МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Західноукраїнський національний університет

Факультет комп’ютерних інформаційних технологій

Кафедра кібербезпеки

СОРОКА Дмитро Євгенійович

**ВДОСКОНАЛЕНИЙ АЛГОРИМ ГАРНЕРА ДЛЯ МІЖБАЗИСНИХ ПЕРЕХОДІВ В СИСТЕМАХ ЗАХИСТУ ІНФОРМАЦІЇ /IMPROVED GARDNER'S ALGORITHM FOR INTERBASE TRANSITIONS IN INFORMATION SECURITY SYSTEMS**

Спеціальність 125 – Кібербезпека

Освітньо-професійна програма – Кібербезпека

Кваліфікаційна робота

Виконав студент групи КБм-21

Д.Є. Сорока

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Науковий керівник:

К.т.н., доцент, С.В. Івасьєв

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

кваліфікаційну роботу допущено

до захисту

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_р.

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Яцків

Тернопіль 2021

ЗМІСТ

|  |  |
| --- | --- |
| Вступ…………………………………………………………………………….. | 3 |
| 1 Аналіз предметної області………………..…………………………………. | 3 |
| 1.1 Розвиток систем числення………………………………………….. | 5 |
| 1.2 Аналіз існуючих систем числення…………………………………. | 6 |
| 1.3 Перспективи побудови комп’ютерної системи в СЗК. Постановка завдання………………………………………………………………….. | 16 |
| 2 Відновлення десяткового числа за його залишком……................................ | 22 |
| 2.1 Узагальнення китайської теореми про залишки…………………... | 22 |
| 2.2 Міжбазисний перехід за алгоритмом Гауса……………………….. | 37 |
| 2.3 Обгрунтування вибору проектних рішень ………………………… | 38 |
| 3 Програмний засіб відновлення десяткового числа за його залишками ….. | 44 |
| 3.1 Розширений алгоритм Евкліда……………………………………… | 44 |
| 3.2 Реалізація алгоритму Гауса…………………………………………. | 53 |
| 3.2 Реалізація алгоритму відновлення числа за його залишком на основі додавання ……………………………………………………….. | 56 |
| 3.3 Вдосконалений алгоритми відновлення десяткового числа за його залишками………………………………………………………… | 58 |
| 3.4 Тестування та налагодження програмного засобу………………... | 63 |
| Висновки……………………………………………………………………….... | 66 |
| Список використаних джерел…………………………………………………. | 67 |
|  |  |

# ВСТУП

**Актуальність роботи.** Сучасні системи опрацювання інформації та криптографічних перетворень завдяки стрімкому розвитку ІТ технологій вимагають значних ресурсів для забезпечення виконання завдань покладених на них. Можливості двійкової арифметики вичерпують свої можливості, щодо забезпечення ефективного функціонування таких систем. Збільшення швидкодії в опрацюванні багаторозрядних чисел можливе завдяки використанню системи числення залишкових класів, котру не використовують через ряд обмежень. Зокрема відсутності порівняння та високій обчислювальній складності між базисного переходу в десяткову систему числення. Зменшення обчислювальної складності при відновленні десяткового числа з системи залишкових класів дозволить оптимізувати велику кількість обчислювальних алгоритмів.

**Мета роботи** полягає в розробці алгоритму відновлення числа за його залишками, який забезпечує високу швидкодію та детермінований час виконання.

Для досягнення даної мети ставились наступні завдання:

* дослідити існуючі системи числення;
* дослідити перспективи побудови комп’ютерних систем в СЗК та виконати постановку завдання;
* проаналізувати математичні основи китайської теореми про залишки;
* дослідити математичні основи між базисного переходу за алгоритмом Гауса;
* проаналізувати роботу алгоритмів Евкліда, Гауса, відновлення числа за його залишком на основі додавання;
* розробити вдосконалений алгоритм відновлення числа за його залишками.
* провести тестування розробленого програмного засобу на основі запропонованого алгоритму .

**Об’єкт дослідження** - процеси програмного та апаратного опрацювання багато-розрядних чисел.

**Предмет досліджень** – методи між базисних перетворень.

**Наукова новизна одержаних результатів** визначається наступним чином:

- розроблено алгоритм, що дозволяє виконувати перехід з системи залишкових класів в десяткову систему числення в детермінований час та відзначається вищою ефективністю за існуючий.

* побудовано аналітичні вирази запропонованої схеми відновлення десяткового числа за його залишками. Чисельний експеримент показав, що розроблений алгоритм дозволяє збільшити швидкодію між базисного переходу.

**Практична цінність одержаних результатів** полягає в тому, що:

* обґрунтовано підхід та розроблено алгоритм відновлення числа за його залишком;
* реалізовано програмне забезпечення для відновлення числа за його залишком та виконання елементарних арифметичних операцій на мові python.

Публікації та апробація до магістерської роботи. За результатами наукових досліджень, проведених у магістерській роботі, підготовлено тези доповіді на проблемно-науковій міжгалузевій конференції «Кібербезпека та комп’ютерно-інтегровані технології» (КБКІТ – 2021, додаток А).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Розвиток систем числення

У 70 – 80 роки двадцятого сторіччя зусиллями радянських учених С.А. Лебедєва, В.М. Глушкова, М.А. Карцева, І.Я. Акуського, Д.І. Юдицького, Г.Я. Гуськова, В.С. Семеніхіна, І.В. Прангішвілі, Н.Я. Матюхіна та багатьох інших були створені зразки обчислювальної техніки, що перевершують за своїми параметрами, новизні ідей та архітектурі всі світові досягнення. В якості приміром назву лише деякі з обчислювальних машин тих років: серія ЕОМ класу БЕСМ та «Ельбрус», ЕОМ «Стріла», серії ЕОМ М-20, М-220, 5Е76, «Світ», ПС та ін і лише після того, як СРСР став відтворювати ЕОМ класу IBM-360 (ЄС ЕОМ) та копіювати, а не розробляти мікроелектронну елементну основу, доля радянської обчислювальної техніки було вирішено наперед.

Відповідальність за це лежить не на вчених та розробниках, а на керівництво країни.

У ті роки в СРСР колективи вчених досліджували і розробляли різні арифметики, що дозволяли створювати ЕОМ з вищою швидкістю обробки даних у порівнянні з поширеною двійковою системою числення. Так, під керівництвом І.Я. Акушського та Д.І. Юдицького було створено ЕОМ К-340 на основі системи числення в залишкових класах, яка тривалий час випускалася нашою промисловістю та відрізнялася високою продуктивністю та надійністю.

Колективом фахівців на чолі із В.М. Глушковим були створено та запущено у виробництво ЕОМ серії «Мир» з новою архітектурою та системою програмування, що дозволяли проводити обчислення зі змінною (регульованою) розрядністю. Тоді ж під керівництвом І.В. Прангішвілі розробляються та виробляються ЕОМ серії ПС на основі асоціативних процесорів з паралельною архітектурою, а під керівництвом М.А. Карцева - високопродуктивні, високонадійні обчислювальні комплекси великої розрядності (до 512 двійкових розрядів) для спеціальних застосувань.

На початку 80-х років з'явилися перші публікації радянських вчених про Фібоначчієву систему числення [1] та ієрархічну системі числення, які дозволяли створювати більше високопродуктивні та надійні обчислювальні засоби на основі нових елементів мікроелектроніки, у тому числі і багатозначних. Одночасно розробляються та алгоритми виконання арифметичних операцій на ЕОМ, засновані на нових арифметиках.

Коротко опишемо найцікавіші системи числення для обчислювальних пристроїв, швидкодія та надійність яких перевершують аналоги, засновані на двійковій арифметиці.

1.2 Аналіз існуючих систем числення

Знако-розрядна система числення.

Число в знако-розрядній системі числення [2], як і в будь-якій позиційної системи, можна записати у вигляді:

,

де *xi*={-*R*, -*R*+1,...,-1, 0, 1,..., *R*}, *S*=2*R* або *S*=2*R*+1.

Додавання двох чисел у знако-розрядній системі числення виконується у два такти. У першому такті формуються поцифрові проміжні суми  та цифри порозрядних переносів , які можуть набувати значення -1, 0 і +1, тобто. *хi* + *уi* = *w*i + *tiS*.

У другому такті формується остаточна сума шляхом складання цифр проміжних розрядних сум та відповідним їм цифр порозрядних переносів, тобто. *Zi* = *wi*+ *ti*+1. Множення чисел у знако-розрядній системі числення виконується послідовним додаванням (відніманням) та зрушенням вправо результатів множення множника на *S* цифри множника, починаючи з молодшого *S* розряду. Ділення чисел підпорядковується загальним правилам поділу в *S*-іншій системі числення.

Основною перевагою знако-розрядної системи числення є те, що сигнал перенесення при виконанні операції додавання поширюється не далі сусіднього розряду, а час виконання операції залежить від розрядності операндів. Тоість будь-яка операція додавання виконується за два такти (під тактом тут розуміється час обчислення розрядної суми.

Система числення Фібоначчі.

Серед позиційних вагомозначних систем числення є системи, в яких ваги розрядів виражаються не відомим співвідношенням *Di*=*Si*, а іншими, наприклад числами ряду.

Фібоначчі, тобто. *Di* = *Di*-2 + *Di*-1. у цьому випадку система числення називається Фібоначчієвою. Інший приклад позиційної вагомозначної системи числення з нетрадиційним законом формування ваг розрядів – так звана поліадична система зчислення. Ваги розрядів у ній визначаються виразом: *Di=pi Di-1*, де *рi* - взаємно-прості числа.

Зупинимося на Фібоначчієвій системі числення. У роботі [1] показано, що будь-яке натуральне число N може бути представлено в двійковій *р*-системі числення при р≥0, вагами розрядів у якої є числа Фібоначчі. При цьому після кожної одиниці зліва направо слідує не менше *р* нулів. Так, наприклад, при р=1 число 75 у двійковій 1-системі числення можна записати як:

75=1001010100=1×55+0×34+0×21+1×13+0×8+1×5+0×3+1×2+0×1+0×1.

Зазначимо дві особливості складання значущих розрядів у двійковій 1-системі числення. По-перше, при підсумовуванні одиниць виникає перенесення не однієї одиниці (як у класичній двійковій системі числення), а кількох одночасно. По-друге, одиниці можна складати двома способами. У першому способі при додаванні i-х розрядів чисел в i-му розряді проміжної суми записується 1 і виникають перенесення двох одиниць одночасно – у (i-1)-й та у (i-р-1)-й розряди. При другому способі складання одиниць у відповідному (*i*-му) розряді проміжної суми записується 0 (як і в класичній двійковій арифметиці) і виникає перенесення *р*+1 одиниць (одна одиниця – у старший (*i*+1)-й та *р* одиниць – у молодші (*i-р*-1), (i-р2),…,(*i*-2*р*) розряди).

Найбільш раціональний спосіб множення двійкових Фібоначчієвих чисел в 1-системі числення аналогічний множенню в класичній двійковій, хоча і має свою специфікою [1-4].

Основний спосіб розподілу чисел (Z=X/Y) у Фібоначчієвій системі числення: накопичуються кратні числам Фібоначчі значення дільника, тобто. *N*=*Y*×*Kj* (*Kj*=1,2,3,5,...). Кратні дільника порівнюються з ділимим, починаючи з максимального кратного. У Залежно від результату порівняння формується приватне, тобто.



Незважаючи на очевидну непрактичність Фібоначчієвої системи обчислення для конструювання цифрових обчислювальних пристроїв, роботи творця системи та її учнів представляють собою значний науковий результат, що показує недослідженість різноманітності систем числення та необхідність пошуку систем із новими якостями.

Система залишкових класів.

Система залишкових класів (СЗК) – це непозиційна система числення, числа в якій видаються залишками від розподілу на вибрану систему основ *Р*1, *Р*2, ..., *Рn* і є взаємнопростими числами. Операції додавання, віднімання та множення над числами в СЗК виробляються незалежно по кожній підставі без перенесення між розрядами (основами). Діапазон репрезентованих чисел *Р*=*Р*1 ×*Р*2×…×*Рn* [3].

Якщо заданий ряд позитивних взаємно простих чисел *Р*1, *Р*2, ..., *Рn*, то ціле позитивне число *А* на обраній основі *Р*1, *Р*2, ..., *Рn*, можна записати у вигляді *А* = (*a*1, *a*2, ..., *an*),

де [ ] - цілісний поділ. Це і є запис числа в СЗК.



Якщо вихідні числа *А, В*, їхня сума *А+В* та їх добуток *А×В* перебувають у діапазоні (0,*Р*)[5-6], то результати операцій додавання *А*+*В* та множення *А*×*В* можуть бути однозначно представлені відповідно залишками *gi* і *pi* за тими ж основами *Pi*, тобто:

, ,

, ,

,

.

де [ ] - цілісний поділ. Це і є запис числа в СЗК.

Такі операції, як розподіл, порівняння та ін, що вимагають інформації про величину всього числа, в СЗК виконуються по більш складним алгоритмам. І в цьому полягає суттєвий недолік даної системи числення, що стримує її широке застосування як комп'ютерної арифметики. Однак сьогодні навіть у найсучасніших комп'ютерах при роботі з великими та супервеликими числами використовують СЗК, бо тільки ця арифметика дозволяє отримувати
результати обчислень у реальний час. У таких випадках
як основу в СЗК застосовують величини, близькі 2*m* (*m*-двійкова розрядність комп'ютера), наприклад 2*m*-1-1, 2*m*-1, *2m*-1+1 і т.д. Комп'ютер обчислює
результат по одному із модулів за один прохід. Інші сфери застосування СЗК – завадостійке кодування, криптографія і т.п. Починаючи з 1952 року фахівці багатьох країн світу, включаючи і СРСР, займалися проблемою
підвищення швидкості виконання «незручних» операцій у СЗК. Особливу роль у вирішенні даної проблеми зіграв І.Я. Акушська. Чималий
внесок у цю область науки внесли також Д.І. Юдицький, В.М. Амербаєв, А.А. Коляда.

Ієрархічні системи числення.

Наприкінці 80-х – на початку 90-х років народилася ідея з'єднання позиційних та непозиційних систем числення, тобто.

Конструювання ієрархічних систем, які мають поєднувати у собі позитивні сторони включених до них систем числення і бути вільними від своїх недоліків [4, 5]. Принцип побудови ієрархічних систем загалом простий. Вибирається деяка зовнішня система числення *А*=<*a*,*Ω*>, де *a* -алфавіт системи, а *Ω* = *Ω* 0, *Ω* *r* - її сигнатура. Сигнатура складається із двох частин: операційної (*Ω* 0), що містить символи операцій системи та реляційної (*Ω* *r*), що містить символи відносин.

Цифри, тобто. елементи алфавіту А цієї системи записуються в вигляді слів (кодів) іншої (внутрішньої) системи числення *В*=<*b*, *Ω* >. Таку систему позначають *A*[*B*].

Розглянемо приклад. Нехай А – десяткова позиційна система, а *В* – двійкова система. Тоді *a*={0,1,2,..,9}, *b*={0,1}. Двійковий кодування цифр системи А (десяткової) провадиться, наприклад, записами:

0→0000, 1→0001,…9→1001

Тоді число 23 (десяткове) запишеться в ієрархічній системі числення у вигляді двох записів (0010, 0011). Система *A*[*B*] у нашому прикладі – добре відома двійково-кодована десяткова система, що застосовується для представлення десяткових чисел у сучасних ЕОМ.

Вочевидь, що ступінь вкладеності ієрархічної системи може бути і більше двох. Інакше кажучи, існують ієрархічні системи числення А0[A1[A*2*...[A*n*]...] з основами *S*0, *S*1,…, *Sn* , причому *S*,> *S*1> ...> *Sn*. Система числення (двійкова, вісімкова, шістнадцяткова), для яких на рівні уявлення є надмірними. Якщо ця умова не виконується, система надмірна (наприклад, двійковокодована десяткова) [7].

Позиційно-залишкова система числення.

При конструюванні ієрархічних систем числення великий інтерес представляє поєднання систем різних типів.

Розглянемо систему виду *A*[*B*], для якої *А* – позиційна система числення з основою *S*, а *В* – система числення в залишкових класах з базовими модулями *Р*1, *Р*2, ..., *Рr*, такими, що *Р*=*Р*1 ×*Р*2×…× *Рr*. Таку систему називають позиційно-залишковою системою числення [6].

Нерівність *Р*≥*S* – необхідна та достатня умова однозначного подання цифр 0, 1, … , *S*-1 позиційної системи наборами відрахувань за модулями *Р*1, *Р*2, ..., *Рr*. Проте враховуючи необхідність коректної реалізації арифметичних операцій у системі *A*[*B*] (наприклад, формування перенесення та т.п.), можна поставити більш жорстку умову *Р* = *Р*1 × *Р*2 × ... × *Рr* ≥2*S*. Дуже важливим є вибір величини підстави *S* позиційної системи числення у достатніх класах. Віддаючи данину двійковій системі числення, можна вибирати S ≥ 2*m*. У цьому випадку модулі СЗК та їх добутки повинні задовольняти умові Р ≥ 2m+1. Для людини ж найбільш зручні підстави, кратні 10 (100, 1000 та і т.д.).

Двійково-кодована десяткова система – певною мірою компроміс між людиною та комп'ютером. Але її відносна надмірність – 26,5%. Щоб подолати цей недолік, ряд дослідників пропонують для арифметики з плаваючою комою замість основи 10 використовувати 100 [8].

Тоді для зберігання двох десяткових цифр достатньо мати сім двійкових розрядів замість восьми (надмірність становитиме – 22,7%). Перехід до основи 1000 дозволяє розміщувати три десяткові цифри у 10 двійкових розрядах замість 12 (надмірність уявлення – 2,35%). Розрахунок за економічне подання чисел при переході до підстав у вигляді 10*n*- більш складні алгоритми кодування та декодування таких чисел.

Однак на рівні машинного уявлення арифметика все одно залишається двійковою.

Арифметичні операції у позиційно-залишковій системі числення виконуються окремо над цифрами зовнішньої та внутрішньої системи. Така ступінчаста реалізація операцій дозволяє практично без змін переносити алгоритми зовнішньої системи числення на операції у системі *A*[*B*]. При цьому "цифрові" операції системи числення *А* замінюються процедурами системи числення *В* [9].

Знако-розрядна позиційно-залишкова система числення.

Ще один приклад ієрархічної системи числення – знакорозрядна система з основою S, цифри якої представляються в системі залишкових класів з базовими модулями *Р* = *Р*1, *Р*2, ..., *Рr*. Перевага цієї системи числення – висока швидкість виконання арифметичних операцій над розрядними цифрами та мінімальна довжина шляху розповсюдження перенесення між *S*-ими розрядами (не далі сусіднього розряду).

Висока швидкодія досягається за рахунок того, що при підсумовуванні у кожному *S*-тому розряді (*S*>2) одночасно формуються три величини:

*xi+yi, xi+yi*-1, *xi+yi*+1.

Потім одна з них вибирається як результат залежно від значення сигналу перенесення *ti* приймає значення -1, 0, +1 [7,8,9,10].

Таким чином, з'являється можливість паралельної обробки на кількох комп'ютерах великих чисел із основами *S*=2*m*.

Обробляти великі числа в реальному часі здатні навіть двійкові персональні комп'ютери, що працюють за алгоритмів знако-розрядної позиційно-залишкової системи обчислення.

Нові західні технології, що з'являються на ринку, в сукупності з вітчизняними розробками в області недвійкових комп'ютерних арифметик та синтезу нових способів та алгоритмів прискорення обчислень відкривають перед розробниками обчислювальних систем нові можливості.

Система залишкових класів (СЗК) (від , інша назва Модулярна арифметика) - Непозиційна система числення. Представлення числа в системі залишкових класів засноване на понятті заликів та китайської теореми про залишки. СЗК визначається набором попарно взаємно простих модулів , тобто таких, що , званих базисом , з добутком , отже кожному цілому числу *x* з відрізка  ставиться у відповідність набір відрахувань , де:





….



У цьому китайська теорема про залишки дозволяє досягнути однозначність (єдиний розв’язок ) представлення цілих невідємних чисел з відрізка .

Переваги системи залишкових класів.

У СЗК арифметичні операції (додавання, віднімання, множення, ділення) виконуються покомпонентно, якщо про результат відомо, що він є цілим і також належить .

Формула для складання: , де:





….



Аналогічно виконуються віднімання, множення та ділення. Необхідно зауважити, що на ділення накладаються додаткові обмеження. Ділення має бути цілочисельнним, тобто дільник повинен націло ділити ділене. Дільник має бути взаємопростим з усіма модулями базису.

Виділимо недоліки системи залишкових класів. Серед них є можливість подання лише обмеженої кількості чисел.

Відсутність ефективних алгоритмів порівняння багаторозрядних чисел, які представлені в СЗК. Порівняння зазвичай відбувається через переведення аргументів із СЗК у змішану систему числення на основі: .

Також до недоліків можна віднести низьку швидкодію роботи та вимагають роботи з великими числами, висока обчислювальна складність реалізації алгоритму переведення з позиційної системи числення до СЗК і назад.

Складні алгоритми ділення (для випадку, коли результат є цілим). Також є проблеми виявлення переповнення розрядної сітки.

Застосування системи залишкових класів має ряд алгоритмів. Часто СЗК використовують в криптографії. СЗК широко використовується в мікроелектроніці в спеціалізованих пристроях ЦОС, де потрібно виконувати: контроль за помилками, за рахунок введення додаткових надлишкових модулів.

Використання СЗК надає ряд переваг, серед яких: висока швидкість роботи, яку забезпечує паралельна реалізація базових арифметичних операцій [10].

Інформаційна безпека такоє ключовим фактором використання СЗК.

У модулярній арифметиці створені спеціальні набори простих модулів, які дозволяють частково усунути недоліки та для яких існують ефективні алгоритми порівняння чисел та алгоритми для прямого та зворотнього переведення модулярних чисел у позиційну систему числення. Одним з найбільш вживаш набором систем модулів є набір із трьох взаємно простих чисел, що мають вигляд .

Розглянемо приклад такого набору модулів. Розглянемо СЗК з базисом (2;3;5;7). У цьому базисі можна однозначно представити числа з проміжку від 0 до 210, враховуючи що *M*=2×3×5×=210. Таблиця відповідності чисел із позиційної системи числення та системи залишкових класів:

0=(0;0;0;0);1=(1;1;1;1);2=(0;2;2;2);3=(1;0;3;3) ;4=(0;1;4;4);

5=(1;2;0;5);6=(0;0;1;6); 7=(1;1;2;0); 8=(0;2;3;1); 9=(1;0;4;2);

10=(0;1;0;3); 11=(1;2;1;4); 12=(0;0;2;5);13=(1;1;3;6); 14=(0;2;4;0);

15=(1;0;0;1); 16=(0;1;1;2) ;17=(1;2;2;3) ;18=(0;0;3;4); 19=(1;1;4;5);

20=(0;2;0;6); 21=(1;0;1;0); 22=(0;1;2;1); 23=(1;2;3;2); 24=(0;0;4;3);

25=(1;1;0;4); 26=(0;2;1;5); 27=(1;0;2;6); 28=(0;1;3;7); 29=(1;2;4;8);

…

Приклад додавання чисел використовуючи СЗК. Додамо два числа 8 і 13 у базисі (2; 3; 5; 7). Їх представлення в заданому базисі 8=(0;2;3;1) та 13=(1;1;3;6). Скористаємося формулою для додавання: (q1,q2,q3,q4) = (0,2,3,1) + (1,1,3,6).









 (q1, q2, q3) = (1, 0, 1,0) – перевіряємо за попередніми даними, що результат дорівнює 21.

Наведемо приклад множення в СЗК. Помножимо два числа 5 і 6 у базисі (2; 3; 5;7). Їх представлення в заданому базисі 5=(1;2;0;2) та 6=(0;0;1;6). Скористаємося формулою для множення: (q1,q2,q3)=(2,3,5,7)∙(0,0,1,6).









(q1, q2, q3,q4) = (0, 0,0,2) - за таблицею переконуємося, що результат дорівнює 30.

Якби множили чи додавали числа, котрі дали в результаті множення число більше або рівне M=210, то отриманий результат , де REAL - результат операції у позиційній системі числення [11].

Приклад ділення, за умови, що можливий поділ націло. Ділення може бути виконано аналогічно множенню, але тільки якщо дільник ділить ділене націло, без залишку. Для модулів (29;31;32) розділимо число 1869 на 3.

1869=(13,9,32)

3=(3,3,3)

Скористаємося формулою:



Тут треба сказати, що , що не та сама операція, що й ділення x на y. Виконаємо цю операцію.







q1≡(13∗10)(mod 29)= 14;

q2≡(9∗21)(mod 31)=15;

q3≡(32∗25)(mod 32)=0;

Це і є правильний результат - число 623. Проте такий результат можна отримати, тільки якщо відомо, що ділення проводиться без остачі.

1.3 Перспективи побудови комп’ютерної системи в СЗК. Постановка завдання

З початку розвитку засобів обробки цифрової інформації та управління одними з головних їх характеристик стали швидкодія (продуктивність) і надійність (відмовостійкість). Хоча різні автори по-різному визначають продуктивність комп’ютерних систем, проте, користувачів завжди цікавить час виконання лише своєї, конкретної завдання, тобто. користувальницька продуктивність, яка і надалі мається на увазі під поняттям продуктивність засобів обробки цифрової інформації. Існують різні способи підвищення продуктивності комп’ютерних систем. Одні з них ґрунтуються на застосуванні багатомашинних комплексів та багатопроцесорних систем. Такі методи, підвищуючи системну продуктивність, істотно не впливають на користувальницьку продуктивність комп’ютерних систем [12-15].

Специфічність завдань розв'язуваних у процесі обробки інформаційних потоків у системах управління реального часу істотно впливає на особливості побудови та експлуатації комп’ютерних систем, які можна згрупувати за такими ознаками.

Постійність і циклічність розв'язуваних завдань. Для універсальних обчислювальних комплексів тип розв'язуваних завдань не має великого значення. Більшість їх вирішується, зазвичай, одноразово. На противагу цьому для комп’ютерних систем характерним є саме заздалегідь певний обмежений клас завдань та алгоритмів, які залишаються майже постійними для всього періоду експлуатації. Таким чином, однією з ознак комп’ютерних систем є сталість та циклічність класу завдань, які вирішуються в процесі всього часу експлуатації.

Функціонування комп’ютерних систем у часі можна оцінювати наступним чином. У комп’ютерних систем, що функціонують, наприклад, в контурі управління АСУ ТП АЕС, БЦВМ космічних апаратів і балістичних ракет, БПЛА пр. цикли розрахунку керуючих впливів не можуть вибиратися довільно. Запізнення в обробці даних призведе до порушення правильного функціонування систем управління, що може призвести до катастрофічних наслідків. Ця обставина зумовлює жорсткі вимоги щодо швидкодії комп’ютерних систем. Як правило, при проектуванні комп’ютерних систем особлива увага звертається на забезпечення високошвидкісного процесу обробки інформації, що передбачає високий ступінь розпаралелювання виконання операцій.

Умови експлуатації комп’ютерних систем суттєво відрізняються від умов експлуатації універсальних обчислювальних комплексів. Умови експлуатації обчислювальних комплексів загального призначення можна вважати практично постійними і стабільними, оскільки дестабілізуючі фактори (температура, тиск, вологість, підвищена радіація тощо) змінюються в порівняно в невеликих межах. Для умов експлуатації комп’ютерних систем характерний ширший спектр впливів зазначених вище несприятливих факторів та значний діапазон їх зміни. Це накладає обмеження на їх масогабаритні, енергетичні та інші важливі характеристики (наприклад, для КС літальних апаратів та ін.). Необхідність забезпечення жорстких вимог до спеціалізованих систем управління змушують проектувальників особливо суворо підходити до вибору параметрів комп’ютерних систем. Функціонування технічних систем у реальному часі вимагає від комп’ютерних систем вирішення завдань одночасно з високою користувальницькою продуктивністю, необхідної точністю і надійністю.

 Підвищення вимог до надійності комп’ютерних систем. На відміну від відмови універсальних обчислювальних машин, коли можна використовувати тимчасове резервування, відмову або збій (порушення працездатності) комп’ютерних систем може призвести (і, як правило, призводить) до катастрофічних наслідків (наприклад, АСУ ТП АЕС або АСУ аеропорту, що управляє повітряним рухом). Ця обставина обумовлює необхідність вжиття жорстких заходів щодо забезпечення надійності та відмовостійкості комп’ютерних систем [16-18].

Забезпечення стійкості до відмови комп’ютерних систем. Поняття отказоустойчивости тісно пов'язані з поняттям надійності комп’ютерних систем. Під відмовостійкістю, відповідно до ДСТУ 2506-94, будемо розуміти властивість, закладену при проектуванні комп’ютерних систем, яка дозволяє зберігати повну або часткову її працездатність за наявності елементів комп’ютерних систем, що перебувають у стані відмови. Відмовостійкість є найважливішою характеристикою комп’ютерних систем. Для комп’ютерних систем, що функціонує в двійковій позиційній системі числення, методи підвищення надійності та відмовостійкості ґрунтуються на використанні різних видів резервування: структурного, інформаційного, тимчасового, навантажувального та функціонального. Основна ідея організації стійкості до відмови комп’ютерних систем полягає в наступному. За рахунок наявної (або штучно введеної) надмірності спочатку визначається блок комп’ютерних систем, що відмовив, а далі відмова (збій) усувається шляхом виключення цього блоку з процесу функціонування. Необхідно ще раз акцентувати увагу, що для комп’ютерних систем реального часу процес виявлення відмов (збоїв) і усунення їх повинен проводитися без переривання обчислювального процесу, тобто. у процесі функціонування.

Таким чином, для комп’ютерних систем характерні і дуже важливі вимоги щодо забезпечення високої користувальницької продуктивності, надійності, відмови стійкості і живучості. Результати досліджень підвищення ефективності функціонування комп’ютерних систем реального часу, отримані як в Україні, так і за кордоном, показали, що існуючі методи підвищення продуктивності, надійності, відмовостійкості та живучості функціонування, засновані на використанні двійкових кодів, не завжди повною мірою задовольняють зростаючим вимогам до систем обробки інформації спеціального призначення. Дана обставина і обумовлює необхідність формулювання концепцій, пошуку нетрадиційних шляхів і методів підвищення продуктивності, надійності, відмовостійкості та живучості комп’ютерних систем об'єктами критичного застосування, що функціонують в реальному часі.

Більш перспективними методами підвищення продуктивності комп’ютерних систем є методи, що ґрунтуються на використанні деяких властивостей завдань (алгоритмів) певного класу, (природний паралелізм суміжних операцій, природний паралелізм безлічі об'єктів тощо), а також методи, що дозволяють штучно розпаралелити певні обчислювальні алгоритми . Однак сфера застосування обмежується класом (типом) розв'язуваних завдань. Крім того, штучне розчленування алгоритмів, визначення та виділення незалежних гілок не завжди можна здійснити, до того ж цей процес вимагає великих попередніх трудозатрат.

Перелічені методи підвищення продуктивності комп’ютерних систем, що функціонують у позиційних системах числення, мають загальний недолік - неможливість розпаралелити розв'язувані алгоритми на рівні елементарних операцій (мікрооперацій). Це зумовлено в першу чергу наявністю в позиційних системах числення міжрозрядних зв'язків в оброблюваних операндах. Розвиток сучасної мікроелектронної бази, заснованої на застосуванні БІС, НВІС, ПЛМ та ПЛІС, дало поштовх до дослідження можливості застосування в двійкових кодах табличних методів обробки інформації. Застосування цих методів може забезпечити надвисоку продуктивність (в результаті розпаралелювання елементарної операції) і надійність комп’ютерних систем, а також високий рівень регулярності та однорідності структури пристроїв, для їх реалізації.

Істотним недоліком табличних методів переробки інформації, що застосовуються в двійкових кодах, є необхідність використання значної кількості обладнання, що суттєво ускладнює їх практичну реалізацію.

Складність, масштаби та обсяги розв'язуваних комп’ютерних систем завдань управління складними технічними системами критичного застосування вимагає розширення функцій та можливостей засобів обчислювальної техніки, що тягне за собою збільшення кількості обладнання обчислювальних засобів та систем, ускладнює структуру та математичне забезпечення комп’ютерних систем. Це у свою чергу викликає необхідність вживання додаткових заходів щодо забезпечення високої надійності та відмовостійкості функціонування комп’ютерних систем і високої живучості .

Існує два основних методи підвищення надійності комп’ютерних систем, що функціонують у двійкових кодах: підвищення надійності окремих логічних елементів (використання нової елементної бази) та введення різних типів надмірності (застосування різних видів резервування, що впливають як на конструктивну, так і на функціональну надійність комп’ютерних систем). Оскільки надійність логічних елементів комп’ютерних систем визначається рівнем розвитку технології, очевидно, що запровадження надмірності під час використання будь-якої елементної бази є найефективнішим шляхом підвищення надійності комп’ютерних систем. Різноманітність умов та жорсткості вимог (необхідність забезпечення високого ступеня точності обчислень, високої продуктивності та відмовостійкості функціонування комп’ютерних систем у реальному часі, оперативна відновлюваність апаратури після відмов та збоїв тощо), що накладаються на режим функціонування та експлуатації керуючих комп’ютерних систем, не завжди дозволяють застосовувати тимчасове та інформаційне резервування. Внаслідок цього один з ефективних практичних методів підвищення надійності комп’ютерних систем, що функціонують в двійкових кодах, є структурне резервування, наприклад, на рівні тройованої мажоритарної структури [19]. Однак застосування структурного резервування в двійкових кодах ускладнює структуру обчислювального комплексу, підвищує його енергоспоживання, призводить до збільшення масогабаритних та інших характеристик, що зрештою підвищує вартість його створення та експлуатації, а також обмежує сферу його застосування в різних технічних системах. Ця обставина обумовлює необхідність розробки та застосування принципово нових методів підвищення продуктивності, надійності та стійкості до відмови комп’ютерних систем і, зокрема, методів заснованих на застосуванні кодів у системі залишків.

2 ВІДНОВЛЕННЯ ДЕСЯТКОВОГО ЧИСЛА ЗА ЙОГО ЗАЛИШКОМ

2.1 Узагальнення китайської теореми про залишки

Відома Китайська теорема про залишки (див., наприклад, [1]) стверджує, що для заданих m ≥ 2 натуральних чисел *ai*, *i* ∈ 1 : *m*, кожні два з яких взаємно прості, і довільних цілих залишків *ci*, *i* ∈ 1 : *m*, існують такі цілі числа *y* та *xi*, *i* ∈ 1 : *m*, що *aixi* + *ci* = *y*, *i* ∈ 1: *m*.

Вводячи матрицю:



цю систему можна переписати у вигляді:

*Ax = r*, (1)

де *x* = (*x*1, *x*2, · · ·, *xm*, *y*) і *r* = (-*c*1, -*c*2, · · ·, -*cm*).

У термінах перманентів матриць ми виводимо умови цілісної роздільної здатності системи (1) у більш загальному випадку, коли матриця A має базисну схему знаків, визначену у [20]. Більше того, зіставляючи схему матриці A логічну формулу формули у вигляді кон'юнктивної нормальної форми (КНФ) так, як це описано в [3], можна розширити поданий висновок умов цілісної разрешимости системи (1) на випадок, коли відповідна КНФ є мінімально нездійсненною. Зауважимо, що базисній схемі відповідає поліноміально перевіряєма мінімально нездійсненна КНФ [21].

Наступна лема, доведена та використана в [22] для доказу Китайської теореми, виявляється суттєвою і в нашому випадку.

Лемма 1. Нехай дані натуральні числа a, b і c. Рівняння ax − by = c можна в цілих числах x, y тоді і тільки тоді, коли вільний член c ділиться на найбільший загальний дільник коефіцієнтів *a*, *b*.

1. Базова матриця, згідно [23], визначається індуктивно за кількістю рядків *m*.

За m = 1 базисною матрицею називається матриця зі знаковою схемою (+−). Матриця B із числом рядків m ≥ 2 називається базисною, якщо вона має наступну структуру:

, (2)

де всі елементи підрядка *b*1 (співвідн. *b*2) невід'ємні і *b*1 ≠ 0, *b*2 ≠ 0. Якщо елементи *b*1 складаються рівно з одного позитивного елемента, то матриця B1 (співвідн. B2) порожня, інакше B1 (відповідно. B2) - базова матриця. Рядок (*b*1, *b*2) називається провідним. Розміром базисної матриці називатимемо число рядків. Матрицю зі схемою (+−) називатимемо найпростішою.

 Базисна матриця називається насиченою, якщо як у її провідному рядку, так і у провідних рядках усіх її базисних підматриць немає нульових елементів.

Базова матриця називається маргінальною, якщо в кожному її рядку є рівно один негативний і рівно один позитивний елемент.

Прикладом маргінальної базової матриці є матриця *A* з Китайської теореми. Перший рядок *A* є провідним, причому *b*1 = (*a*1); *b*2 = (0, · · · 0, −1). Всі базисні підматриці мають аналогічну схему. Легко показати [24], що з базисної матриці *B*[*M*, *N*] виконано |*N*| = | *M* | + 1.

Виведемо достатні умови цілісної роздільної здатності системи *Bx* = *r* з цілими елементами за умови, що B - або маргінальна базова матриця, чи насичена. Для цього нам знадобиться поняття перманенту матриці.

Нагадаємо, що перманентом матриці D[1: n, 1: n] називається:



де *Pn* - безліч всіх перестановок чисел 1, . . ., *n*. Перманент матриці *D* будемо позначати perm(D).

Якщо *B* = *B* [1 : *m*, 1 : *m* + 1] - базисна матриця, то існує рішення *x* [1 : *m* + 1] системи *Bx* = *O* таке, що *x* [*j*] = perm (*B-j*), *j* ∈ 1 : *m* + 1, де *B-j* - матриця *B* без *j*-го стовпця.

Підтвердженням індукції є те що основа індукції буде тоді, коли за *m* = 1 маємо *B* = (*a*11 − *a*12), де *a*11, *a*12 > 0.

Вочевидь, що *x*1 = *a*12 = perm(*B*−1), *x*2 = *a*11 = perm(*B*−2) задовольняють рівняння *a*11*x*1 – *a*12*x*2 = 0. Індуктивний перехід буде відбуватись за умови, що лема правильна, якщо розмір базисної матриці менший за *m*.

Треба перевірити, чи вектор x задовольняє компонентами *x*[*j*] = perm(*B*−*j*),*j* ∈ 1 : *m* + 1, системі *Bx* = *O*.

У припущенні, що *B* має структуру (1) та *i* — індекс провідного рядка, розіб'ємо *x* на *x*1 та *x*2 так, щоб умови, що перевіряються, мали вигляд:



де *B*1 = *B*[1 : *i* − 1, 1 : *i*], *B*2 = *B*[*i* + 1 : *m*, *i* + 1 : *m* + 1].

Так само як при виведенні уявлення визначника блочно-трикутної матриці, можна довести, що якщо матриця має вигляд (1), то:

, при ; (3)

, при . (4)

Використовуючи (2) отримуємо:



оскільки вектор з компонентами perm(*B*1−*j*), *j* ∈ 1 : *i*, за індуктивним припущенням, - рішення системи *B*1*x* = O. Аналогічно з (3) випливає:





Нехай B - цілісна базисна матриця, що має вигляд (1). Будемо говорити, що для *B* та цілого числа *r* виконано умову кратності, якщо *r* кратне:



Нехай *r* [1 : *m*] - цілісний вектор. Під вектором *Ri* за будь-якого *i* ∈ 1 : *m* будемо розуміти вектор *Ri*[1 : *m*] з компонентами:



Нехай *B* = *B*[1 : *m*, 1 : *m* + 1] - цілочисленна базисна матриця, що має вигляд (1), і *i* - номер її провідного рядка. Система *Bx* = *Ri* можна розв'язати в цілих числах, якщо *B* і *ri* виконано умова кратності. Систему *Bx* = *Ri* можна представити у вигляді :



Згідно 2 цілі *xj* = perm (*B*−*j*1), *j* ∈ 1 : *i* і *xj* = perm(*B*2−j), *j* ∈ *i* + 1 : *m* + 1, задовольняють системам *B*1*x*[1 : *i*] = *O* та *B*2*x*[*i* + 1 : *m* + 1] = *O*. Більше того, за будь-яких цілих числах *λ*1 і *λ*2 мають:

 і .

Розглядаючи *λ*1 і *λ*2, як змінні при даних значеннях *xj*, *j* ∈ 1 : *m* + 1, отримуємо, що *i*-е рівняння має вигляд:

,

або згідно з значеннями *x*[1 : *m* + 1] та визначенням перманенту:

.

З 1 випливає, що це рівняння можна розв'язати в цілих числах *λ*1,*λ*2, якщо *ri* кратно:

.

Звідси випливає, що з виконанні з однорідності умови кратності, *λ*1*x*[1 : *i*] є цілим рішенням системи *B*1*x*[1 : *i*] = *O*, а *λ*2*x*[*i*+ 1 : *m*+ 1] - цілим рішенням *B*2*x*[*i* + 1 : *m* + 1] = *O*. Таким чином, вектор (*λ*1*x*[1 : *i*], *λ*2*x*[*i* + 1 : *m* + 1]) є цілим рішенням досліджуваної системи.

2. Покажемо, що для маргінальних базисних матриць B достатня умова цілісної розподільчості системи *Bx* = *r* збігається з умовою Китайської теореми.

 Елемент *B*[*i*, *j*] базисної матриці *B* називається висячим, якщо *B*[*i*, *j*] ≠ 0 і за *B*[*i*, *j*] > 0 з нього порожня базисна підматриця, а за *B*[*i*, *j*] < 0 під ним порожня базисна підматриця.

Наступне твердження легко доводиться індукцією за розміром базисної матриці.

У кожному стовпці базисної матриці є один висячий елемент. Нехай *B* - маргінальна базова матриця. Тоді для будь-якого рядка *i*0 існує перестановка рядків та стовпців така, що нова матриця залишається базисною та рядок *i*0 стає провідним в ній.

Підтвердження індукцією за розміром базисної матриці. При *m* = 1 очевидно. Нехай *m* ≥ 2. Оскільки *B* є маргінальною базисною матрицею, вона має вигляд (1).

Нехай *i* номер її провідного рядка, *B*1 = *B* [*I*1, *J*1], *B*2 = *B* [*I*2, *J*2]. Схематично маємо:

. (5)

Оскільки *m* ≥ 2, хоча одне з множин *I*1 чи *I*2 не порожньо. Припустимо, що *I*1 ≠ ∅ та *i*0 ∈ *I*1. За індуктивним припущенням рядки з *I*1 і стовпці з *J*1 можна переставити так, що *i*0 стане номером провідного рядка *B1*.

Припустимо, що *j* - номер стовпця в новому порядку, для якого *b*1 [*j*] > 0. Очевидно, що матриця:



базисна. Для стовпця *j* ∈ *J*1 у матриці *B*1 по (4) існує рядок *k* ∈ *I*1 такий, що *B* [*k*, *j*] - висячий елемент.

Нехай *J*1 = {1, 2, ···, *j*, *j* + 1, ···, *i*}, *I*1 = {1, 2, ···, *k* − 1, *k*, *k* + 1, ···, *i* − 1}.

Якщо *B*[*k*, *j*] > 0, то зробимо перестановку рядків та стовпців у матриці (4) так, що для стовпців отримаємо порядок 1, 2,…, *j*, *J*2, *j* + 1,…, *i*, для рядків - порядок

1, 2,…, *k* − 1, i, *I*2, *k*,…, *i* − 1, де в *J*2 та *I*2 збережено колишній порядок. Бо в рядку *i* всі елементи, що не входять до матриці *Bi*2, рівні нулю, перебудована матриця має структуру базисної матриці *B*1 з провідним рядком *i*0 та з вбудованим базисним блоком *Bi*2, що зберігає базову структуру *B*1.

Якщо *B*[*k*, *j*] < 0, то стовпці переставляються так само, як у випадку *B* [*k*, *j*] > 0, а для рядків отримаємо новий порядок: 1, 2,…, *k*, *i*, *I*2, *k* + 1,…, *i* - 1.

Якщо *i*0 ∈ *I*2 то схема перестановок симетрична. Наприклад, маргінальна базисна матриця A з китайської теореми за симетричною схемою перебудовується при *i*0 = 2 в матрицю:



Нехай *B* = *B* [1 : *m*, 1 : *m* + 1] - цілочисленна маргінальна базисна матриця. Система *Bx* = *r* має ціле рішення за будь-якого цілочисного вектора *r* якщо всі ненульові елементи *B* попарно взаємно прості. Виведемо по індукції два твердження:

1. Нехай *B* = *B* [1 : *m*, 1 : *m* + 1] - маргінальна базисна матриця і *l* = l [1 : *m* + 1] -

вектор-рядок, в якому рівно один ненульовий елемент. Перманент матриці **так як і матриці **, дорівнює добутку (*m* + 1) модулів ненульових елементів, взятих по одному з кожного стовпця та кожного рядка.

2. Якщо з маргінальної базисної матриці B[1: m, 1: m + 1] викреслити стовпець, то перманент матриці, що залишилася, дорівнює добутку *m* модулів ненульових елементів, взятих по одному з кожного стовпця та кожного рядка.

За *m* = 1 маємо найпростішу матрицю *B* = (*a*1, −*a*2), *a*1, *a*2 > 0.

1. Очевидно, що дорівнює *a*2 · | *l* [1] |, якщо *l* [1]≠ 0, або *a*1 · | *l* [1] |, якщо *l* [2] ≠ 0.

3. Якщо з найпростішої матриці викреслити будь-який стовпець, то залишиться один елемент, а перманент одноелементної матриці дорівнює модулю цього елемента. Нехай *m* ≥ 2 і для всіх маргінальних базисних матриць меншого розміру обидва твердження виконані.

4. Розглянемо маргінальну базисну матрицю *B* = *B*[1 : *m*, 1 : *m* + 1] виду (4) з приписаним рядком *l*. Нехай *j* — номер ненульового елемента *l*, *i* — провідний рядок в *B*. Для певності припустимо, що *j* ∈ 1: *i*. Тоді виходить:



**,**

Якщо B2 - порожня матриця, то  дорівнює модулю елемента, що становить *b*2. Інакше, через маргінальність, рядок *b*2 містить рівно один ненульовий елемент. Тому до  можна застосувати перше індуктивне припущення. До perm(*B*1[1 : *i* − 1, 1 : *i* \ {*j*}]) можна застосувати друге індуктивне припущення.

Таким чином,  це добуток (*m* + 1) ненульових елементів, взятих по одному з кожного стовпця та кожного рядка. Випадок матриці  симетричний розглянутому. Твердження доведено.

5. Розглянемо маргінальну базисну матрицю *B* [1: *m*, 1: *m* + 1] виду (4) і викреслимо з неї довільний стовпець *j*. Допустимо *j* ∈ 1 : i. Тоді:

****

Перший співмножник за твердженням 1 є добутком ненульових елементів, взятих по одному з кожного стовпця та кожного рядка матриці , а другий співмножник, за індуктивним припущенням твердження, що доводиться, дорівнює добутку ненульових елементів, взятих по одному з кожного стовпця та кожного рядка матриці *B*1 [1 : *i* − 1, 1 : *i* \ {*j*}]. Твердження доведено.

Зауважимо, якщо всі ненульові елементи маргінальної матриці *B* виду (4) попарно взаємно прості, тобто , тому що  і  — два добутки, що складаються з різних елементів матриці за доведеним твердженням 1. Тому умова кратності виконана для провідного рядка.

Нехай *M* = 1: *m*, *N* = 1: *m* + 1 і *B*(*i*0) — базисна матриця, отримана перетворенням маргінальної базисної матриці *B* = *B*[*M*, *N*] за рядком *i*0 ∈ *M* відповідно до теореми 1. За (3) для будь-якого *i*0 ∈ *M* систему:

 (6)

можна розв'язати в цілих числах, якщо виконано умову кратності для провідного рядка. Оскільки *B*(*i*0) — маргінальна базисна матриця, із зауваження випливає, що цю система можна розв'язати в цілих числах, якщо всі ненульові елементи *B*(*i*0) попарно взаємно прості. Але матриця *B*(*i*0) відрізняється від *B* лише перестановкою рядків та стовпців.

Тому всі ненульові елементи *B*(*i*0) попарно прості взаємно.

Нехай  — ціле рішення системи (6). При перестановці стовпців матриці *B* порядок їх нумерації міг змінитися, але множина *N* залишилася тоюж. Тому вектор  задовольняє системі (6) і для матриці *B*. Підсумовуючи отримані рівності отримаємо:

****

Вектор  має цілі коефіцієнти і, за побудовою .

Зауважимо, що китайська теорема є тривіальним наслідком доведеної теореми.

6. Покажемо, що насичену базисну матрицю *B*[*M*, *N*], фіксуючи в ній довільний рядок *i*0 ∈ *M*, можна звести до нової базисної матриці *Hi*0 так, що у *Hi*0 провідною буде рядок *i*0. При цьому цілочисленна роздільність системи *Bx* = *r* виявиться наслідком цілісної роздільності всіх систем *Hi*0 y = *Ri*0, *i*0 ∈ *M*. І тому визначимо наступне перетворення матриці B.

Визначення 6. Нехай *B*[*M*, *N*] — базова матриця та *i*0 ∈ *M*. Позначимо:

,, , .

)-перетворенням матриці *B*[*M*,N] по рядку *i*0 називатимемо матрицю:

.

Нехай є базова матриця:

.

Тут *M* = {1, 2, 3, 4, 5}, *N* = {1, 2, 3, 4, 5, 6}, провідною є рядок 4.

Перетворюватимемо за другим рядком (*i*0 = 2): *,;*

.

Якщо *B*[*M*, *N*] - насичена базисна матриця, то *Hi*0(B)-перетворення матриці *B*[*M*, *N*] є базисною матрицею при будь-якому рядку *i*0 ∈ *M*.

Матриці  і  є, за побудовою, насиченими базовими матрицями. Справді,  виходить з *B* викреслюванням стовпців *j* ∈ *N*, для яких *B*[*i*0, *j*] < 0, та рядків

*i* ∈ {*i*0} ∪ **. У рядках *i* матриці *B* ненульові елементи можуть бути тільки викреслюваних стовпчиках. Це випливає з базисності матриці B. У кожному з рядків, що залишилися, I матриці *B* виявилися викресленими або лише нульові елементи або лише ненульові елементи одного знака. В останньому випадку з насиченості випливає, що в провідних рядках всіх базисних підматриць, що залишилися, не виявиться нульових елементів. Таким чином, матриця  зберігає базисну структуру і задовольняє умову насиченості. Для матриці  міркування аналогічні.

Оскільки підрядок  відповдно - невід'ємна (співвідв. за визначенням множин  і ), матриця  - базисна.

Нехай *B* = *B*[*M*, *N*] — цілочисленна насичена базова матриця і *r* = *r*[*M*] – цілочисельний вектор. Система:



має ціле рішення, якщо при кожному *i*0 ∈ *M* для *Hi*0(*B*)-перетворення матриці *B* та числа *ri*0 виконано умову кратності.

Нехай *H* [*L*, *T*] - матриця, що є *Hi*0(*B*)-перетворенням базисної матриці B = B[M, N] за рядком *i*0 ∈ *M*.

З визначення слідує, що елементи рядка *l* ∈ *L* матриці *H* визначаються елементами деякого рядка *i* ∈ *M*. Нехай *Li* – множина рядків l ∈ *L*, елементи яких визначаються рядком *i*. Для рядка *i*0 маємо | *Li*0| = 1, а решти *i* ∈ *M*, за побудовою, 1 ≤ |*Li*| ≤ 2. Отже, розмір матриці *H* трохи більше, ніж подвоюється проти розміром *B*.

Нехай *Li*0 = {l0}, де *l*0 - номер провідного рядка в матриці *H*. Побудуємо вектор *R*′*i*0 = R′*i*0[L] з компонентами:



З твердження 3 випливає цілочислова роздільність системи , якщо виконано умову кратності для матриці *H* і числа *r*[*i*0]. Нехай y *i*0 - ціле рішення цієї системи. Побудуємо матрицю *C*[*M*, *T*], вважаючи для *i* ∈ *M:*



Вочевидь, що всіх рядків *i* ≠*i*0 виконується *C*[*i*, *T* ]*yi*0 = 0, а для *i* = *i*0 маємо *C*[*i*0, *T*]*yi*0 = *r*[*i*0]. З побудови матриці *H* [*L*, *T*] слід, що при складанні рядків *H*[*L*, *T*] стовпці з матриці (відповідно ) подовжуються до стовпців матриці *B*[*M*, *N*] нулями. При цьому деякі стовпці *B* в матриці *C* повторюються.

Нехай *Nj* = {*t* ∈ *T* | *C* [*M, t*] = *B* [*M, j*]}. Побудуємо вектор , вважаючи для *j* ∈ *N:*



Для цього вектора при  маємо:



і аналогічно, при *і=і*0:

.

Звідси випливає, що якщо умова кратності виконана при кожному *i*0 ∈ *M* для відповідних *Hi*0(*B*)-перетворень та числа *r*[*i*0], то при кожному *i*0 ∈ *M* існує цілісний вектор *xi*0 [*N*], що задовольняє умові:

.

Оскільки  для цілочислового вектора  маємо *Bx* = *r*.

Умови цілісної разрешимости системи *Bx* = *r* у разі, коли *B* — довільна базисна матриця, виводяться аналогічно висновків у теоремах 2 і 4. Тільки у випадку *Hi*0(*B*)-перетворення враховує «зайві» нулі у провідних рядках на відміну насичених базисних матриць. Маргінальні та насичені базисні матриці охоплюють два крайні складності випадку серед базисних матриць. У маргінальному випадку при перетворенні розмір матриці не збільшується, а в насиченому збільшується не більше ніж у два рази (див. доказ теореми 4). Подвоєння досягається, якщо як новий провідний рядок береться рядок з двома висячими елементами. У ненасиченій базисній матриці зберігається оцінка «не більше, ніж у два рази" [26-27].

На закінчення зауважимо, що відповідні умови для матриць, схема яких описується небазовими мінімально нездійсненними КНФ, визначаються семантичним висновком КНФ. Складність побудови цих умов збігається зі складністю виведення з точністю до підрахунку перманенту. Для «важких» КНФ ця складність експонентна, а для КНФ з поліноміальним висновком вона поліноміальна.

2.2 Міжбазисний перехід за алгоритмом Гауса

Сучасні досягнення в галузі оптимізації обчислень та мікропроцесорної техніки породжують нові можливості застосування системи залишкових класів для роботи з багаторозрядними числами [1]. Обмеженням використання системи залишкових класів є відсутність операції порівняння та висока обчислювальна складність отримання десяткового представлення числа. Розробка та дослідження алгоритмів отримання десяткового представлення з набору залишків є актуальною наково-прикладною задачею.

Нехай *m* - натуральне число, *m*1, *m*2, ..., *mt* - взаємно прості натуральні числа, добуток яких більше або дорівнює *m*.

Будь-яке число *x*: 0<=*x*<=*m* може бути однозначно представлено у вигляді послідовності *r*(*x*) = (*r*1, *r*2, ..., *rt*), де *ri*= *x*(mod *mi*).

Для будь-яких чисел *r*1.. *rt*, таким чином, існує одне *x*(mod *m*), таке що **** Більше того, будь-який *x* набору такого виразу має вигляд:

****

де 

Наведене вище формулювання - Китайської теореми про залишки в класичному вигляді [28].

Зауважимо, що число  взаємно просте з , а значить зворотне число у формулі для  завжди існує. Крім того, мають місце рівності:





З теорієї кілець відомо, що: . Суми , як випливає з рівностей вище - ортогональні ідепотенти в кільці . Інакше кажучи, кільце **розкладається на пряму суму:



де **

Послідовність  називається модульним уявленням . Набір модульних уявлень для всіх  називається системою відрахувань . Сума уявлень - послідовність [29-30]. Добуток - послідовність .

Виконаємо алгоритм згортання в двійкове число, системи залишків на мові python.

Проведемо дослідження роботи алгоритмів перетворення з системи залишків у десяткове число. Очевидний алгоритм отриманий з класичного виразу, якщо обчислювати *x* за формулою, даною в теоремі:

На вході:

додатні взаємно прості m1 , ..,mt

цілі r1 , .., rt

На виході:

Ціле число x:

x = ri (mod mi ), 1 <= i <= t

0 <= x <= m, m = m1 \*..\*mt

1. Обчислити m = m1 \*..\*mt оголосимо x=0.

2. for i=1, 2, .., t do

 обчислити yi = m/mi

 обчислити розширеним алгоритмом Евкліда si = yi -1 mod mi

 ci = ri \* si mod mi

 x = x + ci \* yi (mod m)

3. Повернути x.

Основним недоліком алгоритму є висока обчислювальна складність розширеного алгоритму Евкліда, що робить його не достатньо швидким [4].

2.2 Міжбазисний перехід за алгоритмом Гарнера

Нехай усі . Тоді будь-яке число 0 <= x < m може бути однозначно представлено у вигляді:

,

де 0 <= x i < m i +1 , i = 0, 1, .., t-1.

Для *xi* правильне співвідношення:



Таким чином, *x*i можуть бути обчислені один за одним. Алгоритм, що був отриманий, носить ім'я Гарнера (Garner)[31]. Він також придатний для аналогічних операцій з поліномами (попередній алгоритм потребує деяких змін).

1. For i from 2 to t {

 1.1 C i := 1;

 1.2 For j from 1 to (i-1) {

 u: = m j -1 mod m i ;

 C i := u\*C i mod m i ;

 }

 }

2. u: = r 1 ; x: = u;

3. For i from 2 to t {

 u := (r i -x) Ci mod m i ;

 x := x + u\* [Добуток m j від j = 1 до i-1];

 }

4. return (x);

Наведемо приклад. Нехай m 1 =5, m 2 =7, m 3 =11, m 4 =13,

m = m 1 \* m 2 \* m 3 \* m 4 = 5 \* 7 \* 11 \* 13 = 5005,

r (x ) = (2, 1, 3, 8).

Константи C i : C 2 =3, C 3 =6, C 4 =5.

Значення (i, u, x), обчислені на кроці 3: (1, 2, 2); (2, 4, 22); (3, 7, 267) та (4, 5, 2192).

Таким чином, модульне уявлення r(x) відповідає x = 2192.

Знаходження m -1 - зворотного елемента за модулем можна здійснити знову за алгоритмом Евкліда [32].

Дослідження реалізації алгоритмів переведення в десяткове число з залишків системи модулів надає можливість виділити обчислювальні операції з великою обчислювальною складністю та провести їх спрощення. Алгоритм Гарнера має простішу реалізацію та меншу обчсилювальну складність, що надає можливості для подальшої оптимізації.

2.3 Обгрунтування вибору проектних рішень

Мова програмування Python останнім часом все частіше використовується для аналізу даних як у науці, так і комерційній сфері. Цьому сприяє простота мови, а також велика різноманітність відкритих бібліотек.

Розберемо простий приклад дослідження та класифікації даних з використанням деяких бібліотек на Python . Для дослідження, нам знадобиться вибрати цікавий для нас набір даних ( DataSet ). Різноманітні набори Dataset можна скачати з сайту . DataSet зазвичай є файлом з таблицею у форматі JSON або CSV. Для демонстрації можливостей досліджуємо простий набір даних з інформацією про спостереження НЛО . Наша мета буде не отримати вичерпні відповіді на всі головні питання, а показати простоту обробки досить великого обсягу даних засобами Python.  Отже, таблиця зі спостереженнями має такі стовпці:

* + datetime - дата появи об'єкта
	+ city — місто, в якому з'явився об'єкт
	+ state - штат
	+ country - країна
	+ duration (seconds) — час, на який з'явився об'єкт у секундах.
	+ duration (hours/min) — час, на який з'явився об'єкт у годинах/хвилинах
	+ shape - форма об'єкта
	+ comments — коментар
	+ date posted — дата публікації
	+ latitude - широта
	+ longitude - довгота

Для тих, хто хоче пробувати нуль, підготуємо робоче місце. У мене на домашньому ПК стоїть Ubuntu, тож покажу для неї. Для початку необхідно встановити сам інтерпретатор Python3 з бібліотеками. В убунту подібному дистрибутиві це буде:

* + sudo apt-get install python3
	+ sudo apt-get install python3-pip
	+ pip — це система керування пакетами, яка використовується для встановлення та керування програмними пакетами, написаними на Python. З її допомогою встановлюємо бібліотеки, які використовуватимемо:
	+ sklearn - бібліотека, алгоритмів машинного навчання, вона знадобиться нам надалі для класифікації досліджуваних даних,
	+ matplotlib - бібліотека для побудови графіків,
	+ pandas - бібліотека для обробки та аналізу даних. Будемо використовувати для первинної обробки даних,
	+ numpy - математична бібліотека з підтримкою багатовимірних масивів,
	+ google-translate — бібліотека для перекладу тексту через yandex API (для використання потрібно отримати API ключ в яндексі),
	+ pycountry — бібліотека, яку будемо використовувати для перетворення коду країни на повну назву країни,

Використовуючи pip пакети ставляться просто:

pip3 install sklearn

pip3 install matplotlib

pip3 install pandas

pip3 install numpy

pip3 install yandex-translate

pip3 install pycountry

Файл DataSet - scrubbed.csv повинен лежати у робочій директорії, де створюється файл програми.

Тож приступимо. Підключаємо модулі, що використовуються нашою програмою. Модуль підключається за допомогою інструкції:

 import <назва модуля>

Якщо назва модуля занадто довга, та/або не подобається з міркувань зручності або політичних переконань, то за допомогою ключового слова as для нього можна створити псевдонім:

 import <назва модуля> as <псевдоним>

Тоді, щоб звернутися до певного атрибуту, визначеного в модулі

<назва модуля>.<Атрибут> або <псевдонім>.<Атрибут>

Для підключення певних атрибутів модуля використовується інструкція from . Для зручності, щоб не писати назви модуля, при зверненні до атрибуту можна підключити потрібний атрибут окремо.

 from <Название модуля> import <Атрибут>

Підключення потрібних нам модулів:

import pandas as pd

import numpy as np

import pycountry

import matplotlib.pyplot as plt

from matplotlib import cm

from mpl\_toolkits.mplot3d import Axes3D

from google\_translate

 import GoogleTranslate

Використовуєм класс GoogleTranslate з модуля google\_translate.

from google\_translate

import GoogleTranslateException

Використовуєм класс GoogleTranslateException з модуля GoogleTranslateException.

Для того, щоб покращити наочність графіків, напишемо допоміжну функцію для генерації колірної схеми. На вході функція приймає кількість кольорів, яку потрібно згенерувати. Функція повертає зв'язковий список із квітами.

PLOT\_LABEL\_FONT\_SIZE = 14

def getColors(n):

 COLORS = []

 cm = plt.cm.get\_cmap('hsv', n)

 for i in np.arange(n):

 COLORS.append(cm(i))

 return COLORS

Для перекладу деяких назв з англійської на російську мову створимо функцію translate . І так, нам знадобиться інтернет, щоб скористатися API перекладача від Google .

Функція приймає на вхід аргументи:

string — рядок, який потрібно перекласти,

translator\_obj — об'єкт у якому реалізовано перекладач, якщо дорівнює None , то рядок не переводиться і повертає перекладений українську мову рядок.

def translate(string, translator\_obj=None):

 if translator\_class == None:

 return string

 t = translator\_class.translate(string, 'en-ua')

 return t['text'][0]

Ініціалізація об'єкта перекладача має бути на початку коду.

GOOGLE\_API\_KEY = тут має бути визначено ключ АРІ ключ !!!!!'

try:

 translate\_obj = GoogleTranslate(GOOGLE\_API\_KEY)

except GoogleTranslateException:

 translate\_obj = None

Розробка виконувалась в середовищі python. Головне вікно середовища розробки приведене на рисунку 2.1.



Рисунок 2.1. Головне вікно середовища розробки.

Серед середовищ розробки варто виділити середовище Visual Studio та Visual Studio Code, які забезпечують інтеграцію мови python в платформу NET та продукти Microsoft. Серед додаткових чинників вибору мови програмування була можливість python працювати з багаторозрядними числами не використовуючи сторонніх бібліотек.

3 ПРОГРАМНИЙ ЗАСІБ ВІДНОВЛЕННЯ ДЕСЯТКОВОГО ЧИСЛА ЗА ЙОГО ЗАЛИШКАМИ

3.1 Розширений алгоритм Евкліда

Алгоритм обчислення найбільшого спільного дільника (НСД) був описаний давньогрецькими математиками і зараз він відомий, як алгоритм на основі віднімання. І хоча реалізації цього алгоритму є ще в Аристотеля, згодом його почали називати алгоритмом Евкліда.  Найбільший спільний дільник, його властивості та методи обчислення розглянуті у [1].

Нагадаємо, що НСД двох чисел можна обрахувати використовуючи наступну рекурентність:



Алгоритмічно функційю обчислення НСД можна відобразити наступним чином:

int evk( int a, int modul)

{

  if (modul == 0) return a;

  return evk (modul, % modul);

}

 Алгоритм Евкліда можна розширити знаходження по заданим *a* і *b* таких цілих чисел *x* і *y* , що виконується рівність  , де змінна *d* – представляє найбільший спільний дільник чисел a і b .

Нехай для додатніх цілочисельних значень *a* і *b* ( *a* > *b* ) відомі значення , а також числа  *x* ' і *y* ', для яких: .

Тоді значення чисел  *x* і *y* , що є развязками рівняння обчислюються з співвідношень:

 (6)

  Через  будемо позначати цілу частину числа .

Оскільки , то 

де позначено значення 

На рисунку 3.1 приведено схему алгоритму знаходження НСД Евкліда.



Рисунок 3.1 – Алгоритм знаходження НСД Евкліда

   Функція gcd\_ext (int a , int b , int \* d , int \* x , int \* y ), код якої наведено нижче, з аргументами *a* і *b* знаходить значеня  і такі x , y що задовільняють рівність . Для пошуку невідомих значень *x* та *y* необхідно рекурсивно запустити функцію gcd\_ext( b , a mod b , d ,x , y ) і перерахувати значення x і y за запропонованою спочатку формулою. Рекурсія закінчується, тоді коли  значення . При  та , тому будемо вважати що .

void gcd\_ext ( int number, int modul, int \*d, int \*x, int \*y)

{

  int s;

  if (modul == 0)

  {

    \*d = number; \* x = 1; \* y = 0;

    return ;

  }

  gcdext(modul,a% modul,d,x,y);

  s = \* y;

  \* y = \* x - (number / modul) \* (\* y);

  \* x = s;

}

Для послідовного виконання кроків з розширеного алгоритму Евкліда зручно скористатися табличкою з чотирма стовпцями (як показано у прикладі), що відповідають значенням *a , b , x , y*. Процес обчислення алгоритму розділимо на три послідовні етапи:

1) Спочатку обчислимо , використовуючи перших два стовпчики таблиці та рівності запропонованої в (1). В останньому рядку в колонці а знаходитиметься результат обчислень .

 2) Внесемо обчислені результати 1 і 0 відповідно до колонок *x* та *y* останнього рядка.

3) Будемо вносити інформацію послідовно в рядки для колонок x і y знизу догори. Значення змінних x та y внесемо в останній рядок (етап б). Припустимо, що в поточному рядку котрий вже заповнений знаходяться значення чисел ( *x*', *y* '), обчислення можна виконати за допомогою формул (2), далі будемо перераховувати і записувати значення змінних( *x* , *y* ) у відповідні позиції вище рядка, що стоїть.

Знайдемо рішення рівняння рівняння . Обчислення значеннята знаходження відповідних чисел *x , y* виконаємо в таблиці:



Рисунок 3.2 – Процес обчислення

   Порядок та напрямок обчислень вказані стрілками та літерами.

   З таблиці знаходимо, що , тобто .

   Розглянемо невеликий наприклад, як було отримано значення  для першого рядка. З другого рядка будем обрати значення . За наступними формулами (2) буде отримано:





   Лінійним порівнянням називається рівняння виду . Воно має рішення і тоді, коли *b* ділиться на . Якщо , то рівняння можна спростити, замінивши його на , де . Після такого перетворення числа *a '* та *n*' є взаємно простими.

Алгоритм розв'язання рівняння  із взаємно простими *a '* і *n* ' складається з двох частин:

1. Вирішуємо рівняння . Для цього за допомогою розширеного алгоритму Евкліда шукаємо рішення ( x 0 , y 0 ) рівняння . Взявши за модулем *n '* рівність, отримаємо .

2. Помножимо на *b '* рівність . Отримаємо , звідки рішенням вихідного рівняння  буде .

  Якщо , то рівність  еквівалентно .

Доведення. З  випливає, що  ділиться на *m* . Але оскільки k і m взаємно прості, *a – b* ділиться на *m* , тобто . Розв'язати рівняння .

   Значення  є дільником 6 тому рішення існує. Після спрощення отримаємо рівняння . Що згідно з твердженням еквівалентно . Вирішуючи рівняння  за допомогою розширеного алгоритму Евкліда, отримаємо x = -1, y = 1. Звідки . Тобто x = 3 буде як розв'язування рівняння , і .

Діофантовими називаються рівняння виду ,

де  - многочлен з цілими коефіцієнтами.

У цьому розділі розглянемо алгоритм знаходження рішення лінійного діофантового рівняння з двома невідомими видами  (далі для простоти опускати знаки множення і писати прото  ).

Рівняння  має рішення в цілих числах тоді і лише тоді, коли c поділяється на .

Якщо пара ( x 0 , y 0 ) є рішенням рівняння ax + by = c , то вся множина його рішень ( x , y ) описується формулою:



де *k* Є *Z*. Очевидно, якщо , то  для будь-якого цілого *k* .

   Для знаходження часткового рішення  рівняння  слід спочатку знайти рішення  рівняння  ( *d* – найбільший спільний дільник *a* та *b* ) за допомогою розширеного алгоритму Евкліда, після чого помножити його на *c* / *d* . Тобто:

.

Для прикладу знайдемо множину розв'язків рівняння .

1. Рівняння має рішення, тому що 7 ділиться на .

2. Знаходимо рішення рівняння  за допомогою розширеного алгоритму Евкліда: .

3. Знаходимо рішення  вихідного діофантових рівнянь:



Відповідно до теореми 2 безліч рішень вихідного діофантового рівняння має вигляд:.

Для тестування розробленого алгоритму відновлення числа за його залишками необхідно мати доступ до бази багаторозрядних простих чисел. Для побудови систем модулів, щ обули використані в тестах створено алгоритм пошуку простих чисел та відповідне програмне забезпечення. Алгоритм пошуку простих чисел приведений на рисунку 3.1. При створенні програмних модулів для тестування запропонованої схеми згортання залишків чисоа було реалізовано цілий ряд алгоритмів пошуку та підбору простих модулів.



Рисунок 3.3 − Алгоритм пошуку простих чисел

Пошук оберненого елемента за модулем присутній в багатьох схемах відновлення десяткового числа за залишком. Також ця операція поширена в асиметричних криптосистемах. Алгоритм приведений на рисунку 3.4.



Рисунок 3.4 − Блок − схема алгоритму знаходження оберненого числа за модулем

3.2 Реалізація алгоритму Гауса

Китайська теорема про залишки використовується для спрощення виразів, за доказом тотожностей, теорем, наприклад, у теорії чисел [2]. Ця теорема допомагає звести деяке порівняння по модулю до системи простіших порівнянь, і навпаки, звести систему порівнянь одного порівнянню. Розглянуто приклади різних завдань, розв'язання яких знаходиться за допомогою упорядкування системи порівнянь.

При створенні програмних модулів було побудовано блок-схему алгоритму для відновлення числа за його залишком алгоритмом Гауса, яка приведена на рисунку 3.5.

Наприклад: одного разу середній товариш підійшов до розумного товариша і попросив його знайти число, яке при розподілі на 4 дає в залишку 1, при розподілі на 5 дає в залишку 3, а при розподілі на 7 дає в залишку 2.

У криптографії як відкритий ключ використовують дуже великі числа, які є творами великих простих чисел порядку 500-600 розрядів. Так як розкласти таке число на прості множники дуже складно, система працює добре.

При створенні програмного засобу було реалізовано алгоритм відновлення десяткового представлення з системи залишків мовою python.

Для програмної реалізації спочатку необхідно виконати підключення бібліотек.

import math

Надалі буде реалізовано введення простих модулів та їхніх залишків, щоб сформувати систему простих модулів та їхніх залишків.

print("Введіть значення а1, а2, а3, р1, р2, р3:")

a1=int(input("a1= "))

a2=int(input("a2= "))

a3=int(input("a3= "))

p1=int(input("p1= "))

p2=int(input("p2= "))

p3=int(input("p3= "))

print("Знайдемо значенння p:")



Рисунок 3.5 – Схема алгоритму для відновлення числа за його залишком алгоритмом Гауса

Обчислюється добуток простих модулів, що і буде границею розрядної сітки.

p=p1\*p2\*p3

print("p=",p)

Пошук середньозважених коефіцієнтів відбувається шляхом поділу p на кожен залишок з відкиданням дробової частини.

print("\nЗнайдемо значення m1, m2, m3:")

m1=math.trunc(p/p1)

m2=math.trunc(p/p2)

m3=math.trunc(p/p3)

print("m1=",m1,",","m2=",m2,",","m3=",m3)

print("\nЗнайдемо значення M1, M2, M3:")

print("За алгоритмом Евкліда:")

#M1=((m1)^(-1))mod(p1))=g1 mod(p1)

Для реалізації алгоритму Гауса необхідно використати розширений алгоритм Евкліда.

def evklid(num1, num2):

 if num1 == 0:

 return (num2, 0, 1)

 else:

 div, g, z = evklid(num2 % num1, num1)

 return (div, z - (num2 // num1) \* g, g)

З використанням розширеного алгоритму Евкліда обчислимо M1=((m1)^(-1))mod(p1))=g1 mod(p1).

k = evklid(m1, p1)

print("g1=",k[1],",","z1=",k[2])

M1=k[1]%p1

print("Тоді M1=",M1,"\n")

#M2=((m1)^(-1))mod(p2))=g2 mod(p2)

h = evklid(m2, p2)

print("g2=",h[1],",","z2=",h[2])

M2=h[1]%p2

print("Тоді M2=",M2,"\n")

#M3=((m1)^(-1))mod(p3))=g3 mod(p3)

r = evklid(m3, p3)

print("g3=",r[1],",","z3=",r[2])

M3=r[1]%p3

print("Тоді M3=",M3,"\n")

print("Тоді значеня х, буде дорівнювати алгебраїчній сумі m i-тих, M i-тих, p i-тих:")

Процес обчислення алгебраїчної суми m i-тих, M i-тих, p i-тих буде мати наступний вигляд.

a=m1\*M1\*a1

print("m1=",m1)

print("a=",a)

b=m2\*M2\*a2

print("b=",b)

c=m3\*M3\*a3

print("c=",c)

f=a+b+c

print("f, ",f," p ",p)

x=f%p

print("x=",x)

Програмний код було реалізовано у вигляді модуля та використано при оцінці часових характеристик. Код програмного засобу наведено в додатку А.

3.2 Реалізація алгоритму відновлення числа за його залишком на основі додавання

Розроблено схему роботи алгоритму відновлення десяткового представлення на основі додавання (рисунок 3.6).



Рисунок 3.6 - Блок-схема алгоритму відновлення десяткового числа на основі операції додавання

Для реалізації модуля відновлення числа за його модуле створено спеціальний програмний модуль мовою python. Для реалізації алгоритму створено функцію попарного скручування залишків по парах модулів. Найбільшої ефективності алгоритм набуває при найбільших перших модулях, якими відбувається скручування.

def calc(m1,m2,res1,res2):

 result=res1

 for i in range (m2):

 if result%m2==res2:

 break

 result=result+m1

 return (result,m1\*m2)

count=int(input (" Введіть кількість модулів "))

m=[]

res=[]

Далі реалізовано цикл котрий попарно зкручує залишки по простих модулях і на завершальному етапі отримуємо десяткове представлення.

for i in range(count):

 m.append(int(input ("Введіть модуль "+str(i)+" ")))

 res.append(int(input ("Введіть залишок "+str(i)+" ")))

temp=calc(m[0],m[1],res[0],res[1])

print(temp)

for i in range(2,count):

 print(" Для i ",i)

 temp=calc(temp[1],m[i],temp[0],res[i])

 print(temp)

 Після скручування усіх модулів отримаємо повноцінне представлення десяткового числа.

3.3 Вдосконалений алгоритми відновлення десяткового числа за його залишками

Алгоритми опрацювання багаторозрядних чисел мають ряд недоліків, найбільш вагомими з яких є: неможливість розпаралелення обчислення, складність усунення переповнення розрядної сітки, проблеми при використанні спеціалізованих бібліотек для опрацювання багато розрядних чисел, що перевищують розрядність процесора[1]. Наведені недоліки можна усунути за допомогою використання системи залишкових класів (СЗК), проте зазначена система незважаючи на переваги має не повністю досліджений математичний апарат, а саме відсутність операції порівняння, висока обчислювальна складність «скручування» числа в десяткову форму.

Описаний у [2] алгоритм Гарнера має наступні теоретичні положення. Нехай усі . Тоді будь-яке число 0 <= x < m може бути однозначно представлено у вигляді:

,

де 0 <= x i < m i +1 , i = 0, 1, .., t-1.

Для *xi* правильне співвідношення:



Зазначені теоретичні положення дозволяють уникнути пошуку оберненого елемента за модулем.

В роботі [2] приведено алгоритм відновлення десяткового числа за його залишком. Важливою перевагою алгоритму скручування на основі додавання є відсутність необхідності використання взаємопростих модулів для побудови системи.

Програмна реалізація запропонованого алгоритму, яка була використанна для оцінки часових характеристик алгоритму, приведена нижче:

def calc(m1,m2,res1,res2):

 result=res1

 for i in range (m2):

 if result%m2==res2:

 break

 result=result+m1

 return (result,m1\*m2)

count=int(input (" Введіть кількість модулів "))

m=[]

res=[]

for i in range(count):

 m.append(int(input ("Введіть модуль "+str(i)+" ")))

 res.append(int(input ("Введіть залишок "+str(i)+" ")))

temp=calc(m[0],m[1],res[0],res[1])

print(temp)

for i in range(2,count):

 print(" Для i ",i)

 temp=calc(temp[1],m[i],temp[0],res[i])

 print(temp)

Алгоритм приведений в [2] представлений в лінійній ітераційній формі. На відміну від схеми алгоритму, що приведена в роботі [2], пропонується використання можливостей багатопотчності для наведеним математичних положень. Приведемо схему скручування залишків та модулів (рисунок 3.7).

Res1

Mod1

Res2

Mod2

Res3

Mod3

Res4

Mod4

Res5

Mod1\*Mod2

Res6

Mod3\*Mod4

Res7

Mod1\*Mod2\*Mod3\*Mod4

Рисунок 3.7 - Схема скручування в десяткову форму

Згідньо цієї схеми створено код програмного засобу на мові програмування Python.

import mythread

def calc(m1,m2,res1,res2):

 result=res1

 for i in range (m2):

 if result%m2==res2:

 break

 result=result+m1

 return (result,m1\*m2)

count=int(input (" Введіть кількість модулів "))

m=[]

res=[]

for i in range(count):

 m.append(int(input ("Введіть модуль "+str(i)+" ")))

 res.append(int(input ("Введіть залишок "+str(i)+" ")))

mt=[]

rest=[]

for i in range(2,count,2):

 create\_thread\_(calc(m[i],m[i+1],res[i+1]),mt,rest)

while (len(mt)>1):

 create\_thread\_next(calc(m[i],m[i+1],res[i+1]),mt,rest)

print(mt[0])

Розроблено алгоритм роботи програмного засобу, що реалізує запропонований підхід (рисунок 3.8).



Рисунок 3.8 - Блок-схема вдосконаленого алгоритму відновлення десяткового числа за його залишками.

Рисунок 3.9 - Порівння часових залежностей роботи алгоритмів у відповідності до розрядностей

Проаналізувавши графіки залежностей можна зробити висновок про високу ефективність алгоритму на основі додавання, що описаний в [2]. Схема розпаралелення дозволяє підвищити ефективність алгоритму на основі додавання. Тестування для кожної розрядності проводилось над випадковими величинами та з різними наборами модулів, тому результати не відображають функціональної залежності, що співвідноситься з обчислювальною складністю.

Апробація результатів проведених досліджень та розробленого алгоритму приведена в додатку А. Розроблено алгоритм, що дозволяє розпаралелити метод скручування на основі операції додавання. Проведено експериметальні дослідження наведених алгоритмів, що доводять ефективність запропонованих рішень.

3.4 Тестування та налагодження програмного засобу

Для перевірки виконання операції відновлення числа за його залишками створено відповідне програмне забезпечення, головне вікно якого приведене на рисунку 3.10.



Рисунок 3.10 − Головне вікно програмного засобу

Програмний засіб дозволяє виконати перетворення числа в систему залишкових класів та зворотню дію за обраною системою модулів. Система модулів вибрана таким чином, щоб їхній добуток був більший за число, що переводиться (розрядна сітка).

В прикладі наведено переведення багаторозрядного числа, розрядністю більше 256 бітів та час затрачений на його переведення. Швидкодію забезпечує порядок більших модулів по яких відбувається перший етап роботи алгоритму згортання числа. На рисунку 3.11 приведено результат зворотньої операції та час затрачений на перетворення з СЗК в позиційну систему числення.



Рисунок 3.11 − Переведення числа з системи залишкових класів

Виконаємо перетворення для ще одного прикладу з більшим аргументом. Результат виконання приведено на рисунку 3.12.



Рисунок 3.12 − Переведення числа з системи залишкових класів, тест 2

Код основних розроблених модулів приведений в додатку Б.

Для перевірки роботи описаних алгоритмів та доведення ефективності модулярної арифметики, розроблено програмний засіб виконання базових операцій в системі залишкових класів (рисунок 3.13).



Рисунок 3.13 − Виконання базових операцій в СЗК

При розробці системи обрано систему взаємно простих модулів, які покриваю 512 бітну розрядну сітку.

Тестування програмного засобу було проведене на ПК intel core I3 3110m (2.4 ГГц) / RAM 12 ГБ / Windows 10. В результаті тестування було експериментально досліджено розроблену схему відновлення десяткового числа за його залишками та доведено її ефективність.

ВИСНОВКИ

Досліджено математичні основи існуючих систем числення, що дало можливість виявити їхні переваги та недоліки для використання при опрацюванні багаторозрядних чисел.

Досліджено перспективи побудови комп’ютерних систем в СЗК що дало можливість обґрунтувати актуальність та практичну цінність дослідження і виконати постановку завдання.

Проаналізовано математичні основи китайської теореми про залишки, що дозволило проаналізувати властивості системи числення залишкових класів на основі яких розроблені алгоритми між базисних переходів.

Дослідити математичні основи між базисного переходу за алгоритмом Гауса, що дозволяє оцінити та провести експериментальні часові дослідження вказаного алгоритму.

Досліджено роботу алгоритму Евкліда, котрий часто використовується в алгоритмах між базисних переходів та впливає на їхню обчислювальну складність. Проведено дослідження та аналіз алгоритму для відновлення числа за його залишком на основі додавання.

Розроблено вдосконалений алгоритм відновлення числа за його залишками та проведено тестування розробленого програмного засобу на основі запропонованого алгоритму. Запропонований підхід визначається високою швидкодією про що свідчать результати чисельних експериментів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ишмухаметов Ш.Т. Методы факторизации натуральных чисел: учебное пособие / Ш.Т. Ишмухаметов.– Казань: Казан. ун. 2011.– 190 с. ,c. 52
2. Ivasiev S., Yakymenko I., Kasianchuk M., Shevchuk R., Tymoshenko L. The Method of Factorizing Multi-Digit Numbers Based on the Operation of Adding Odd Numbers. Advanced Computer Information Technology (ACIT–2018): Proceedings of the International Conference. Ceske Budejovice (Czech Republic). 2018. P. 232-235.
3. Persson A., Bengtsson L. Forward and reverse converters and moduli set selection in signed-digit Residue Number Systems. J. Signal Process. Syst. 2009. Vol. 56 (1). P. 1–15.
4. Краснобаев В.А., Янко А.С., Кошман С.А., Курчанов В.Н., Бендес Ю.П. Расчет и сравнительный анализ производительности компьютерной системы обработки целочисленных данных, представленных в системе остаточных классов. Системи обробки інформації. 2015. В 3 (128). С. 57-61.
5. Касянчук М., Карпінський М., Казмірчук С. Методологія опрацювання багаторозрядних чисел в асиметричних криптосистемах/ Захист інформації. 2019. Т.21, №2. С. 65- 73.
6. Касянчук М.М., Якименко І.З., Івасьєв С.В., Момотюк О.В. Експериментальне дослідження програмної реалізації методів пошуку оберненого елемента за модулем. Інформатика та математичні методи в моделюванні. 2017. T.7, №3. С. 178–186.
7. Ишмухаметов Ш.Т. Методы факторизации натуральных чисел: учебное пособие / Ш.Т. Ишмухаметов.– Казань: Казан. ун. 2011.– 190 с. ,c. 52
8. Persson A., Bengtsson L. Forward and reverse converters and moduli set selection in signed-digit Residue Number Systems. J. Signal Process. Syst. 2009. Vol. 56 (1). P. 1–15.
9. Краснобаев В.А., Янко А.С., Кошман С.А., Курчанов В.Н., Бендес Ю.П. Расчет и сравнительный анализ производительности компьютерной системы обработки целочисленных данных, представленных в системе остаточных классов. Системи обробки інформації. 2015. В 3 (128). С. 57-61.
10. Zadiraka V., Yakymenko I., Kasianchuk M., Ivasiev S. Theoretical and numerical Krestenson’s basis and its application to problems of cryptographic protection and factorization of multidigit numbers, Computer technologies in information security: collective monograph, By edited V.Zadiraka, Ya.Nykolaichuk, Ternopil: Kart-blansh, 2015. P. 216-260. Ch. 5.
11. Ivasiev S., Yakymenko I., Kasianchuk M., Shevchuk R., Karpinski M., Gomotiuk O. Effective algorithms for finding the remainder of multi-digit numbers. Advanced Computer Information Technology (ACIT–2019): Proceedings of the International Conference. Ceske Budejovice (Czech Republic). 2019. P. 175-178
12. Ivasiev S., Yakymenko I., Kasianchuk M., Shevchuk R., Tymoshenko L. The Method of Factorizing Multi-Digit Numbers Based on the Operation of Adding Odd Numbers. Advanced Computer Information Technology (ACIT–2018): Proceedings of the International Conference. Ceske Budejovice (Czech Republic). 2018. P. 232-235.
13. Rajba T. Klos-Witkowska A., Ivasiev S., Yakymenko I., Kasianchuk M. Research of Time Characteristics of Search Methods of Inverse Element by the Module. Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications: Proceedings of the 2017 IEEE 9 th International Conference. Bucharest, Romania. V.1. September, 2017. P.82– 85 (Scopus).
14. Касянчук М., Карпінський М., Казмірчук С. Методологія опрацювання багаторозрядних чисел в асиметричних криптосистемах/ Захист інформації. 2019. Т.21, №2. С. 65- 73.
15. Касянчук М.М., Якименко І.З., Івасьєв С.В., Момотюк О.В. Експериментальне дослідження програмної реалізації методів пошуку оберненого елемента за модулем. Інформатика та математичні методи в моделюванні. 2017. T.7, №3. С. 178–186.
16. Івасьєв С.В., Лісковецький Д.В., Шпак В.Б. Метод збереження простих великорозрядних чисел у базисі Радемахера. Збірник матеріалів проблемно-наукової міжгалузевої конференції «Автоматизація та комп’ютерно-інтегровані технології» (АКІТ - 2019), Тернопіль, 2019. С. 156-160.
17. Глинська М.Л., Лісковецький Д.В., Івасьєв С.В. Збірник матеріалів наукової конференції «Кібербезпека та комп’ютерноінтегровані технології» (КБКІТ - 2019). –Тернопіль. –2019. С. 21-24.
18. Архангельский А.Я. Программирование в С++ Builder. – М.: Изд. «Бином», 2010 г. 447 с.
19. Ахо Альфред, В. Хопкрофт, Джон, Ульман, Джеффри. Структуры данных и алгоритмы.: Пер.с англ.: М.: Издательский дом “Вильямс”, 2001.- 384с.
20. Ачасова С. М., Бандман О. Л. Корректность параллельных вычислительных процессов. Новосибирск: Наука. Сиб. отд-ние, 1990. – 253 с.
21. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии. М.:МЦНМО, 2003.—328 с.
22. Вербіцький О.В. Вступ до криптології. – Львів: ВНТЛ, 1998 г. – 248 с.
23. Горшков С.И. Производственная эргономика. АМН.–М.: Медицина, 1979. 312 с.
24. Гулямов С.С., Зайнидинов Х.Н. Синтез параллельно-конвейерных вычислительных структур для выполнения быстрых преобразований Хаара. Методы и модели систем обработки. Сб. научн. трудов НПО "Кибернетика" АН РУЗ. Ташкент. -1994. - С.124-127.
25. Карацуба А.А. Сложность вычислений. Тр.МИАМ, 1995.Т.211. С.186‑ 202.
26. Касянчук М. М., Николайчук Я. М., Якименко І. З. Теорія алгоритмів перетворень китайської теореми про залишки в матрично-розмежованому базисі Радемахера-Крестенсона. Вісник національного університету «Львівська політехніка». 2011. № 688. С. 118–124.
27. Айерлэнд К., Роузен М. [Классическое введение в современную теорию чисел](http://ega-math.narod.ru/Books/Ireland.htm), М.: Мир.– 1987- 416 с.
28. Акушский И.Я., Юдицкий Д.И., Машинная арифметика в остаточных классах. М: Сов.радио, 1968. – 440 с.
29. Касянчук М.М. Якименко І.З., Волинський О.І., Івасьєв С.В. Теоретичні основи аналітики та алгоритми оптимізації обчислень простих чисел. Поступ в науку. Збірник наукових праць Бучацького інституту менеджменту і аудиту. Матеріали Міжнародної проблемно-наукової міжгалузевої конференції «Інформаційні проблеми комп’ютерних систем, юриспруденції, енергетики, економіки, моделювання та управління (ПНМК-2010)». В.6, т.1.- с.33-36.
30. Касянчук М.М. Теорія та математичні закономірності досконалої форми системи залишкових класів. Праці Міжнародного симпозіуму „Питання оптимізації обчислень (ПОО–ХХХV)”. Т.1. Київ–Кацивелі. 2009.– С. 306–310.
31. Лабунец В.Г. Теоретико-числовые преобразования над полями алгебраических чисел. Применение ортогональных методов при обработке сигналов и анализе систем. Свердловск: УПИ, 1981, с.44-54.
32. Ленг С. Алгебра. М.: Мир, 1968. - 564 с.
33. Маккелан Дж. X., Рэйдер Ч.М. Применение теории чисел в цифровой обработке сигналов. М.: Радио и связь, 1983. - 264 с.
34. Манин Ю.И., Панчишкин А.А. Введение в теорию чисел. М.: ВИНИТИ, 1990. – Т. 49. – 341 с.
35. Николайчук Я.М., Волинський О.І., Кулина С.В. Теоретичні основи побудови спецпроцесорів у базисі Крестенсона. Вісник Хмельницького національного університету. 2007. № 3. Т.І(93). С. 85-90.
36. Смарт Н. Криптография. Москва: Издательский дом «ТЕХНОСФЕРА», - 2005 г. 526 с.
37. Стенли Б. Липпман. C++ для начинающих, Пер. с англ. 2тт. Москва: Унитех. Рязань: Гэлион, 1992, 304-345 с.
38. Николайчук Я.М. Теорія джерел інформації. Тернопіль: ТзОВ„Терно–граф”, 2010. – 536 с.
39. Николайчук Я.Н., Божнев В.П., Зевелев С.Я. Применение методов теории чисел для сжатия измерительной информации в системах телеконтроля процессов бурения. Материалы Всесоюзной конференции молодых ученых нефтяных ВУЗов. М.:МИНХиГП, 1975. С. 134-138.
40. Якименко І.З., Касянчук М.М., Тимошенко Л.М., Івасьєв С.В., Николайчук Я.М. Алгоритм знаходження системи модулів модифікованої досконалої форми системи залишкових класів, Матеріали МНПК СІЕТ.- Одеса. 2014. С. 115-117.
41. Ященко В.В., Варновский Ю.В. , Нестеренко Г.А. Кабатянский Введение в криптографию. Питер, 2001. 271 с.
42. Kozaczko D., Kasianchuk M., [Yakymenko](http://link.springer.com/search?facet-author=%22I.+Z.+Yakymenko%22) I., Ivasiev S.Vector Module Exponential in the Remaining Classes System. Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS–2015). Warsaw, Poland. V.1, September, 2015. P.161–163.
43. Lakhani G. Some Fast Residual Arithmetic Adders. International Journal of Electronics. - 1994. - P. 225-240.
44. Lenstra H. W. Divisors in residue classes. Math. Comput. 1984. Vol. 42. N 165. P.331-340.
45. Nykolaychuk Ya. M., Kasianchuk M.M., Yakymenko I.Z., [Theoretical Foundations of the Modified Perfect form of Residue Number System](http://link.springer.com/10.1007/s10559-016-9817-2). [Cybernetics and Systems Analysis](http://link.springer.com/journal/10559), 2016, P. 219-223.
46. Tsmots I., Teslyuk V., Teslyuk T., Ihnatyev I. Basic Components of Neuronetworks with Parallel Vertical Group Data Real-Time Processing. Advances in Intelligent Systems and Computing II. CSIT 2017. Advances in Intelligent Systems and Computing, vol. 689, Springer, Cham. P. 558 – 576.
47. Yakymenko I., Kasyanchuk M., Nykolajchuk Ya. Matrix Algorithms of Processing of the Information Flow in Computer Systems Based on Theoretical and Numerical Krestenson’s Basis. Proceedings of the X–th International Conference ”Modern Problems of Radio Engineering, Telecommunications and Computer Science” (TCSET–2010). L’viv–Slavske. 2010. – P.241.