Іванна Леонова

РОЗВ'ЯЗУВАННЯ КОНГРУЕНЦІЙ ПЕРШОГО ТИПУ ЗА ДОПОМОГОЮ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Анотація. *У статті описано основні теоретичні аспекти розв'язування конгруентностей. Узагальнено основні означення і теореми, що стосуються конгруентностей. Наведено приклад роза'язання конгруентностей першого типу за допомогою метода Ейлера, а також продемонстровано код програми, що обчислює конгруентність першого типу за заданими значеннями.*

Ключові слова: *конгруентність першого типу, метод Ейлера, розв'язкування конгруентностей в Python.*

В алгебрі конгруентність — це відношення еквівалентності, яке «зберігає» структуру алгебри. Це означає, що якщо два елементи алгебри конгруентні, то вони будуть конгруентними після застосування будь-якої операції над цією алгеброю. Конгруентність використовується для побудови факторіальних структур, які є новими алгебраїчними структурами, отриманими з вихідної алгебри шляхом об'єднання її елементів у класах еквівалентності.

Конгруентність також використовується в теорії чисел, щоб виразити, що два числа мають однаковий залишок при діленні на одне й те саме число (модуль).

Розглянемо основні означення та теореми, що стосуються конгруентності чисел та методів їх розв'язання.

Означення 1. [1]. Числа a і b називаються конгруентними за модулем m, якщо остачі при діленні їх на число m рівні між собою, тобто $a = mq + r$, $b = mq_1 + r$ і $0 \le r \le m$.

Записують це так:

$$a \equiv b \left(\mathrm{mod}\, m\right).$$

Якщо розглядається кілька чисел, конгруентних між собою за тим самим модулем m, то роблять такий запис:

$$a \equiv b \equiv c \equiv d \left(\mathrm{mod}\, m\right).$$

Теорема 1. [1]. Для того щоб числа a і b були конгруентні за модулем m, необхідно і достатньо, щоб різниця $a-b$ ділилася на m, або що теж саме, $a=b+mt$, де t – довільне ціле число.

Доведення. Необхідність.

$$a \equiv b(\bmod m) \Rightarrow a = mq + r \wedge b = mq_1 + r \Rightarrow a - b = m(q - q_1),$$

тобто $(a-b) \vdots m$, а позначаючи $q - q_1 = t,$ дістанемо $a = b + mt.$

Достатність.

$$(a-b) \vdots m \wedge a = mq + r \Rightarrow$$

$$a = b + mt \wedge a = mq + r \Rightarrow b = m(q - t) + r \Rightarrow b = mq_1 + r.$$

Отже, число b, як і число a, при діленні на m має остачу, рівну r, тобто $a \equiv b(\bmod m)$. У зв'язку з тим, що конгруенції за теоремою 1 тісно пов'язані з рівностями, вони мають багато властивостей, аналогічних властивостям рівностей.

Означення 2. [3]. Функцією Ейлера $\varphi(m)$ називається функція, визначена на множині натуральних чисел; значення $\varphi(m)$ є кількість невід'ємних чисел, менших за m і взаємно простих з m.

Розглянемо деякі властивості (теореми) функції Ейлера, які використовуються до розв'язування різного типу конгруентностей [2, 3]:

Теорема 2. Функція Ейлера $\varphi(m)$ мультиплікативна.

Теорема 3. Якщо p – просте число, а k – натуральне число, то

$$\varphi\left(p^k\right) = p^k\left(1 - \frac{1}{p}\right).$$

Теорема 4. Якщо канонічний розклад числа m має вигляд

$$m = p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots \cdot p_s^{k_s},$$

$$\text{то } \varphi(m) = m\prod_{i=1}^{s}\left(1 - \frac{1}{p_i}\right).$$

Теорема 5. Сума значень функції Ейлера для всіх дільників d_i числа m дорівнює m:

$$\sum_i \varphi(d_i) = m.$$

Теорема 6.(*Ейлера*). Якщо m – натуральне число і $m > 1, (a,m) = 1,$ то

$$a^{\varphi(m)} \equiv 1 (\mathrm{mod}\, m).$$

Перейдемо безпосередньо до написання коду для розв'язання конгруенцій першого типу способом Ейлера. У загальному випадку розв'язок зводиться до знаходження оберненого елемента $a^{-1} \pmod{m}$, тобто такого a^{-1}, що:

$$a \cdot a^{-1} \equiv 1 \pmod{m}.$$

Цей обернений елемент можна знайти за допомогою розширеного алгоритму Евкліда. Якщо існує, тоді: $x \equiv a^{-1} \cdot b \pmod{m}$.

Розглянемо наступний приклад. Користуючись методом Ейлера, розв'язати конгруенцію $27x \equiv 24 \pmod{102}$.

Безпосередньо, розв'язання матиме наступний вигляд: знайдемо $(27,102) = 3$. Задана конгруенція матиме три розв'язки, оскільки $24 \vdots 3$. Тоді ділимо обидві частини і модуль цієї конгруенції на 3: $9x \equiv 8 \pmod{34}$.

Оскільки тепер $a = 9$, $b = 8$, $m = 34$, $34 = 2 \cdot 17$, то знайдемо функцію Ейлера $\varphi(m) = \varphi(34)$:

$$\varphi(34) = 2 \cdot 17 \cdot \left(1 - \frac{1}{2}\right) \cdot \left(1 - \frac{1}{17}\right) = 15.$$

Тоді $x = 8 \cdot 9^{15} \pmod{34}$. Число $8 \cdot 9^{15}$ замінимо найменшою невід'ємною остачею за модулем 34. Тоді отримаємо

$$x = 8 \cdot 9^{15} = 8 \cdot 3^{30} = 8 \cdot 3^{14} = 8 \cdot (2187)^2 = 8 \cdot 11^2 = 16 (\mathrm{mod}\, 34).$$

Отже, $x = 16 (\mathrm{mod}\, 34)$ і є розв'язком конгруенції $9x \equiv 8 \pmod{34}$.

Тоді конгруенція $27x \equiv 24 \pmod{102}$ має розв'язки:

$$x = 16 \pmod{102},$$
$$x = 50 \pmod{102},$$
$$x = 84 \pmod{102},$$

або ж $x = 16;50;84 \pmod{34}$.

Використаємо дану конгруенцію, як основну, для написання коду в програмі Python. У написанні коду використаємо такі кроки:

- використаємо фукцію extended_gcd(a, b) — розширений алгоритм Евкліда, який знаходить $d = \gcd(a,b)$ та коефіцієнти x і y, що задовольняють рівність: $a \cdot x + b \cdot y = d$ (обернений елемент $a^{-1} \pmod{m}$ можна отримати саме з цього алгоритму, якщо $\gcd(a,m) = 1$);
- використаємо функцію solve_congruence(a, b, m) — знаходимо найбільший спільний дільник $d = \gcd(a,b)$. Це перший тест на існування розв'язків;
- перевіряємо сумісність — якщо d не ділить b, то рівняння немає розв'язків
- зводимо до простої форми — спрощуємо всі коефіцієнти поділивши на d, щоб отримати конгруенцію $a_1 x \equiv b_1 \pmod{m_1}$, з якою легше працювати;
- знаходимо обернений елемент — використовуємо знову розширений алгоритм Евкліда, щоб знайти обернений елемент $a_1^{-1} \pmod{m_1}$.
- знаходимо основний розв'язок — перший розв'язок зведеної конгруенції x_0 у модулі m_1;
- формуємо повний набір розв'язків — якщо $d > 1$, то конгруенція має d розв'язків: $x \equiv x_0 + k \cdot \frac{m}{d} \pmod{m}$, $k = 0,1,...,d-1$.

Для попередньо вказаного рівняння, тобто для значень $a = 27$, $b = 24$, $m = 102$ програма буде мати вигляд (рис. 1):

```python
def extended_gcd(a, b):
    if b == 0:
        return a, 1, 0
    d, x1, y1 = extended_gcd(b, a % b)
    x, y = y1, x1 - (a // b) * y1
    return d, x, y

def solve_congruence(a, b, m):
    d, x, y = extended_gcd(a, m)
    if b % d != 0:
        print("Немає розв'язків (несумісна конгруенція).")
        return []

    # Основний розв'язок
    a1, b1, m1 = a // d, b // d, m // d
    _, inv_a1, _ = extended_gcd(a1, m1)
    x0 = (inv_a1 * b1) % m1

    # Усі розв'язки
    solutions = [(x0 + k * m1) % m for k in range(d)]
    return solutions

# Приклад: розв'язати 27x ≡ 24 (mod 102)
a, b, m = 27, 24, 102
sols = solve_congruence(a, b, m)
print(f"Розв'язки конгруенції {a}x ≡ {b} (mod {m}): {sols}")
```

Рис.1. Код програми для обчислення конгруенції першого типу (за конкретними даними)

Результат розв'язання конгруенції виводитиметься на екран у вигляді (рис. 2):

```
Розв'язки конгруенції 27x ≡ 24 (mod 102): [16, 50, 84]
```

Рис. 2. Результат виконання програми

Цей приклад демонструє як використати функцію для конкретної задачі.

Список використаних джерел:

1. Алгебра та теорія чисел: конгруенції та їх застосування (завдання для самостійної роботи та методичні вказівки до їх виконання)/ В. М. Дармосюк, О. Ю. Пархоменко: посібник для самостійної та дистанційної роботи студентів. Миколаїв: Миколаївський національний університет ім. В.О. Сухомлинського, 2021, 152 с.

2. Ізюмченко Л.В. Практикум з теорії чисел / Л.В. Ізюмченко. Кіровоград: КДПУ ім. В. Винниченка, 2014. 76 с.

3. Оглобіна О.І. Елементи теорії чисел : навч. посіб. / О. І. Оглобліна, Т. С. Сушко, Ю. В. Шрамко. Суми : Сумський державний університет, 2015. 186 с.

SOLVING FIRST-KIND CONGNUENCY PROBLEMS USING SOFTWARE

Abstract. *The article describes the main theoretical aspects of solving congruences. The main definitions and theorems related to congruences are summarized. An example of solving congruences of the first type using Euler's method is given, and the program code that calculates the congruence of the first type from given values is also demonstrated..*

Keywords: *congruence of the first type, Euler's method, solving congruences in Python.*