МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Навчально-науковий інститут інноваційних освітніх технологій
Кафедра комп'ютерної інженерії

ПУНДОР Юрій Олегович

Алгоритми пошуку найбільшого спільного дільника на основі теоретико-числового базису Радемахера-Крестенсона / Greatest Common Divisor Search Algorithms Based on Rademacher-Krestenson Numerical Basis

спеціальність: 123 - Комп'ютерна інженерія
магістерська програма - Комп'ютерна інженерія

Магістерська робота

Виконав студент групи КІзм-21
Ю. О. Пундор
_____

Науковий керівник:
к.т.н., І. З. Якименко
_____

Магістерську роботу допущено
до захисту:
"11" 01 2018 р.

Завідувач кафедри
_____ О. М. Березький

ТЕРНОПІЛЬ - 2018

Тернопільський національний економічний університет
Навчально-науковий інститут інноваційних освітніх технологій
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії
Освітній ступінь «магістр»
спеціальність: 123 - Комп'ютерна інженерія
магістерська програма - Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.М. Березький
"21" 11 2016 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ
Пундору Юрію Олеговичу
(прізвище, ім'я, по батькові)

1. Тема магістерської роботи «Алгоритми пошуку найбільшого спільного дільника на основі теоретико-числового базису Радемахера-Крестенсона/ Greatest Common Divisor Search Algorithm Based on Rademacher-Krestenson Numerical Basis»
керівник роботи к.т.н., І.З. Якименко
затверджені наказом по університету від 23 жовтня 2017 р. № 1325.

2. Строк подання студентом роботи «15» січня 2018 року

3. Вихідні дані до магістерської роботи
Об'єкт дослідження – процеси оптимізації обчислень основних операцій при шифруванні/дешифруванні в асиметричних системах захисту інформації.
Предмет дослідження – визначення складності виконання алгоритмів пошуку найбільшого спільного дільника в теоеретико-числових базисах Радемахера та Крестенсона.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
– аналіз, вибір методів розв'язання задач пошуку найбільшого спільного дільника;
– удосконалення алгоритму пошуку найбільшого спільного дільника на основі алгоритму Евкліда з використанням ТЧБ Радемахера-Крестенсона;
– розробка паралельного алгоритму пошуку найбільшого спільного дільника з використанням ТЧБ Радемахера-Крестенсона;
– дослідження часових складностей розроблених алгоритмів з класичними;
– провести програмну реалізації розроблених алгоритмів пошуку найбільшого спільного дільника.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
– актуальність теми дослідження;

- мета та задачі дослідження;
- об'єкт, предмет та методи досліджень;
- наукова новизна отриманих результатів;
- практичне значення отриманих результатів;
- функціональні можливості ТЧБ Радемахера та Крестенсона;
- метод пошуку залишків чисел великої розрядності з використаням ТЧ Радемахера-Крестенсона;
- часова складність алгоритму Евкліда та удосконалення його реалізації;
- матричний метод знаходження найбільшого спільного дільника (НСД з застосуванням ТЧБ Крестенсона;
- складності існуючих та розроблених методів пошуку НСД двох чисел;
- інтерфейс програми для обчислення модулярного множення двох чисе та пошуку НСД.

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Антиплагіат	Мельник Г.М., доцент		
Нормо-контроль	Гураль І. В., викладач		

7. Дата видачі завдання «21» листопада 2018 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Приміт
1	Аналіз методів пошуку найбільшого спільного дільника	3.11.2016 – 1.01.2017	Виконан
2	Алгоритми пошуку найбільшого спільного дільника на основі теоретико-числових базисів Радемахера-Крестенсона	2.01.2017 – 31.05.2017	Виконан
3	Програмна реалізація та дослідження алгоритмів пошуку найбільшого спільного дільника	1.06.2017 – 25.01.2018	Виконан
4	Нормоконтроль, попередній захист	16.01.2018 – 1.02.2018	Виконан
5	Захист	2.02.2018	

Студент _____ (підпис) Пундор Ю.О.

Керівник магістерської роботи _____ (підпис) к.т.н., доцент, І.З. Якименк

ВСТУП

Актуальність теми. Знаходження найбільшого спільного дільника (НСД) є важливою фундаментальною задачею теорії чисел, успішне вирішення якої дозволяє вдосконалити алгоритми широкого класу прикладних задач, особливо задач захисту інформаційних потоків в комп'ютерних системах з використанням асиметричної криптографії (алгоритмів RSA, Рабіна, Ель–Гамаля, електронного цифрового підпису [1]–[3], дослідження порядку еліптичної кривої за допомогою алгоритму Шуфа) [4]. Це зумовлено необхідністю використання, як правило, взаємно простих чисел, НСД яких дорівнює 1. В зв'язку з цим актуальною проблемою досліджень є розробка теоретичних основ пошуку НСД з використанням теоретико-числових базисів Радемахера та Крестенсона [5], застосування яких дозволяє зменшити обчислювальну складність та ефективно використовувати при побудові сучасних систем захисту інформації.

У даній роботі проведений аналіз, запропоновані та реалізовані в середовищі C++ нові алгоритми пошуку найбільшого спільного дільника. Побудовані аналітичні вирази часових складностей розроблених підходів, порівняно з відомими.

Мета і завдання дослідження. Метою досліджень є удосконалення методів пошуку найбільшого спільного дільника на основі використання теоретико-числових базисів Радемахера та Крестенсона.

Для досягнення поставленої мети необхідно ефективно розв'язати низку взаємопов'язаних завдань:

1) аналіз, вибір методів розв'язання задач пошуку найбільшого спільного дільника;

2) удосконалення алгоритму пошуку найбільшого спільного дільника на основі алгоритму Евкліда з використанням ТЧБ Радемахера-Крестенсона;

3) розробка паралельного алгоритму пошуку найбільшого спільного дільника з використанням ТЧБ Радемахера-Крестенсона;

4) дослідження часових складностей розроблених алгоритмів з класичними;

5) провести програмну реалізації розроблених алгоритмів пошуку найбільшого спільного дільника.

Об'єкт дослідження – процеси оптимізації обчислень основних операцій при шифруванні/дешифруванні в асиметричних системах захисту інформації.

Предмет дослідження – визначення складності виконання алгоритмів пошуку найбільшого спільного дільника в теоеретико-числових базисах Радемахера та Крестенсона.

Методи дослідження. Для проведення поставлених в даній роботі досліджень, використовуються результати з таких областей знань: для дослідження та аналізу основних операцій в асиметричних криптоалгоритмах - теорія чисел та алгебра Евкліда; теорія алгоритмів для дослідження часових складностей методів пошуку найбільшого спільного дільника.

Наукова новизна отриманих результатів. Під час досліджень отримано такі наукові результати:

- удосконалено реалізацію алгоритму Евкліда пошуку найбільшого спільного дільника на основі використання теоретико-числових базисів Радемахера-Крестенсона, що дозволило зменшити часову складність на 40 %.
- розроблено алгоритм пошуку найбільшого спільного дільника з використанням теоретико-числового базису Крестенсона, який на відміну від існуючих, дозволяє паралельно в мультипрограмному режимі розв’язувати задачу та зменшувати часову складність.

Практичне значення отриманих результатів. Запропоновано новий засіб реалізації алгоритму пошуку найбільшого спільного дільника на основі використання теоретико-числових базисів Радемахера та Крестенсона, це

дозволяє скоротити час виконання на 60%, та економити обчислювальні ресурси.

Публікації та апробація ДР. Результати досліджень, що включено до дипломної роботи, апробовано на науковій конференції молодих вчених ACIT-2016.

1 АНАЛІЗ МЕТОДІВ ПОШУКУ НАЙБІЛЬШОГО СПІЛЬНОГО ДІЛЬНИКА

1.1 Способи кодування даних на основі теоретико-числових базисів Радемахера-Крестенсона

1.1.1 Система функцій Радемахера та двійкові коди.

Система функцій Радемахера утворюється на основі екстракції *sin*-складових наборів дискретно-фазових функцій (рисунок 1.1).

$$Rad(n,\theta) = Dyf(n,\theta,0) = sign(\sin(2^n \pi\theta)). \qquad (1.1)$$

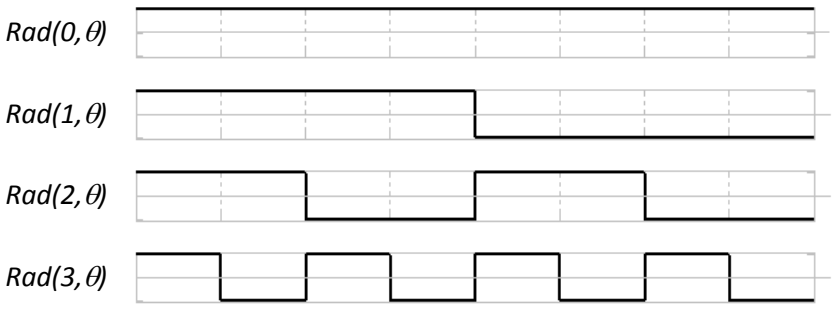


Рисунок 1.1 – Дискретно-фазові функції Радемахера

Крім того, система дискретно-фазових функцій Радемахера породжує двійкову систему числення.

Залежність значень функцій у точках $\theta_s = s/2^n$, $s=0,1,...,2^n-1$ та їхнім записом у двійковому коді $\theta_s = r_n r_{n-1} \ldots r_0$ обчислюється на основі співвідношення:

$$r_k = (1 - Rad(n-k, \theta_s))/2, \qquad (1.2)$$

де r_k – значення розрядів двійкового коду, $k = 0,1,...,n$.

Якщо взяти наприклад $n=3$, то чотирьом функціям будуть відповідати наступні елементи кодової матриці розмірності 4×8

$$
\begin{array}{lcccccccc}
Rad(0,\theta) & \to & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
Rad(1,\theta) & \to & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
Rad(2,\theta) & \to & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
Rad(3,\theta) & \to & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
s & \to & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7.
\end{array}
$$

До основних властивостей системи дискретно-фазових функцій Радемахера слід віднести:

- дискретно-фазових функцій Радемахера ортонормовані на відрізку $[0,1)$, оскільки:

$$\int_0^1 Rad(n,\theta)Rad(k,\theta)d\theta = 0 \text{ та } \int_0^1 Rad(n,\theta)Rad(n,\theta)d\theta = 1.$$

- система дискретно-фазових функцій Радемахера утворює в просторі інтегровних із квадратом функцій $L_2[0,1)$ неповну систему ортонормованих функцій, оскільки не виконується означення повноти системи для довільного n:

$$\int_0^1 Rad(n,\theta)Rad(1,\theta)Rad(2,\theta)d\theta = 0,$$

Це означає, що існує така ортогональна до всіх функцій системи функція $Rad(1,\theta)Rad(2,\theta)$ тотожно не рівна нулю на інтервалі $[0,1)$.

Саме ця властивість дискретно-фазових функцій Радемахера не дозволяє її застосовувати при поданні даних з використанням ортогональних перетворень. Крім того, існування наскрізних переносів в міжрозрядному позиціонуванні формує певні функціональні обмеження при застосуванні системи Радемахера в задачах теорії чисел та багато розрядній арифметиці, яка є основою асиметричних криптоалгоритмів .

1.1.2 Теоретичні основи базису Крестенсона та його застосування

Теоретико-числовий базис Крестенсона породжує систему числення залишкових класів (СЗК). Метод зменшення надлишковості технологічних

сигналів на основі СЗК базується на основі теорії діофантових рівнянь і залишків [1]:

$$x_i = a_i \cdot p + b_i,$$

де a_i – ранг;

p – модуль;

b_i – найменший невід'ємний залишок.

Діофантове рівняння:

$$x_i \equiv b_i (\text{mod } p),$$

або операція прямого кодування по залишках:

$$b_i = res\ x_i (\text{mod } p),$$

де res – символ операції отримання залишку.

Зворотня операція [18]:

$$x_i = \overset{\vee}{E}\left[\frac{x_{i-1} - b_i}{p} + 0{,}5\right] \cdot p + b_i, \qquad (1.3)$$

де $\overset{\vee}{E}[\cdot]$ – цілочисельна функція з округленням до меншого цілого.

Умова однозначності кодування методом залишків виконується, якщо

$$\Delta x_{\max} \le \frac{p-1}{2}, \qquad (1.4)$$

для $p = 5$, $\Delta x_{i\max} \le 2$, $p = 7$, $\Delta x_{i\max} \le 3$.

Інформація на рисунку 1.2 закодована у вигляді залишків. При виконанні умови (1.4) процес x_i можна однозначно представити послідовністю залишків b_i.

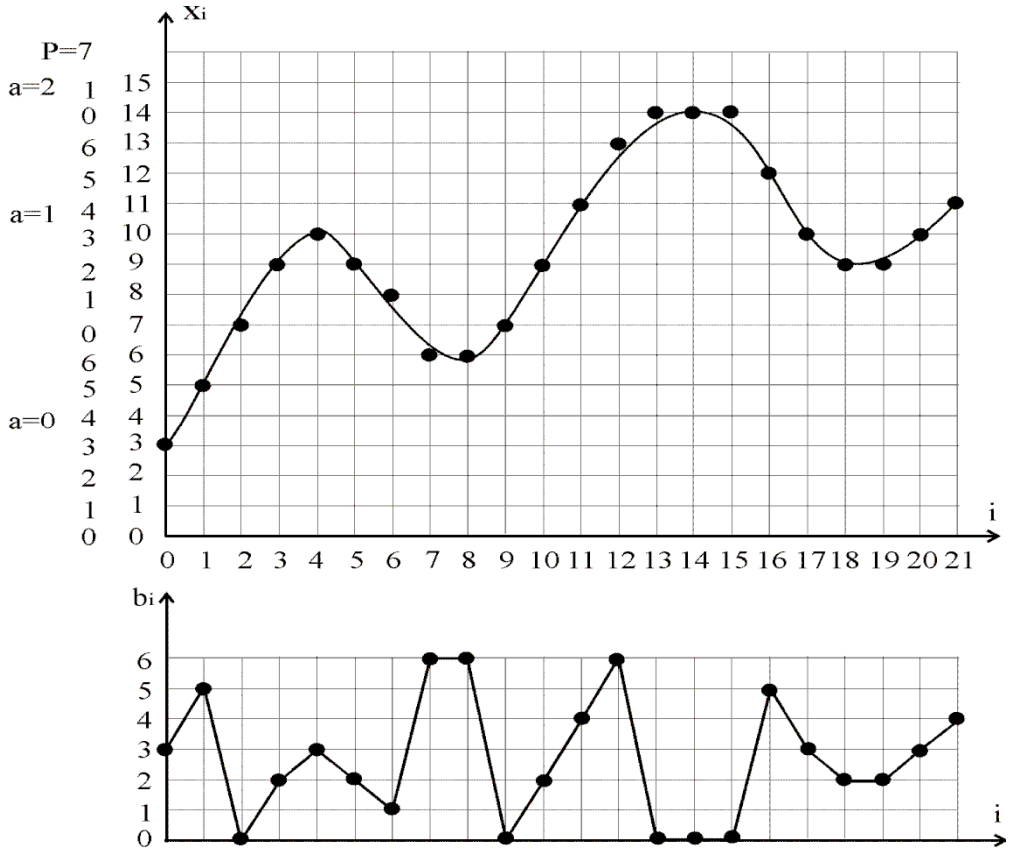


Рисунок 1.2 – Кодування аналогового сигналу методом залишків.

Ефект стиснення даних досягається за рахунок представлення інформаційних відліків відповідними залишками меншої розрядності.

Розрядність коду x_i визначається за формулою Хартлі:

$$n = \hat{E}\left[\log_2 A\right],$$

де n – розрядність двійкового коду для представлення величини x_i;

$\hat{E}[\cdot]$ – цілочисельна функція з округленням до більшого цілого;

A – діапазон квантування сигналу $0 \le x_i \le A$.

Розрядність коду залишків b_i визначаємо за формулою:

$$n_z = \hat{E}\left[\log_2 p\right].$$

Отже, формула для коефіцієнту стиснення буде мати наступний вигляд:

$$k = \frac{n}{\hat{E}\left[\log_2 p\right]}.$$

При кодуванні аналогового сигналу, представленого на рисунку 1.2, об'єм масиву становить:

$$V = n \cdot m = 4 \cdot 22 = 88 \text{ біт.}$$

Після кодування методом залишків об'єм масиву становить:

$$V = n_z \cdot m = 3 \cdot 22 = 66 \text{ біт.}$$

Декодування даних відбувається за формулою (1.3), при $x_0 = 3$ визначаємо x_1:

$$x_1 = \overset{\vee}{E}\left[\frac{x_0 - b_1}{p} + 0.5\right] \cdot p + b_1 = \overset{\vee}{E}\left[\frac{3-5}{7} + 0.5\right] \cdot 7 + 5 = 5;$$

$$x_2 = \overset{\vee}{E}\left[\frac{5-0}{7} + 0.5\right] \cdot 7 + 0 = 7;$$

$$x_3 = \overset{\vee}{E}\left[\frac{7-2}{7} + 0.5\right] \cdot 7 + 2 = 9;$$

$$x_4 = \overset{\vee}{E}\left[\frac{9-3}{7} + 0.5\right] \cdot 7 + 3 = 10;$$ і. т.д.

З приведених розрахунків значень сигналу $x_1 \div x_4$ по відповідних залишках видно, що отримані значення відповідають значенням інформаційних відліків до виконання операції кодування.

Отже, для представленого на рисунку 1.2 сигналу при використанні методу кодування та зменшення надлишковості даних на основі залишків коефіцієнт стиснення дорівнює $k = 1{,}33$ рази.

Апаратна реалізація принципу перетворення аналогового сигналу $x(t)$ в цифровий сигнал, представлений в системі залишкових класів, приведена на рисунку 1.3 [33].

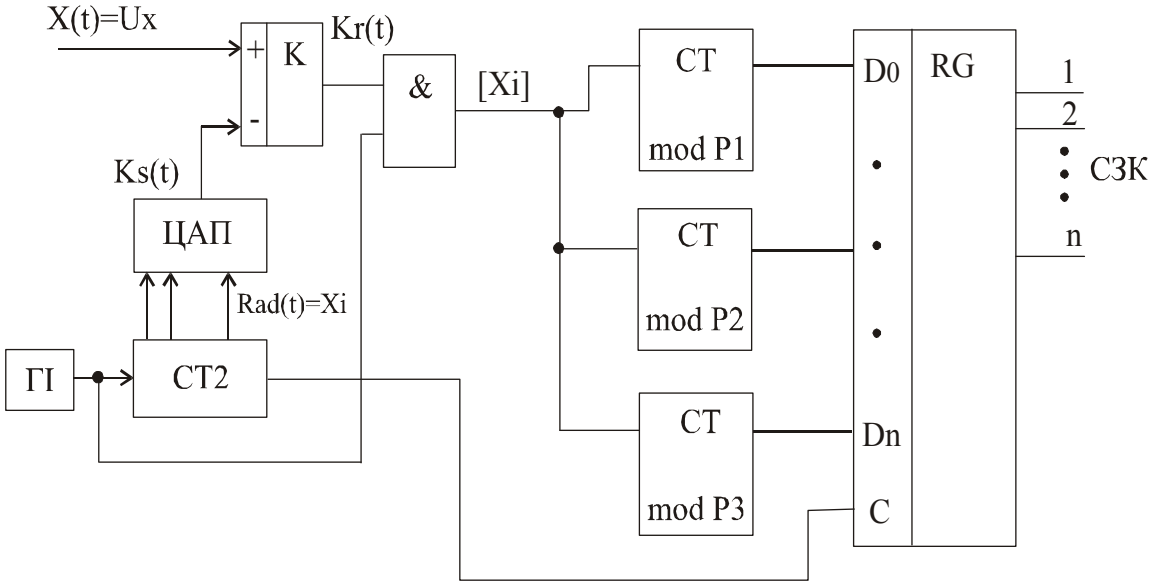


Рисунок 1.3 - Структурна схема кодера в базисі Крестенсона по модулях P1, P2, P3.

Робота представленої структурної схеми (рисунок 1.3) відбувається наступним чином: в початковий момент, який відповідає часу $t = 0$, лічильник СТ2 і регістр RG знаходиться в нульовому стані. Вхідний аналоговий сигнал $x(t)$ представляється у вигляді пропорційної напруги U_x, яка подається на вхід компаратора, на опорний вхід якого поданий вихідний сигнал цифро-аналогового перетворювача (ЦАП). При цьому відбувається порівняння значення ступінчатої функції $Ks(t)$ зі значенням аналогової функції $x(t)$.

В результаті отримаємо x_i:

$$x_i = \overset{\vee}{E}\left[\frac{x(t)}{\delta}\right].$$

Високочастотний сигнал, який модулює генератор ГІ, подається одночасно на вхід лічильника СТ2 і на один з входів логічної схеми “І”. При цьому в процесі формування двійкового коду на виходах лічильника СТ2, що відповідає генеруванню системи функцій базису Радемахера $Rad(t)$, формується двійковий код x_i, який в ЦАП перетворюється у відповідну ступінчасту функцію $Ks(t)$ базису Крестенсона $x(t) \le Ks(t)$ [33].

В момент виконання умови $U_x \le Ks(t)$ на виході компаратора, який підключений до входу логічного елемента “І”, формується нульовий сигнал і, фактично, формується широтно-імпульсна функція базису Крейга $Kr(t)$.

В цей момент на виході логічного елемента “І” припиняється формування числа імпульсів унітарного коду $[x_i] = x(t)$, що представляє собою код унітарного базису x_i.

Для перетворення унітарного коду x_i в систему залишкових класів, унітарний код x_i поступає на лічильники, які працюють по модулю Р1, Р2, Р3. На паралельних виходах лічильників одержуємо код в системі залишкових класів по вибраних модулях. Запис даних в регістр пам’яті здійснюється сигналом переносу лічильника СТ2.

Перевагою використання аналого-цифрового перетворення (АЦП) в СЗК є незалежне утворення інформаційних розрядів, що дає можливість їх паралельної обробки.

Перетворення аналогового сигналу методом залишків по одному модулю (рисунок 1.4) відбувається аналогічно перетворенню по трьох модулях (рисунок 1.3).

Коефіцієнт стиснення методом залишків залежить від вибраного модуля Р та розрядності даних (рисунок 1.5).

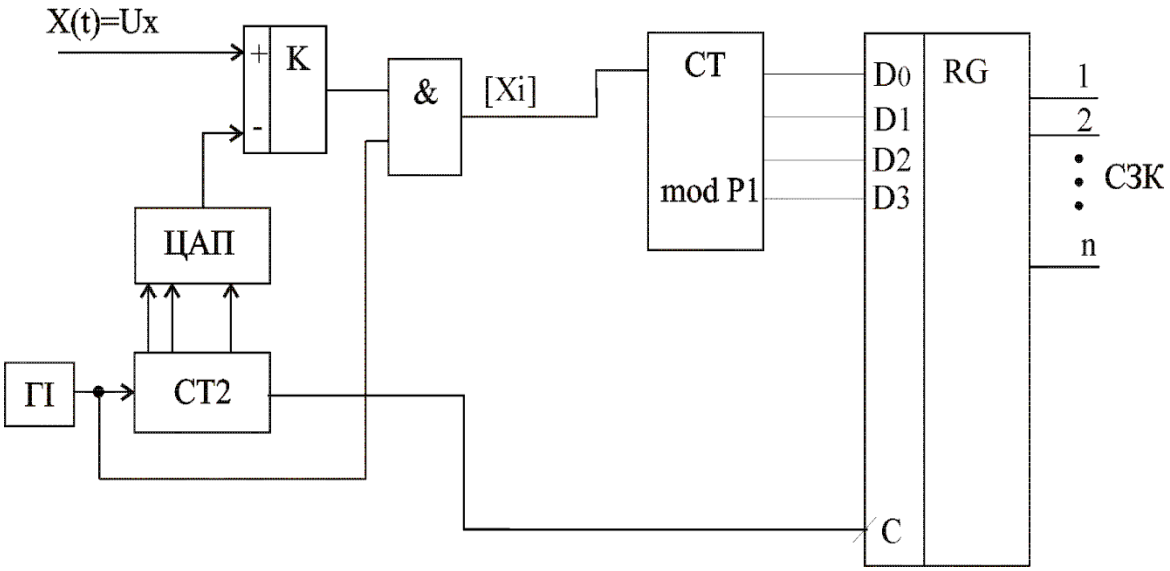


Рисунок 1.4 - Структурна схема кодера по модулю Р1.

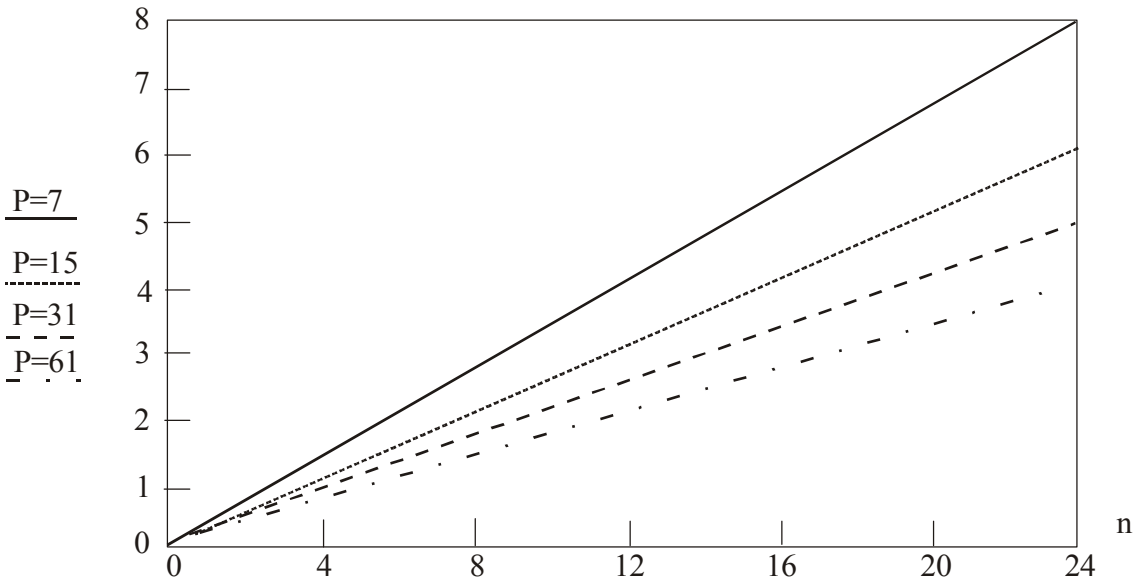


Рисунок 1.5 - Коефіцієнт стиснення даних в залежності від модуля Р.

Метод залишків доцільно використовувати для кодування процесів з низькою динамікою (телефонна розмова, вимірювання температури та інші).

Представлення даних в системі залишкових класів дає змогу здійснювати паралельну обробку інформації без значного ускладнення обчислювальних засобів.

Особливістю СЗК залишається простота реалізації прямого та зворотного перетворень.

1.1.3 Досконала форма системи залишкових класів теоретико-числового базису Крестенсона

Основною операцією, яка збільшує часову складність алгоритмів перетворень в цілочисельній та нормалізованій СЗК є процедура множення при отриманні парних добутків $b_j \cdot B_j$ та $[b_j] \cdot m_j$. Тому її виключення з аналітики названих перетворень дозволяє добитися суттєвих спрощень як в алгоритмічному так і в апаратурному аспектах.

Для рішення цієї задачі запишемо вираз для зворотнього перетворення СЗК з одиничними воговими коефіцієнтами:

$$N_k = \begin{cases} res \sum_{j=1}^{k} b_j (\pm 1) \frac{\rho}{P_j} (\text{mod } \rho), \quad \dot{N}_k \le 0; \\ \rho - res \sum_{j=1}^{k} b_j (\pm 1) \frac{\rho}{P_j} (\text{mod } \rho), \quad \dot{N}_k > 0. \end{cases} \tag{1.3}$$

Зворотнє перетворення для таких СЗК визначається виразом:

$$\pm \dot{N}_k = res \sum_{j=1}^{k} b_j (\pm 1) \frac{\rho}{P_j} (\text{mod } \rho), \tag{1.4}$$

а пряме перетворення виконується згідно виразу:

$$b_j = \begin{cases} res \dot{N}_k (\text{mod } P_j), \quad N_k \le 0; \\ res(\rho - \dot{N}_k) (\text{mod } P_j), \quad N_k > 0. \end{cases} \tag{1.5}$$

Умови (1.3) та (1.4) дозволяють зформувати вимоги до наборів модулів СЗК, в яких виключається надлишковість подання $b_j \cdot B_j$ при заданих $\{m_j\}$, тобто:

$$
\begin{aligned}
&0)\ \{m_j\} = \{1,1,...,1\}; \quad \{P_j > m_j\};\\
&1)\ \{m_j\} = \{1,1,..., P_k - 1\}; \quad P_k = m_k + 1;\\
&2)\ \{m_j\} = \{1,1,..., P_{k-1} - 1, P_k - 1\}; \quad P_{k-1} = m_{k-1} + 1; P_k = m_k + 1;\\
&\ldots \qquad\qquad\qquad \ldots\\
&k)\ \{m_j\} = \{P_j - 1\}; \quad P_j = m_j + 1.
\end{aligned}
\tag{1.6}
$$

Розглянемо випадок (1.6.0).

Підставивши $m_j = 1$ у вираз для B_j і дальше визначимо умову існування СЗК з набором m_j перейшовши від порівняння до лінійного рівняння:

$$\sum_{j=1}^{k} \prod_{i=j}^{k} P_i = r_1 \prod_{j=1}^{k} P_j + 1, \tag{1.7}$$

де $r_1 = 0,1,2,...$ - ранг СЗК.

При зміні параметра $k = 1,2,...$ згідно останнього виразу маємо:

$$
\begin{aligned}
&k = 1; \quad P_1 = r_1 P_1 + 1;\\
&k = 2; \quad P_1 + P_2 = r_1 P_1 \cdot P_2 + 1;\\
&k = 3; \quad P_1 \cdot P_2 + P_1 \cdot P_3 + P_2 \cdot P_3 = r_1 P_1 P_2 P_3 + 1;\\
&\ldots \qquad\qquad \ldots
\end{aligned}
$$

Очевидно, що при $k = 1$ ш $k = 2$ не може бути знайдено ні однієї СЗК з властивістю. В інших випадках (1.5) доцільно привести до вигляду

$$x_k \sum_{j=1}^{k-1} \prod_{i \neq j}^{k} P_i + \prod_{i \neq k}^{k} P_i = r_1 x \prod_{k\ j \neq k}^{k} P_j + 1,$$

або

$$x_k = \frac{\prod_{i \neq k}^{k} P_i - 1}{r_1 \prod_{j \neq k}^{k} P_j - \sum_{j=1}^{k} \prod_{i \neq j}^{k} P_i}, \qquad (1.8)$$

де $x_k = P_k$ -й модуль СЗК, який задовольняє умові (1.6) і обчислюється на основі відомих $k-1$ модулів.

Наприклад:

Нехай заданий набір модулів СЗК:

$$\{P_j\} = \{2,3,7,x\}.$$

Необхідно знати таке x, щоб $\{m_j\} = \{1,1,1,1\}$. Підставляючи числові значення цього приклад в (1.8) отримаємо:

$$x = \frac{42-1}{42-(6+14+21)} = \frac{41}{42-41}$$

і при $r_1 = 1,\ x = 41$.

Так як отримане значення x просте число, рішення є єдиним.

Таким чином отримуємо набір модулів досконалої СЗК:

$$\{P_j\} = \{2,3,7,41\}, \quad \rho = 1722.$$

Для перевірки правильності рішення (1.8) розрахуємо ортогональні базиси СЗК з таким набором модулів:

$$\dot{B}_1 = \frac{1722}{2} m_1 \equiv 1 (\mathrm{mod}\, 2);\ m_1 = 1;$$
$$\dot{B}_2 = \frac{1722}{3} m_2 \equiv 1 (\mathrm{mod}\, 3);\ m_2 = 1;$$
$$\dot{B}_3 = \frac{1722}{7} m_3 \equiv 1 (\mathrm{mod}\, 7);\ m_3 = 1;$$
$$\dot{B}_4 = \frac{1722}{41} m_1 \equiv 1 (\mathrm{mod}\, 41);\ m_4 = 1.$$

отже,

$$\dot{N}_k = (\dot{B}_1 + \dot{B}_2 + \dot{B}_3 + \dot{B}_4)(\mathrm{mod}\,\rho) = 1,$$

тобто для нашого прикладу:

$$\dot{N}_k = (861 + 574 + 246 + 42)(\mathrm{mod}1722) = 1,$$

що відповідає умові $\{m_j\} = \{1,1,1,1\}$.

Аналогічно знайдемо вирази існування СЗК для певних умов:

$$\begin{aligned}
&1)\ \sum_{j=1}^{k-1}\prod_{i\neq j}^{k-1} P_i - \prod_{j\neq k}^{k} P_i = \pm r_1 \prod_{j=1}^{k} P_j + 1;\\
&2)\ \sum_{j=1}^{k-2}\prod_{i\neq j}^{k-2} P_i - \sum_{j=k-1}^{k}\prod_{i\neq j}^{k} P_i = \pm r_1 \prod_{j=1}^{k} P_j + 1;\\
&\cdots \qquad\qquad \cdots\\
&k)\ \sum_{j=1}^{k-2}\prod_{i\neq j}^{k-2} P_i - \sum_{j=1}^{k}\prod_{i\neq j}^{k} P_i = \pm r_1 \prod_{j=1}^{k} P_j + 1.
\end{aligned} \qquad (1.9)$$

Дослідження отриманих виразів показує, що умова (1.9.к) повністю співпадає з (1.6), а також відображає параметри існування СЗК з одиничними ваговими коефіцієнтами $\{m_j\} = \{-1,-1,...,-1\}$.

Приведення СЗК досконалої форми дозволяє повністю виключити модульні операції у його зворотньому перетворенні.

Умову утворення досконалої без рангової СЗК можна отримати у вигляді нерівності, підставивши граничні значення $b_j = m_j = P_{j-1}$, тобто:

$$\sum_{j=1}^{k}(P_j-1)\frac{(P_j-1)\cdot\rho}{P_j})<\prod_{j=1}^{k}P_j\,, \qquad (1.10)$$

Звідси після елементарних перетворень отримаємо:

$$\sum_{j=1}^{k}(P_j-2+\frac{1}{P_j})<1.$$

Очевидно, що у цьому випадку не існує ні одного набору цілочисельних взаємопростих модулів $P_j \geq 2$, $k \geq 2$, які утворюють досконалу без рангову СЗК.

Враховуючи, що коли $m_j = P_j - 1$, то можна замінити $m_j = -1$ і записати у вигляді:

$$\sum_{j=1}^{k}(P_j-1)\frac{\rho}{P_j}>-\prod_{j=1}^{k}P_j \qquad (1.11)$$

або

$$\sum_{j=1}^{n}(1-\frac{1}{P_j})<1$$

Звідки згідно (1.10) при $k \geq 2$ та коли не існує ні однієї без рангової СЗК.

В якості одного з найбільш простих способів представлення СЗК до досконалої без рангової форми можна застосувати метод надлишкового розширення системи модулів початкової СЗК[318, 335].

Умова існування досконалої без рангової СЗК для розширеної системи модулів визначається згідно виразу:

$$\sum_{j=1}^{k-e}\prod_{i\neq j}^{k-e}P_i-\sum_{j=e}^{k-\nu}P_i \lhd \pm\prod_{j=1}^{k}P_j\,,$$

де e, ν - відповідно кількість основних і надлишкових модулів.

Розглянемо приклад:

Нехай маємо набір модулів досконалої СЗК

$$\{P_j\} = \{2, 3, 5\}; \quad \rho = 30; \quad \overset{0}{B}_1 = 15; \quad \overset{0}{B}_2 = 10; \quad \overset{0}{B}_3 = 6;$$

$m_1 = m_2 = m_3 = 1; \quad N_k = 26_{(10)} = 11010_{(2)}$.

Представимо число N_k у СЗК, тобто:

N_k → res26(mod2)=0, res26(mod3)=2, res26(mod5)=1 } =(0, 2,1) (b_1, b_2 b_3)

Запишемо залишки b_1, b_2, b_3, у нормалізованій формі:

$$[b_1]_0 = 0/2 = 0; \quad [b_2]_0 = 2/3 = 0{,}66; \quad [b_3]_0 = 1/5 = 0{,}2$$

Виконуємо зворотнє перетворення досконалої нормалізованої форми СЗК згідно виразу

$$[N_k]_0 = (0 + 0{,}66 + 0{,}2) \bmod 1 = 0{,}86$$

Отримаємо цілочисельне значення $[N_k]_0$

$$N_k = \hat{E}[[N_k]_0 \cdot \rho] = \hat{E}[0{,}86 \cdot 30] = 26$$

Аналогічно у двійковій системі числення «0» «1» виконуємо операції перетворень ДНСЗК:

$$[b_1]_0 = 0{,}00000_{(2)}; \quad [b_2]_0 = 0{,}10101_{(2)}; \quad [b_3]_0 = 0{,}00110_{(2)};$$

$[N_k]$= 0,00000 + 0,10101 = 0,11011

x 0,11011 × 11110 = 110110, 11011, 11011, 11011

$N_k = \hat{E}[1100101010] = 26_{10}$

Труднощі пошуку досконалої форми СЗК а також їх відносна рідкість вимагають розвитку інших методів отримання досконалих перетворень базису Крестенсона-Галуа.

Одним з таких методів є отримання скороченої СЗК з ваговими коефіцієнтами $\{m_j\}=\{1,1,...,1\}$ шляхом виключення одного або кількох модулів у перетвореннях СЗК.

Умова існування таких СЗК має вигляд:

$$\sum_{j=1}^{k-\nu}\prod_{i\neq 1}^{k-\nu}P_i=\prod_{j=1}^{k}P_j-1-\sum_{j-\nu}^{k}\prod_{i\neq j}^{k}P_i$$

Наприклад виберемо в якості початкового набору модулів СЗК наступні $\{P_j\}=\{5,7,8,9,19\}$, для якого $\{m_j\}=\{1,1,1,1,8\}$ і виключено з нього модуль $P_5=19$ прийнявши $b_5=0$.

Тоді всі операції перетворення в СЗК можуть виконуватись по алгоритмах без рангових досконалих форм СЗК.

Іншим універсальним практичним методом приведення до досконалої форми СЗК для будь яких наборів модулів $\{P_j\}$ є використання властивості (1.9).

Суть такого методу приведення в СЗК до досконалої форми полягає в тому, що при виконанні зворотнього перетворення СЗК апріорно задається набір $\{m_j\}=\{1,1,...,1\}$ для всіх модулів $\{P_j\}$. При цьому легко показати, що однозначність перетворення СЗК не порушується, але при виконанні зворотного перетворення необхідно коректувати значення добутих N_k залишків b_j з врахуванням конкретних значень $\{m_1, m_2, ..., m_j\}$ для заданого набору $\{P_1, P_2, ..., P_j\}$ згідно виразу:

$$b_j = res\, \overset{0}{b}_j \cdot \bmod P_j,$$

де $\overset{0}{b}_j$ - залишок отримання із числа $\overset{0}{N}_k$, тобто

$$\overset{0}{b}_j = res\, \overset{0}{N}_k \,(\mathrm{mod}\, P_j),$$

а число $\overset{0}{N}_k$ досконалого перетворення СЗК розраховується згідно виразу

$$\overset{0}{N}_k = \left(\overset{0}{B}_1 b_1 + \overset{0}{B}_2 b_2 + ... + \overset{0}{B}_k b_k \right) (\mathrm{mod}\, \rho),$$

де $\overset{0}{B}_0 = \rho / P_j \cdot 1$. Тобто: всі $m_j = 1; \quad j \in \overline{1, k}.$

Нехай заданий набір модулів СЗК:

$$\{P_j\} = \{P_1, P_2, ..., P_k\}, \quad \rho = \prod_{j=1}^{k} P_j,$$

Якому відповідають набори:

$$\{m_j\} = \{m_1, m_2, ..., m_k\}, \quad \{B_j\} = \left\{ \frac{\rho}{P_j} \cdot m_j \right\},$$

при чому не виконується ні одна з умов (1.9) і ні одним з розглянутих раніше методів СЗК з таким набором модулів не може бути приведене до досконалої форми.

Тоді, якщо прийняти всі $\left\{ \overset{0}{m}_j \right\} = \{1, 1, ..., 1\}$ незалежно від їх фактичних значень $\{m_j\}$, то однозначність перетворень СЗК не порушується при виконанні їх згідно виразів:

$$b_j = res \left[res\, \overset{0}{N}_k \left(\mathrm{mod}\, P_j \right) \cdot m_j \right] \mathrm{mod}\, P_j,$$

$$\overset{0}{N}_k = res \left(\overset{0}{B}_1 \cdot b_j + \overset{0}{B}_2 \cdot b_2 + ... + \overset{0}{B}_k \cdot b_k \right) \mathrm{mod}\, \rho.$$

В нормалізованій досконалій формі СЗК аналогічні перетворення виконуються згідно виразів:

$$\left[\overset{0}{b_j}\right]_0 = b_j / P_j; \quad \left[\overset{0}{N_k}\right]_0 = res \sum_{j=1}^{k} \left[\overset{0}{b_j}\right]_0 (\mathrm{mod}\, 1);$$

$$\overset{0}{N_k} = \overset{\vee}{E}\left[\left[\overset{0}{N_k}\right]_0 \cdot \rho\right];$$

$$b_j = \overset{\vee}{E}\left[\left[\overset{0}{b_j}\right] \cdot resP_j \cdot m_j (\mathrm{mod}\, P_j)\right].$$

В роботах [5,7] нами показано, що опрерація залишків $\overset{0}{b_j}$ з врахуванням $\{m_j\}$ виконується шляхом циклічного зсуву на величину нормалізованих значень з модулів P_j.

Наприклад:

Нехай маємо набір модулів СЗК:

$P_1=5$; $P_2=7$; $P_3=11$; тоді $\rho = 5 \cdot 11 \cdot 13 = 385;$

$$B_1 = 77 \cdot m_1 \equiv 1(\mathrm{mod}\, 5); \quad m_1 = 3; \quad B_1 = 231; \quad \overset{0}{B_1} = 77;$$

$$B_2 = 55 \cdot m_2 \equiv 1(\mathrm{mod}\, 7); \quad m_2 = 6; \quad B_2 = 330; \quad \overset{0}{B_2} = 55;$$

$$B_3 = 35 \cdot m_3 \equiv 1(\mathrm{mod}\, 11); \quad m_3 = 6; \quad B_3 = 210; \quad \overset{0}{B_3} = 35.$$

Таблиці значень залишків b_j для заданої (а) та досконалої форми СЗК (б) зваженої цілочисельної форми СЗК мають наступний вигляд:

b_i	P_1	P_2	P_3
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	–	5	5
6	–	6	6
7	–	–	7
8	–	–	8
9	–	–	9
10	–	–	10

а)

$\Rightarrow$

$\overset{o}{b}_i$	P_1	P_2	P_3
0	0	0	0
1	3	6	6
2	4	1	7
3	0	2	8
4	1	2	9
5	–	4	10
6	–	5	1
7	–	–	2
8	–	–	3
9	–	–	4
10	–	–	5

б)

Таким чином у цілочисельній формі СЗК число N_k=1 обчислюється згідно виразу:

$$N_k = (231 \cdot 1 + 330 \cdot 1 + 210 \cdot 1)(\mathrm{mod}\, 385) = 1,$$

або

$$\overset{0}{N}_k = (77 \cdot 3 + 55 \cdot 6 + 35 \cdot 6)(\mathrm{mod}\, 385) = 1.$$

Аналогічно складаються таблиці нормалізованих значень $\left[\overset{0}{b}_j\right]_0$ у десятковій та двійковій системах числення.

1.2 Методи пошуку найбільшого спільного дільника та застосування в системах захисту інформації

Знаходження найбільшого спільного дільника (НСД) є важливою фундаментальною задачею теорії чисел, успішне вирішення якої дозволяє

вдосконалити алгоритми широкого класу прикладних задач, особливо задач захисту інформаційних потоків в комп'ютерних системах з використанням асиметричної криптографії (алгоритмів RSA, Рабіна, Ель–Гамаля, електронного цифрового підпису [1]–[3], дослідження порядку еліптичної кривої за допомогою алгоритму Шуфа) [4]. Це зумовлено необхідністю використання, як правило, взаємно простих чисел, НСД яких дорівнює 1.

В зв'язку з цим актуальною проблемою досліджень є розробка теоретичних основ пошуку НСД з використанням теоретико-числових базисів Радемахера та Крестенсона [5], застосування яких дозволяє зменшити часову складність.

Найбільш розповсюдженим методом знаходження НСД є один з найдавніших математичних алгоритмів – алгоритм Евкліда, згідно якого для знаходження НСД двох чисел необхідно декілька разів від більшого числа відняти менше, поки різниця не стане меншою від'ємника. Тоді цю ж саму процедуру потрібно виконати з від'ємником та різницею. Процес віднімання буде тривати до тих пір, поки від'ємник та різниця не стануть однакові. Оскільки числа, над якими виконуються операції, на кожному кроці зменшуються, то такий процес не може тривати нескінченно, а закінчиться через деяке число кроків.

Означення. Число $d \in \mathbb{Z}$, ділить одночасно числа a, b, c, ..., $k \in \mathbb{Z}$, називається спільним дільником цих чисел. Найбільше d з такою властивістю називається найбільшим спільним дільником. Позначення: $d = (a, b, c, ..., k)$.

Теорема. Якщо $(a, b) = d$, то знайдуться такі цілі числа u і v, що $d = au+bv$.

Доведення. Розглянемо безліч $P = \{au + bv \mid u, v \in Z\}$. Очевидно, що $P \subseteq Z$, а знавці алгебри можуть перевірити, що P - ідеал в Z. Очевидно, що $a,b,0 \in P$. Нехай x, y $\in$ P і y $\neq 0$. Тоді залишок від ділення x на y належить P. дійсно:

$$x = yq + r\ ,\ 0 \le r < y,$$

$$r = x - yq = (au_1 + bv_1) - (au_2 + bv_2)\ q = a\ (u_1 - u_2q) + b(v_1 - v_2q) \in P.$$

Нехай $d \in$ P - найменше додатнє число з P. Тоді a ділиться на d. Справді, $a = dq + r_1$, $0 \le r_1 < d$, $a \in$ P, $d \in$ P, значить $r_1 \in$ P, отже $r_1 = 0$. Аналогічними міркуваннями виходить, що b ділиться на d, значить d - спільний дільник a і b.

Далі, оскільки $d \in$ P, то $d = au_0 + bv_0$. Якщо тепер d_1 - спільний дільник a і b, то $d_1 \mid (au_0 + bv_0)$, тобто $d_1 \mid d$. Значить $d \ge d_1$ і d - найбільший спільний дільник.

Означення. Цілі числа a і b називаються взаємно простими, якщо $(a, b)=1$.

Легко помітити, що два числа a і b є взаємно простими тоді і тільки тоді, коли знайдуться цілі числа u і v такі, що $au + bv = 1$.

Нехай дано два числа a і b; $a \ge 0$, $b \ge 0$, вважаємо прості і a> b. Символом : = в записі алгоритму позначаємо присвоювання.

Алгоритм:

1. Ввести a і b.

2. Якщо $b = 0$, то Відповідь: a. Кінець.

$$a = bq_1 + r_1 \qquad 0 \le r_1 < b$$
$$b = r_1 q_2 + r_2 \qquad 0 \le r_2 < r_1$$
$$r_1 = r_2 q_3 + r_3 \qquad 0 \le r_3 < r_2$$
$$r_2 = r_3 q_4 + r_4 \qquad 0 \le r_4 < r_3$$

. . . . . .

$$r_{n-3} = r_{n-2} q_{n-1} r_{n-1} \qquad 0 \le r_{n-1} < r_{n-2}$$
$$r_{n-2} = r_{n-1} q_n + r_n \qquad 0 \le r_n < r_{n-1}$$
$$r_{n-1} = r_n q_n + 1 \qquad r_n + 1 = 0$$

3. Присвоїти r : = "Залишок від ділення a на b", a: = b, b: = r.

4. Перейти на 2.

Тобто алгоритм Евкліда можна записати наступним чином: $a > b$; $a, b \in Z$.

Маємо: $b > r_1 > r_2 > ... > r_n > 0$, отже процес завершується максимум через b кроків. Постає також запитання, про те, при яких значеннях алгоритм Евкліда працює особливо довго. Давайте зараз покажемо, що $r_n = (a, b)$. Переглянемо послідовно рівності зверху вниз: всякий дільник a і b ділить $r_1, r_2, ..., r_n$. Якщо ж переглядати цей ланцюжок рівностей від останнього до першого, то видно, що $r_n \mid r_{n-1}$, $r_n \mid r_{n-2}$, і так дальше, тобто r_n ділить a і b. Тому r_n - найбільший спільний дільник чисел a і b.

Алгоритм Евкліда дозволяє в практичний спосіб знаходити числа u і $v \in Z$, для яких виконується $r_n = au + bv = (a, b)$.

Дійсно, з ланцюжка рівностей отримується:

$r_n = r_{n-2} - r_{n-1} q_n = r_{n-2} - (r_{n-3} - r_{n-2} q_{n-1}) q_n = ... = au + bv = (a, b)$.

Приклад. Нехай a = 525, b = 231. (нижче наводиться запис ділення ланцюгом, і кожен раз то, що було в куточку, тобто дільник, приписується до залишку від ділення з лівого боку, а залишок, як новий дільник, береться в куточок).

```
        _   525 |231
            462 |2
      _  231 | 63
            189 |3
    _    63 | 42
         42 |1
   _42 |21
   42 |  2
   0
```

Запис того ж самого на основі алгоритму Евкліда:

$525 = 231 \cdot 2 + 63$

$231 = 63 \cdot 3 + 42$

$63 = 42 \cdot 1 + 21$

$42 = 21 \cdot 2$

Таким чином найбільшим спільним дільником чисел (525, 231) = 21. Лінійне представлення найбільшого спільного дільника:

$21 = 63 - 42 \cdot 1 = 63 - (231 - 63 \cdot 3) \cdot 1 =$

$= 525 - 231 \cdot 2 - (231 - (525 - 231 \cdot 2) \cdot 3) =$

$= 525 \cdot 4 - 231 \cdot 9,$

і наші u і $v \in Z$ рівні, відповідно, 4 і - 9.

Приступимо тепер до виконання другої частини назви цього пункту - аналізу алгоритму Евкліда. Нас буде цікавити найгірший випадок - коли алгоритм працює особливо довго? Запитаємо точніше: які два найменших числа необхідно подати в алгоритм Евкліда, щоб він працював в точності заданої кількості кроків? Відповідь на це питання дає наступна теорема.

Теорема (Ламе, 1845 г.). Нехай $n \in N$, і нехай $a > b > 0$ такі, що алгоритму Евкліда для обробки a і b необхідно виконати точно n кроків (ділень із залишком), причому a - найменше з такою властивістю. Тоді $a = \Phi_{n+2}$, $b = \Phi_{n+1}$, де Φ_k - k- е число Фібоначі.

Якщо натуральні числа a і b не перевищують $n \in N$, то число кроків (операцій ділення із залишком), необхідних алгоритму Евкліда для обробки a і b не перевищує $\lceil \log_\Phi (\sqrt{5}\, N) \rceil - 2$, де $\lceil \alpha \rceil$ – верхнє ціле α, $\Phi = (1 + \sqrt{5}) / 2$ - більший корінь характеристичного рівняння послідовності Фібоначі.

Доведення. Максимальне число кроків n досягається при $a = \Phi_{n+2}$, $b = \Phi_{n+1}$, де n - найбільший номер такий, що $\Phi_{n+2} < N$. Розглядаючи формулу для n -ого члена послідовності Фібоначі, легко зрозуміти, що Φ_{n+2} - найближче ціле до $(1 / \sqrt{5})\, \Phi_{n+2}$. Значить $(1 / \sqrt{5})\, \Phi_{n+2} < N$, отже, $n + 2 < \log_\Phi (\sqrt{5}\, N)$, звідки слідує, що $n < \lceil \log_\Phi (\sqrt{5}\, N) \rceil - 3$ (саме "мінус три", адже розглядається верхнє ціле).

$\log_\Phi (\sqrt{5}\, N) \approx 4{,}785 \cdot \lg N + 1{,}672$, тому, наприклад, з будь-якою парою чисел, менших мільйона, алгоритм Евкліда розбирається не більше, ніж за $\lceil 4{,}785 \cdot 6 + 1{,}672 \rceil - 3 = 31 - 3 = 28$ кроків.

Лістинг алгоритму Евкліда на мові С

```
// Узагальнений алгоритм Евкліда для пошуку найбільшого загального
// дільника gcd = НСД (u, v) цілих позитивних чисел u і v
// і коефіцієнтів a і b рівняння a * u + b * v = gcd
// Все числа покладаються типу long
// Підстановки спрощують запис вихідного тексту
#define isEven (x) ((x & 0x01L) == 0)                // x - парне?
#define isOdd (x) ((x & 0x01L))                      // x - непарне?
#define swap (x, y) (x ^ = y, y ^ = x, x ^ = y) // обмін значень x і y
void GenEuclid (long * u, long * v, long * a, long * b, long * gcd)
{
int k;       // Параметр циклів
long a1, b1, g1;    // Допоміжні змінні
// Алгоритм передбачає, що u> v, якщо u <v, то вони переставляються
if (* u <* v) swap (* u * v);
// Якщо u = n * 2 ^ k1 або v = m * 2 ^ k2, то перед пошуком НОД
// виробляємо скорочення u = u / (2 ^ k), v = v / (2 ^ k),
// де k - мінімальне з k1, k2. Показник k запам'ятовуємо.
for (k = 0; isEven (* u) && isEven (* v); ++ k) {
  * U >> = 1; * V >> = 1;
}
// Задання початкових значень
* A = 1; * B = 0; * Gcd = * u; a1 = * v; b1 = * u - 1; g1 = * v;
do {
  do {
          if (isEven (* gcd)) {
                  if (isOdd (* a) || isOdd (* b)) {
                          * A + = * v; * B + = * u;
                  }
                  * A >> = 1; * B >> = 1; * Gcd >> = 1;
```

```
            }
            if (isEven (g1) || * gcd <g1) {
                    swap (* a, a1); swap (* b, b1); swap (* gcd, g1);
            }
    } While (isEven (* gcd));
    while (* a <a1 || * b <b1) {
            * A + = * v; * B + = * u;
    }
    * A - = a1; * B - = b1; * Gcd - = g1;
} While (g1> 0);
while (* a> = * v && * b> = * u) {
    * A - = * v; * B - = * u;
}
// здійснюємо множення коефіцієнтів рівняння
// на скорочений раніше множник 2 ^ k
* A << = k; * B << = k; * Gcd << = k;
}
```

Розширений алгоритм Евкліда і співвідношення Безу

Формули для r_i можуть бути переписані таким чином:

$r_1 = A + b (- q_0)$

$r_2 = B - r_1 q_1 = a (- q_1) + b (1 + q_1 q_0)$

. . .

$(A, b) = r_n = as + bt$

тут s і t цілі.

Це представлення найбільшого загального дільника називається співвідношенням Безу, а числа s і t - коефіцієнтами Безу. Співвідношення Безу є ключовим в доказі основної теореми арифметики.

1.3 Постановка задач дослідження.

Метою досліджень у дипломній роботі є удосконалення методів пошуку найбільшого спільного дільника на основі використання теоретико-числових базисів Радемахера та Крестенсона.

Для досягнення поставленої мети необхідно ефективно розв‘язати низку взаємопов‘язаних завдань:

1) аналіз, вибір методів розв’язання задач пошуку найбільшого спільного дільника;

2) удосконалення алгоритму пошуку найбільшого спільного дільника на основі алгоритму Евкліда з використанням ТЧБ Радемахера-Крестенсона;

3) розробка паралельного алгоритму пошуку найбільшого спільного дільника з використанням ТЧБ Радемахера-Крестенсона;

4) дослідження часових складностей розроблених алгоритмів з класичними;

5) провести програмну реалізації розроблених алгоритмів пошуку найбільшого спільного дільника.

Для проведення поставлених в даній роботі досліджень, використовуються результати з таких областей знань: для дослідження та аналізу основних операцій в асиметричних криптоалгоритмах - теорія чисел та алгебра Евкліда; теорія алгоритмів для дослідження часових складностей методів пошуку найбільшого спільного дільника.

Висновки до розділу 1.

1. Проведений аналіз способів кодування даних на основі теоретико-числових базисів Радемахера-Крестенсона, який вказує на перспективи застосування їх для розв’язку різної складності задач.

2. Наведені теоретичні основи системи залишкових класів теоретико-числового базису Крестенсона та його модифікації.

3. Проведена постановка завдань на дипломну роботу розв’язання яких дозволить досягнути мету досліджень.

2 АЛГОРИТМИ ПОШУКУ НАЙБІЛЬШОГО СПІЛЬНОГО ДІЛЬНИКА НА ОСНОВІ ТЕОРЕТИКО-ЧИСЛОВИХ БАЗИСІВ РАДЕМАХЕРА-КРЕСТЕНСОНА

2.1 Теоретичні засади виконання операцій в базисі Крестенсона

Аналіз наукових тенденцій розвитку теорії та перспективних інформаційних технологій покращення ефективності опрацювання інформаційних потоків в комп'ютерних мережах, потребує поглибленого дослідження теоретичних засад базисів Крестенсона та Галуа. Слід зауважити, що найбільш фундаментально досліджено цілочисельну форму в СЗК, яка утворюється на основі прямого перетворення ТЧБ Крестенсона.

Тому є доцільним дослідити інші форми СЗК, які можуть бути використані для реалізації високопродуктивних алгоритмів опрацювання і захисту інформаційних потоків, а також виконати порівняльний аналіз названих ТЧБ з базисом Радемахера, який породжує двійкову систему числення на основі відповідних критеріїв.

Відомо, що двійкова система числення, яка використовується в сучасних КС, має певні недоліки – наявність міжрозрядних зв'язків та велику розрядність [46]. Тому актуальним є розвиток і застосування непозиційних систем числення, в яких відсутні вказані недоліки. Прикладом може бути СЗК, або, як її ще називають, представлення чисел у базисі Крестенсона [29], [47]. Хоча вона не набула значного поширення у зв'язку з необхідністю визначення умов переповнення, складністю та громіздкістю зворотнього перетворення чисел у десяткову систему числення, а також складнощами реалізації операцій ділення та порівняння, але СЗК можна ефективно використовувати у мультибазисних процесорах, спеціалізованих обчислювальних машинах для виконання операцій додавання, віднімання та множення, наприклад, у задачах лінійної алгебри (матрично–векторні

операції) тощо. Необхідно відмітити, що ця система особливо ефективна при обчисленнях з великими числами [37], [48].

Фундаментальною основою СЗК є теорія чисел [51], [52], зокрема, властивості китайської теореми про залишки. Будь–яке ціле додатнє число N у десятковій системі числення представляється в СЗК у вигляді набору найменших додатніх залишків від ділення цього числа на фіксовані цілі додатні попарно взаємно прості числа $p_1, p_2, \dots, p_n$ ($N_{10}=(b_1, b_2, \dots, b_n)_{p1, p2, \dots, pn}$, де $b_i=N \bmod p_i$), які називаються модулями (n – кількість модулів). При цьому повинна виконуватись умова $0 \le N \le P-1$, де $P=\prod_{i=1}^{n} p_i$.

На відміну від позиційних систем числення, де величина визначеного розряду суми, різниці або множення залежить не тільки від значень відповідних, але і від попередніх розрядів доданків або множників, в СЗК додавання, віднімання та множення цілих чисел виконується окремо по кожному модулю і переноси між розрядами відсутні. Отже, такі операції в СЗК є модульними [54].

Нехай два десяткові числа A і B, записані в СЗК за вибраними модулями: $A_{10}=(a_1, a_2, \dots, a_i, \dots, a_n)_{p1, p2, \dots, pi, \dots, pn}$, $B_{10}=(b_1, b_2, \dots, b_i, \dots, b_n)_{p1, p2, \dots, pi, \dots, pn}$. Тоді:

$$A_{10} \pm B_{10}=C_{10}=(c_1, c_2, \dots, c_i, \dots, c_n)_{p1, p2, \dots, pi, \dots, pn};$$

$$A_{10} \times B_{10}=D_{10}=(d_1, d_2, \dots, d_i, \dots, d_n)_{p1, p2, \dots, pi, \dots, pn},$$

де $c_i= a_i \pm b_i$, $d_i= a_i \times b_i$.

Останні рівності справедливі лише в тому випадку, коли результат операції не виходить за межі інтервалу $\prod_{i=1}^{n} p_i - 1$.

Зворотнє перетворення із базису Крестенсона у десяткову систему числення є досить громіздким і грунтується на використанні китайської теореми про залишки [51]:

$$N = \left(\sum_{i=1}^{n} b_i B_i \right) mod\ P, \tag{2.1}$$

де $B_i = M_i m_i$, $M_i = \frac{P}{p_i}$, m_i шукається з виразу $(M_i m_i) mod\ p_i = 1$, при цьому повинна виконуватись умова $\left(\sum_{i=1}^{n} B_i \right) mod\ P = 1.$

Слід зазначити, що при переведенні чисел із СЗК у десяткову систему числення значну обчислювальну складність становить пошук коефіцієнтів $m_i = M_i^{-1} mod\ p_i$. У роботі [29] було запропоновано досконалу форму СЗК (ДФ СЗК), у якій підбір модулів такий, що m_i=1, тобто

$$M_i\ mod\ p_i = 1. \tag{2.2}$$

Крім того, було підібрано декілька наборів для чотирьох та п’яти модулів ДФ СЗК. Подальший розвиток ДФ СЗК отримала у роботах [63, 64], у яких було встановлено правила побудови наборів з будь–якої кількості модулів ДФ СЗК для будь–якого діапазону десяткових чисел. Шукані модулі повинні отримуватися з такої умови:

$$\begin{cases} p_1 = 2 \\ p_i = p_1 p_2 \ldots p_{i-1} + 1,\ 1 < i < n. \\ p_n = p_1 p_2 \ldots p_{n-1} - 1. \end{cases} \tag{2.3}$$

Слід зазначити, що запропонована система не вичерпує всіх можливих наборів для базису Крестенсона при заданих n. Наприклад, при n=5 набір модулів, отриманий за допомогою системи (2.3), буде P_{51} = 2, 3, 7, 43, 1805. Однак відомі також набори P_{52} = 2, 3, 7, 83, 85 та P_{53} = 2, 3, 11, 17, 59. При n=6 набір модулів, отриманий з (2.3), буде таким: P_{61} = 2, 3, 7, 43, 1807, 3263441. Усі можливі набори модулів для ДФ СЗК базису Крестенсона при n=6, відповідні їм діапазони десяткових чисел та розрядність у двійковій системі наведені у таблиці 2.1. Як видно з таблиці 2.1, набір модулів, отриманий за допомогою системи (2.3), найоптимальніший, оскільки в цьому випадку величина P є максимальна, що дозволяє розглядати найбільший діапазон десяткових чисел. При цьому досягається зменшення розрядності вдвічі.

Крім того, у цих роботах запропонована напівдосконала форма СЗК (m_i=±1), яку зручно використовувати у випадку обмеженої кількості модулів та необхідності розгляду великих чисел.

Таблиця 2.1 – Можливі набори модулів при n=6 для ДФ СЗК та відповідні їм діапазони десяткових чисел (в дужках – розрядність у двійковій системі)

№	p_1, p_2	p_3	p_4	p_5	p_6	P
1	2, 3 (2)	7 (3)	43 (6)	1807 (11)	3263441 (22)	$1{,}0650050423922\times10^{13}$ (44)
2				1811 (11)	654133 (20)	$2{,}139450562578\times10^{12}$ (41)
3				1819 (11)	252701 (18)	$8{,}30151592914\times10^{11}$ (41)
4				1825 (11)	173471 (18)	$5{,}7175174245\times10^{11}$ (40)
5				1871 (11)	51985 (16)	$1{,}7565866661\times10^{11}$ (38)
6				1901 (11)	36139 (16)	$1{,}24072631634\times10^{11}$ (37)

Продовження таблиці 2.1

№	p_1, p_2	p_3	p_4	p_5	p_6	P
7				1945 (11)	25271 (15)	$8{,}876868357\times10^{10}$ (37)
8				2053 (12)	15011 (14)	$5{,}5656554898\times10^{10}$ (36)
9				2167 (12)	10841 (14)	$4{,}2427359282\times10^{10}$ (36)
10				2501 (12)	6499 (13)	$2{,}9354722194\times10^{10}$ (35)
11				3041 (12)	4447 (13)	$2{,}4423128562\times10^{10}$ (35)
12		11 (4)		3611 (12)	3613 (12)	$2{,}3562056658\times10^{10}$ (35)
13			47 (6)	395 (9)	779729 (20)	$6{,}0797809317\times10^{10}$ (36)
14				481 (9)	2203 (12)	$2{,}091735282\times10^{9}$ (31)
15			53 (6)	271 (9)	799 (10)	$4{,}81993554\times10^{8}$ (29)
16			71 (7)	103 (7)	61429 (16)	$1{,}8867671634\times10^{10}$ (35)
17			23 (5)	31 (5)	47057 (16)	$2{,}214408306\times10^{9}$ (32)

Напівдосконала форма дозволяє побудувати систему з двох модулів, що неможливо у ДФ СЗК. Для цього необхідно вибрати два будь–які послідовні числа p_1 та $p_2=p_1+1$, які завжди будуть взаємно простими, оскільки для них виконується умова:

$$\begin{cases} (p_1+1) mod\ p_1 = 1 \\ p_1\ mod(p_1+1) = -1. \end{cases} \qquad (2.4)$$

Загальна система для визначення набору з будь–якої кількості модулів напівдосконалої форми СЗК має вигляд:

$$\begin{cases} p_2 = p_1 + 1 \\ p_i = p_1 p_2 \ldots p_{i-1} \pm 1, \end{cases} \qquad (2.5)$$

де $i = 3, \ldots, n$.

Перспективними модифікаціями СЗК, які на даний час глибоко досліджуються науковою школою професора Я.М.Николайчука, є нормалізована та розмежована форми СЗК, описані в [65].

Часова складність алгоритму формування структуризованих даних на низових рівнях КС в нормалізованій досконалій формі в базисі Крестенсона включає характеристики часової складності виконання операцій додавання, множення та модульної операції, що реалізовуються за допомогою універсальних процесорів різної розрядності n. Оцінка функціональних можливостей реалізації арифметики та базових функцій над числами в базисах Радемахера, Крестенсона представлена в таблиці 2.2.

Таблиця 2.2 – Функціональні можливості досліджуваних теоретико-числових базисів Радемахера та Крестенсона

№	Базові операції	Радемахер	Крестенсон
1	2	3	4
1	Додавання	$2nv$	v
2	Зсув	v	v
3	Множення	$2v(2n+1)$	v
4	Рівності	v	v
5	Знакова(старшинства)	nv	-
6	Віднімання	$(3n+5)v$	-
7	Ділення	n^2v	-
8	Модульна	n^2v	$2nv$

У таблиці n – розрядність представлення чисел, v – тривалість спрацювання мікроелектронного вентиля. Експериментальні дослідження

проводилися на основі Sempron 3000+, для якого $\nu = \frac{1 \cdot c}{2000 млн.опер} = 5 \cdot 10^{-10} \frac{c}{опер}$.

Результати досліджень часової складності реалізації модульної операції над числами в базисах Радемахера, Крестенсона на основі Sempron 3000+ приведені на рисунку 2.1.

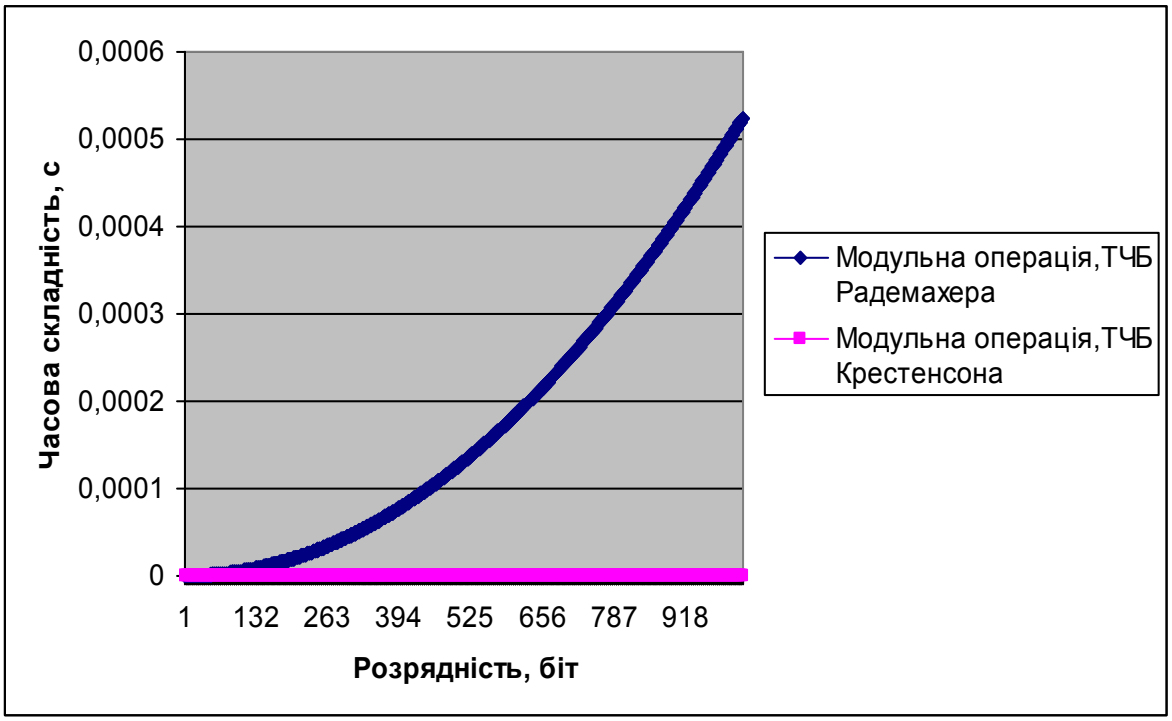


Рисунок 2.1 – Часова складність реалізації модульної операції над числами в базисах Радемахера та Крестенсона в залежності від розмірності.

З рисунку 2.1 видно, що алгоритм обчислення залишку у базисі Крестенсона характеризується суттєвим збільшенням швидкодії, що є важливою перевагою його застосування на низових рівнях розподілених комп'ютерних систем шляхом використання спеціалізованих процесорів та контролерів, які призначені для роботи в промислових умовах при дії впливів вібрації, широкого діапазону температур, вибухобезпечності, мобільного виконання та інших факторів.

2.2 Удосконалення реалізації алгоритму Евкліда з використанням розмежованої системи числення Радемахера – Крестенсона

Для знаходження найбільшого спільного дільник можна скористатися класичним алгоритмом Евкліда [83]

Алгоритм 2.1 (Евкліда) – знаходження найбільшого спільного дільника двох чисел X, Y.

Нехай X, Y цілі числа, не рівні одночасно нулю, і послідовність чисел $X, Y, r_1 > r_2 > \ldots > r_i > \ldots > r_n$ визначена так, що кожне r_i - залишок від ділення передпопереднього числа на попереднє, а передостаннє ділиться націло на останнє, тобто:

$$
\begin{gathered}
X = Y \cdot q_0 + r_1; \\
Y = r_1 \cdot q_1 + r_2; \\
r_1 = r_2 \cdot q_2 + r_3; \\
r_2 = r_3 \cdot q_3 + r_4; \\
\ldots \\
r_i = r_{i+1} \cdot q_{i+1} + r_{i+2}; \\
\ldots \\
r_{n-1} = r_n \cdot q_n.
\end{gathered}
$$

Тоді НСД(X, Y) рівний r_n, останньому ненульовому члену даної послідовності.

В [28] оцінено складність цього алгоритму і вона не перевищує 5κ операцій ділення з остачею, де κ - кількість цифр в десятковому записі більшого з чисел X, Y. Оскільки операція ділення двох чисел розрядності n має обчислювальну складність $O((n+1)^2)$, що значно збільшує час $O1(17,5n(n+1)^2)$ виконання операції пошуку найбільшого спільного дільника з використанням алгоритму Евкліда в базисі Радемахера (двійковій формі).

Алгоритм 2.2 – Удосконалення реалізації алгоритму Евкліда з використанням розмежованої системи числення Радемахера – Крестенсона.

Для знаходження залишків r_i з використанням розмежованої системи числення Радемахера – Крестенсона операція ділення замінюється операцією додавання залишків степеневих коефіцієнтів по заданому модулі наступним чином:

1. Представляємо X, Y в базисі Радемахера:

$$X = x_{n-1} \cdot 2^{n-1} + x_{n-2} \cdot 2^{n-2} + \ldots + x_i \cdot 2^i + \ldots + x_0 \cdot 2^0;$$
$$Y = y_{n-1} \cdot 2^{n-1} + y_{n-2} \cdot 2^{n-2} + \ldots + y_i \cdot 2^i + \ldots + y_0 \cdot 2^0,$$

де y_i, x_i =0, 1, n–розрядності X, Y.

Для виконання 1 кроку алгоритму Евкліда, потрібно знайти значення $r_1 = res\left(X \bmod(Y)\right), r_2 = res\left(Y \bmod(r_1)\right), \ldots, \; r_{n-1} = res\left(r_{n-1} \bmod(r_n)\right)$ згідно формул:

$$r_1 = \left(\sum_{i=1}^{n} res\left(x_i \cdot 2^i\right) \bmod Y \right) \bmod Y;$$

$$r_2 = \left(\sum_{i=1}^{n} res\left(y_i \cdot 2^i\right) \bmod r_1 \right) \bmod r_1;$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (2.6)$$

$$r_{n-1} = \left(\sum_{i=1}^{n} res\left(r_i \cdot 2^i\right) \bmod r_n \right) \bmod r_n.$$

Таким чином, з врахуванням кількості кроків знаходимо всі залишки в алгоритмі Евкліда зі складністю $O\left(17{,}5n \cdot \left(\log_2 \frac{n}{2} \right)\right)$. На рисунку 2.1 подано оцінки складностей алгоритму Евкліда та його удосконалення з використанням розмежованої системи числення Радемахера – Крестенсона.

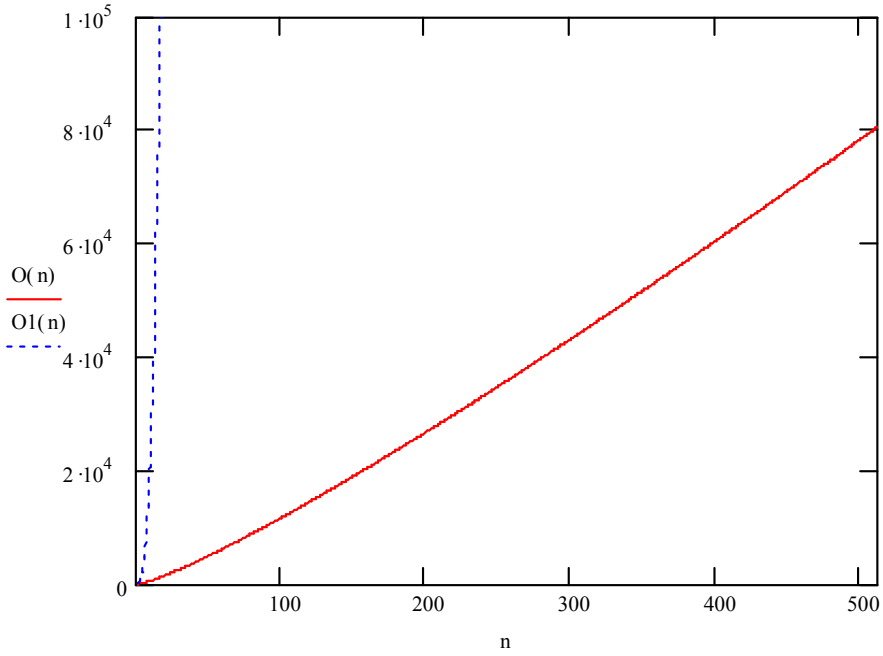


Рисунок 2.1 – Складності алгоритму Евкліда

Результати чисельного експерименту показали, що використання розмежованої системи числення дозволяє значно зменшити складність. Тому для зменшення складності паралельного алгоритму Шуфа доцільно скористатися СЗК, яка може бути використана для реалізації високопродуктивних алгоритмів формування і опрацювання інформаційних потоків, а також застосувати ТЧБ Крестенсона та Радемахера, який породжує двійкову систему числення на основі відповідних критеріїв для знаходження НСД. Для цього доцільно використати запропонований алгоритм знаходження НСД. Незважаючи на переваги, слід зазначити, що основним недоліком цього алгоритму є його послідовність і неможливість розпаралелення.

2.3 Знаходження найбільшого спільного дільника з застосуванням теоретико-числових базисів Радемахера–Крестенсона

З метою реалізації алгоритму на основі розпаралелення обчислювальних операцій, розроблений алгоритм на базі використання ТЧБ Радемахера-Крестенсона, який базується на виявленні та порівнянні нульових залишків в системі простих модулів і дозволяє вирішувати дві задачі, а саме:

1) факторизація чисел (розклад на прості множники);
2) знаходження найбільшого спільного дільника.

Алгоритм 2.3 – знаходження НСД з застосуванням ТЧБ Радемахера–Крестенсона.

Вхід: X, Y.

Вихід: $Z = НСД(X, Y)$.

1. Представляємо X, Y у базисті Радемахера в двійковій системі числення:

$$X = x_{n-1} \cdot 2^{n-1} + x_{n-2} \cdot 2^{n-2} + \ldots + x_i \cdot 2^i + \ldots + x_0 \cdot 2^0,$$
$$Y = y_{n-1} \cdot 2^{n-1} + y_{n-2} \cdot 2^{n-2} + \ldots + y_i \cdot 2^i + \ldots + y_0 \cdot 2^0. \quad (2.7)$$

де n – розрядність процесора які представляють X, Y.

2. Представляємо двійкові коди чисел X, Y в розмежованій системі чисел залишкових класів у вигляді векторів залишків згідно виразів:

$$res(x_i \cdot 2^i) \bmod p_j = a_i,$$
$$res(y_i \cdot 2^i) \bmod p_j = b_i. \quad (2.8)$$

В результаті коди чисел X, Y можна представити у вигляді векторів

залишків у системі простих модулів p_j ,які утворюють відповідні матриці (2.9), причому $E[\log_2 p_k - 1] \le E[\log_2 2^n] = n$, тобто:

$$\left|\begin{array}{cccccc|c|cccccc} a_{n-1,1} & a_{n-2,1} & \cdots & a_{i,1} & \dots & a_{0,1} & \bmod p_1 & b_{n-1,1} & b_{n-2,1} & \cdots & b_{i,1} & \dots & b_{0,1} \\ a_{n-1,2} & a_{n-2,2} & \dots & a_{i,2} & \dots & a_{0,2} & \bmod p_2 & b_{n-1,2} & b_{n-2,2} & \dots & b_{i,2} & \dots & b_{0,2} \\ \dots & \dots & \dots & \dots & \dots & & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n-1,j} & a_{n-2,j} & \dots & a_{i,j} & \dots & a_{0,j} & \bmod p_j & b_{n-1,j} & b_{n-2,j} & \dots & b_{i,j} & \dots & b_{0,j} \\ \dots & \dots & \dots & \dots & \dots & & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n-1,k} & a_{n-2,k} & \dots & a_{i,k} & \dots & a_{0,k} & \bmod p_k & b_{n-1,k} & b_{n-2,k} & \dots & b_{i,k} & \dots & b_{0,k} \end{array}\right| \qquad (2.9)$$

Приведене представлення X, Y у вигляді векторів залишків окремих бітів можна узагальнити наступним чином:

$X = \|a_{ij}\|$, $Y = \|b_{ij}\|$ і $i \in \overline{0,n}$, $j \in \overline{0,k}$ та $0 \le a_{ij}, b_{ij} \le p_j - 1$.

Причому

$$1 \le Z \le \prod_{j=1}^{k} p_j \ . \qquad (2.10)$$

3. Для отримання значення (2.11) потрібно представити матриці X і Y у вигляді двох векторів згідно виразів:

$$a_j = res\left(\sum_{i=1}^{n-1} a_{ij} (\bmod p_j)\right);$$

$$b_j = res\left(\sum_{i=1}^{n-1} b_{ij} (\bmod p_j)\right), \qquad (2.11)$$

для всіх $j \in \overline{0,k}$.

З (2.11) отримуємо два вектори, які представляють числа X і Y у цілочисельній системі залишкових класів в базисі Крестенсона:

$$\begin{aligned} X &= (a_1 \quad a_2 \quad \dots \quad a_j \quad \dots \quad a_k); \\ Y &= (b_1 \quad b_2 \quad \dots \quad b_j \quad \dots \quad b_k); \\ p_j &= \quad p_1 \quad p_2 \quad \dots \quad p_j \quad \dots \quad p_k. \end{aligned} \tag{2.12}$$

Для визначення компонентів (модулів) найбільшого мультиплікативного спільного дільника Z необхідно виконати порівняння $a_i = b_i = 0, \forall i \in \overline{1,k}$. Тоді найбільший мультиплікативний дільник знаходимо згідно виразу:

$$z = \prod_{j=1}^{k} p_j \,, \tag{2.13}$$

та умови

$$p_j = \begin{cases} p_j, a_j = b_j = 0 \\ 1, a_j \neq b_j \end{cases}. \tag{2.14}$$

4. Для пошуку НСД для компонентів p_j які відповідають умові (2.14) проводиться представлення вхідних чисел X і Y в розмежованій системі числення згідно (2.9) створюємо матрицю залишків по $p_j^{\,m}$, де m - показник степеня, при якому виконується умова (2.14), тобто отримуємо вектори:

$$\begin{bmatrix} a_j^{(2)} = res\left(\sum_{i=1}^{n-1} a_{ij} (\mathrm{mod}\, p_j^{\,2})\right) \\ a_j^{(3)} = res\left(\sum_{i=1}^{n-1} a_{ij} (\mathrm{mod}\, p_j^{\,3})\right) \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ a_j^{(m)} = res\left(\sum_{i=1}^{n-1} a_{ij} (\mathrm{mod}\, p_j^{\,m})\right) \end{bmatrix}; \quad \begin{bmatrix} b_j^{(2)} = res\left(\sum_{i=1}^{n-1} b_{ij} (\mathrm{mod}\, p_j^{\,2})\right) \\ b_j^{(3)} = res\left(\sum_{i=1}^{n-1} b_{ij} (\mathrm{mod}\, p_j^{\,3})\right) \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ b_j^{(m)} = res\left(\sum_{i=1}^{n-1} b_{ij} (\mathrm{mod}\, p_j^{\,m})\right) \end{bmatrix}. \tag{2.15}$$

для всіх $j \in \overline{1,k}$.

Перевірка умови (2.5) для степенів $p_j^{\,m}$ виконується згідно порівнянь

$$\begin{array}{lllll} X^{(m)} = & (a_j{}^{(1)} & a_j{}^{(2)} & \dots & a_j{}^{(k)}); \\ Y^{(m)} = & (b_j{}^{(1)} & b_j{}^{(2)} & \dots & b_j{}^{(k)}); \\ \{p_j{}^{k}\} = & p_j{}^{1} & p_j{}^{2} & \dots & p_j{}^{k}. \end{array} \qquad (2.16)$$

Тоді значення Z обчислюється згідно наступної мультиплікативної функції $Z = \prod_{j=1}^{k} p_j{}^{m_j}$.

В порівнянні з відомим алгоритмом Евкліда запропонований алгоритм знаходження НСД характеризується наступними перевагами:

1.Обчислення матриць $a_j{}^{m}, b_j{}^{m}$ двох векторів можна обчислити паралельно з використанням двох процесорів.

2.Виконується операція порівняння двох компонент $X^{(m)}$ і $Y^{(m)}$.

3.Отримання добутків $p_j{}^{m_j}$ які дорівнюють шуканому НСД.

Алгоритм 2.4 – Удосконалення алгоритму 2.3.

Запропонований алгоритм можна суттєво удосконалити шляхом скорочення кроків алгоритму вилучення 3 кроку згідно виразів (2.11) з виконанням додаткової умови:

$$\min j : a_j = b_j = 0 . \qquad (2.17)$$

Для пошуку найбільшого спільного дільника для компонентів p_j які відповідають умові (2.17) проводиться представлення вхідних чисел X і Y в розмежованій системі числення згідно (3.16) створюємо матрицю залишків по $p_j{}^{m}$, де m - показник степеня, при якому виконується умова $a_j{}^{(m)} = b_j{}^{(m)} = 0$, тобто отримуємо вектори:

$$\begin{cases} a_j^{(2)} = res\left(\sum_{i=1}^{n-1} a_{ij} (\mathrm{mod}\, p_j^{\,2})\right) \\ a_j^{(3)} = res\left(\sum_{i=1}^{n-1} a_{ij} (\mathrm{mod}\, p_j^{\,3})\right) \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_j^{(m)} = res\left(\sum_{i=1}^{n-1} a_{ij} (\mathrm{mod}\, p_j^{\,m})\right) \end{cases} \quad \begin{cases} b_j^{(2)} = res\left(\sum_{i=1}^{n-1} b_{ij} (\mathrm{mod}\, p_j^{\,2})\right) \\ b_j^{(3)} = res\left(\sum_{i=1}^{n-1} b_{ij} (\mathrm{mod}\, p_j^{\,3})\right) \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ b_j^{(m)} = res\left(\sum_{i=1}^{n-1} b_{ij} (\mathrm{mod}\, p_j^{\,m})\right) \end{cases}. \tag{2.18}$$

Після цього знаходимо значення залишків по $p_{j,k}^{\,m_s} = p_j^{\,m} \cdot p_{j+k}^{\,s}$, де s - показник степеня, при якому виконується умова $a_{j,k} = b_{j,k} = 0$, k - наступний після j-го простого модуля, для якого виконується умова $a_{j,k} = b_{j,k} = 0$, тобто:

$$\begin{cases} a_{j,k}^{(2)} = res\left(\sum_{i=1}^{n-1} a_{ij} (\mathrm{mod}\, p_{j,k}^{\,m_2})\right) \\ a_{j,k}^{(3)} = res\left(\sum_{i=1}^{n-1} a_{ij} (\mathrm{mod}\, p_{j,k}^{\,m_3})\right) \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{j,k}^{(m_s)} = res\left(\sum_{i=1}^{n-1} a_{ij} (\mathrm{mod}\, p_{j,k}^{\,m_s})\right) \end{cases} \begin{cases} b_{j,k}^{(2)} = res\left(\sum_{i=1}^{n-1} b_{ij} (\mathrm{mod}\, p_{j,k}^{\,m_2})\right) \\ b_{j,k}^{(3)} = res\left(\sum_{i=1}^{n-1} b_{ij} (\mathrm{mod}\, p_{j,k}^{\,m_3})\right) \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ b_{j,k}^{(m_s)} = res\left(\sum_{i=1}^{n-1} b_{ij} (\mathrm{mod}\, p_{j,k}^{\,m_s})\right) \end{cases}. \tag{2.19}$$

Таким чином, добуток всіх модулів до $\sqrt{Y}$ степеня для яких виконується умова $a_{j,k} = b_{j,k} = 0$, буде НСД, тобто:

$$Z = \prod_{j=1}^{k} p_j^{\,m_j} \ . \tag{2.20}$$

Основною перевагою цього алгоритму є отримання добутків $p_j^{\,m_j}$, пропустивши 3 крок алгоритму 2.3, що суттєво пришвидшить роботу алгоритму.

Складність запропонованих алгоритмів визначається обчислювальною складністю наступних операцій: знаходженні залишків a_j, b_j чисел X, Y по

простих модулях $p_j^{m_j}$ для яких виконується умова $a_j = b_j = 0$; обчислення добутку модулів $Z = НСД(X,Y) = \prod_{j=1}^{k} p_j^{m_j}$.

Приклади застосування алгоритмів пошуку НСД.

Нехай потрібно обчислити НСД(3843, 1449).

1. Стандартний алгоритм Евкліда:

3843=1449·1+945

1449=945·1+504

945=504·1+441

504=441·1+63

441=63·7+0

Отже, НСД(3843, 1449)=63.

2. Алгоритм Евкліда в розмежованій системі числення зручно представити у вигляді таблиці 2.2.

Таблиця 2.2 – Алгоритм Евкліда в розмежованій системі числення

1	3843	1	1	1	1	0	0	0	0	0	0	1	1
2	2^imod1449	599	1024	512	256	128	64	32	16	8	4	2	1
3	1449		1	0	1	1	0	1	0	1	0	0	1
4	2^imod945		79	512	256	128	64	32	16	8	4	2	1
5	945			1	1	1	0	1	1	0	0	0	1
6	2^imod504			8	256	128	64	32	16	8	4	2	1
7	504				1	1	1	1	1	1	0	0	0
8	2^imod441				256	128	64	32	16	8	4	2	1
9	441				1	1	0	1	1	1	0	0	1
10	2^imod63				4	2	1	32	16	8	4	2	1

З рядка 2 видно, що (599+1024+512+256+2+1)mod 1449=945.

З рядка 4: (79+256+128+32+8+1)mod 945=504.

З рядка 6: (8+256+128+32+16+1)mod 504=441.

З рядка 8: (256+128+64+32+16+8)mod 441=63.

З рядка 10: (4+2+32+16+8+1)mod 63=0.

Таким чином можна отримати НСД, уникнувши громіздкої операції ділення.

3. Пошук НСД в базисі Крестенсона.

Дану задачу також зручно представити у вигляді таблиці 2.3, врахувавши, що $\sqrt{1449} \approx 38$.

Таблиця 2.3 – Знаходження залишків по простих модулях.

1		2048	1024	512	256	128	64	32	16	8	4	2	1
2	3843	1	1	1	1	0	0	0	0	0	0	1	1
3	1449		1	0	1	1	0	1	0	1	0	0	1
4	2^imod2	0	0	0	0	0	0	0	0	0	0	0	1
5	2^imod3	2	1	2	1	2	1	2	1	2	1	2	1
6	2^imod5	3	4	2	1	3	4	2	1	3	4	2	1
7	2^imod7	4	2	1	4	2	1	4	2	1	4	2	1
8	2^imod11	2	1	6	3	7	9	10	5	8	4	2	1
9	2^imod13	7	10	5	9	11	12	6	3	8	4	2	1
10	2^imod17	8	4	2	1	9	13	15	16	8	4	2	1
11	2^imod19	15	17	18	9	14	7	13	16	8	4	2	1
12	2^imod23	1	12	6	3	13	18	9	16	8	4	2	1
13	2^imod29	18	9	19	24	12	6	3	16	8	4	2	1
14	2^imod31	2	1	16	8	4	2	1	16	8	4	2	1

Продовження таблиці 2.3

15	$2^i \bmod 37$	13	25	31	34	17	27	32	16	8	4	2	1
16	$2^i \bmod 9$	5	7	8	4	2	1	5	7	8	4	2	1
17	$2^i \bmod 27$	23	25	26	13	20	10	5	16	8	4	2	1

З таблиці 2.3 шукаються залишки по простих модулях.

Рядок4: 3843 mod 2=1; 1449 mod 2=1;

Рядок 5: 3843 mod 3=(2+1+2+1+2+1) mod 3=0; 1449 mod 3=(1+1+2+2+2+1) mod 3=0;

Рядок 6: 3843 mod 5=(3+4+2+1+2+1) mod 5=3; 1449 mod 3=(4+1+3+2+3+1) mod 5=4;

Рядок 7: 3843 mod 7=(4+2+1+4+2+1) mod 7=0; 1449 mod 7=(2+4+2+4+1+1) mod 7=0;

Рядок 8: 3843 mod 11=(2+1+6+3+2+1) mod 11=4; 1449 mod 11=(1+3+7+10+8+1) mod 3=8;

Рядок 9: 3843 mod 13=(7+10+5+9+2+1) mod 13=8; 1449 mod 3=(10+9+11+6+8+1) mod 13=6;

Рядок 10: 3843 mod 17=(8+4+2+1+2+1) mod 17=1; 1449 mod 17=(4+1+9+15+8+1) mod 17=4;

Рядок 11: 3843 mod 19=(15+17+18+9+2+1) mod 19=5; 1449 mod 19=(17+9+14+13+8+1) mod 19=5;

Рядок 12: 3843 mod 23=(1+12+6+3+2+1) mod 23=2; 1449 mod 23=(12+3+13+9+8+1) mod 23=0;

Рядок 13: 3843 mod 29=(18+9+19+24+2+1) mod 29=15; 1449 mod 29=(9+24+12+3+8+1) mod 29=28;

Рядок 14: 3843 mod 31=(2+1+16+8+2+1) mod 31=30; 1449 mod 31=(1+8+4+1+8+1) mod 31=23;

Рядок 15: 3843 mod 37=(13+25+31+34+2+1) mod 37=32; 1449 mod 37=(25+34+17+32+8+1) mod 37=6.

Дані розрахунки показують, що спільними простими дільниками є числа 3 та 7. Для знаходження НСД потрібно перевірити їх степені:

Рядок 16: 3843 mod 3^2=(5+7+8+4+2+1) mod 3^2=0; 1449 mod 3^2 =(7+4+2+5+8+1) mod 3^2=0;

Рядок 17: 3843 mod 3^3=(23+25+26+13+2+1) mod 3^3=9; 1449 mod 3^3 =(25+13+20+5+8+1) mod 3^3=18.

Число 7^2=49>38 і його можна не перевіряти. Отже, НСД(3843, 1449)= 3^2·7=63. Даний метод дозволяє провести факторизацію чисел, наприклад 1449=3^2·7·23. Крім того, обчислення можна виконувати паралельно по різних модулях.

4. Удосконалений алгоритм пошуку НСД в базисі Крестенсона.

Будується таблицю 2.4.

Таблиця 2.4 – Знаходження залишків в удосконаленому алгоритмі.

1		2048	1024	512	256	128	64	32	16	8	4	2	1
2	3843	1	1	1	1	0	0	0	0	0	0	1	1
3	1449		1	0	1	1	0	1	0	1	0	0	1
4	2^imod2	0	0	0	0	0	0	0	0	0	0	0	1
5	2^imod3	2	1	2	1	2	1	2	1	2	1	2	1
6	2^imod9	5	7	8	4	2	1	5	7	8	4	2	1
7	2^imod27	23	25	26	13	20	10	5	16	8	4	2	1
8	2^imod45	23	34	17	31	38	19	32	16	8	4	2	1
9	2^imod63	32	16	8	4	2	1	32	16	8	4	2	1
10	2^imod441	284	142	71	256	128	64	32	16	8	4	2	1
11	2^imod693	662	331	512	256	128	64	32	16	8	4	2	1

Аналізуємо таблицю 2.4.

Рядок 4: 3843 mod 2=1; 1449 mod 2=1;

Рядок 5: 3843 mod 3=(2+1+2+1+2+1) mod 3=0; 1449 mod 3=(1+1+2+2+2+1) mod 3=0;

Рядок 6: 3843 mod 9=(5+7+8+4+2+1) mod 9=0; 1449 mod 9 =(7+4+2+5+8+1) mod 9=0;

Рядок 7: 3843 mod 27=(23+25+26+13+2+1) mod 27=9; 1449 mod 27 =(25+13+20+5+8+1) mod 27=18;

Рядок 8: 3843 mod 45=(23+34+17+31+2+1) mod 45=18; 1449 mod 45=(34+31+38+32+8+1) mod 3=8;

Рядок 9: 3843 mod 63=(32+16+8+4+2+1) mod 63=0; 1449 mod 63=(16+4+2+32+8+1) mod 63=0;

Рядок 10: 3843 mod 441=(284+142+71+256+2+1) mod 441=315; 1449 mod 441=(142+256+128+32+8+1) mod 441=126;

Рядок 11: 3843 mod 693=(662+331+512+256+2+1) mod 693=378; 1449 mod 693=(331+256+128+32+8+1) mod 693=63.

З розрахунків також випливає, що НСД(3843, 1449)=63.

Висновки до розділу 2.

1. Проведені дослідження виконання операцій в базисі Крестенсона та його модифікацій, які вказують на ефективність застосування в прикладних задачах теорії чисел.

2. Удосконалено реалізацію алгоритма Евкліда пошуку найбільшого спільного дільника на основі використання теоретико-числових базисів Радемахера-Крестенсона.

3. Розроблені теоретичні основи пошуку найбільшого спільного дільника з використанням теоретико-числових базисів Радемахера та Крестенсона, який на відміну від існуючих дозволяє паралельно в мультипрограмному режимі розв’язувати даного класу задачі та зменшувати часову складність.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ПОШУКУ НАЙБІЛЬШОГО СПІЛЬНОГО ДІЛЬНИКА

3.1 Розробка та реалізація структурної організації алгоритмів пошуку найбільшого спільного дільника

Розробку структурної схеми організації пошуку найбільшого спільного дільника виконаємо згідно запропонованих в розділах 2.2, 2.3 алгоритмів та структурної організації програмних засобів поданих на рисунку 3.1.

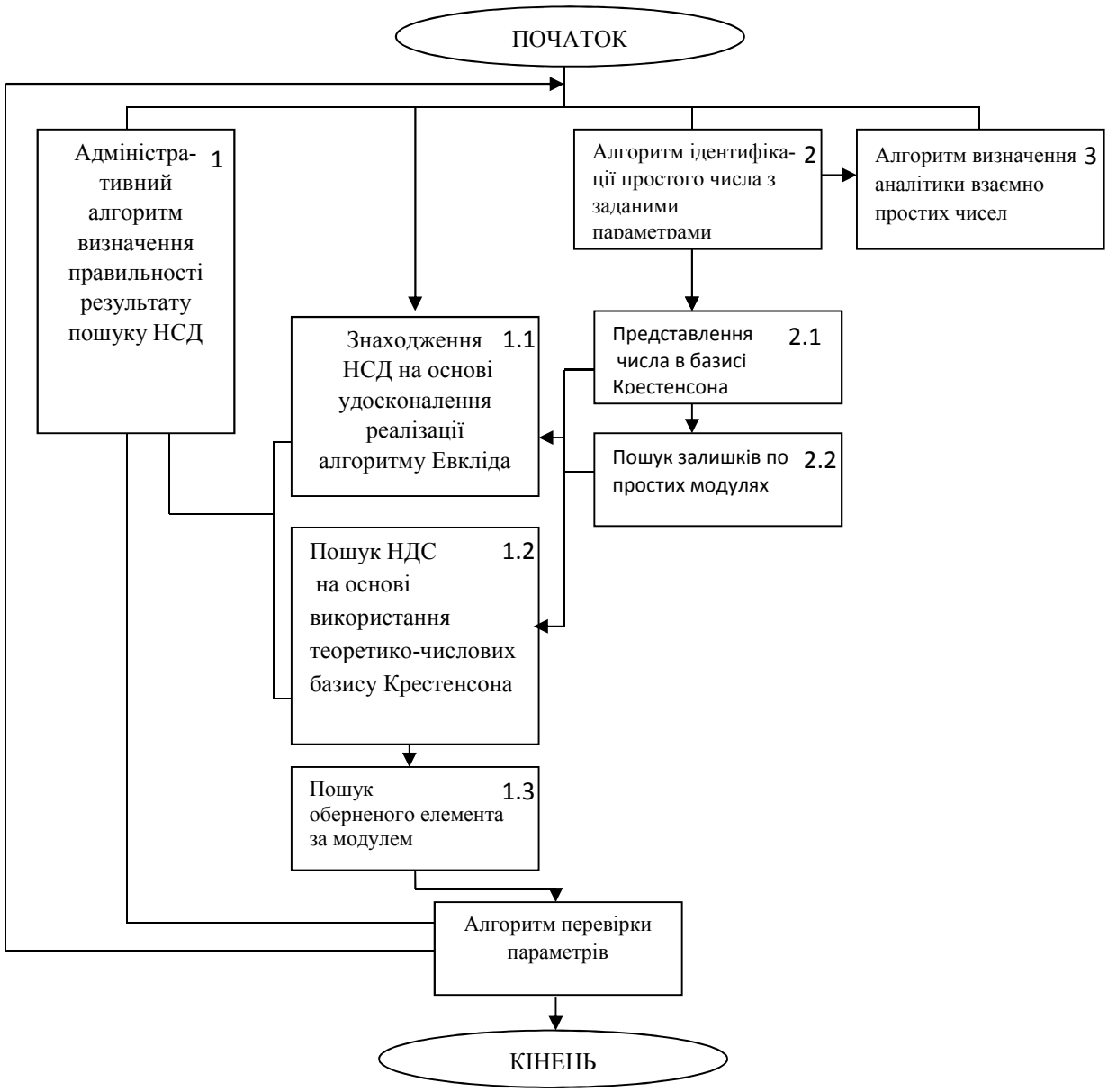


Рисунок 3.1 – Структурна організація пошуку найбільшого спільного дільника на основі використання теоретико-числових базисів Радемахера та Крестенсона.

На рисунку 3.1 програмний модуль (1) виконує адміністративні функції координації та управління виконання обчислювальних процесів компонентними програмними модулями пошуку найбільшого спільного дільника. Алгоритми ідентифікації простого числа (2) та (3) виконуються паралельно в мультипрограмному режимі, або відповідним числом сопроцесорів. Програмні модулі (1.1-2.3) та (2.1-2.2) виконуються в конвеєрному режимі згідно їх часової координації на основі моделі мережевих та матричних моделей руху даних [17].

Для реалізації поставленого завдання було обрано мову програмування C++, а середовище програмування C++ Buider 6.0.

Перевагами мови програмування C++ є гнучкість використання, інкапсуляція, поліморфізм, структурованість. Великою перевагою є тонкість програмного коду, що дозволяє зменшити затрати процесорного часу. Велику роль при обранні мови програмування відіграла наявність зовнішніх модулів, що забезпечують виконання операцій на великими числами.

C++ Buider 6.0 може бути використане всюди, де необхідно доповнити існуючі додатки розширеним стандартом C++, збільшити швидкодію і придати користувачу інтерфейсу професійного рівня.

В таблиці 3.1 представлені характеристики, які вирізняють і вказують на переваги середовища C++ Buider 6.0 серед інших об’єктно-орієнтованих засобів програмування і є одним з найперспективніших для швидкої розробки сучасного математичного забезпечення. Слід відмітити, що C++ Buider 6.0 випускається в трьох варіантах: Standart, Professional і Client/Server Suite.

Слід відмітити, що в таблиці 3.1 представлені основні характеристики, які є важливими при проектування програмних засобів та дозволяють оптимізувати часові характеристики.

Таблиця 3.1 – Характеристики середовища проектування C++ Buider 6.0

Характеристики	Standart	Professional	Client/Server Suite
Язик C++ з розширеною підтримкою стандартів ANSI/ISO	+	+	+
Високопродуктивний 32 – розрядний оптимізуючий компілятор	+	+	+
Швидкий інкрементальний компоновщик додатків	+	+	+
Інтегроване середовище розробки IDE	+	+	+
Механізми двох направленої розробки	+	+	+
Інструменти командної стрічки	+	+	+
Створення бібліотек DLL, LIB і використовуваних програмних файлів EXE	+	+	+
Об’єкти модулів даних	+	+	+

Продовження таблиці 3.1

Повний доступ до Windows API	+	+	+
Хранилище об'єктів	+	+	+
Контролери і сервери OLE Automation	+	+	+
Компоненти для роботи з базами даних	+	+	+
Підтримка з'єднання ODBC		+	+
32- розрядний одно користувацький сервер Local InterBase		+	+
Генератор дистрибутивів InstallShield Express		+	+
Internet Solutions Pack для розробки Web-додатків		+	+
Драйвери SQL Links для баз даних Oracle, Sybase, Informix, DB2, InterBase			+
SQL Monitor			+
Visual Query Builder			+

Крім вказаних в таблиці 3.1 переваг слід відмітити, що між програмними продуктами C++ Buider 6.0 і Borland C++ існує повна і взаємна функціональна сумісність. C++ Buider 6.0 добавляє процесу програмування нову якість – швидку візуальну розробку додатків C++.

Крім того, всі компоненти, форми і модулі даних, які накопили програмісти в Delphi можуть бути використані з використанням додатків C++ Buider 6.0 для Windows без будь-яких змін. Тому C++ Buider 6.0 ідеально підходить тим розробникам, які хочуть забезпечити потужність C++ і зберегти продуктивність Delphi.

3.2 Реалізація компонентів алгоритмів пошуку найбільшого спільного дільника

3.2.1 Алгоритм ідентифікації простого числа з використанням базису Крестенсона

Основними перевагами даного алгоритму ідентифікації простого числа n є використання системи залишкових класів по всіх простих модулях до $\sqrt{n}+1$, якщо один з залишків рівний нулю, то до залишків додаємо 2 по всіх модулях, ця операція припиняється коли по одному з модулів отримаємо 0. Тоді пропускаємо числа кратні модулю, по якому залишок 0 і знаходимо послідовність простих чисел. Це дозволяє значно зменшити складність алгоритму пошуку простих чисел, за рахунок введення циклу D.

Особливості даної структури, яка по ефективності переважає відомі алгоритми завдяки введення блоку (3) швидкого перетворення з базису Радемахера в базис Крестенсона та модуля пошуку залишків в розмежованій системі числення. Схема алгоритму ідентифікації простого числа з використанням базису Крестенсона представлена на рисунку 3.2.

Алгоритм дозволяє оперативно знаходить прості числа в діапазоні до 2^{20} за 60 с, а в діапазоні 2^{30} за 180 с, що вказує на переваги застосування системи залишкових класів базису Крестенсона для розв'язання даного класу задач. Після того, як проведена ідентифікація простих чисел, які широко використовуються в асиметричних системах захисту інформаційних потоків

необхідно визначити ще одну властивість чисел – взаємно простоту, для цього необхідно знайти найбільший спільний дільник.

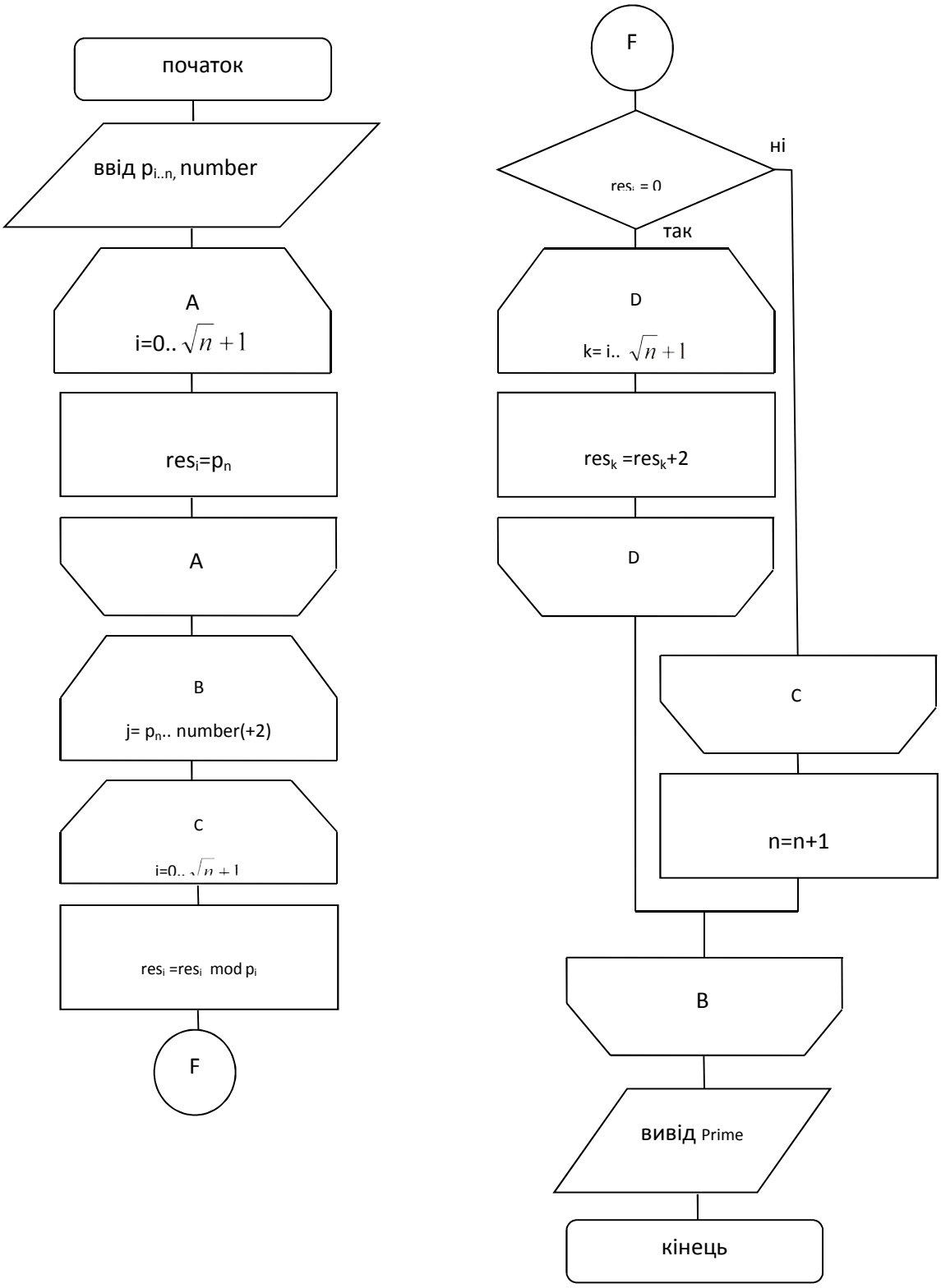


Рисунок 3.2 – Схема алгоритму ідентифікації простих чисел втеоретико-числовому базисі Крестенсона.

3.2.1 Алгоритм визначення простих та взаємнопростих чисел

Відомі алгоритми визначення простих та взаємно простих чисел базуються на використанні решета Еретосфена, ймовірносному тесту на простоту, для визначення взаємно простоти – алгоритмом Евкліда. Функціональними обмеженнями даних алгоритмів є використання для обчислень багаторівневого базису Радемахера (десяткової системи числення), які характеризуються алгоритмічно складними операціями модульного ділення, множення та сумування з наскрізними переносами. Тому, при використання багато розрядних чисел необхідно використовувати систему числення залишкових класів теоретико-числового базису Крестенсона, який дозволяє проводити розрахунки над числами значно меншої розмірності в залежності від обраних модулів.

Запропонований алгоритм базується на рекурентному обчисленні залишків по заданому модулю шляхом обчислення значення на основі попереднього:

$$b_{i+1} = 2 \cdot b_i (\mathrm{mod}\, p). \tag{3.1}$$

При чому, стартова позиція рекурентної перевірки подільності числа на прості множники визначається згідно виразу:

$$res\, 2^i (\mathrm{mod}\, p) + res \sum_{j=0}^{n} 2^j (\mathrm{mod}\, p) \equiv 0 (\mathrm{mod}\, p). \tag{3.2}$$

Результати реалізації даного алгоритму представлено в таблиці 3.1.

Отримана аналітика простих чисел, які не задовольняють умову (3.2) ні по одному з простих модулів менше половини розрядності шуканого p приведена в таблиці 3.2.

Таблиця 3.1 - Пошук простих чисел виду 2^n+3, розклад на прості множники.

131072		65536		32768		16384		8192		4096		2048		1024		512		256		128		64		32		16		8		4		2		1	
18		17		16		15		14		13		12		11		10		9		8		7		6		5		4		3		**2**		**1**	
5,7	1	107	1		1	7	1	5,11, 146	1		1	7	1	13, 79	1	5, 103	1	7, 37	1	131	1	67	1	5,7	1	19	1	11	1	7	1		**2**		**1**
7	1	13, 109	1	5,11	1	7, 131	1	83	1		1	5,7	1		1		1	7	1	5	1		1	7,11	1	13, 19	1	5	1	7, 59	1	29, 101	1		1
5,7, 11	1		1		1	7	1	79,5	1		1	7, 29	1	13	1	5	1	7, 37	1	11, 53	1		1	5,7	1	19, 97	1		1	7	1	5	1	61	1
7	1	13	1	5, 109	1	7	1		1		1	5,7	1		1	11	1	7, 137	1	5	1	103	1	7	1	13, 19	1	5, 131	1	7	1		1		1
5,7	1	79	1		1	7,	1	5	1		1	7, 11, 139	1	13, 131	1	5	1	7, 37	1	59	1		1	5,7,	1	19	1	29	1	7	1	5,11	1	67	1
7, 131	1	13	1	5	1	7, 109	1	11,29 53,97	1		1	5,7	1		1		1	7	1	5,83	1	61	1	7	1	13, 19	1	5,11	1	7	1		1		1
5,7	1	79	1	11	1	7	1	5	1		1	7, 101	1	13	1	5	1	7, 37	1		1		1	5,7, 11	1	19	1	103	1	7	1	5	1		1
7,11	1	13	1	5	1	7	1	109	1	67	1	5,7, 59	1		1		1	7	1	5, 11, 107	1	131	1	7,29	1	13, 19	1	5	1	7	1	79	1		1
5,7	1		1	29	1	7	1	5, 131	1	61	1	7	1	13	1	5,11	1	7, 37	1		1		1	5,7	1	19, 97	1		1	7	1	5	1		1
7	1	13	1	5	1	7	1		1	109	1	5,7, 11	1		1		1	7	1	5	1		1	7,53	1	13, 19, 79	1	5	1	7	1	11, 83	1	103	1
5,7, 139	1		1	59	1	7	1	5,11	1		1	7	1	13	1	5	1	7, 37	1	29	1		1	5,7	1	19	1	11	1	7, 131	1	5	1		1
7,29	1	13, 107	1	5,11, 103	1	7	1	97	1		1	5,7, 109, 137	1		1	131	1	7, 37	1	5, 79	1	67	1	7,11	1	13, 19	1	5	1	7	1		1		1
5,7, 11	1	131	1		1	7	1	5,83	1		1	7	1	13	1	5, 53	1	7	1	11	1		1	5,7	1	19	1	101	1	7	1	5	1	61	1
7	1	13	1	5	1	7	1		1		1	5,7	1	79, 109	1	11,29	1	7, 37	1	5	1		1	7	1	13, 19	1	5	1	7	1		1		1

Слід зазначити, що в результаті заповнення табличних даних комірки, які залишилися незаповненими, відповідають простим числам виду 2^n+3, а всі інші числа є складеними.

Таблиця 3.2 -Аналітика простих та взаємно простих чисел.

Прості числа	Вирази виду 2^n+1, які діляться на прості числа	Вирази виду 2^n+3, які діляться на прості числа	Вирази виду 2^n+5, які діляться на прості числа	Вирази виду 2^n+11, які діляться на прості числа	Вирази виду 2^n+13, які діляться на прості числа
3	$2^{2n+1}+1$	-	-	-	-
5	$2^{4n+2}+1$	$2^{4n+1}+3$	-	-	-
7	-	$2^{3n+2}+3$	-	-	-
11	$2^{10n+5}+1$	$2^{10n+3}+3$	$2^{10n+9}+5$	-	-
13	$2^{12n+6}+1$	$2^{12n+10}+3$	$2^{12n+3}+5$	$2^{12n+2}+11$	-
17	$2^{8n+4}+1$	-	-	-	$2^{8n+3}+13$
19	$2^{18n+9}+1$	$2^{18n+4}+3$	$2^{18n+7}+5$	$2^{18n+4}+11$	$2^{18n+15}+13$
23	-	-	$2^{11n+6}+5$	$2^{11(n+1)}+11$	-
29	$2^{28n+14}+1$	$2^{28n+19}+3$	$2^{28n+8}+5$	$2^{28n+12}+11$	$2^{28n+5}+13$
31	-	-	-	-	-
37	$2^{36n+18}+1$	$2^{36n+8}+3$	$2^{36n+5}+5$	$2^{36n+13}+11$	$2^{36n+30}+13$
41	$2^{20n+10}+1$	-	$2^{20n+17}+5$	-	-
43	$2^{14n+7}+1$	-	-	$2^{14n+6}+11$	-
47	-	-	$2^{23n+9}+5$	$2^{46n+18}+11$	$2^{23n+8}+13$
53	$2^{52n+26}+1$	$2^{52n+43}+3$	$2^{52n+21}+5$	$2^{52n+32}+11$	$2^{52n+51}+13$
59	$2^{58n+29}+1$	$2^{58n+21}+3$	$2^{58n+35}+5$	$2^{58n+55}+11$	$2^{58n+17}+13$
61	$2^{60n+30}+1$	$2^{60n+36}+3$	$2^{60n+52}+5$	$2^{60n+46}+11$	$2^{60n+11}+13$
67	$2^{66n+33}+1$	$2^{66n+6}+3$	$2^{66n+48}+5$	$2^{66n+27}+11$	$2^{66n+53}+13$
71	-	-	-	$2^{35n+12}+11$	-
73	-	-	-	-	-
79	-	$2^{39n+10}+3$	-	-	-
83	-	$2^{82n+31}+3$	$2^{82n+51}+5$	$2^{82n+49}+11$-	-
89	-	-	-	$2^{11n+9}+11$	-
97	$2^{45n+24}+1$	$2^{45n+40}+3$	-	-	-
101	$2^{100n+50}+1$	$2^{100n+19}+3$	$2^{100n+74}+5$	$2^{100n+65}+11$	$2^{100n+17}+13$

Слід зазначити, що на основі використання таблиці 3.1 було побудовано таблицю 3.2, яка дозволяє знаходити взаємно прості числа та вирішувати задачу факторизації чисел виду $2^n + 3$.

3.3 Програмна реалізація алгоритмів пошуку найбільшого спільного дільника

3.3.1 Організація діалогової системи реалізації основних компонентів алгоритму пошуку найбільшого спільного дільника.

Реалізація основних модулів, тобто пошуку простих та взаємно простих чисел, знаходження залишку по модулю, найбільшого спільного дільника відбувається в програмному середовищі C++ Buider 6.0.

Слід зазначити, що при генеруванні простих чисел було використано високопродуктивний алгоритм 3.1. Основна робота якого починається з завантаження користувачем головної форми за допомогою головного меню і надається можливість отримати доступ до основних функцій додатку. На цій формі знаходяться три вкладки:

- „Пошук простих чисел”
- „Діаграми”
- „Таблиця залишкових класів”

Кожна вкладка має специфічне функціональне навантаження. Основні можливості додатку розкриваються лише після того, як відбудеться пошук простих чисел. Для відбору простих чисел передбачено обмеження, пошук буде відбуватися до вказаного в параметрах числа.

На рисунку 3.5 показано процес пошуку простих чисел менших від 1000000. Список простих чисел буде відображений на екрані з вказаним часом початку пошуку та в кінці списку вказаним часом завершення пошуку.

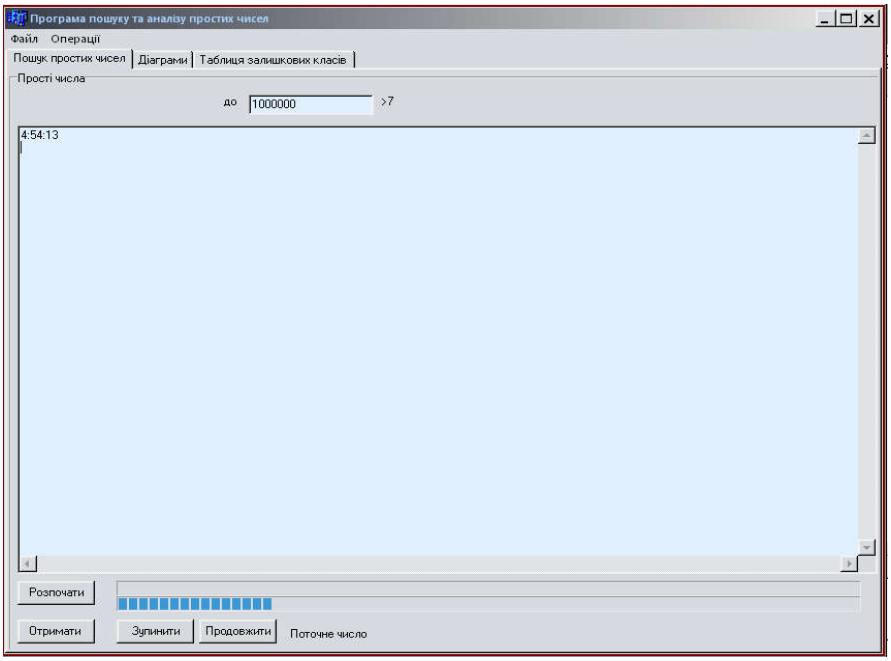


Рисунок 3.5 – Процес відбору простих чисел

Також в списку простих чисел будуть відмічені всі прості числа „Мерсена” та числа „Ферма”, які представляють особливий інтерес при побудові високопродуктивних спецпроцесорів. Після кожного десяткового представлення через пропуск відображено двійкове представлення чисел (рисунок 3.6).

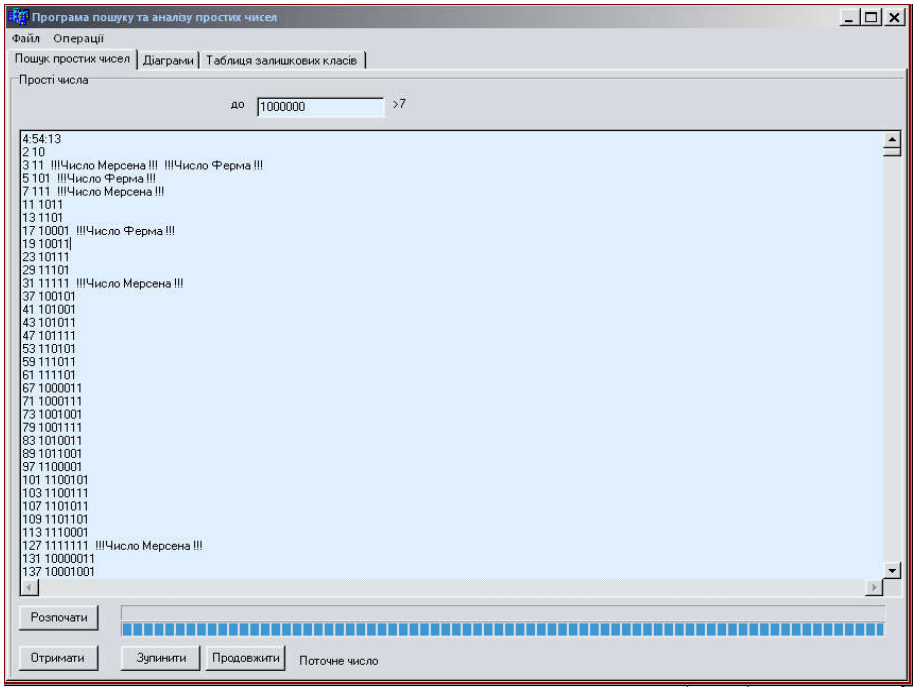


Рисунок 3.6 – Результати пошуку простих чисел

Процес пошуку простих чисел відбувається з допомогою створення нового потоку, що дозволяє процесорові розподіляти час між частинами додатку, що в свою чергу забезпечує стабільну роботу, як додатку так і ОС в цілому під час обчислень.

Процес відбору простих чисел можна зупинити на певному етапі, та отримати результати пошуку не очікуючи повної пробірки заданого інтервалу. Про стан пошуку інформує також “Status bar” який знаходиться в нижній частині, як показано на рисунку 3.6.

Вкладка „Діаграми” за функціональністю поділена на дві частини:

- „Налаштування параметрів”
- „Діаграми”

Перша вкладка дозволяє вводити критерії відбору простих чисел для побудови діаграми їх так званої швидкості зміни (рисунок 3.7).

Для кожного графіку є можливість встановлення певного кольору для графічної ідентифікації обмеженого інтервалу чисел.

Також серед простих чисел для побудови графічно відображеної залежності є можливість відбору за приналежністю до множини чисел „Мерсена” та „Ферма”. Серед специфічних функціональних можливостей є також вибір лише вказаних розрядів, що надає змогу проаналізувати їх зміну для певного виду чисел.

Також, як створювати діаграми інтервалу чисел, можна виконувати операцію їхнього видалення.

На закладці діаграми (рисунок 3.8) відображаються раніше створені графіки швидкості зміни простих чисел відносно чисел Мерсена. Слід відмітити, що часова залежність є лінійною і зі збільшенням розмірності чисел збільшується час пошуку.

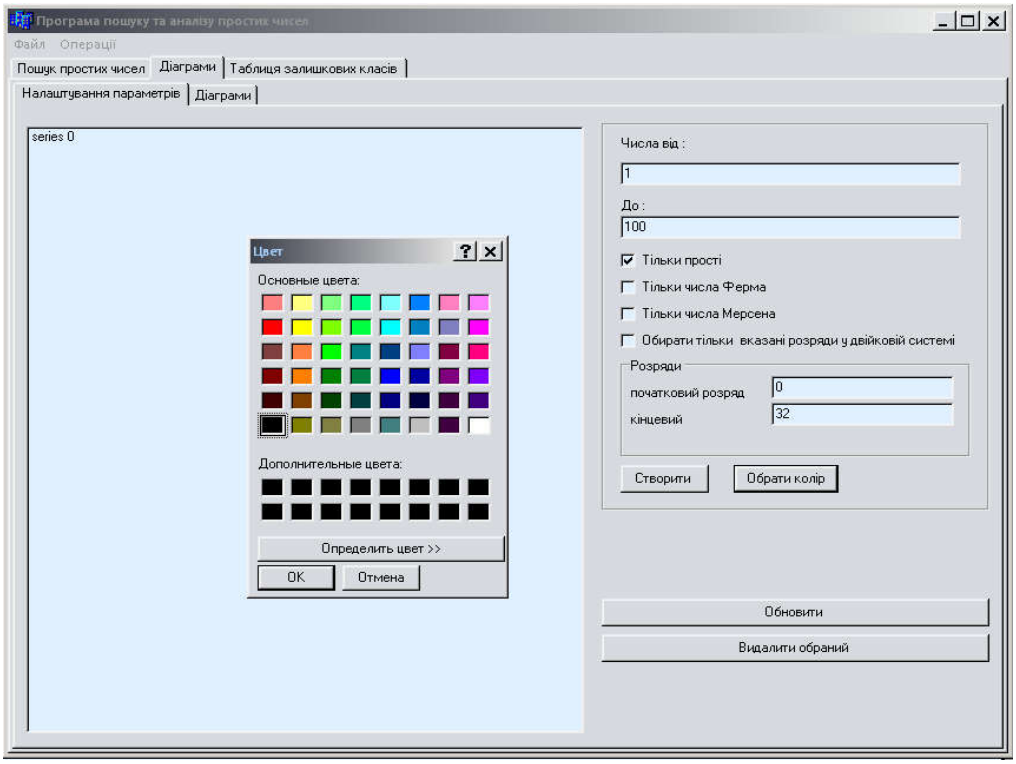


Рисунок 3.7 – Налаштування та створення діаграми швидкості зміни простих чисел та їх відбір згідно критеріїв

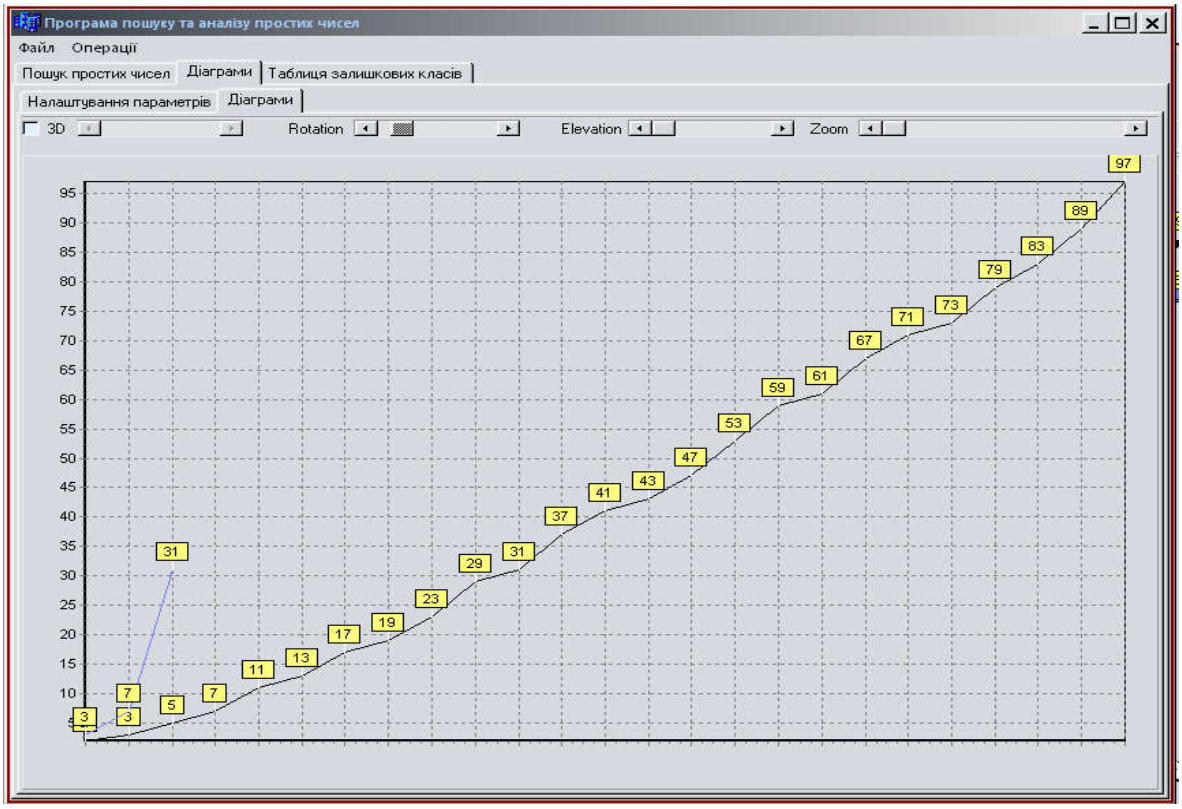
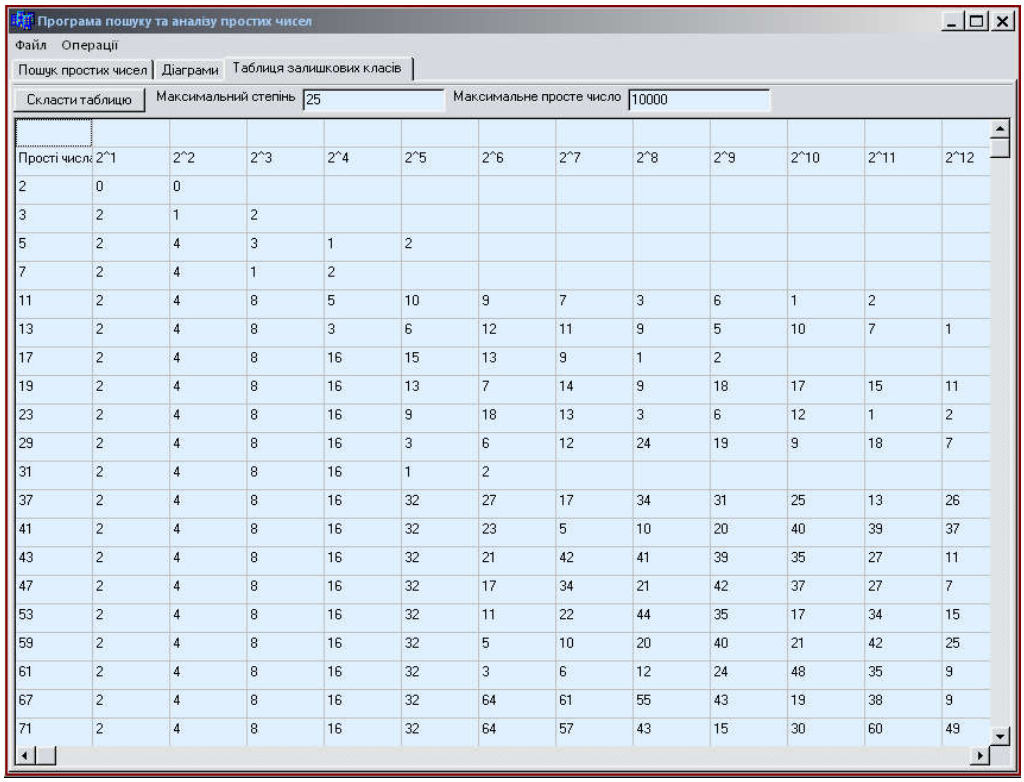


Рисунок 3.8 – Діаграма швидкості зміни простих чисел відносно чисел Мерсена

Для зручного перегляду та аналізу передбачено масштабування, 3-D перегляд, відображення під різними кутами, як по вертикалі так і по горизонталі.

На рисунку 3.9 зображена форма, що містить таблицю залишкових класів, її можна побудувати лише після пошуку простих чисел оскільки вона прямо від них залежить. При побудові таблиці використовуються два обмеження, по максимальному показнику степеня і по максимальному простому числу для, якого будуть обчислюватись залишки.



Прості числа	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^10	2^11	2^12
2	0	0										
3	2	1	2									
5	2	4	3	1	2							
7	2	4	1	2								
11	2	4	8	5	10	9	7	3	6	1	2	
13	2	4	8	3	6	12	11	9	5	10	7	1
17	2	4	8	16	15	13	9	1	2			
19	2	4	8	16	13	7	14	9	18	17	15	11
23	2	4	8	16	9	18	13	3	6	12	1	2
29	2	4	8	16	3	6	12	24	19	9	18	7
31	2	4	8	16	1	2						
37	2	4	8	16	32	27	17	34	31	25	13	26
41	2	4	8	16	32	23	5	10	20	40	39	37
43	2	4	8	16	32	21	42	41	39	35	27	11
47	2	4	8	16	32	17	34	21	42	37	27	7
53	2	4	8	16	32	11	22	44	35	17	34	15
59	2	4	8	16	32	5	10	20	40	21	42	25
61	2	4	8	16	32	3	6	12	24	48	35	9
67	2	4	8	16	32	64	61	55	43	19	38	9
71	2	4	8	16	32	64	57	43	15	30	60	49

Рисунок 3.9 – Згенерована таблиця залишкових класів

Функція побудови таблиці припиняє обчислювати залишки для певного простого числа при виході за рамки вказаних умов або ж при виявленні закономірної циклічності повтору.

З головного меню програми надається можливість виконання операцій на основі модулярної арифметики з допомогою розроблених алгоритмів.

На рисунку 3.10 показано приклад обчислення модуля числа з використання базису Радемахера-Крестенсона. Ця операція є досить важливою, оскільки використовується в більш складних функціях і від її алгоритмічної складності та часових характеристик залежить ефективність обчислень.

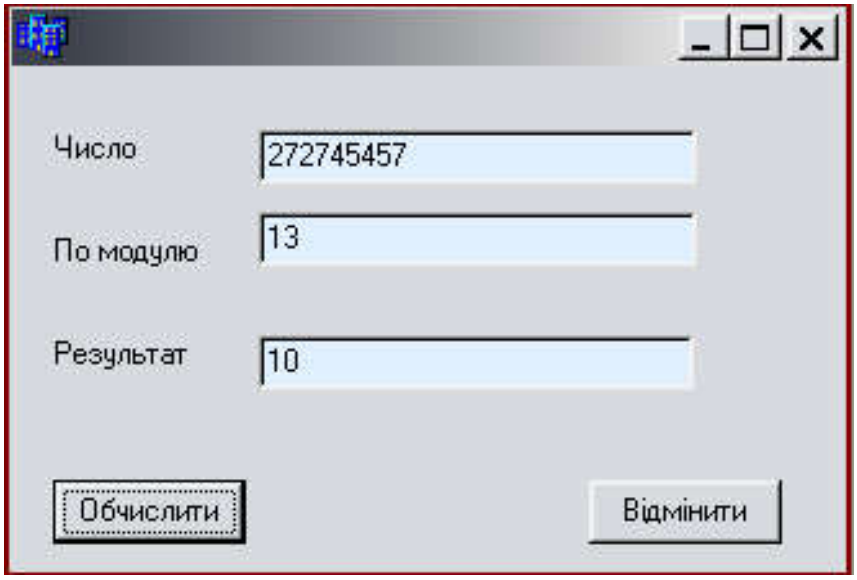


Рисунок 3.10 – Обчислення модуля числа

Рисунок 3.11 наводить приклад пошуку найбільшого спільного дільника з використанням високопродуктивного алгоритму 3.3 на базі використання системи числення залишкових класів базису Крестенсона.

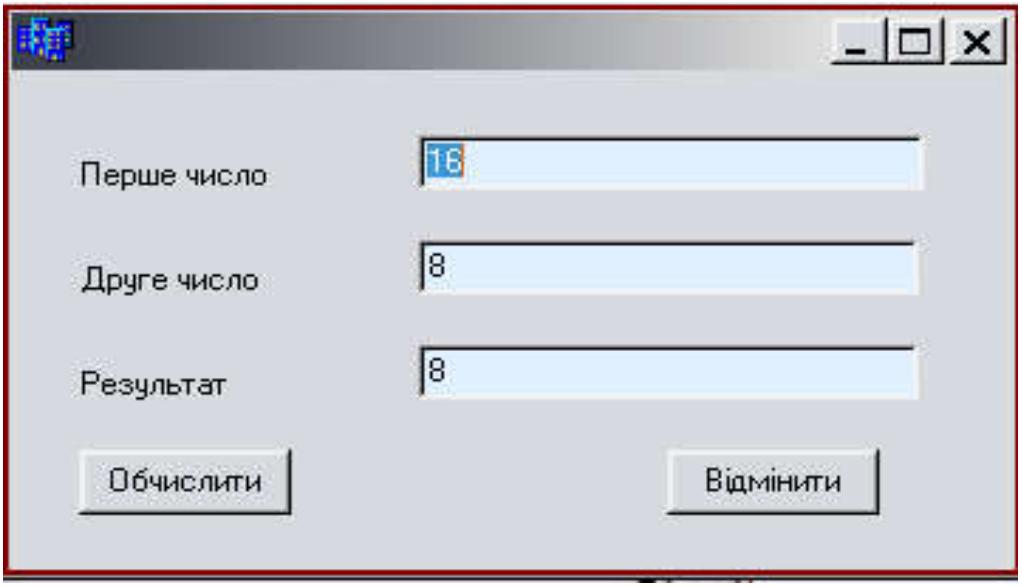


Рисунок 3.13 – Обчислення найбільшого спільного дільника.

Отже, реалізовано в програмному середовищі C++ Buider 6.0 основні компоненти які відповідають теоретично розрахованим параметрам і підтверджують правильність та результативність запропонованого наукового підходу по вдосконаленню методів та алгоритмів опрацювання інформаційних потоків в комп'ютерних мережах за умови застосування ЕК.

3.3 Дослідження часових складностей алгоритмів пошуку найбільшого спільного дільника

Крім того, зазначимо, що в двох останніх алгоритмах не обов'язково шукати залишки від обох чисел *a* та *b*. Досить знайти залишки від меншого числа і тільки при їх рівності 0 перевіряти друге число.

В таблиці 3.3 подано оцінки часової складності основних операцій алгоритму 3.3, що дозволяє зробити порівняльний аналіз з вищенаведеними алгоритмами пошуку найбільшого спільного дільника.

Таблиця 3.3 – Обчислювальна складність основних операцій алгоритму 3.3 та удосконаленого алгоритму 3.4.

№	Основні операції	Обчислювальна складність
1.	$p_j^{\,m_j}$	$O\left(\log_2 n \cdot \left(\log_2 n + \frac{n}{2}\right)\right)$
2.	$a_j^{(m)} = res\left(\sum_{i=1}^{n-1} a_{ij} (\mathrm{mod}\, p_j^{\,m})\right)$ $b_j^{(m)} = res\left(\sum_{i=1}^{n-1} b_{ij} (\mathrm{mod}\, p_j^{\,m})\right)$	$O\left(\log_2 n/2\right)$
3.	$Z = \prod_{j=1}^{k} p_j^{\,m_j}$	$O(k \cdot \log_2 n)$

де k - кількість модулів для яких виконується умова $a_j = b_j = 0$.

З врахуванням даних таблиці 3.7, загальна обчислювальна складність запропонованого алгоритму 3.3, та удосконаленого алгоритму 3.4 буде визначатися сумою складностей основних операцій, а саме: $O\left(\log_2 n \cdot \left(\log_2 n + \frac{n}{2} + k \cdot \log_2 n\right) + n \cdot \log_2 n/2\right)$ і $O3\left(\log_2 n\left(\log_2 n + k \cdot \log_2 n + \frac{n}{2}\right) + \frac{n}{2} \cdot \log_2 \frac{n}{2}\right)$ відповідно.

Слід зауважити, що застосування форм СЗК, які використовуються для реалізації високопродуктивних алгоритмів опрацювання і захисту інформаційних потоків, а саме: знаходження модулярного множення та піднесення до степеня, знаходження найбільшого спільного дільника, оберненого елемента по модулю і застосування для китайської теореми про лишки, дозволяє зменшувати алгоритмічну складність та проводити обчислення з використанням паралельної технології.

На рисунку 3.14 показані графіки які характеризують складності існуючого та запропонованих алгоритмів в залежності від розрядності компонентів Z .

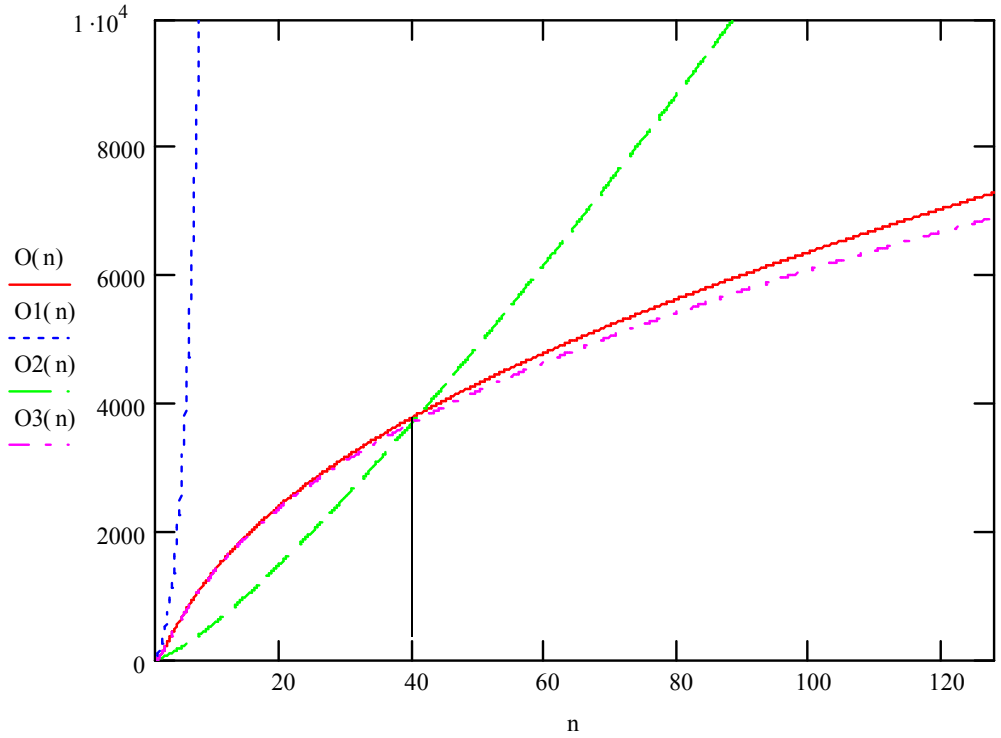


Рисунок 3.14 – Складності алгоритмів пошуку НСД(X, Y).

Результати досліджень показали, що для пошуку НСД двох чисел існуючий алгоритм Евкліда, який традиційно використовується для пошуку НСД для чисел великої розрядності, при сучасному рівні комп'ютерної техніки стає практично нездійсненний, оскільки потребує роботи процесорів на інтервалі близько 100 років. Чисельний експеримент оцінки складностей запропонованих алгоритмів пошуку НСД показує, що в діапазоні двійкових розрядів від 0 до 40 бітів, слід використовувати удосконалення реалізації алгоритму Евкліда 2.2, а при збільшенні розрядності чисел потрібно застосовувати алгоритм 2.3 та удосконалений алгоритм 2.4.

Висновки до розділу 3

1. Розроблено структурну схему організації пошуку найбільшого спільного дільника згідно розроблених алгоритмів та обґрунтовано середовище проектуваня для реалізації.

2. Проведена програмна реалізація запропонованих методів пошуку найбільшого спільного дільника з використанням додатків C++ Buider 6.0 та бібліотеки великих чисел Ленстри.

3. Проведено дослідження пошуку чисел виду $2^n + 3$, що дозволяє ефективно вирішувати задачі не тільки пошуку простих чисел, але й факторизації чисел даного виду та пошуку взаємно простих чисел.

4. Проведено дослідження обчислювальної складності алгоритмів пошуку найбільшого спільного дільника які показують, що в діапазоні двійкових розрядів від 0 до 40 бітів, слід використовувати удосконалення реалізації алгоритму Евкліда, а при збільшенні розрядності чисел алгоритми з використанням теоретико-числового базису Крестенсона.

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Задірака В. Комп'ютерна криптологія: Підручник. / В.Задірака, О. Олексюк – К.:2002. – 504 с.

2. Романец Ю.В. Защита информации в компьютерных системах и сетях / Ю.В. Романец, П.А. Тимофеев, В.П. Шаньгин; Под ред. В.Ф.Шаньгина.– М.: Радио и связь, 1999. – 328с.

3. Фергюсон Н. Практическая криптография: Пер. с англ. / Н.Фергюсон, Б. Шнайер – М.: – Вильямс, 2005. – 424 с.

4. Якименко І.З. Прискорення алгоритму Шуфа методом паралельних обчислень./ І.З. Якименко, А.А. Хомінчук // Матеріали дванадцятої наукової конференції Тернопільського державного технічного університету імені Івана Пулюя. – Тернопіль, ТДТУ.–14-15 травня 2008 р. – С:116.

5. Николайчук Я.М. Теоретичні основи базисних перетворень СЗК / Я.М.Николайчук, Ю.С. Федорович // Матеріали наукової конференції «Автоматика 2000». – Львів, 2000. – С. 120.

6. Бородін О.І. Теорія чисел. / О.І. Бородін – К.: Вища школа, 1970. – 275 с.

7. Варновский Н. П. Криптография и теория сложности. / Н.П.Варновский // Математическое просвещение. – 1998. – №2. – С. 71 – 86.

8. Вербіцький О.В. Вступ до криптології. / О.В. Вербінський – Львів:ВНТЛ, 1998.–.248 с.

9. Бухштаб А.А. Теория чисел. / А.А. Бухштаб – М.: Просвещение, 1966. – 384 с.

10. Волинський О.І. Методи порівняння та сумування в розмежованій системі числення./ О.І. Волинський // Поступ в науку. Збірник праць Бучацького інституту менеджменту і аудиту.– Бучач. – 2009. - №4. Т1. – С. – 91-94.

11. Якименко І.З. Розмежована система числення залишкових класів та спецпроцеси на її основі. / І.З. Якименко, О.І. Волинський // Поступ в науку. Збірник праць Бучацького інституту менеджменту і аудиту.– Бучач. – 2009. – №4. Т1. – С. – 94-98.