered}\left(A_0^{(k)}, p\right) = \left(B_1^{(k)}, p\right) = 1, A_i^{(k)} \equiv B_j^{(k)} \equiv 0(modp), \\ i = 1,2, \dots, k+1, j = 0,2,3, \dots, k,\ \textit{якщо k непарне ціле};\end{gathered} \tag{4.3}$$

$$\begin{gathered}\left(A_1^{(k)}, p\right) = \left(B_0^{(k)}, p\right), A_i^{(k)} \equiv B_j^{(k)} \equiv 0(modp), \\ i = 0,2,3, \dots, k+1, j = 1,2, \dots, k, \textit{якщо k парне ціле}.\end{gathered}$$

До того ж, для $k = 0,1,2, \ldots$

$$\begin{cases} A_0^{(k+2)} = (a+b^2)A_0^{(k)} + abB_0^{(k)} \\ A_1^{(k+2)} = 2bcA_0^{(k)} + (a+b^2)A_1^{(k)} + acB_0^{(k)} + abB_1^{(k)} \\ A_i^{(k+2)} = c^2A_{i-2}^{(k)} + 2bcA_{i-1}^{(k)} + (a+b^2)A_i^{(k)} + acB_{i-1}^{(k)} + abB_i^{(k)}, \\ i = 2,3, \ldots, k+1; \\ A_{k+2}^{(k+2)} = 2bcA_{k+1}^{(k)} + c^2A_{k+1}^{(k)} \\ A_{k+3}^{(k+2)} = c^2A_{k+1}^{(k)} \\ B_0^{(k+2)} = aB_0^{(k)} + bA_0^{(k)} \\ B_j^{(k+2)} = aB_j^{(k)} + bA_j^{(k)} + cA_{j-1}^{(k)}, j = 1,2, \ldots, k; \\ B_{k+1}^{(k+2)} = bA_{k+1}^{(k)} + cA_k^{(k)} \\ B_{k+2}^{k+2} = cA_{k+1}^{(k)} \\ A_0^{(0)} = 0, B_0^{(0)} = 1, A_1^{(0)} = 1, B_1^{(0)} = 0, A_i^{(0)} = B_i^{(0)}, i > 1; \\ A_0^{(1)} = a, B_0^{(1)} = 0, A_1^{(1)} = b, B_1^{(1)} = 1, A_2^{(1)} = c, B_2^{(1)} = 0, \\ A_i^{(1)} = B_i^{(1)} = 0, i > 2; \\ A_0^{(2)} = ab, A_1^{(2)} = (a+ac+b^2), A_2^{(2)} = 2bc, A_3^{(2)} = c^2, B_0^{(2)} = a, B_1^{(2)} = b, \\ B_2^{(2)} = c, A_i = B_i = 0, i > 3, j > 2. \end{cases} \quad (4.4)$$

ДОВЕДЕННЯ. Формула (4.2) та співвідношення (4.4) можуть бути отримані індукцією по k в формулі (4.1). Тоді (4.3) отримується із (4.4).

□

Лема 4.2 Нехай $A_i^{(k)}$, $B_i^{(k)}$ будуть такі ж самі, як і в Лемі 4.1 і нехай $bc \equiv 0(\mathrm{mod} p^m)$, $\nu_p(b) \leq \nu_p(c)$. Тоді мають місце формули

$$\Psi_{2k}(\omega) = \frac{A_0^{(2k)} + A_1^{(2k)}\omega}{B_0^{(2k)} + B_1^{(2k)}\omega + B_2^{(2k)}\omega^2}, \quad (4.5)$$

$$\Psi_{2k+1}(\omega) = \frac{C_0^{(2k+1)} + C_1^{(2k+1)}\omega + C_2^{(2k+1)}\omega^2}{D_0^{(2k+1)} + D_1^{(2k+1)}\omega}, \quad (4.6)$$

де

$$\begin{gathered} A_0^{(2k)} = ka^k b + \overline{A}_0^{(2k)} b^3, A_1^{(2k)} = a^k + k\overline{A}_1^{(2k)} b^2; \\ B_0^{(2k)} = a^k + \overline{B}_0^{(2k)} b^2, B_1^{(2k)} = ka^{k-1} b + \overline{B}_1^{(2k)} b^3, B_2^{(2k)} = ka^{k-1} c; \\ C_0^{(2k+1)} = a^{k+1} b + \overline{C}_0^{(2k+1)} b^2, C_1^{(2k+1)} = (k+1)a^k b + \overline{C}_1^{(2k+1)} b^3, \\ C_2^{(2k+1)} = (k+1)a^k c; \\ D_0^{(2k+1)} = ka^k b + \overline{D}_0^{(2k+1)} b^3, D_1^{(2k+1)} = a^k + ka^k c + \overline{D}_1^{(2k+1)} b^2. \end{gathered} \quad (4.7)$$

ДОВЕДЕННЯ. Позначимо

$$A = \begin{pmatrix} a + b^2 & ab \\ b & a \end{pmatrix}, B = \begin{pmatrix} 0 & ac \\ c & 0 \end{pmatrix}$$

Так як $bc \equiv 0(mod p^m)$ і $\nu_p(b) \leq \nu_p(c)$, ми маємо $c^2 \equiv 0(mod p^m)$.

Тому по Лемі 4.1 після невеликих обчислень ми можемо зробити висновок, що

$$\begin{pmatrix} A_0^{(2k+2)} \\ B_0^{(2k+2)} \end{pmatrix} = A^{k+1} \begin{pmatrix} A_0^{(0)} \\ B_0^{(0)} \end{pmatrix} = A^{k+1} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (4.8)$$

$$\begin{gathered} \begin{pmatrix} A_1^{(2k+2)} \\ B_1^{(2k+2)} \end{pmatrix} = A \begin{pmatrix} A_1^{(2k)} \\ B_1^{(2k)} \end{pmatrix} + B \begin{pmatrix} A_0^{(2k-2)} \\ B_0^{(2k-2)} \end{pmatrix} = \\ = A^k \begin{pmatrix} A_1^{(2)} \\ B_1^{(2)} \end{pmatrix} + ka^{k-1} c \begin{pmatrix} A_0^{(1)} \\ B_0^{(1)} \end{pmatrix} = A^k \begin{pmatrix} a + ac + b^2 \\ b \end{pmatrix} + ka^{k-1} c \begin{pmatrix} a \\ 0 \end{pmatrix} \end{gathered} \quad (4.9)$$

За формулою (4.7) ми можемо отримати

$$A = a(E + A_1), E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, A_1 = \begin{pmatrix} a_1 b & b \\ a_1 & 0 \end{pmatrix}, a_1 \equiv ba^{-1}(modp^m)$$
$$A^k \equiv a^k \left(E + kA_1 + \cdots + \binom{k}{M} A_1^M \right)(modp^m), \quad (4.10)$$
$$A_1^j \equiv 0(modp^{\nu(j-1)}),$$

де $\nu = \nu_p(b)$, $M = \left[\frac{m}{\nu}\right]$.

Таким чином з формул (4.1), (4.8)-(4.10) легко випливає твердження Леми 4.2.

□

Зауваження. У загальному випадку, нехай $b^{\ell}c \equiv 0(modp^m)$, $b^{\ell-1}c \not\equiv 0(modp^m)$, $\ell \geq 2$ $\nu_p(b) = \nu \leq \nu_p(c)$. Тоді подібним чином ми отримуємо

$$\Psi_{2k}(\omega) = \frac{A_0^{(2k)} + A_1^{(2k)}\omega + \cdots + A_\ell^{2k}\omega^\ell}{B_0^{(2k)} + B_1^{(2k)}\omega + \cdots + B_{\ell+1}^{(2k)}\omega^\ell},$$

$$\begin{pmatrix} A_i^{(2k+2)} \\ B_i^{(2k+2)} \end{pmatrix} = A^{k+1-i} \begin{pmatrix} A_i^{(2i)} \\ B_i^{(2i)} \end{pmatrix} + B_k \begin{pmatrix} A_{i-1}^{(2i-2)} \\ B_{i-1}^{(2i-2)} \end{pmatrix} + C_k \begin{pmatrix} A_{i-2}^{(2i-4)} \\ B_{i-2}^{(2i-4)} \end{pmatrix}, a \leq i \leq \ell,$$

де матриці B_k, C_k задовільняють конгруенції $B_k \equiv C_k(modp^\nu)$.

Лема 4.3 Для кожного $k = 1,2, \ldots$, відображення $\Psi_k(\omega)$ - є підстановка на R_m^*.

ДОВЕДЕННЯ. Для $c \equiv 0(modp^m)$ твердження леми є очевидним.

Нехай $\nu_p(c) < m$. Із $b \equiv c \equiv 0(modp)$ випливає, що $(\Psi_k(\omega), p) = 1$ для кожного $\omega \in R_m^*$.

Далі, нехай $k = 1$. Із співвідношення

$$a\omega_1^{-1} + b + c\omega_1 \equiv a\omega_2^{-1} + b + c\omega_2(modp^m)$$

випливає, що $a\omega_1^{-1} \equiv a\omega_2^{-1}$ і тому $\omega_1 \equiv \omega_2(modp)$.

Покладемо $\omega_2 = \omega_1 + p\delta_1$. Тоді $\omega_2^{-1} \equiv \omega_1^{-1}(1 - p\delta_1\omega_1^{-1})(modp^2)$ (тут $\omega_1^{-1}\omega_1 \equiv 1(modp^m)$).

Тому

$$a\omega_1^{-1} + b + c\omega_1 \equiv a(\omega_1^{-1} - p\delta_1\omega_1^{-2}) + b + c(\omega_1 + p\delta_1)(modp^2) \Rightarrow$$

$$p\delta_1\omega_1^{-2} \equiv 0(modp^2) \Rightarrow \delta_1 = p\delta_2 \Rightarrow \omega_2 = \omega_1 + p^2\delta_2 \Rightarrow$$

$$\Rightarrow \omega_2^{-1} \equiv \omega_1^{-1} - p^2\delta_2\omega_1^{-2}(modp^3).$$

Тепер із співвідношення

$$a\omega_1^{-1} + b + c\omega_1 \equiv a(\omega_1^{-1} - p^2\delta_2\omega_1^{-2}) + b + c(\omega_1 + p^2\delta_2)(modp^3)$$

відповідним чином ми отримуємо $\omega_2 = \omega_1 + p^3\delta_3$. Через m ітерацій ми отримаємо $\omega_2 \equiv \omega_1(modp^m)$, тобто для k=1 твердження леми доведено.

Припустимо, що для $k-1$ ми маємо

$$\Psi_{k-1}(\omega_1) \equiv \Psi_{k-1}(\omega_2)(modp^m) \Leftrightarrow \omega_1 \equiv \omega_2(modp^m).$$

Тоді розглянемо співвідношення

$$\frac{a}{\Psi_{k-1}(\omega_1)} + b + c\omega_1 \equiv \frac{a}{\Psi_{k-1}(\omega_2)} + b + c\omega_2(modp^m).$$

Якщо $\omega_1 \equiv \omega_2(modp^{m-\nu_p(c)})$, тоді ми маємо

$$\Psi_{k-1}(\omega_1) \equiv \Psi_{k-1}(\omega_2)(modp^m) \Rightarrow \omega_1 \equiv \omega_2(modp^m).$$

Припустимо, що $\omega_1 \not\equiv \omega_2(modp^{m-\nu_p(c)})$.

Але тоді ми маємо

$$\frac{a}{\Psi_{k-1}(\omega_1)} + b + c\omega_1 \equiv \frac{a}{\Psi_{k-1}(\omega_2)} + b + c\omega_2(modp^{\nu_p(c)}) \Rightarrow$$

$$\frac{a}{\Psi_{k-1}(\omega_1)} \equiv \frac{a}{\Psi_{k-1}(\omega_2)}(modp^{\nu_p(c)}) \Rightarrow \omega_1 \equiv \omega_2(modp^{\nu_p(c)}) \Rightarrow$$

$$\frac{a}{\Psi_{k-1}(\omega_1)} + b + c\omega_1 \equiv \frac{a}{\Psi_{k-1}(\omega_2)} + b + c\omega_2(modp^{2\nu_p(c)}) \Rightarrow$$

$$\Rightarrow \omega_1 \equiv \omega_2(modp^{2\nu_p(c)}).$$

Отже, через $\left[\frac{m}{\nu_p(c)}\right]$ кроків ми отримуємо протиріччя із припущенням $\omega_1 \not\equiv \omega_2(modp^{m-\nu_p(c)})$,

доводячи таким чином твердження Леми 4.3.

□

Лема 1.4 Нехай $q > 1$ натуральне, $p > 3$ просте, $a \in Z$, $m \in N, m \geq 2$ і нехай $f(x) = A_1x + A_2x^2 + p(A_3x^3 + \cdots)$ поліном над Z, окрім того, $(A_1, A_2, p) = 1$. Тоді

$$\sum_{x=0}^{q-1} e^{2\pi i\frac{ax}{q}} = \begin{cases} q & ,якщо a \vdots q, \\ 0 & ,якщо a \nmid q. \end{cases} \qquad (4.11)$$

$$\left|\sum_{x\in R_m} e^{2\pi i\frac{f(x)}{p^m}}\right| = \begin{cases} 0 & ,якщо (A_1, p) = 1, A_2 \vdots p, \\ p^{\frac{m}{2}} & ,якщо (A_2, p) = 1. \end{cases} \qquad (4.12)$$

Твердження леми 4.4 добре відомі.

Лема 4.5 Нехай $f(\omega) = B\omega + C\omega^2 + p(D\omega^3 + \cdots)$ є поліном над Z, і нехай $(C, p) = 1$. Тоді для кожного $A \in Z$ ми маємо

$$T = \left|\sum_{\omega\in R_m^*} e_{p^m}(A\omega + f(\omega^{-1}))\right| \leq 4p^{\frac{m}{2}}. \qquad (4.13)$$

ДОВЕДЕННЯ. Візьмемо

$$\omega^{-1} = u + p^{m-1}z,$$

$$\omega \equiv u^{-1} - p^{m-1}u^{-2}z (mod p^m),$$

$$\omega^{-2} = u^2 + 2p^{m-1}uz.$$

Звідси ми отримуємо

$$A\omega + f(\omega^{-1}) \equiv (Au^{-1} + Bu + Cu^2 + p(Du^3 + \cdots)) +$$

$$+p^{m-1}(-u^{-2}A + B + 2Cu)z (mod p^m).$$

Тоді по Лемі 4.4

$$T = \left| \sum_{u \in R_{m-1}^*} e_{p^m}(Au^{-1} + Bu + Cu^2 + p(Du^3 + \cdots)) \times \right.$$

$$\left. \times \sum_{z \in R_1} e_p((-Au^{-2} + B + 2Cu)z) \right| \leq$$

$$\leq p \left| \sum_{\substack{u_0 \in R_1^* \\ Au_0^{-1} - Bu_0 - 2Cu_0^2 \equiv 0 (mod p)}} e_{p^m}(Au_0^{-1} + Bu_0 + Cu_0^2 + p(Du^3 + \cdots)) \times \right.$$

$$\left. \times \sum_{v \in R_{m-2}} e_{p^{m-2}}(A_1 v + A_2 v^2 + p(A_3 v^3 + \cdots)) \right| \leq 4p^{\frac{m}{2}},$$

(тут, $A_1 = \frac{-Au_0^{-1} + B + 2Cu_0}{p} \in Z$, $A_2 = C$, $A_j \in Z$, $j = 3,4, \ldots$). □

Розглянемо послідовність $\{\omega_n\}$, $n = 0,1,2, \ldots$, яка визначається із рекурентного співвідношення

$$\omega_n = \frac{a}{\omega_{n-1}} + b + c\omega, \omega_0 = \omega \in R_m^*,$$

тобто $\omega_n = \Psi_n(\omega)$.

З лем 4.1 та 4.3 випливає, що ця послідовність є виключно періодичною із найменшою довжиною періоду $\tau \leq p^{m-1}(p-1)$, де τ є парне ціле. У випадку $c \equiv 0 (mod p^m)$ В.-С. Чоу дослідив залежність τ від a, b, ω. Нижче ми вивчаємо τ, як функцію від a, b, c та ω.

Для позитивного цілого h ми визначаємо

$$\sigma_{k,\ell}(h, p^m) = \sigma_{k,\ell} := \sum_{\omega \in R_m^*} e_{p^m}\left(h\big(\Psi_k(\omega) - \Psi_\ell(\omega)\big)\right). \qquad (4.14)$$

Лема 4.6 Нехай k, ℓ ненегативні цілі, що приймають значення різної парності, та нехай $h \in Z$, $(h, p^m) = p^\delta$, $\delta < m$. Тоді ми маємо

$$|\sigma_{k,\ell}| \leq 2p^{\frac{m+\delta}{2}}$$

ДОВЕДЕННЯ. Не відходячи від загального ми можемо вважати, що $(h, p^m) = 1$ і $k = 2k_1$, $\ell = 2\ell_1 + 1$.

Із Леми 4.1 достатньо легко випливає

$$\Psi_{2k_1}(\omega) = \frac{k_1 a^{k_1} b + \left(a^{k_1} + p^\nu A_1\right)\omega + p^\nu (A_1\omega^2 + pA_3\omega^3 + \cdots)}{(a^{k_1} + p^\nu B_0) + (ka^{k_1} b + p^\nu B_2)\omega + p^\nu (B_2\omega^2 + \cdots)}$$

$$\Psi_{2\ell_1+1}(\omega) = \frac{\left(a^{\ell_1+1} + C_0 p^\nu\right) + \left((\ell_1+1)a^\ell b + c_1 p^{2\nu}\right)\omega + \left((\ell+1)a^\ell c + p^{\nu_p(c)+\nu} C_2\right)\omega^2 + \cdots}{(\ell_1 a^{\ell_1} b + D_0 b^3) + \left(a^{\ell_1} + p^{min(\nu_p(c),2\nu)} D_1\right)\omega + p^{\nu+\nu_p(c)}(D_2\omega^2 + \cdots)}, \quad (4.15)$$

де $\nu = \nu_p(b) \le \nu_p(c)$.

Помножимо чисельник і знаменник $\Psi_{2k_1}(\omega)$ на $(a^{k_1} + p^\nu B_0)^{-1}(mod p^m)$, а також помножимо чисельник і знаменник $\Psi_{2\ell_1+1}(\omega)$ на $((a^{\ell_1} + p^{min(\nu_p(c),2\nu_p(b))})\omega)^{-1}(mod p^m)$.

З конгруенції

$$(1 + p(A\omega + \cdots))^{-1} \equiv 1 - p(A\omega + \cdots) + p^2(A\omega + \cdots)^2 + \cdots (mod p^m)$$

$$(1 + pB\omega^{-1} + pB'\omega + \cdots)^{-1} \equiv 1 - p(B\omega^{-1} + B'\omega + \cdots) +$$

$$+p^2(B\omega^{-1} + B'\omega + \cdots)^2 + \cdots (mod p^m)$$

після невеликих обчислень ми отримуємо

$$\Psi_{2k_1}(\omega) - \Psi_{2\ell_1+1}(\omega) \equiv E_0 + E_1\omega + E_{-1}\omega^{-1} + p^\nu G(\omega, \omega^{-1})(mod p^m),$$

де $E_1 \equiv 1(mod p^\nu)$, $E_{-1} \equiv a(mod p^\nu)$, $G(x, y)$ є поліном від x, y.

Позначимо $E(\omega) = \Psi_{2k_1}(\omega) - \Psi_{2\ell_1+1}(\omega)$.

Візьмемо $\omega = u + p^{m-1}z$, $u \in R^*_{m-1}$, $z = 0,1, \ldots, p-1$.

Тоді

$$\omega^j \equiv u^j + jp^{m-1}u^{j-1}z, \omega^{-j} \equiv u^{-j} - jp^{m-1}u^{-j-1}z(mod p^m).$$

Звідси, після простих обчислень, ми маємо

$$\sigma_{k,\ell} = \sum_{u\in R^*_{m-1}} \sum_{z=0}^{p-1} e_{p^m}\left(E_1 u + E_{-1}u^{-1} + p^{\nu}\left(E_{2'}u^2 + E'_{-2}u^{-2} + \cdots\right) + \right.$$

$$\left. + p^{m-1}(E_1 - E_{-1}u^{-2})z\right) = \qquad (4.16)$$

$$= p \sum_{\substack{u\in R^*_{m-1} \\ u^2 \equiv E_1^{-1}E_{-1}(mod p)}} e_{p^m}\left(E_1 u + E_{-1}u^{-1} + p^{\nu}\left(E_{2'}u^2 + E'_{-2}u^{-2} + \cdots\right)\right).$$

Нехай u_1, u_2 - це два розв'язки конгруенції

$$u_i^2 \equiv E_{-1}E_1^{-1}(mod p^{m-1}), i = 1{,}2.$$

Покладемо $u = u_i + pv$, $v \in R_{m-2}$. Тоді маємо

$$\sigma_{k,\ell} = p\sum_{i=1}^{2}\sum_{v\in R_{m-2}} e^{2\pi i h\frac{F_1^{(i)}(v)+F_2^{(i)}(v)}{p^{m-2}}}, \qquad (4.17)$$

де $F_1^{(i)} = \frac{E_1 - E_{-1}u_i^{-2}}{p} + 2(-E_1 u_i + E_{-1}u_i^{-3})(mod p^{m-2})$,

$F_2^{(i)}(v) = E_2 v^2 + p(E_3 v^3 + \cdots),, E_2 \equiv E_{-1}(mod p), (E_2, p) = 1.$

За Лемою 4.4 внутрішня сума по v в рівнянні (4.17) оцінюється як $p^{\frac{m-2}{2}}$. Тоді, $|\sigma_{k,\ell}| \le 2p^{\frac{m}{2}}$.

□

Наслідок: найменший період послідовності $\Psi_k(\omega)$ не може бути непарним цілим.

Лема 4.7 Нехай k, ℓ ненегативні цілі, що приймають значення однакової парності, $h \in Z$, $\nu_p(h, p^m) = \delta$, $\nu_p(k - \ell) = \kappa$ і нехай $bc \equiv 0(\mathrm{mod} p^m)$, $1 \le \nu_p(b) \le \nu_p(c) < m$. Тоді

$$\left|\sigma_{k,\ell}(h)\right| \le \begin{cases} p^m, \text{якщо} m - \delta - \kappa - \nu \le 0, \\ 2p^{\frac{m+\delta+\kappa+\nu}{2}}, \text{якщо} m - \delta - \kappa - \nu > 0, \end{cases} \qquad (4.18)$$

де $\nu_p(b) = \nu$.

ДОВЕДЕННЯ. Спочатку припустимо, що $b^2 \equiv 0(\mathrm{mod} p^m)$. Так як b і c задовільняють умові $bc \equiv 0(\mathrm{mod} p^m)$, по Лемі 4.1 для $k = 2,3, \ldots$ ми маємо

$$\Psi_{2k}(\omega) = \frac{ka^k b + (a^k + ka^k c)\omega}{a^k + ka^{k-1}b\omega + ka^{k-1}c\omega^2}, \tag{4.19}$$

$$\Psi_{2k+1}(\omega) = \frac{a^{k+1} + (k+1)a^k b + (k+1)a^k c\omega^2}{ka^k b + (a^k + ka^k c)\omega}. \tag{4.20}$$

Звідси,

$$\Psi_{2k}(\omega) - \Psi_{2\ell}(\omega) = \frac{E_0 + E_1\omega + E_2\omega^2 + E_3\omega^3}{F_0 + F_1\omega + F_2\omega^2}, \tag{4.21}$$

де

$$E_0 = (k-\ell)a^{k+\ell}b,$$

$$E_1 = (k-\ell)a^{k+\ell}c,$$

$$E_2 = (k-\ell)a^{k+\ell-1}b,$$

$$E_3 = -(k-\ell)a^{k+\ell-1}c,$$

$$F_0 = a^{k+\ell}, F_1 = (k+\ell)a^{k+\ell-1}b, F_2 = (k+\ell)a^{k+\ell-1}c.$$

Аналогічно, можна довести, що

$$\Psi_{2k+1}(\omega) - \Psi_{2\ell+1}(\omega) = \frac{E_{0'} + E_{1'}\omega + E_{2'}\omega^2 + E_{3'}\omega^3}{F_{0'} + F_{1'}\omega + F_{2'}\omega^2} \tag{4.22}$$

де

$$\begin{cases} E_{0'} = (k-\ell)a^{k+\ell-1}b, E_{1'} = -(k-\ell)a^{k+\ell-1}c, \\ E_{2'} = (k-\ell)a^{k+\ell}b, E_{3'} = (k-\ell)a^{k+\ell}c, \\ F_{0'} = 0, F_{1'} = (k+\ell)a^{k+\ell}b, F_{2'} = a^{k+\ell}(1 + (k+\ell)c) \end{cases}$$

Тепер, аналогічно доведенню Леми 4.6, ми маємо, що за модулем p^m

$$\begin{cases} \Psi_{2k}(\omega) - \Psi_{2\ell}(\omega) \equiv (k-\ell)(b + c\omega - a^{-1}b\omega^2 - a^{-1}c\omega^3) \\ \Psi_{2k+1}(\omega) - \Psi_{2\ell+1}(\omega) \equiv (k-\ell)\omega^{-2}(1-(k+\ell)c) \times \\ \times (a^{-1}b - a^{-1}c\omega - b\omega^2 + c\omega^3)(1-(k+\ell)b\omega^{-1}) \equiv \\ \equiv (k-\ell)(c\omega + b - a^{-1}c\omega^{-1} - a^{-1}b\omega^{-2}). \end{cases} \tag{4.23}$$

У загальному випадку для $b \equiv c \equiv 0(modp)$, $bc \equiv 0(modp^m)$, $b^2 \not\equiv 0(modp^m)$, ми маємо, що для $k = 2,3, \ldots$

$$\Psi_{2k}(\omega) = \frac{(ka^k b + A_k a^k b^3) + (a^k + ka^k c + B_k a^{k-1} b^2)\omega}{(a^k + C_k a^{k-1} b^2) + (ka^{k-1} b + D_k a^{k-1} b^3)\omega + ka^{k-1} c\omega^2}, \tag{4.24}$$

$$\Psi_{2k+1}(\omega) = \frac{(a^{k+1} + C_k a^k b^2) + \left((k+1)a^k b + D_{k+1} a^{k-1} b^3\right)\omega + (k+1)a^k c\omega^2}{(ka^k b + A_k a^k b^3) + (a^k + ka^k c + B_k a^{k-1} b^2)\omega} \tag{4.25}$$

де

$$A_k = \frac{k(k+1)(k+2)}{12} - 1 + p^{2\nu} F_A(k);$$

$$B_k = \frac{k(k+1)}{2} + p^{2\nu} F_B(k);$$

$$C_k = \frac{(k-1)k}{2} + p^{2\nu} F_C(k);$$

$$D_k = \frac{k(k^2-1)}{6} + p^{2\nu} F_D(k);$$

$F_A(k), F_B(k), F_C(k), F_D(k) \in Z[k]$.

Тому, як і в формулі (1.23) ми отримуємо

$$\Psi_{2k}(\omega) - \Psi_{2\ell}(\omega) \equiv$$

$$\equiv (k-\ell)[f_0 + f_1\omega + f_2\omega^2 + f_3\omega^3 + p^{2\nu}\omega^4 G_0(\omega)](modp^m)$$

$$\Psi_{2k+1}(\omega) - \Psi_{2\ell+1}(\omega) \equiv \tag{4.26}$$

$$\equiv (k-\ell)[e_1\omega + e_0 + e_{-1}\omega^{-1} + e_{-2}\omega^{-2} + p^{2\nu}\omega^{-3} G_1(\omega^{-1})](modp^m)$$

де

$f_0 \equiv b, f_1 \equiv c, f_2 \equiv a^{-1}b, f_3 \equiv -a^{-1}c(modp^{2\nu})$;

$e_1 \equiv c, e_0 \equiv b, e_{-1} \equiv -a^{-1}c, e_{-2} \equiv a^{-1}b(modp^{2\nu})$;

$G_0(\omega), G_1(\omega) \in Z[\omega]$.

Позначимо $\nu_p(c) = \mu$, $m_0 = m - \nu - \delta - \kappa$, $c_0 = cp^{-\mu}$, $b_0 = bp^{-\nu}$, $h + 0 = hp^{-\delta}$.

Якщо $\nu < \mu$, тоді по формулі (4.23) або (4.26), використовуючи Леми 4.4 та 4.5, для парних чисел k, ℓ виводимо

$$\left|\sigma_{k,\ell}(h)\right| \le p^{\delta+\kappa+\nu}\left|\sum_{\omega\in R^*_{m_1}} e_{p^{m_0}}\left(h_0\left(c_0p^{\mu-\nu}\omega + a^{-1}b_0\omega^2 + pG_0(\omega)\right)\right)\right| \le \tag{4.27}$$

$$\le \begin{cases} p^{\frac{m+\delta+\kappa+\nu}{2}}, \text{якщо} \delta+\kappa+\nu < m, \\ p^m, \text{якщо} \delta+\kappa+\nu \ge m, \end{cases}$$

або для непарних чисел k, ℓ

$$|\sigma_{k,\ell}(h)| \le$$

$$\le p^{\delta+\kappa+\nu}\left|\sum_{\omega\in R^*_{m_0}} e_{p^{m_0}}\left(h_0\left(c_0p^{\mu-\nu}\omega - a^{-1}c_0p^{\mu-\nu} + a^{-1}b_0\omega^{-2} + pG(\omega^{-1})\right)\right)\right| \le \tag{4.28}$$

$$\le \begin{cases} 2p\dfrac{m+\delta+\kappa+\nu}{2}, \text{якщо} \delta+\kappa+\nu < m, \\ p^m, \text{якщо} \delta+\kappa+\nu \ge m. \end{cases}$$

Застосовуючи леми 4.4 та 4.5 ми отримаємо твердження леми 4.7.

□

Зауваження. Оцінка $|\sigma_{k,\ell}|$ справедлива і в загальному випадку, коли $b \equiv c \equiv 0(\mathrm{mod}p)$, $b \not\equiv 0(\mathrm{mod}p^m)$. Лише у цьому випадку представлення для $\Psi_k(\omega)$ стає дуже громіздким.

Лема 4.8 Нехай $b \equiv c \equiv 0(\mathrm{mod}p)$, $bc \equiv 0(\mathrm{mod}p^m)$ та нехай τ є найменшою довжиною періоду послідовності $\Psi_k(\omega)$, $k = 0,1,2,\ldots$, за модулем p^m. Тоді $\tau = 2p^{m-\nu}$, якщо $ab + ac\omega + b\omega^2 - c\omega^3 \not\equiv 0(\mathrm{mod}p^{\nu+1})$

ДОВЕДЕННЯ. Якщо $ab - ac\omega + b\omega^2 - c\omega^3 \equiv 0(\mathrm{mod}p^{\nu+1})$, тоді послідовність $\{\Psi_k(\omega)\}$ має максимальну довжину періоду серед усіх

інверсних конгруенціальних псевдовипадкових генераторів (4.1) за модулем p^m. Окрім того, $\tau = 2p^{m-\nu}$.

Це твердження випливає з (4.20)-(4.26) і умови

$$\omega_{2k} - \omega_{2\ell} \equiv 0 (\mathrm{mod} p^m)$$

В цьому випадку береться до уваги, що

$$B_k - B_\ell, D_k B_\ell - D_\ell B_k, A_k - A_\ell, A_k C_\ell - A_\ell C_k, \ell C_k - k C_\ell, C_k - C_\ell$$

ділиться на $k - \ell$.

Так як найменша довжина періоду τ є парним цілим, ми маємо $\tau = 2p^{m-\nu}$.

РОЗДІЛ 5
ПРОЕКТУВАННЯ ТА РОЗРОБКА ГЕНЕРАТОРУ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ НА ОСНОВІ ІНВЕРСНОГО КОНГРУЕНТНОГО МЕТОДУ

Однією з головних цілей цієї роботи є розробка генератора псевдовипадкових чисел на основі інверсного конгруентного методу. Цей метод є особливо цікавим з кількох причин:

1). Простота реалізації: інверсний конгруентний метод використовує просту математичну формулу для генерації чисел, що робить його виконання в програмах досить простим і ефективним.

2). Періодичність: інверсний конгруентний метод має довгий період генерації псевдовипадкових чисел перед тим, як він почне повторюватися. Це означає, що генератор може створювати значну кількість унікальних чисел, що зручно для багатьох застосувань.

3). Властивості розподілу: інверсний конгруентний метод може генерувати числа, які мають рівномірний розподіл у великому діапазоні значень. Це важливо для багатьох задач, де необхідно отримати рівномірно розподілені випадкові числа.

4). Можливість повторного використання: інверсний конгруентний метод дозволяє зберігати поточний стан генератора і використовувати його пізніше для продовження генерації псевдовипадкових чисел. Це робить його зручним для використання в алгоритмах, які вимагають багато випадкових чисел з одного генератора.

Таким чином, інверсний конгруентний метод є привабливим вибором для розробки генератора псевдовипадкових чисел, оскільки він поєднує простоту реалізації, довгий період, властивості розподілу та можливість повторного використання.

5.1 Постановка задачі

До основних задач програмного застосунку відносяться:

1) генерація псевдовипадкових чисел на основі введених користувачем початкових значень x та y, та y кількості генерації цих чисел, введеної користувачем;

2) збереження випадкових значень x та y і їх відображення на графіку;

3) окреме вираховування розподілу Гаусса-Кузьміна, середнього значення та дисперсії.

Основними критеріями гарної роботи програми буде рівномірне заповнення точками координатної площини та максимально щільне її заповнення точками зі збільшенням їх кількості генерації. Саме такі критерії забезпечать високу «якість» ймовірності.

5.2 Інструменти реалізації

Для реалізації даного проекту були обрані середовище розробки Visual Studio Code та високорівнева мова програмування Python.

Visual Studio Code (VS Code) є однією з найпопулярніших серед розробників середовищ розробки з відкритим вихідним кодом. Це легковагове і потужне середовище, яке надає широкий спектр функцій для програмістів.

Використання VS Code для розробки програми має кілька переваг:

– **Легкість використання**: VS Code має простий та інтуїтивно зрозумілий інтерфейс, що спрощує роботу з ним. Він пропонує функціональність, яка покриває всі основні аспекти розробки програмного забезпечення, такі як підсвічування синтаксису, автодоповнення, відладка та керування версіями.

– **Велика кількість розширень**: VS Code має велику кількість розширень, які дозволяють налаштувати середовище розробки під свої

потреби. Для Python існує безліч розширень, які допомагають поліпшити продуктивність, надають підказки, відлагоджувальні засоби та багато іншого.

– **Інтеграція з Git**: VS Code має вбудовану підтримку Git, що дозволяє зручно працювати з системою контролю версій. Це дозволяє розробникам легко відстежувати зміни в коді, комітити зміни, розглядати різницю між версіями і багато іншого.

Загалом, використання VS Code для розробки програми на Python дозволяє ефективно працювати з мовою програмування, надає широкий функціонал та зручне середовище для розробки програмного забезпечення.

Мова програмування Python є однією з найбільш популярних мов для розробки програмного забезпечення. Вона має простий і зрозумілий синтаксис, що полегшує розробку програм та знижує час їх написання. Python також має велику спільноту розробників, що забезпечує багато корисних бібліотек і інструментів. Саме ці бібліотеки значно полегшують реалізацію програмного застосунку (наприклад, у мене використані бібліотеки matplotlib і numpy, які надають зручні засоби для візуалізації даних і роботи з числовими обчисленнями).

5.3 Створення програми

У даній програмі можна виділити дві основні функції:

1). `inverse_congruential_generator`: ця функція відповідає за генерацію псевдовипадкових чисел з використанням обернено-конгруентного методу. Вона приймає початкове значення `seed`, множник `a`, приріст `c`, модуль `m` та кількість чисел `n`, які потрібно згенерувати. Функція повертає список згенерованих чисел.

```
def inverse_congruential_generator(seed, a, c, m, n):
    random_numbers = []
    x = seed
```

```
    for _ in range(n):
        inv_a = pow(a, -1, m)
        x = (inv_a * x + c) % m
        random_numbers.append(x)
    return random_numbers
```

Лістинг 5.1 – функція для генерації псевдовипадкових чисел

2). `plot_random_points`: ця функція будує графік випадкових точок на основі координат x і y. Вона приймає списки координат `x_coords` і `y_coords` і відображає точки на графіку. Функція використовує бібліотеку `matplotlib.pyplot` для створення графіку та виводить його на екран.

```
def plot_random_points(x_coords, y_coords):
    plt.scatter(x_coords, y_coords, s=5)
    plt.title("Случайные точки")
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.show()
```

Лістинг 5.2 – функція для побудови графіка випадкових точок

Обидві ці функції відіграють важливу роль у програмі: перша відповідає за генерацію випадкових чисел, а друга - за їх візуалізацію. Повний лістинг програми можна подивитися у Додатку А.

5.4 Інструкція користувача

Після відкриття програми main.py у середовищі прогромування необхідно скомпілювати та запустити її (рис 5.1).

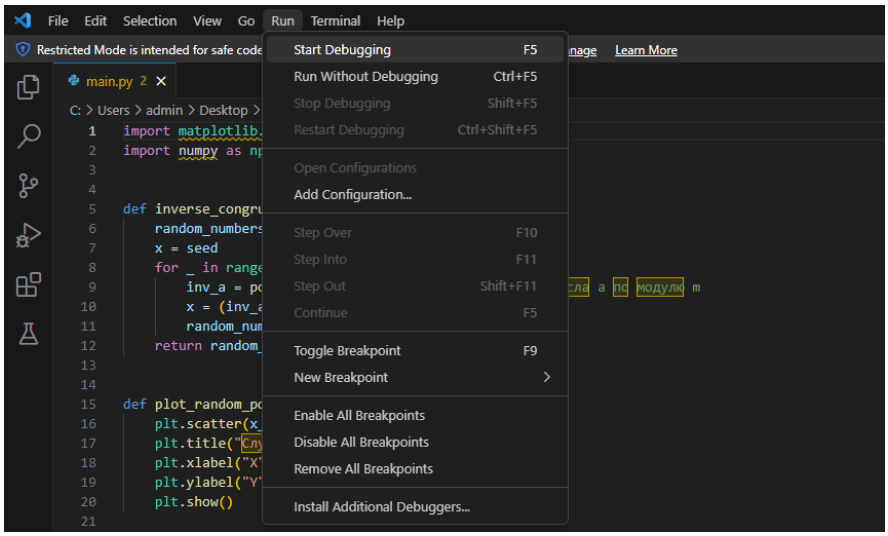

Рисунок 5.1 – Для компіляції програми натискаємо Run->Start Debugging

Після компіляції, програма запуститься автоматично, та запропонує ввести кількість згенерованих точок, початкове значення x та початкове значення y (рис. 5.2).

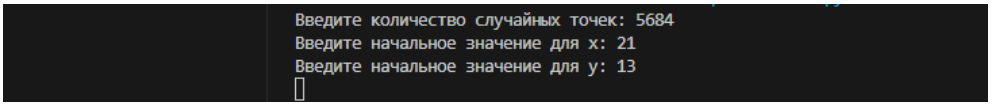

Рисунок 5.2 - Вхідні значення

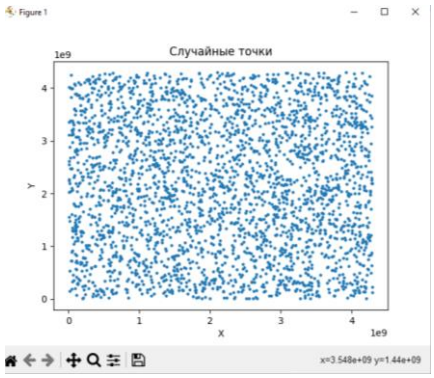

Рисунок 5.3 - Результат роботи на графіку при вхідних значеннях n=5684, x=21, y=13

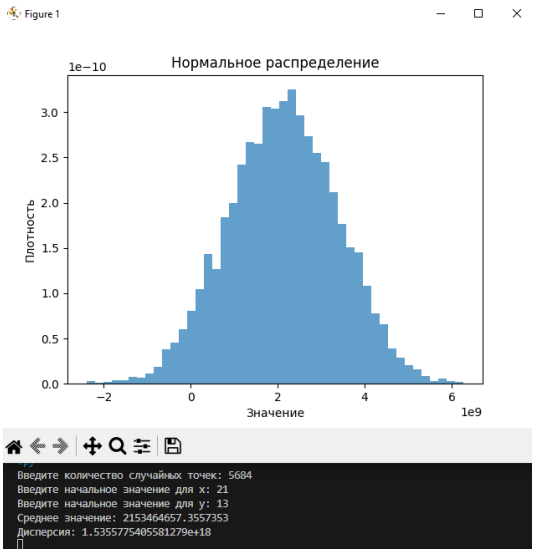

Рисунок 5.4 – Графік розподілу Гаусса-Кузьміна, розрахунок середнього значення та дисперсії при вхідних значеннях n=5684, x=21, y=13

Спробуємо повторити цей процес, збільшивши кількість згенерованих точок до 50000 (рис. 5.5). Графік у цьому випадку має максимально щільно заповнитися точками, і саме такий результат можна бути вважати гарним.

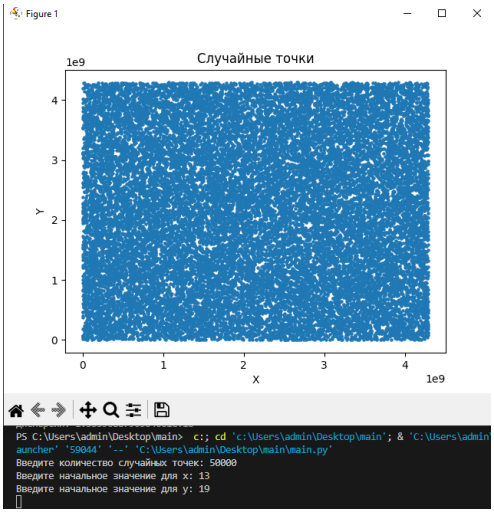

Рисунок 5.5 – Результат роботи для значень n=50000, x=13, y=19

Як можна побачити на рисунку вище, графік дійсно заповнився точками максимально щільно, а отже програма працює правильно.

ВИСНОВОК

У цій роботі ми детально дослідили тему псевдовипадкових числел і розглянули різні аспекти їх структури, алгоритмів генерації та використання, зокрема в криптографії. А також нами було спроектовано та побудовано власні генератор псевдовипадкових чисел на основі інверсного конгруентного методу, результати якого виявился цілком пристойними для роботи з ним в реальному житті. Результати нашого дослідження підтверджують важливість і актуальність цієї теми в сучасному світі.

Псевдовипадкові числа є важливим інструментом у багатьох областях, зокрема в криптографії. Їх використання допомагає забезпечити конфіденційність, цілісність та доступність інформації. Однак, важливо пам'ятати, що вибір відповідного генератора псевдовипадкових чисел та правильне його налагодження є критичними для забезпечення безпеки криптографічних систем.

Майбутні дослідження в цій області можуть бути спрямовані на розробку нових методів генерації псевдовипадкових чисел, враховуючи сучасні вимоги до безпеки і ефективності. Також важливо проводити аналіз і валідацію існуючих генераторів, щоб виявляти потенційні слабкі місця та удосконалювати їх.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дональд Кнут. Искусство программирования, том 2. Получисленные алгоритмы = The Art of Computer Programming, vol.2. Seminumerical Algorithms. — 3-е изд. — М.: «Вильямс», 2007. — С. 832. — ISBN 0-20189684-2

2. The RAND Corporation (Author). A Million Random Digits with 100,000 Normal Deviates Paperback – October 23, 2001.

3. A. Menezes, P. van Oorschot, S. Vanstone. Handbook of Applied Cryptography. — CRC-Press, 1996. — 816 p. — (Discrete Mathematics and Its Applications).

4. Intel® Digital Random Number Digital Random Number Generator Generator(DRNG). Software Implementation Guide. Revision 1.1. August 7, 2012. [Електронний ресурс]. – Режим доступу: https://software.intel.com/sites/default/files/m/d/4/1/d/8/441_Intel_R__DRNG_Software_Implementation_Guide_final_Aug7.pdf. Дата 29.07.2016.

5. Аппаратный генератор случайных чисел ГСЧ-6. [Електронний ресурс]. – Режим доступу: http://tegir.ru/ml/k66.html. Перевірено 29.07.2016.

6. ГОСТ Р ИСО 28640-2012. [Електронний ресурс]. – Режим доступу: http://files.stroyinf.ru/cgi-bin/ecat/ecat.fcgi?b=0&i=53898&pr=1. Перевірено 29.07.2016.

7. http://www.noisecom.com. [Електронний ресурс]. – Режим доступу: Перевірено 29.07.2016.

8. Шнайер Б. 14.1 Алгоритм ГОСТ 28147-89 // Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си = Applied Cryptography. Protocols, Algorithms and Source Code in C. — М.: Триумф, 2002. — С. 373-377.

9. Barker E., Kelsey J. NIST Special Publication 800-90A. Recommendation for Random Number Generation Using Deterministic Random Bit Generators.

10. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си = Applied Cryptography. Protocols, Algorithms and Source Code in C. — М.: Триумф, 2002. — 816 с.

11. Lock-in effect in cascades of clock-controlled shift-registers. In Christoph G. Gunther, editor, Advances in Cryptology—EUROCRYPT 88, volume 330 of Lecture Notes in Computer Science, pages 331–344.

12. G. Mayhew, R. Frazee, and M. Bianco, “Kinetic Protection Device”, Proceedings of the 15th National Computer Security Conference, NIST, 1994, pp. 147154.

13. Ross J. Anderson. On Fibonacci Keystream Generators [Електронний ресурс]. – Режим доступу: http://www.iacr.org/cryptodb/data/paper.php?pubkey=2963. Заголовок з екрану. Перевірено 20.07.2016.

14. Н. Смарт. Криптография. – Москва: Техносфера, 2005. 528 с. ISBN 594836-043-1.

15. Recommendation for Block Cipher Modes of Operation. NIST Special Publication 800-38A. Technology Administration U.S. Department of Commerce. 2001 Edition.

16. S. Wolfram, “Random Sequence generation by Cellular Automata”, Advances in Applied Mathematics, v. 7, 1986, pp.123-164.

17. W. Meier and O. Staffelbach, “Fast Correlation Attack on Stream Ciphers”, Journal of Cryptology v I n. 3, 1989, pp.159-176.

18. Биркгоф Г., Барти Т. Современная прикладная алгебра. – М.: Мир, 1976.

ДОДАТОК А

Лістинг програми

```
import matplotlib.pyplot as plt
import numpy as np

def inverse_congruential_generator(seed, a, c, m, n):
    random_numbers = []
    x = seed
    for _ in range(n):
        inv_a = pow(a, -1, m)
        x = (inv_a * x + c) % m
        random_numbers.append(x)
    return random_numbers

def plot_random_points(x_coords, y_coords):
    plt.scatter(x_coords, y_coords, s=5)
    plt.title("Случайные точки")
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.show()

def plot_histogram(data):
    plt.hist(data, bins='auto', edgecolor='black')
    plt.title("Гистограмма распределения")
    plt.xlabel("Значение")
    plt.ylabel("Частота")
    plt.show()

def plot_line_chart(x, y):
    plt.plot(x, y)
    plt.title("Линейный график")
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.show()
```

```
    def generate_random_numbers(seed, a, c, m, n):
        random_numbers = inverse_congruential_generator(seed, a,
c, m, n)
        return random_numbers

    def calculate_mean(data):
        mean = np.mean(data)
        return mean

    def calculate_variance(data):
        variance = np.var(data)
        return variance

    def calculate_standard_deviation(data):
        std_deviation = np.std(data)
        return std_deviation

    def generate_normal_distribution(mean, std_deviation, size):
        normal_distribution = np.random.normal(mean,
std_deviation, size)
        return normal_distribution

    def plot_normal_distribution(data):
        plt.hist(data, bins='auto', density=True, alpha=0.7)
        plt.title("Нормальное распределение")
        plt.xlabel("Значение")
        plt.ylabel("Плотность")
        plt.show()

    def calculate_median(data):
        median = np.median(data)
        return median

    def calculate_min_max(data):
```

```
        min_value = np.min(data)
        max_value = np.max(data)
        return min_value, max_value

    def plot_boxplot(data):
        plt.boxplot(data)
        plt.title("Boxplot")
        plt.xlabel("Данные")
        plt.ylabel("Значение")
        plt.show()

    def plot_scatter_regression(x, y):
        plt.scatter(x, y, s=5)
        coeffs = np.polyfit(x, y, 1)
        regression_line = np.polyval(coeffs, x)
        plt.plot(x, regression_line, color='red')
        plt.title("Scatter plot с линией регрессии")
        plt.xlabel("X")
        plt.ylabel("Y")
        plt.show()

    def plot_pie_chart(labels, sizes):

        plt.pie(sizes, labels=labels, autopct='%1.1f%%',
startangle=90)
        plt.axis('equal')
        plt.title("Круговая диаграмма")
        plt.show()

    def main():
        seed = 1
        a = 1664525
        c = 1013904223
        m = pow(2, 32)
```

```
        n = int(input("Введите количество случайных точек: "))
        x_seed = int(input("Введите начальное значение для x:
"))
        y_seed = int(input("Введите начальное значение для y:
"))

        random_numbers = generate_random_numbers(seed, a, c, m,
2 * n)

        x_coords = random_numbers[:n]
        y_coords = random_numbers[n:2 * n]

        plot_random_points(x_coords, y_coords)

        mean = calculate_mean(random_numbers)
        variance = calculate_variance(random_numbers)
        print("Среднее значение:", mean)
        print("Дисперсия:", variance)

        normal_distribution = generate_normal_distribution(mean,
np.sqrt(variance), n)

        plot_normal_distribution(normal_distribution)




    if __name__ == "__main__":
        main()
```