



Algebraic Structures

Author: Sakura

Санкт-Петербург, 2025

ПРЕДИСЛОВИЕ

“The darker the night, the brighter the stars.”

“Even the darkest night will end and the sun will rise.”

Привет! Да, я хочу начать наше общение с чего-то не слишком формального, но и не пренебрежительно фамильярного. Тем не менее, я буду обращаться на «вы».

Ваше направление существует второй год — столько же времени я взаимодействую со студентами в качестве наставника, учителя, или, иными словами, кого-то, кто направляет и учит. За это время я слышал множество волнующих вопросов о том, зачем вообще изучать тот или иной предмет. Полагаю, что и вы за время пребывания в университете не раз зададитесь вопросом: «А зачем нам всё это?» В таких случаях предлагаю обратиться к философу. Изучая древние мифы, Клод Леви-Стросс писал, что это не загадки, требующие ответов, а ответы на ещё не заданные вопросы.

Совсем недавно вы были школьниками, а сейчас — студенты. Можно сказать, вы начали новую жизнь, перерождение. Не случайно в качестве иллюстрации к данной лабораторной работе был выбран цветок ликориса (паучьей лилии). Он цветёт около недели в начале осени. В японской мифологии это символ смерти, а вместе с тем — новой жизни.

Для меня существенное отличие университета (высшей школы) от школы состоит в необходимости самостоятельного принятия решений, а главное — в ответственности за них. Решения могут быть самыми разными: от выбора материалов для изучения предмета до выбора между посещением пары и сном. Если бы меня спросили, что является главным на этом пути, я бы без сомнения ответил — воля. Воля действовать вопреки обстоятельствам, как бы тяжело и трудно ни было, идти вперёд.

При чём тут новая жизнь, перерождение, самостоятельное принятие решений и воля? Дело в том, что личность рождается дважды: первый раз — внутри общества, а второй — самостоятельно, сделав личностный выбор. Например, выбор университета может стать вторым рождением личности; выбор жизненного партнёра, выбор профессии — всё это акты самосозидания. Главное — чтобы выбор был самостоятельным.

При первом рождении личности появляется структура мотивов, но она не осознаётся — в этом нет необходимости. При втором — она осознаётся, и личность принимает решения, руководствуясь собственной структурой мотивов. Осталось сказать про волю. Воля — инструмент разрешения мотивационного конфликта, зачастую между мотивами природными (биологическими) и личностными. К примеру, выбор между просмотром коротких видео и выполнением дела по учёбе или работе.

Эта работа не получилась бы такой, какая она есть, без помощи и вдохновения, полученных от *Hermitian operator*, И. А. Кобченко и Р. А. Попкова.

ПРАВИЛА ВЫПОЛНЕНИЯ РАБОТЫ

1. **Теоретические задания** требуют развёрнутых ответов с математическими выкладками и пояснениями. Односложные ответы («да/нет», «возможно») считаются некорректными.
2. **Работу можно выполнять на любом языке программирования.** Только для обязательных уровней *easy* и *normal* разрешается использование любой системы компьютерной алгебры, например, Wolfram|Alpha, Sage, Singular, SymPy.
3. Работа выполняется и сдаётся в электронном виде. Допускается использование Jupyter Notebook и/или LaTeX с вставками кода. Кроме того, весь код необходимо продублировать в облачной системе контроля версий, например, GitHub или GitLab.



Easy level

Ключевые слова:

Палиндромы и их степени, простые числа и циклические перестановки, распределение простых, факторизация и функция Эйлера

EASY LEVEL

Для разминки, окунёмся немного в мир теории чисел, знакомый со школьной скамьи.

Если $a = \overline{a_{n-1} \dots a_1 a_0}$, то реверсированное $\text{rev}(a) = \overline{a_0 a_1 \dots a_{n-1}}$. Если $a = \text{rev}(a)$, то a называется палиндромом.

Easy:

1. Найдите все палиндромы $a < 10^5$, такие что a^2 — также палиндром.
2. Найдите все простые числа $p < 10^6$, все циклические перестановки цифр которых также являются простыми.

```
from typing import List

def palindromic_squares_and_circular_primes() -> tuple[List[int], List[int]]:
    """
    Возвращает:
        tuple:
            - список всех палиндромов а < 100000, для которых а^2 - палиндром;
            - список всех простых р < 1000000, все циклические перестановки цифр которых
              → просты.
    """
    pass
```

Easy:

1. Найдите все палиндромы $a < 10^5$, такие что a^3 — также палиндром.
2. Найдите все палиндромические простые числа $p \leq 10\,000$.

```
from typing import List

def palindromic_cubes_and_palindromic_primes() -> tuple[List[int], List[int]]:
    """
    Возвращает:
        tuple:
            - список всех палиндромов а < 100000, для которых а^3 - палиндром;
            - список всех простых р <= 10000, которые являются палиндромами.
    """
    pass
```

Easy: Найдите первые 100 простых чисел, состоящих только из:

- цифр 1 и 3,
- цифр 1 и 5,
- цифр 1 и 7,
- цифр 1 и 9.

Проанализируйте, какие из этих типов встречаются реже всего и почему.

```
from typing import Dict, List

def primes_with_two_digits() -> Dict[str, List[int]]:
    """
    Возвращает словарь вида:
    {
        '13': [список первых 100 простых из {1,3}],
        '15': [список первых 100 простых из {1,5}],
        '17': [список первых 100 простых из {1,7}],
        '19': [список первых 100 простых из {1,9}]
    }
    """
    pass
```

Easy: Найдите первые 1000 пар простых-близнецов $(p, p+2)$. Исследуйте, как меняется отношение количества пар близнецов $\leq n$ к общему числу простых $\leq n$ при росте n .

```
from typing import List, Tuple

def twin_primes_analysis(limit_pairs: int = 1000) -> Tuple[List[Tuple[int, int]], 
→ List[float]]:
    """
    Возвращает:
        - список первых `limit_pairs` пар близнецов (p, p+2);
        - список значений отношения  $\pi_2(n) / \pi(n)$  для n, соответствующих последним
        → элементам каждой пары,
        где  $\pi_2(n)$  - количество пар близнецов  $\leq n$ ,  $\pi(n)$  - количество простых  $\leq n$ .
    """
    pass
```

Easy: Для $n = 2, 3, \dots, 50$ вычислите разложение числа $n! + 1$ на простые множители. Определите:

- максимальное количество *различных* простых делителей среди всех $n! + 1$;
- случаи, в которых $n! + 1$ содержит «большой» простой множитель (например, $> 10^6$).

```
from typing import Dict

def factorial_plus_one_factors() -> Dict[int, Dict[int, int]]:
    """
    Возвращает словарь вида:
        { n: {простой_делитель: степень, ...}, ... }
    для n от 2 до 50, где ключ - n, значение - разложение n! + 1 на простые множители.
    """
    pass
```

Easy: Реализуйте два способа вычисления функции Эйлера $\varphi(n)$:

1. Прямой перебор: $\varphi(n) = |\{1 \leq k \leq n : \gcd(k, n) = 1\}|$;
2. Через разложение на простые множители: если $n = p_1^{e_1} \cdots p_k^{e_k}$, то $\varphi(n) = n \prod_{i=1}^k \left(1 - \frac{1}{p_i}\right)$.

Сравните время работы обеих реализаций и встроенной функции `euler_phi` из систем компьютерной алгебры, например, Wolfram|Alpha, Sage, Singular, SymPy.

```
import time
from typing import List

def euler_phi_direct(n: int) -> int:
    """Вычисляет (n) прямым перебором."""
    pass

def euler_phi_factor(n: int) -> int:
    """Вычисляет (n) через разложение на простые множители."""
    pass

def compare_euler_phi_methods(test_values: List[int]) -> dict:
    """
    Сравнивает время работы трёх методов на заданных значениях.
    Возвращает словарь с тремя списками времён (в секундах).
    """
    pass
```



Normal level

Ключевые слова:

Симметрическая группа, подгруппы, смежные классы, порядок элемента, циклическая подгруппа, образующий элемент, изоморфизм групп, конечные поля, корни полиномов, приводимость, разложение на неприводимые множители, расширенный алгоритм Евклида

NORMAL LEVEL

Перейдём к материалу, который изучают на линейной алгебре в университете.

ГРУППЫ

Пусть $N = \text{ISU} \bmod 20$. Для вычислительной реализуемости определим:

$$m = 4 + (N \bmod 5), \quad n = 2 + (N \bmod 10), \quad k = 1 + (N \bmod 7),$$

$$n_1 = N \bmod 6, \quad n_2 = (N + 1) \bmod 6, \quad n_3 = (N + 2) \bmod 6.$$

Значения p, s, r, t зависят от $N \bmod 5$:

$$p = \begin{cases} 29, & N \equiv 0 \pmod{5}, \\ 31, & N \equiv 1 \pmod{5}, \\ 37, & N \equiv 2 \pmod{5}, \\ 23, & N \equiv 3 \pmod{5}, \\ 19, & N \equiv 4 \pmod{5}, \end{cases} \quad s = \begin{cases} 5, & N \equiv 0 \pmod{5}, \\ 4, & N \equiv 1 \pmod{5}, \\ 3, & N \equiv 2 \pmod{5}, \\ 17, & N \equiv 3 \pmod{5}, \\ 15, & N \equiv 4 \pmod{5}, \end{cases}$$

$$r = \begin{cases} 59, & N \equiv 0 \pmod{5}, \\ 60, & N \equiv 1 \pmod{5}, \\ 38, & N \equiv 2 \pmod{5}, \\ 45, & N \equiv 3 \pmod{5}, \\ 44, & N \equiv 4 \pmod{5}, \end{cases} \quad t = \begin{cases} 9, & N \equiv 0 \pmod{5}, \\ 8, & N \equiv 1 \pmod{5}, \\ 7, & N \equiv 2 \pmod{5}, \\ 12, & N \equiv 3 \pmod{5}, \\ 14, & N \equiv 4 \pmod{5}. \end{cases}$$

Normal: Найдите все подгруппы симметрической группы S_m . Выведите их количество и одну случайную подгруппу. Для подгруппы с индексом $N \bmod |S_m|$ (число подгрупп) постройте левые и правые смежные классы, определите её индекс и проверьте, является ли она нормальной.

```
def subgroups_of_Sm(N: int) -> dict:  
    pass
```

Normal: В группе S_m возьмите элемент g с индексом $N \bmod |S_m|$. Найдите порядки элементов $g^{n_1}, g^{n_2}, g^{n_3}$ и порядки циклических подгрупп, ими порождаемых.

```
def element_powers_in_Sm(N: int) -> dict:  
    pass
```

Normal: В группе S_m найдите все решения уравнения

$$\sigma^n = (1 \ 2 \ 3 \ \dots \ m-1).$$

Выведите количество решений и три случайных решения. Опишите, что у них общего.

```
def solve_sigma_power_eq(N: int) -> dict:  
    pass
```

Normal: В циклической группе порядка m найдите:

- все элементы g , такие что $g^k = e$,
- все элементы порядка k .

```
def elements_of_order_k_in_cyclic_group(N: int) -> dict:
    pass
```

Normal: Найдите все подгруппы мультипликативной группы \mathbb{Z}_m^* .

```
def subgroups_of_Zm_star(N: int) -> list:
    pass
```

Normal: Пусть $s \in \mathbb{Z}_p^*$. Найдите порядок элемента s^r в \mathbb{Z}_p^* .

```
def order_of_sr(N: int) -> int:
    pass
```

Normal: Найдите порядок элемента t в группе \mathbb{Z}_p^* . Является ли t образующим (примитивным корнем)?

```
def order_and_primitivity_of_t(N: int) -> dict:
    pass
```

Normal: Найдите все образующие (примитивные корни) циклической группы \mathbb{Z}_m^* .

```
def generators_of_Zm_star(N: int) -> list:
    pass
```

Normal: В аддитивной группе \mathbb{Z}_m найдите циклическую подгруппу, порождённую элементом $t \bmod m$. Определите её порядок и все порождающие элементы. Опишите связь с исходным t .

```
def cyclic_subgroup_in_Zm_additive(N: int) -> dict:
    pass
```

Normal: В мультиликативной группе \mathbb{Z}_m^* найдите циклическую подгруппу, порождённую элементом $t \bmod m$. Определите, какой циклической подгруппе в симметрической группе S_d (где d — порядок подгруппы) она изоморфна.

```
def isomorphism_of_cyclic_subgroup_Zm_star(N: int) -> dict:
    pass
```

ПОЛИНОМЫ

Определим коэффициенты:

$$\begin{aligned} a_i &= (i+N) \bmod 4, & b_j &= (j+N) \bmod 7, & c_k &= (k+N) \bmod 5, \\ d_l &= (l+N) \bmod 9, & r_m &= (m+N) \bmod 11, & s_t &= (t+N) \bmod 11. \end{aligned}$$

Параметры конечного поля:

$$p = \begin{cases} 5, & N \equiv 0 \pmod{5}, \\ 3, & N \equiv 1 \pmod{5}, \\ 2, & N \equiv 2 \pmod{5}, \\ 13, & N \equiv 3 \pmod{5}, \\ 11, & N \equiv 4 \pmod{5}, \end{cases} \quad m = \begin{cases} 3, & N \equiv 0 \pmod{5}, \\ 4, & N \equiv 1 \pmod{5}, \\ 7, & N \equiv 2 \pmod{5}, \\ 2, & N \equiv 3 \pmod{5}, \\ 2, & N \equiv 4 \pmod{5}. \end{cases}$$

Normal: Найдите все корни полиномов:

- $f(x) = x^9 + \sum_{i=0}^8 a_i x^i \in \mathbb{F}_4[x]$;
- $f(x) = \sum_{i=0}^6 b_i x^i \in \mathbb{F}_7[x]$.

Normal: Исследуйте полиномы на приводимость. Приводимые полиномы разложите на неприводимые множители:

- $f(x) = x^5 + \sum_{i=0}^4 c_i x^i \in \mathbb{F}_5[x]$;
- $f(x) = x^4 + \sum_{i=0}^3 d_i x^i \in \mathbb{F}_9[x]$.

Normal: Пусть

$$f(x) = \sum_{i=0}^7 r_i x^i, \quad g(x) = \sum_{i=0}^3 s_i x^i$$

— полиномы над полем \mathbb{F}_{11} . Найдите $\gcd(f, g)$ и его линейное представление:

$$\gcd(f, g) = u(x)f(x) + v(x)g(x).$$

Normal: Пусть

$$f(x) = s_2 x^2 + s_1 x + s_0, \quad g(x) = x^8 + x^4 + x^3 + 6x + 2$$

— полиномы над полем \mathbb{F}_{13} . Найдите обратный элемент $f^{-1} \pmod{g}$, то есть такой полином h , что

$$f(x)h(x) \equiv 1 \pmod{g(x)}.$$

Normal: Реализуйте алгоритм генерации всех неприводимых полиномов степени d над простым конечным полем \mathbb{F}_q , где q — простое. Протестируйте его для $q = 2, 3, 5$ и $d = 2, 3, 4$.

```
def generate_irreducible_polynomials(q: int, d: int) -> list:
    """
    Возвращает список всех неприводимых полиномов степени d над F_q.
    """
    pass
```



Hard level

Ключевые слова:

Идеал, главный идеал, порождающий идеала

HARD LEVEL

Я рад, что вы отважились заглянуть в этот раздел. Нас ждёт необычное путешествие в идеальный мир. Это не будет просто, отнюдь, лишь стойкий духом сможет пройти этот путь. Интересно? Тогда только вперёд!

ЗОВ ИДЕАЛЬНОГО МИРА

В вселенной полиномов есть королевства – идеалы. Подданные королевства связаны общей судьбой. Они могут умножаться на подданных других королевств, складываться с полиномами из своего и всё равно останутся верными королю.

Пусть R — это ассоциативное коммутативное кольцо с единицей.

Подмножество $I \subseteq R$ называется **идеалом**, если:

$$\begin{aligned} 0 &\in I, \\ \forall a, b \in I \quad &(a - b \in I), \\ \forall r \in R, \forall a \in I \quad &(ra \in I). \end{aligned}$$

Идеал, **порождённый** множеством $S \subseteq R$, обозначается (S) и определяется как:

$$(S) = \left\{ \sum_{i=1}^n r_i s_i \mid n \in \mathbb{N}, r_i \in R, s_i \in S \right\}.$$

Если идеал порождён (может быть порождён) одноэлементным S , то он называется главным.

ТЁМНЫЕ ВРЕМENA

Однако в королевстве настали тёмные времена, недобросовестные полиномы начали притворяться подданными. Король призвал на помощь. Как определить, кто перед нами, подданный или нарушитель?

Пусть $R = \mathbb{K}[x_1, \dots, x_n]$ — ассоциативное коммутативное кольцо с единицей. Дан порождённый идеал $I = (S)$. Нужно определить для произвольного $f \in R$, верно ли, что $f \in I$?

ПУТЕШЕСТВИЕ В МИР ПОРЯДКА. ПОИСК ВДОХНОВЕНИЯ

В вселенной полиномов царит хаос. Не может же во всех вселенных быть один хаос? Так...? Да, есть целые числа и полиномы от одной переменной. Возможно, это озарит нас.

Задача (Hard): Принадлежность идеала и его порождающий в кольцах главных идеалов

Рассмотрим два случая: кольцо целых чисел \mathbb{Z} и кольцо полиномов от одной переменной $\mathbb{K}[x]$, где \mathbb{K} — поле (например, \mathbb{Q} или \mathbb{R}).

Случай 1: \mathbb{Z} . На вход подаётся конечный список целых чисел a_1, \dots, a_k . Они порождают идеал

$$I = (a_1, \dots, a_k) \subseteq \mathbb{Z}.$$

Требуется:

- определить, является ли I главным;
- в случае утвердительного ответа, найти порождающий элемент $d \in \mathbb{Z}$, такой что $I = (d)$.

Случай 2: $\mathbb{K}[x]$. На вход подаётся конечный список полиномов $f_1(x), \dots, f_k(x) \in \mathbb{K}[x]$, порождающих идеал

$$I = (f_1, \dots, f_k) \subseteq \mathbb{K}[x].$$

Требуется:

- определить, является ли I главным;
- в случае утвердительного ответа, найти порождающий полином $d(x) \in \mathbb{K}[x]$, такой что $I = (d(x))$.

Подсказка. В голове крутится одно слово, Евкл...

Прототип класса для работы с кольцами:

```

from typing import Union, List

class RingElement:
    """Базовый класс для элементов колец: целых чисел и полиномов."""
    def __init__(self, data: Union[int, List[float]]):
        """
        data:
            - если это целое число → элемент кольца Z;
            - если это список коэффициентов [a0, a1, ..., an] → полином a0 + a1*x + ... +
                ↪ an*x^n.
        """
        self.data = data
        self.is_polynomial = isinstance(data, list)

    def __repr__(self) -> str:
        if self.is_polynomial:
            terms = [f"{c}*x^{i}" if i > 0 else str(c) for i, c in enumerate(self.data) if
                    c != 0]
            return " + ".join(terms) if terms else "0"
        return str(self.data)

    def gcd_ring_elements(elements: List[RingElement]) -> RingElement:
        """
        Возвращает порождающий главного идеала, порождённого заданными элементами.
        - Для Z: НОД целых чисел.
        - Для K[x]: НОД полиномов (с коэффициентами в поле K).
        """
        pass

```

Требования к реализации:

- Реализуйте классический алгоритм Евклида для целых чисел.
- Реализуйте алгоритм Евклида для полиномов от одной переменной (деление с остатком).
- Убедитесь, что функция `gcd_ring_elements` корректно обрабатывает оба случая.

Как нам поможет знание порождающего в главном идеале ответить на вопрос, принадлежит ли произвольный элемент кольца этому идеалу?

Подсказка. Возможно, здесь поможет определение идеала.



Expert level

Ключевые слова:

Мономиальный порядок, редукция, нормальная форма, базис Грёбнера, алгоритм Бухбергера, редуцированный базис Грёбнера

EXPERT LEVEL

А тем временем мы возвращаемся в нашу вселенную полиномов.

СУРОВАЯ РЕАЛЬНОСТЬ. ХАОС. КОНЕЦ?

Итак, что мы делали в алгоритме деления? Мы находили старший член. Старший, а что такое старший член у полинома от многих переменных, они же все такие равнозначные, но в то же время разные. Решено, нужно ввести закон, навести порядок в мире хаоса.

Пусть $R = \mathbb{K}[x_1, \dots, x_n]$. **Порядок на мономах** — это отношение \prec , обладающее свойствами:

- (1) $a \prec b, b \prec c \Rightarrow a \prec c$,
- (2) $\forall a, b, a = b$ или $a \prec b$ или $b \prec a$,
- (3) $a \prec b \Rightarrow ac \prec bc$.

Так в королевстве появился порядок. Каждый моном обрёл своё место. Примеры порядков:

- **Лексикографический порядок (lex):** $x_1^{a_1} \cdots x_n^{a_n} \succ x_1^{b_1} \cdots x_n^{b_n}$, если в первой позиции, где $a_i \neq b_i$, выполняется $a_i > b_i$.
- **Порядок по степени (grlex):** сравниваются сначала суммы степеней, а при равенстве — лексикографически.

Задача (Expert): Докажите или опровергните утверждение.

Приведённые примеры порядков — отношения порядка на мономах.

Пусть задан мономиальный порядок в кольце $\mathbb{K}[x_1, \dots, x_n]$. Для ненулевого полинома $f \in \mathbb{K}[x_1, \dots, x_n]$ обозначим:

- $\text{LM}(f)$ (**leading monomial**) — моном наибольшего порядка, входящий в f с ненулевым коэффициентом;
- $\text{LC}(f)$ (**leading coefficient**) — коэффициент при $\text{LM}(f)$;
- $\text{LT}(f) = \text{LC}(f) \cdot \text{LM}(f)$ (**leading term**) — моном $\text{LM}(f)$ вместе с его коэффициентом;
- $\text{in}(f) = \text{LT}(f)$ — альтернативное обозначение старшего члена полинома (от англ. **initial term**).

Для нулевого полинома все перечисленные величины считаются неопределенными.

Пример. Пусть $f = 3x^2y - 5xy^3 + 7$. При лексикографическом порядке с $x > y$ имеем: $\text{LM}(f) = x^2y$, $\text{LC}(f) = 3$, $\text{LT}(f) = 3x^2y$, $\text{in}(f) = 3x^2y$.

ОХОТА НА ВЕДЬМ

Но вместе с порядком пришли и подозрения. Король, опасаясь мятежа, созвал **Инквизицию**. Он дал им сакральные артефакты — **порождающие полиномы** g_1, \dots, g_m , и приказал: «Проверяйте каждого!»

Инквизиторы разработали свой ритуал — **редукцию**.

Если $\text{LM}(g_i)$ делит $\text{LT}(f)$, выполняется преобразование:

$$f \longrightarrow f - \frac{\text{LT}(f)}{\text{LT}(g_i)} g_i.$$

Ритуал повторяется, пока больше не находится ни один g_i , который способен сократить f . Иными словами, пока результат редукции не станет неизменным.

Когда редукция завершалась, полином открывал свою **нормальную форму** — сущность, очищенную от всего лишнего, навязанного извне. Но что означали результаты ритуала?

- Если редукция привела к 0, полином признавался верным подданным ($f \in I$).

- Если остался ненулевой остаток, его объявляли еретиком ($f \notin I$). У него были какие-то ценности, отличные от порождающих королевства.

«Но разве истина так проста?»

Задача (Expert): Дайте ответы на вопросы. Что же это значит. Как нам интерпретировать результаты ритуала?

- если получился 0?
- если получился не 0?
- существует ли порядочный подданный, который не сможет пройти ритуал?
- зависит ли результат ритуала от того, в каком порядке инквизиторы будут проводить ритуал?

ПРОЗРЕНИЕ. ОРДЕН ГРЁБНЕРА

Некоторые невинные полиномы были изгнаны, и в их сердцах зародилось сомнение. Они решили искать более совершенный закон — **закон внутренней гармонии**, в котором редукция не зависит от произвола инквизиторов. Так возник Орден Грёбнера.

Для идеала $I \subseteq R$ обозначим его начальный идеал, идеал старших членов:

$$\text{in } I = (\text{in } f \mid f \in I \setminus \{0\}).$$

Множество $\{g_1, \dots, g_m\}$ называется **базисом Грёбнера** идеала I , если

$$\text{in } I = (\text{in } g_1, \dots, \text{in } g_m),$$

или эквивалентно:

$$\forall f \in I, \exists i : \text{in}(g_i) \mid \text{in}(f).$$

Задача (Expert): Дайте ответ на вопрос.

Любой ли набор полиномов – базис Грёбнера? Иными словами, для любых ли $\{g_1, \dots, g_m\}$:

$$\text{in } I = (\text{in } g_1, \dots, \text{in } g_m)?$$

Кажется, это может решить проблему с невинными? Или нет?

Задача (Expert): Дайте ответы на вопросы. Что если провести ритуал с базисом Грёбнера вместо артефактов? Как теперь интерпретировать результаты?

- если получился 0?
- если получился не 0?
- существует ли порядочный подданный, который не сможет пройти ритуал?
- зависит ли результат ритуала от того, в каком порядке инквизиторы будут проводить ритуал?

НОВАЯ СЛОЖНОСТЬ

Инквизиция пользуется набором сакральных артефактов для проверки. Но не каждый набор позволяет провести корректный ритуал, как же получить набор, который будет базисом Грёбнера? Гораздо же проще обвинить, чем искать истину. Задавшись этим вопросом, в глубинах монастыря ордена разработали новый инструмент — *S-полином*, символ равенства и испытания двух артефактов.

Пусть $f_i, f_j \in R$. Тогда их *S*-полином определяется как:

$$S(f_i, f_j) = m_{ij}f_i - m_{ji}f_j,$$

где m_{ij}, m_{ji} — мономы минимальной степени такие, что

$$m_{ij} \text{in}(f_i) = m_{ji} \text{in}(f_j) = \text{LCM}(\text{LM}(f_i), \text{LM}(f_j)).$$

Если после редукции *S*-полинома получается ноль, это знак того, что f_i и f_j не противостоят.

воречат законам идеала.

Эквивалентное определение базиса Грёбнера:
 $\{g_1, \dots, g_m\}$ — базис Грёбнера \iff

$$\forall i, j \quad N(S(g_i, g_j)) = 0,$$

где $N(f)$ — нормальная форма при редукции.

ОПТИМИЗАЦИОННЫЕ ОТКРОВЕНИЯ

Члены ордена поняли, что не всякая пара требует проверки.

Если $\gcd(\text{in } f_i, \text{in } f_j) \in \mathbb{K}$, то $N(S(f_i, f_j)) = 0$.

Следовательно, если

$$\forall i \neq j, \quad \gcd(\text{in } f_i, \text{in } f_j) \in \mathbb{K},$$

то $\{f_1, \dots, f_n\}$ уже является базисом Грёбнера.

СОЗИДАНИЕ — ВОТ ПУТЬ ИСТИНЫ

«Борьба противоречий — есть развитие, и их последующее разрешение — есть переход в новое качество».

Алгоритм Бухбергера:

1. Пусть $G = \{f_1, \dots, f_m\}$.
2. Для каждой пары (f_i, f_j) вычислить $S(f_i, f_j)$.
3. Редуцировать $S(f_i, f_j)$ относительно G .
4. Если остаток $r \neq 0$, добавить r в G .
5. Повторять, пока для всех пар $N(S(f_i, f_j)) = 0$.

Так завершается ритуал просветления — когда больше нет противоречий, и каждая редукция приводит к нулю. Королевство идеалов обретает совершенный порядок.

«В тот миг, когда последний S -полином исчез в пыли редукции, все поняли: истина не в уничтожении, а в разрешении противоречий.»

ЭТО НЕ КОНЕЦ?

Орден Грёбнера выработал строгие требования к тому, как должен выглядеть «закон» – базис, достойный называться минимальным и редуцированным. Эти требования уберегают королевство от дубликатов и скрытых противоречий.

$\mathcal{G} = \{g_1, \dots, g_s\}$ - редуцированный базис Грёбнера идеала $I \subseteq \mathbb{K}[x_1, \dots, x_n]$ относительно выбранного мономиального порядка. Если:

1. Ведущие коэффициенты всех g_i равны 1 (возможно доумножением на обратимый скаляр можно установить это всегда).
2. Ведущие мономы $\text{LM}(g_i)$ попарно взаимнопросты (т.е. ни один ведущий моном не делится на другой). Более строго: ни один ведущий моном не делится на ведущий моном другого элемента базиса.
3. Все ненулевые мономы (неначальные члены) каждого g_i являются \mathcal{G} -нормальными (не делятся ни на один из ведущих мономов $\text{LM}(g_j)$, $j \neq i$).

Задача (Expert): Докажите или опровергните утверждение. Любой базис Грёбнера можно привести к редуцированному.

Редуцированный базис Грёбнера единственен с точностью до перестановки элементов.

В ТАЙНОЙ БИБЛИОТЕКЕ ОРДЕНА

Какие секреты она хранит? Вы открываете записи ордена и видите:

«В королевства линейных форм направить в помощь свиток с ритуалом Бухбергера в упрощённой форме. Назвать его ритуалом Г-са»

Какого Г-са, что это за королевства? Почему упрощённой форме? Как много вопросов и как мало ответов, кажется, время может подсказать ответ на этот вопрос.

ДЕЛА ГЕРОЯ

Задача (Expert): Реализуйте алгоритм Бухбергера с оптимизациями

Требования:

- Класс `RingElement` с хранением полинома (словарь моном \mapsto коэффициент) и поддержкой выбранного мономиального порядка.
- Функции: `leading_monomial`, `leading_term`, `normal_form`, `S_polynomial`, `buchberger`.
- Реализовать критерий взаимной простоты и возможность расширять дополнительными критериями.
- Тесты: несколько тривиальных и нетривиальных примеров (для проверки корректности).

Задача (Expert): Напишите функцию проверки полинома на верность королю

Написать функцию `is_in_ideal(f, G)`: редуцировать f по редуцированному базису \mathcal{G} и возвращать `True` если нормальная форма равна нулю, иначе `False`. Используйте нормализованный (ведущие коэффициенты = 1) \mathcal{G} .

ЗАЧЕМ ЕЩЁ НУЖЕН БАЗИС ГРЁБНЕРА (ПРАКТИЧЕСКИЕ ПРИМЕНЕНИЯ)

- Приведение системы полиномиальных уравнений к форме, удобной для последовательного решения (аналог исключения переменных).
- Вычисление размерности фактор-алгебры $\mathbb{K}[x]/I$, поиск нормальных мономов (базиса как векторного пространства), тесты, является ли идеал нулевым, вычисление пересечений/проекций идеалов и др.

Если два множества порождений задают один и тот же идеал I , то соответствующие системы уравнений имеют один и тот же набор общих корней (в алгебраически замкнутом поле), поэтому преобразование порождающих через базис Грёбнера упрощает систему без изменения множества решений.

Раз у нас есть алгоритм для последовательного построения базиса, почему бы не воспользоваться им для построения эквивалентных систем уравнений.

Задача (Expert): Решите несколько систем нелинейных алгебраических уравнений от двух и более переменных с помощью построения базиса Грёбнера. Выполните для каждой системы:

1. Выберите удобный порядок мономов (обоснуйте выбор).
2. Постройте (вручную для простых примеров) или с помощью программы базис Грёбнера.
3. Получите последовательность уравнений для отдельных переменных (исключение переменных).
4. Сравните результаты с выводом CAS (Wolfram|Alpha, Sage, Singular, SymPy): проверяйте эквивалентность идеалов и наборы корней.
5. Оцените вычислительную сложность и опишите, какие оптимизации алгоритма Бухбергера помогли бы ускорить расчёт для данной системы.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. И. В. Аржанцев, Базисы Грёбнера и системы алгебраических уравнений, МЦНМО, Москва, 2003.
2. И. Р. Шафаревич, Основные понятия алгебры, Наука, Москва, 1999.
3. S. D. Galbraith, Mathematics of Public Key Cryptography, Version 2.0, University of Auckland, 2012.
4. Д. Кокс, Дж. Литлл, Д. О'Ши, Идеалы, многообразия и алгоритмы. Введение в вычислительные аспекты алгебраической геометрии и коммутативной алгебры, Пер. с англ., Мир, 2000.