Научная статья
УДК 004.315
https://doi.org/10.35266/1999-7604-2025-3-5

Анализ сложности арифметических операций в модулярной системе счисления квадратичного диапазона

Наталья Сергеевна Золотарева
Сургутский государственный университет, Сургут, Россия

Аннотация. Исследование посвящено изучению особенностей и эффективности модулярной системы счисления квадратичного диапазона, включая реализацию базовых арифметических операций на серийных (позиционных) компьютерах. Основной целью является проанализировать структуру и особенности выполнения различных арифметических операций в модулярной системе счисления квадратичного диапазона и провести сравнение их временной сложности с аналогичными операциями в традиционных позиционных системах счисления. Для достижения поставленной цели были сформулированы следующие задачи: изучение структуры и особенностей модулярной системы счисления квадратичного диапазона, реализация базовых арифметических операций на языке Python, проведение экспериментов и анализ временной сложности операций. Методология исследования включает теоретическое изучение основ модулярной системы счисления квадратичного диапазона, создание алгоритмов выполнения операций на Python, экспериментальное тестирование и анализ результатов. Результатом исследования является создание алгоритма выполнения арифметических операций в модулярной системе счисления квадратичного диапазона, выявление значительного выигрыша в производительности по сравнению с позиционными системами счисления, подтвержденного экспериментально. Таким образом, данное исследование подтверждает перспективность применения модулярной системы счисления для повышения производительности в задачах с высокими требованиями к быстродействию и ресурсоемкости.

Ключевые слова: модулярная система, квадратичный диапазон, арифметические операции, временная сложность, позиционные системы, последовательная обработка, позиционные компьютеры

Для цитирования: Золотарева Н. С. Анализ сложности арифметических операций в модулярной системе счисления квадратичного диапазона // Вестник кибернетики. 2025. Т. 24, № 3. С. 44–54. https://doi.org/10.35266/1999-7604-2025-3-5.

Original article

Arithmetic operations' complexity analysis in modular arithmetic within quadratic range

Natalia S. Zolotareva
Surgut State University, Surgut, Russia

Abstract. The authors study the features and effectiveness of modular arithmetic within a quadratic range, including the implementation of basic arithmetic operations on commodity computers. The research aim is to analyze the structure and features of performing various arithmetic operations in the modular arithmetic within quadratic range and compare their time complexity with similar operations in traditional positional systems. To meet this objective, the authors investigated the structure and characteristics of modular arithmetic within a quadratic range. They implemented arithmetic operations in Python, performed experiments, and assessed the time complexity of operations. The research methods include a theoretical study of the basis of the modular arithmetic within quadratic range, the creation of algorithms for performing operations in Python, experimental testing and analysis of the results. The research result is the creation of an algorithm for performing arithmetic operations in the modular arithmetic within quadratic range, revealing a significant performance gain

compared to positional number systems, confirmed experimentally. Therefore, this study proves that modular arithmetic can improve task productivity that requires speed and resources.

Keywords: modular arithmetic, quadratic range, arithmetic operations, time complexity, positional system, sequential processing, commodity computer

For citation: Zolotareva N. S. Arithmetic operations' complexity analysis in modular arithmetic within quadratic range. *Proceedings in Cybernetics*. 2025;24(3):44–54. https://doi.org/10.35266/1999-7604-2025-5.

ВВЕДЕНИЕ

Модулярная система счисления (МСС) квадратичного диапазона представляет собой расширение традиционной модулярной системы счисления (одинарного диапазона). МСС предназначена для работы в больших числовых областях и специфическими алгоритмами обработки чисел. Она является математической моделью для реализации вычислений на цифровых вычислительных устройствах.

К основным компонентам МСС квадратичного диапазона относятся следующие.

1. Совокупность элементов модулярного квадратичного диапазона.

Элементы этой совокупности представляют собой числа, выраженные в виде вычетов по различным основаниям. Эти основания определяют конкретный диапазон значений, внутри которого проводятся операции над числами. Каждое число записывается в виде остатков от деления на некоторые фиксированные основания – модули.

2. Множество операций.

Операции, выполняемые над представленными в МСС данными, делят на два класса: модульные и немодульные. Модульные характеризуются тем, что при их выполнении не происходит переносов между разрядами. Примерами таких операций являются сложение, вычитание и умножение, деление нацело, умножение на обратный элемент и др. К немодульным операциям отнесены деление, расширение, определение знака, сравнение, определение переполнения, масштабирование, определение ошибки, локализация ошибки, вычисление ранга и др. Для выполнения этих операций, в отличие от модульных, требуется оценка значения числа в целом, которая затруднена в связи с непозиционной природой МСС [1, 2].

3. Алгоритмы выполнения операций.

Алгоритмические процедуры предназначены для оптимизации процесса вычислений, минимизации временных затрат и повышения точности результатов.

Вычисления производятся над числами, представленными в типовом машинном формате, который соответствует возможностям конкретного компьютера. Такие числа задаются в виде цифр, соответствующих позициям позиционной системы счисления. Применяется на практике:

– двоичная система, которая используется благодаря простоте аппаратной реализации и эффективности хранения информации;

– шестнадцатеричная система применяется, когда нужна компактность представления больших объемов данных и удобство восприятия человеком.

МАТЕРИАЛЫ И МЕТОДЫ

В данном исследовании основные арифметические операции в МСС реализованы на серийном (позиционном) компьютере.

Серийный компьютер – это вычислительная система классической архитектуры, в которой операнды представлены в позиционной системе счисления (ПСС), а операции выполняются последовательно. Термин «серийный» подчеркивает традиционный подход к обработке данных, в отличие от специализированных систем, таких как модулярные компьютеры.

К основным характеристикам серийных компьютеров можно отнести следующие.

1. Числа представляются в виде последовательности цифр, каждая из которых имеет вес, зависящий от ее позиции (разряда).

2. Последовательная обработка. Операции выполняются последовательно, например, сложение двух чисел выполняется пошагово, начиная с младшего разряда. Сложение, вы-

читание, умножение и деление выполняются с учетом переносов и заимствований.

3. Традиционная архитектура фон Неймана (один центральный процессор выполняет операции последовательно).

Операции в МСС, реализованные на серийном (позиционном) компьютере, сопровождаются рядом трудностей.

1. Преобразование в ПСС. Числа, представленные в МСС, могут быть преобразованы в ПСС с помощью китайской теоремы об остатках (КТО). После преобразования можно выполнять обычные арифметические операции (сложение, вычитание, умножение и т. д.).

2. Обратное преобразование. После выполнения операций в ПСС результат необходимо снова преобразовать в МСС.

Преобразование между системами счисления требует вычислительных ресурсов.

Введем основные понятия и определения для МСС квадратичного диапазона.

В МСС каждое число представляется в виде набора остатков по соответствующим основаниям (модулям), позволяющим эффективно использовать ресурсы оборудования и минимизировать ошибки округления при выполнении сложных расчетов [3].

Стоит отметить, что в статье модули, по которым вычисляются наименьшие неотрицательные вычеты по модулю (остатки от деления), называются в этой тематике основаниями одинарной или квадратичной МСС.

Определение: Модулярная система счисления квадратичного диапазона – это совокупность математических конструкций и правил, предназначенных для представления и обработки чисел с использованием системы взаимно простых модулей, каждый из которых является квадратом простого числа.

Пусть p_1, p_2, ... p_n – простые числа и пусть $m_i = p_i^2, i = 1, 2, ..., n$ – соответствующие квадратичные модули. Тогда любое целое число X в диапазоне $[0, M-1]$, где $M = m_1 \cdot m_2 \cdot ... \cdot m_n$, может быть однозначно представлено в виде набора остатков:

$$X = (a_1, a_2, ... a_n)(m_1, m_2, ... m_n),$$

где a_i – остаток от деления числа X на модуль m_i, то есть $a_i = X \, mod \, m_i$; пара $a_1, a_2, ... a_n$ называется модулярным представлением числа X.

Определение: Квадратичный модуль – это модуль, задаваемый в виде квадрата простого числа ρ, то есть $m = \rho^2$.

Определение: Диапазон значений – множество чисел, представленных в МСС с квадратичными модулями, ограничено верхним пределом, равным произведению всех модулей.

Китайская теорема об остатках (КТО) – известная в теории чисел. Из теоремы следует: если заданы попарно взаимно простые модули $m_1, m_2, ... m_n$, то любое число Y в диапазоне $[0, M-1]$, где $M = m_1 \cdot m_2 \cdot ... \cdot m_n$, может быть однозначно представлено своими остатками по этим модулям.

Теорема: Пусть $m_1, m_2, ... m_n$ – попарно взаимно простые числа, то есть НОД$(m_i, m_j) = 1$ для всех $i \neq j$. Тогда система сравнений:

$$\begin{cases} Y \equiv \alpha_1 (mod m_1), \\ Y \equiv \alpha_2 (mod m_2), \\ \vdots \\ Y \equiv \alpha_n (mod m_n), \end{cases}$$

имеет единственное решение Y в диапазоне $[0, M-1]$, где $M = m_1 \cdot m_2 \cdot ... \cdot m_n$.

Доказательство.

Пусть модули $m_1, m_2, ... m_n$ попарно взаимно просты, то есть НОД$(m_i, m_j) = 1$ для всех $i \neq j$. Требуется доказать существование и единственность решения Y в указанном диапазоне.

I. Определим

$$M_i = \frac{M}{m_i}, i = 1, 2, \ldots, n.$$

Так как модули попарно взаимно просты, очевидно, что M_i и m_i также взаимно просты для каждого i.

II. Поиск обратных элементов:

Для каждого M_i найдем обратный элемент X_i по модулю m_i, то есть такое число, что:

$$(X_i \cdot M_i) \equiv 1 (mod \, m_i)$$

Обратные элементы можно найти с помощью расширенного алгоритма Евклида.

III. Далее воспользуемся формулой:

$$Y = \sum_{i=1}^{n} (\alpha_i \cdot X_i \cdot M_i) (mod \, M).$$

Рассмотрим арифметические операции в МСС квадратичного диапазона.

1. Сложение, вычитание и умножение.

Сложение и вычитание в МСС квадратичного диапазона выполняются поэлементно по каждому модулю с последующим вычислением остатка [1, 2, 4, 5]. Пусть заданы два числа $A = \alpha_1 \cdot \alpha_2 \cdot ... \cdot \alpha_n$ и $B = \beta_1 \cdot \beta_2 \cdot ... \cdot \beta_n$, где α_i и β_i – остатки по модулям $m_i = p_i^2$, а p_i – простые числа. Тогда:

$$A + B = (\alpha_1 + \beta_1 \ (mod\ m_1), \alpha_2 + \beta_2 \ (mod\ m_2), \ldots, \alpha_n + \beta_n \ (mod\ m_n)),$$

$$A - B = (\alpha_1 - \beta_1 \ (mod\ m_1), \alpha_2 - \beta_2 \ (mod\ m_2), \ldots, \alpha_n - \beta_n \ (mod\ m_n)).$$

Умножение также выполняется поэлементно с последующим взятием остатка:

$$A \cdot B = (\alpha_1 \cdot \beta_1 \ (mod\ m_1), \alpha_2 \cdot \beta_2 \ (mod\ m_2), \ldots, \alpha_n \cdot \beta_n \ (mod\ m_n)).$$

Результаты операций сложения, вычитания и умножения $A + B$, $A - B$ и $A \cdot B$ представлены остатками γ_i σ_i и δ_i по тем же основаниям системы счисления m_i.

$$A + B = (\gamma_1, \gamma_2, \ldots \cdot \gamma_n),$$

$$A - B = (\sigma_1, \sigma_2, \ldots \cdot \sigma_n),$$

$$A \cdot B = (\delta_1, \delta_2, \ldots \cdot \delta_n),$$

где γ_i сравнимо с $\alpha_i + \beta_i$ по модулю m_i, σ_i сравнимо с $\alpha_i - \beta_i$ по модулю m_i, δ_i сравнимо с $\alpha_i \cdot \beta_i$ по тому же модулю, выполняются соотношения:

$$\gamma_i = \alpha_i + \beta_i (mod\ m_i),$$

$$\sigma_i = \alpha_i - \beta_i (mod\ m_i),$$

$$\delta_i = \alpha_i \cdot \beta_i (mod\ m_i).$$

Результатом являются числа:

$$\gamma_i = \alpha_i + \beta_i - \left[\frac{\alpha_i + \beta_i}{m_i} \right] m_i,$$

$$\sigma_i = \alpha_i - \beta_i - \left[\frac{\alpha_i - \beta_i}{m_i} \right] m_i,$$

$$\delta_i = \alpha_i \beta_i - \left[\frac{\alpha_i \beta_i}{m_i} \right] p_i.$$

Пример 1. Рассмотрим выполнение данных операций на примере.

В компьютерах, поддерживающих 32-битные числа, типовой диапазон чисел – от 0 до $2^{32} - 1$. Ставится задача – выбрать подходящий набор квадратичных модулей, который покроет этот диапазон, и выполнить операции сложения, вычитания и умножения в МСС.

В качестве оснований выберем взаимно простые числа, квадраты которых дадут достаточное покрытие диапазона: $m_1 = 17$, тогда $p_1 = 17^2 = 289$, $m_2 = 19$, тогда $p_2 = 19 = 361$, $m_3 = 23$, тогда $p_3 = 23^2 = 529$, $m_4 = 29$, тогда $p_4 = 29^2 = 841$, $m_5 = 31$,тогда $p_5 = 31^2 = 961$. $M = 289 \cdot 361 \cdot 529 \cdot 841 \cdot 961 = 44604646326241 > 2^{32} - 1 = 4294967295$.

Необходимость превышения произведения модулей над машинным диапазоном обусловлена особенностью МСС и требованием полной однозначности представления чисел.

Пусть даны два числа, представленных в МСС: A = (23, 87, 12, 34, 56) и B = (17, 34, 157, 89, 112).

Выполняем поэлементное сложение с последующим взятием остатка по каждому модулю:

$A + B$ = (23 + 17 (*mod* 289), 87 + 24 (*mod* 361), 12 + 157 (*mod* 529), 34 + 89 (*mod* 841), 56 + 112 (*mod* 961)) = (40 (*mod* 289), 121 (*mod* 361), 169 (*mod* 529), 1239 (*mod* 841), 168 (*mod* 961)) = = (40, 121, 169, 123, 168).

Аналогично выполняем поэлементное вычитание со взятием остатка:

$A - B$ = (23−17 (*mod* 289), 87−24 (*mod* 361), 12−157 (*mod* 529), 34−89 (*mod* 841), 56−112 (*mod* 961)) = (6 (*mod* 289), 53 (*mod* 361), 384 (*mod* 529), 786 (*mod* 841), 905 (*mod* 961)) = = (6, 53, 384, 786, 905).

При выполнении операции вычитания в МСС результатом может оказаться отрицательное число. Однако в МСС работают с неотрицательными числами, ограниченными пределами выбранного модуля. Для перевода результата в стандартный вид применяется простая процедура: если число получилось отрицательным, к нему добавляют модуль. Это обусловлено следующими причинами:

1. Периодичность и цикличность.

В МСС действует циклическая арифметика. Если вычитается число и получается отрицательный результат, то это эквивалентно перемещению по кругу на определенное расстояние

влево, и возвращение в положительную зону можно выполнить добавлением модуля.

Периодичность и цикличность в МСС позволяют работать с числами в ограниченном диапазоне, создавая эффект возврата в начало при выходе за пределы. Это важнейшее свойство, лежащее в основе модулярной арифметики и обеспечивающее корректность вычислений.

2. Единственное представление.

В МСС каждое число в диапазоне [0, $M - 1$] должно иметь единственное представление. Появление отрицательного числа нарушает это правило, поэтому нормализация необходима.

3. Последовательность вычислений.

Добавление модуля нормализует результат и восстанавливает нормальное состояние системы, позволяя продолжить вычисления без нарушения законов модулярной арифметики.

На примерах выполним поэлементное умножение с последующим взятием остатка:

$A \cdot B$ = (23 · 17 (*mod* 289), 87 · 34 (*mod* 361), 12 · 157 (*mod* 529), 34 · 89 (*mod* 841), 56 · 112 (*mod* 961)) = (391 (*mod* 289), 2958 (*mod* 361), 1884 (*mod* 529), 3026 (*mod* 841), 6272 (*mod* 961)) = (102, 70, 297, 503, 506).

2. Возведение в степень.

Возведение в степень в МСС квадратичного диапазона выполняется поэлементно с последующим взятием остатка по каждому модулю.

Пусть A = (23, 87, 12, 34, 56) и k = 5. Необходимо найти A^k.

Для каждого модуля m_i возводим соответствующую компоненту числа A в степень k и берем остаток по модулю m_i.

A^5 = (235 (*mod* 289), 875 (*mod* 361), 125 (*mod* 529), 345 (*mod* 841), 565 (*mod* 961)) = = (6436343 (*mod* 289), 4984209207 (*mod* 361), 248832 (*mod* 529), 45435424 (*mod* 841), 550731776 (*mod* 961)) = (24, 254, 202, 399, 935).

3. Нахождение обратного элемента.

Нахождение обратного числа в МСС квадратичного диапазона выполняется поэлементно с использованием расширенного алгоритма Евклида. Это позволяет эффективно обрабатывать числа и выполнять обратные операции в МСС.

Определение: Обратное число A^{-1} по модулю m_i – это такое число, что: $A^{-1} \cdot A \equiv 1$ ($mod\ m_i$).

Для каждого модуля m_i находится обратный элемент X_i для соответствующей компоненты числа A по модулю m_i. Будем использовать расширенный алгоритм Евклида для поиска обратных элементов. Рассмотрим его подробно.

Требуется найти обратный элемент Y для числа A по модулю m, то есть такое число Y, что:

$$A \cdot X \equiv 1 \ (mod\ m).$$

Расширенный алгоритм Евклида основан на классическом алгоритме Евклида для нахождения наибольшего общего делителя (НОД), дополненном поиском коэффициентов Безу.

Алгоритм

I. Начальные условия.

Пусть A и m – заданные числа, причем НОД(A, m) = 1 (условие взаимной простоты). Введем начальные значения: $x_0 = 1, x_1 = 0, y_0 = 0, y_1 = 1$.

II. Основной цикл.

Пока $m \neq 0$, делаем следующие шаги:

$$q = \frac{A}{m},\ \ r = A\left(mod\, m\right),\ \ A = m,\ \ m = r.$$

Обновляем коэффициенты:

$$(x_0, x_1) = (x_1, x_0 - q \cdot x_1), (y_0, y_1) = (y_1, y_0 - q \cdot y_1).$$

III. Завершение цикла.

Когда m = 0, последний ненулевой остаток A – это НОД. Если A = 1, то обратный элемент найден и равен x_0.

Пример 2. Пусть A = (23, 87, 12, 34, 56). Необходимо найти A^{-1}.

Найдем обратный элемент для A = 23 по модулю m = 289.

I. Начальные условия:

$A = 23, m = 289, x_0 = 1, x_1 = 0, y_0 = 0, y_1 = 1$.

II. Основной цикл.

Первый шаг:

$$q = \frac{23}{289} = 0,\ \ r = 23\left(mod\ \ 289\right) = 23.$$

Обновляем коэффициенты:

$A = 289, m = 23, x_0 = 0, x_1 = 1, y_0 = 1, y_1 = 0$.

Второй шаг:

$$q = \frac{289}{23} = 12,\ \ r = 289\left(mod\ \ 23\right) = 13.$$

Обновляем коэффициенты:

$A = 23, m = 13, x_0 = 1, x_1 = -12, y_0 = 0, y_1 = 1$.

Третий шаг:

$$q = \frac{23}{13} = 1,\ \ r = 23\left(mod\ \ 13\right) = 10.$$

Обновляем коэффициенты:
$A = 13, m = 10, x_0 = -12, x_1 = 13, y_0 = 1, y_1 = -1.$
Четвертый шаг:

$$q = \frac{13}{10} = 1,\ \ r = 13\left(mod\ \ 10\right) = 3.$$

Обновляем коэффициенты:
$A = 10, m = 3, x_0 = 13, x_1 = -25, y_0 = -1, y_1 = 2.$
Пятый шаг:

$$q = \frac{10}{3} = 3,\ \ r = 10\left(mod\ \ 3\right) = 1.$$

Обновляем коэффициенты:
$A = 3, m = 1, x_0 = -25, x_1 = 88, y_0 = -2, y_1 = 7.$
Последний шаг:

$$q = \frac{3}{1} = 3,\ \ r = 3\left(mod\ \ 1\right) = 0.$$

Обновляем коэффициенты:

$A = 1, m = 0, x_0 = 88, x_1 = -299, y_0 = -7, y_1 = 23.$

III. Завершение.

Последний ненулевой остаток $A = 1$, следовательно, обратный элемент равен $x_0 = 88$.

Расширенный алгоритм Евклида позволил найти обратный элемент к числу 23 по модулю 289, он равен 88.

Чтобы проверить результат, подставим его в исходное уравнение:

$$A \cdot X \equiv 1\ (mod\ m)$$

$$23 \cdot 88 \equiv 1\ (mod\ 289)$$

Вычислим левую сторону: $23 \cdot 88 = 2024$.

Теперь возьмем остаток от деления на 289: $2024\ (mod\ 289) = 1$.

Таким образом, уравнение выполняется, и 88 действительно является обратным элементом к 23 по модулю 289.

Аналогично, найдем обратные элементы для остальных компонент числа A = (23, 87, 12, 34, 56) по остальным модулям $p_2 = 361$, $p_3 = 529$, $p_4 = 841$, $p_5 = 961$. Используя расширенный алгоритм Евклида для каждого модуля отдельно, получим:

$$87 \cdot 83 \equiv 1\ (mod\ 361)$$

$$12 \cdot 485 \equiv 1\ (mod\ 529)$$

$$34 \cdot 470 \equiv 1\ (mod\ 841)$$

$$56 \cdot 532 \equiv 1\ (mod\ 961)$$

Обратное число для A = (23, 87, 12, 34, 56) по модулям $p_1 = 289$, $p_2 = 361$, $p_3 = 529$, $p_4 = 841$, $p_5 = 961$ соответственно: A^{-1} = (88, 83, 485, 470, 532).

Рассмотрим отдельно случай для компоненты $\alpha_5 = 56$ числа A, по модулю $p_5 = 961$. Расширенный алгоритм Евклида позволил нам найти, что обратный элемент к числу 56 по модулю 961 равен –429. Однако в модулярной арифметике результат должен находиться в диапазоне от $[0, p_5 - 1]$. В нашем случае модуль равен 961.

Чтобы привести результат к нужному диапазону, мы добавляем модуль к отрицательному числу:

$$-429 + 961 = 532.$$

Таким образом, –429 эквивалентно 532 по модулю 961.

Нормализация отрицательного результата путем добавления модуля – это стандартная практика в модулярной арифметике, позволяющая приводить результаты к корректному диапазону.

4. Конвертирование в ПСС.

Для перевода числа из МСС в ПСС используется КТО. Она позволяет построить число Y в ПСС, исходя из остатков по каждому модулю.

Пример 3. Пусть задано число в МСС в виде остатков A = (23, 87, 12), модули $m_1 = 49$, $m_2 = 121$, $m_3 = 169$.

1) Находим произведение модулей:

$$M = 49 \cdot 121 \cdot 169 = 1002001.$$

2) Вычислим:

$$M_1 = \frac{M}{m_1} = \frac{1002001}{49} = 20449,$$

$$M_2 = \frac{M}{m_2} = \frac{1002001}{121} = 8281,$$

$$M_3 = \frac{M}{m_3} = \frac{1002001}{169} = 5929.$$

3) Используя расширенный алгоритм Евклида, находим обратные элементы:

Найдем обратный элемент X_1 для M_1 по модулю $m_1 = 49$:

$X_1 = 46$, так как $46 \cdot 20449 \equiv 1$ (*mod* 49).

Найдем обратный элемент X_2 для M_2 по модулю $m_2 = 121$:

$X_2 = 16$, так как $16 \cdot 8281 \equiv 1$ (*mod* 121).

Найдем обратный элемент X_3 для M_3 по модулю $m_3 = 169$:

$X_3 = 157$, так как $157 \cdot 5929 \equiv 1$ (*mod* 169).

4) Собираем результат:

$Y = (23 \cdot 46 \cdot 20449 + 87 \cdot 16 \cdot 8281 + 12 \cdot 157 \cdot 5929)$ (*mod* 1002001) = 44332430 (*mod* 1002001) = 244386.

Таким образом, число $A = (23, 87, 12)$ в позиционной системе равно 244386.

РЕЗУЛЬТАТЫ И ИХ ОБСУЖДЕНИЕ

На языке программирования Python разработаны программы выполнения рассмотренных операций. Приведены фрагменты алгоритмов и результаты (рис. 1–4).

Исходные данные: в МСС квадратичного диапазона представлены числа $A = (23, 87, 12, 34, 56)$ и $B = (17, 34, 157, 89, 112)$. Модульярные основания $m_1 = 17$, тогда $p_1 = 17^2 = 289$, $m_2 = 19$, тогда $p_2 = 19 = 361$, $m_3 = 23$, тогда $p_3 = 23^2 = 529$, $m_4 = 29$, тогда $p_4 = 29^2 = 841$, $m_5 = 31$, тогда $p_5 = 31^2 = 961$.

Анализ сложности операций в МСС квадратичного диапазона позволяет оценить ресурсы, необходимые для выполнения операций, и служит важным фактором при выборе подходящей методики и оптимизации производительности [6–8].

Оценка сложности алгоритмов – это мера количества ресурсов времени, необходимого для выполнения алгоритма в зависимости от размера входных данных. Сложность позволяет сравнить алгоритмы и выбрать наиболее эффективный.

Обычно сложность оценивается в следующих терминах. Временная сложность – сколько времени займет выполнение алгоритма и пространственная сложность – сколько памяти потребуется для выполнения алгоритма. Чаще всего рассматривают временную слож-

```
from sympy.ntheory.modular import solve_congruence
from sympy.ntheory import isprime
from sympy.ntheory.primetest import mr
from functools import reduce
from operator import mul
```

Рис. 1. Импорт необходимых библиотек

Примечание: составлено автором на основании данных, полученных в исследовании.

```
# Вспомогательные функции
def modular_inverse(a, m):
    """Расширенный алгоритм Евклида для нахождения обратного элемента."""
    t, new_t = 0, 1
    r, new_r = m, a
    while new_r != 0:
        quotient = r // new_r
        t, new_t = new_t, t - quotient * new_t
        r, new_r = new_r, r - quotient * new_r
    if r > 1:
        return None  # Нет обратного элемента
    if t < 0:
        t += m
    return t

def convert_to_positional(rems, mods):
    """Перевод из модулярной системы счисления в позиционную."""
    return solve_congruence(*zip(rems, mods))[0]
```

Рис. 2. Вспомогательные функции

Примечание: составлено автором на основании данных, полученных в исследовании.

```
# Данные
A = (23, 87, 12)
moduli_msc = (49, 121, 169)

# Другие данные для операций
A_full = (23, 87, 12, 34, 56)
B_full = (17, 34, 157, 89, 112)
moduli_full = (289, 361, 529, 841, 961)

# 1. Сложение
addition_result = tuple((a + b) % m for a, b, m in zip(A_full, B_full, moduli_full))
print("Сложение:", addition_result)

# 2. Вычитание
subtraction_result = tuple((a - b) % m for a, b, m in zip(A_full, B_full, moduli_full))
print("Вычитание:", subtraction_result)

# 3. Умножение
multiplication_result = tuple((a * b) % m for a, b, m in zip(A_full, B_full, moduli_full))
print("Умножение:", multiplication_result)

# 4. Возведение в степень
power_result = tuple(pow(a, 5, m) for a, m in zip(A_full, moduli_full))
print("Возведение в степень:", power_result)

# 5. Нахождение обратного элемента
inverses = [modular_inverse(a, m) for a, m in zip(A_full, moduli_full)]
print("Обратные элементы:", inverses)

# 6. Конвертирование из модулярной системы в позиционную
positional_A = convert_to_positional(A, moduli_msc)
print("Перевод из модулярной системы в позиционную:", positional_A)
```

Рис. 3. Основная программа

Примечание: составлено автором на основании данных, полученных в исследовании.

```
Сложение: (40, 121, 169, 123, 168)
Вычитание: (6, 53, 384, 786, 905)
Умножение: (102, 70, 297, 503, 506)
Возведение в степень: (24, 254, 202, 399, 935)
Обратные элементы: [88, 83, 485, 470, 532]
Перевод из модулярной системы в позиционную: 244386
```

Рис. 4. Результат

Примечание: составлено автором на основании данных, полученных в исследовании.

ность, так как она отражает скорость выполнения.

Проведем анализ сложности операций сложения, вычитания, умножения, возведения в степень и нахождения обратного элемента.

1. Сложение и вычитание. В МСС квадратичного диапазона обе операции выполняются поэлементно по каждому модулю с последующим взятием остатка. Память требуется пропорционально размеру входных данных.

Временная сложность операции сложения: $O(n)$, где n – количество модулей.

В ПСС сложность данных операций напрямую зависит от размера чисел, участвующих в расчетах. Операции занимают время порядка $O(n)$, где n – количество цифр в числе (базисная единица измерения сложности).

2. Умножение. В МСС квадратичного диапазона операция умножения также выполняется поэлементно с последующим взятием остатка.

Временная сложность операции сложения: $O(n)$, где n – количество модулей.

Для сравнения в позиционной системе счисления (ПСС) сложность следующая:

– классический метод: сложность $O(n^2)$, где n – количество цифр в числе.

– быстрый метод (Карацуба): сложность $O(n^{\log_2 3})$.

– самый быстрый известный метод (FFT): сложность $O(n \log n)$ где n – размер входных данных.

В ПСС при умножении двух чисел длиной n цифр нужно выполнить n^2 операций сложения и переноса, что и приводит к квадратичной сложности.

В МСС квадратичного диапазона же каждое умножение выполняется отдельно по каждому модулю, что существенно снижается до линейной сложности.

3. Возведение в степень. Используется быстрый алгоритм Binary Exponentiation, который снижает сложность возведения в степень до $O(\log k)$, где k – показатель степени. Однако, поскольку эта операция выполняется для каждого модуля, общая временная сложность составит $O(n \cdot \log k)$.

В ПСС операция возведения числа в степень имеет следующую сложность.

Простой метод: сложность $O(n^k)$, где n – количество цифр, k – показатель степени.

Быстрый метод (Binary Exponentiation): сложность $O(n \cdot \log k)$.

В ПСС возведение в степень приводит к росту числа разрядов, что требует больших ресурсов для обработки.

4. Нахождение обратного элемента. Используется расширенный алгоритм Евклида, который требует $O(n \cdot \log M)$ операций, где M – максимальный модуль. Учитывая, что мы выполняем эту операцию для каждого модуля, общая временная сложность составит $O(n \cdot \log M)$.

Таблица показывает временную сложность арифметических операций для одинарной и квадратичной, а также позиционной систем счисления. Все они используются для отображения чисел из одинаковых числовых диапазонов. Обозначения в МСС: n – количество модулей, k – показатель степени, M – максимальный модуль; в ПСС: n – размер входных данных (количество цифр в числе), k – показатель степени.

Из таблицы можно заметить, что сложность арифметических операций для одинарной и квадратичной МСС одинаковая. Причина заключается в том, что фундаментальная структура операций в обеих системах идентична и ключевым параметром является не сама величина модулей, а их количество. Количество шагов остается постоянным, и разница проявляется лишь в максимальном диапазоне представлений чисел, но не в самой процедуре вычислений.

ЗАКЛЮЧЕНИЕ

Исследование посвящено анализу модулярной системы счисления квадратичного диапазона, ее структуре и особенностям выполнения основных арифметических операций на серийных (позиционных) компьютерах. Оцениваются временные затраты на выполнение основных арифметических опера-

Таблица

Временная сложность выполнения арифметических операций

Система счисления	Основные арифметические операции				
	Сложение	**Вычитание**	**Умножение**	**Возведение в степень**	**Нахождение обратного элемента**
Позиционная система счисления	$O(n)$	$O(n)$	$O(n^2)$ $O(n^{\log_2 3})$ $O(n \cdot \log n)$	$O(n^k)$ $O(n \cdot \log k)$	$O(n^2)$
МСС одинарного диапазона	$O(n)$	$O(n)$	$O(n)$	$O(n \cdot \log k)$	$O(n \cdot \log M)$
МСС квадратичного диапазона	$O(n)$	$O(n)$	$O(n)$	$O(n \cdot \log k)$	$O(n \cdot \log M)$

Примечание: составлено автором на основании данных, полученных в исследовании.

ций в сравнении с позиционными системами счисления. На языке программирования Python разработан алгоритм выполнения основных арифметических операций, приведены результаты его работы.

Проведенное исследование позволяет сделать следующие выводы:

1. Рассмотренные в исследовании арифметические операции характеризуются низкой сложностью и высокой эффективностью. Большинство операций линейны относительно количества модулей, что обеспечивает быструю обработку данных. Как видно из таблицы, это операции сложения, вычитания и умножения. Линейные операции $O(n)$ масштабируются линейно с ростом количества модулей, что делает их очень эффективными и быстрыми даже при большом количестве модулей. К логарифмическим операциям относятся возведение в степень и нахождение обратного элемента. Операция возведения в степень $O(n \cdot \log k)$ эффективна при умеренных показателях степени, но при очень больших степенях может замедляться. Нахождение обратного элемента $O(n \cdot \log M)$ также может замедляться при росте максимального модуля, но в целом остается достаточно эффективной для большинства практических задач.

Большинство операций в МСС квадратичного диапазона имеют низкую сложность и обеспечивают высокую производительность. Линейные операции особенно привлекательны для крупномасштабных вычислений, тогда как операции с логарифмической зависимостью требуют внимания к размерам показателей степени и модулей.

2. Моделирование МСС квадратичного диапазона на серийных компьютерах имеет свои преимущества и недостатки. К преимуществам можно отнести универсальность и простоту реализации. Серийные компьютеры могут выполнять широкий спектр задач, используя стандартные алгоритмы и архитектуры.

К недостаткам реализации МСС квадратичного диапазона на серийном компьютере можно отнести следующие:

– медленная обработка – преобразование между системами счисления требует значительных вычислительных ресурсов. Операции в ПСС выполняются последовательно, что снижает производительность;

– большой объем данных – представление чисел в МСС может потребовать больших объемов памяти для хранения промежуточных результатов;

– ограниченные возможности параллельной обработки – серийные компьютеры не приспособлены для параллельной обработки данных, что делает их менее эффективными для массовых вычислений.

Серийные компьютеры широко используются в жизни и науке. Они универсальны, что делает их популярными в различных областях, от персональных компьютеров до серверов и суперкомпьютеров.

Альтернативой серийным компьютерам являются специализированные многопроцессорные (модулярные) компьютеры.

Специализированные многопроцессорные компьютеры, работающие с МСС, специально разработаны для параллельной обработки данных. Они позволяют эффективно выполнять операции в МСС, распределяя нагрузку между процессорами и используя преимущества параллельных вычислений. Целесообразно рассмотреть сложность выполнения операций в специализированных многопроцессорных компьютерах, которые разработаны для работы с МСС и обеспечивают высокую производительность.

Список источников

1. Акушский И. Я., Юдицкий Д. И. Машинная арифметика в остаточных классах. М. : Советское радио, 1968. 440 с.
2. Амербаев В. М. Теоретические основы машинной арифметики. Алма-Ата : Наука, 1976. 320 с.

References

1. Akushsky I. Ya., Yuditsky D. I. Mashinnaya arifmetika v ostatochnykh klassakh. Moscow: Sovetskoye radio; 1968. 440 p. (In Russ.).
2. Amerbaev V. M. Teoreticheskie osnovy mashinnoy arifmetiki. Alma-Ata: Nauka; 1976, 320 p. (In Russ.).

3. Червяков Н. И., Коляда А. А., Ляхов П. А. и др. Модулярная арифметика и ее приложения в инфокоммуникационных технологиях : моногр. М. : Физматлит, 2017. 400 с.
4. Инютин С. А. Модулярная алгоритмика многоразрядных вычислений. М. : Московский авиационный институт (национальный исследовательский университет), 2020. 160 с.
5. Золотарева Н. С. Обзор методов и оценка сложности алгоритмов операций сравнения в модулярной арифметике и перевода из модулярной системы в позиционную систему счисления // Вестник кибернетики. 2022. № 4. С. 77–90. https://doi.org/10.34822/1999-7604-2022-4-77-90.
6. Инютин С. А. Дробно-рациональные конструкции в компьютерной модулярной арифметике // Информационные технологии. 2019. Т. 25, № 9. С. 515–521.
7. Инютин С. А. Метод вычисления позиционных характеристик модулярного представления с линейной сложностью // Информационные технологии и вычислительные системы. 2024. Вып. 1. С. 109–122. https://doi.org/10.14357/20718632240111.
8. Инютин С. А. Комплексирование систем счисления для многоразрядных вычислительных процессов // Информационные технологии. 2018. Т. 24, № 12. С. 782–790. https://doi.org/10.17587/it.24.782-790.

3. Chervyakov N. I., Kolyada A. A., Lyakhov P. A. et al. Modulyarnaya arifmetika i ee prilozheniya v infokommunikatsionnykh tekhnologiyakh. Monograph. Moscow: Fizmatlit; 2017. 400 p. (In Russ.).
4. Inyutin S. A. Modulyarnaya algoritmika mnogorazryadnykh vychislenii. Moscow: Izd-vo MAI; 2020. 160 p. (In Russ.).
5. Zolotareva N. S. Methods review and complexity estimation of the algorithms for comparison operations in modular arithmetic and transfer operations from a modular number system to a positional number system. *Proceedings in Cybernetics*. 2022;(4):77–90. https://doi.org/10.34822/1999-7604-2022-4-77-90. (In Russ.).
6. Inyutin S. A. Fraction-rational constructions in computer modular arithmetic. *Information Technologies*. 2019;25(9):515–521. (In Russ.).
7. Inyutin S. A. A method for calculating the positional characteristics of a modular representation with linear complexity. *Journal of Information Technologies and Computing Systems*. 2024;(1):109–122. https://doi.org/10.14357/20718632240111. (In Russ.).
8. Inyutin S. A. The aggregation of number systems for multi-digit computational processes. *Information Technologies*. 2018;24(12):782–790. https://doi.org/10.17587/it.24.782-790. (In Russ.).

Информация об авторе

Н. С. Золотарева – аспирант;
https://orcid.org/0000-0001-9751-4232,
zolotareva_ns@surgu.ru

About the author

N. S. Zolotareva – Postgraduate;
https://orcid.org/0000-0001-9751-4232,
zolotareva_ns@surgu.ru