НАО «Восточно-Казахстанский технический университет имени Д. Серикбаева»

Дисциплина «Теоретико-числовые методы в криптографии»

ПРАКТИЧЕСКАЯ РАБОТА 10

СРАВНЕНИЯ ВЫСШИХ СТЕПЕНЕЙ И ПОДНЯТИЕ КОРНЕЙ

Методические указания и задание к выполнению

Тема 10 по рабочей программе: сравнения высших порядков; сведение к сравнениям по модулям степеней простых чисел; поднятие решения от простого модуля к его степеням. **Аудиторная нагрузка:** 3 академических часа, 135 минут.

1 Цель и организация работы

Цель: находить все корни многочленного сравнения по составному модулю, последовательно поднимая корни по простым степеням и объединяя полученные множества китайской теоремой об остатках.

Планируемые результаты: студент объясняет формулу одного шага поднятия; различает единственное продолжение, ветвление и обрыв ветви; сохраняет все найденные корни; проверяет полноту ответа независимым перебором.

Подготовка: сравнения первой степени, КТО, формальная производная многочлена. Порядок работы: сначала разделы 2–4 с методикой, ручными примерами и программой, затем индивидуальные задания и таблица вариантов.

Средства: Python 3.10 или новее; дополнительные пакеты для приведённого кода не требуются. Все коэффициенты, модули и показатели — целые числа. Малые модули выбраны для ручного разбора и полной программной проверки.

Распределение времени: 25 минут — теория и разбор примеров; 35 минут — ручное поднятие корней; 35 минут — программа и КТО; 25 минут — проверка и оформление; 15 минут — защита. Код рекомендуется заранее перенести в рабочий файл.

2 Методические сведения

2.1 Многочленное сравнение и локальные задачи

Пусть $f(x) = a_0 + a_1x + \cdots + a_dx^d \in \mathbb{Z}[x]$. Решить $f(x) \equiv 0 \pmod m$ — значит найти все классы x по модулю m, для которых $m \mid f(x)$. Если $x \equiv y \pmod m$, то $f(x) \equiv f(y) \pmod m$, поэтому условие зависит только от класса.

Ненулевой многочлен степени d над полем имеет не более d различных корней. При переходе от целых коэффициентов к $\mathbb{F}_p$ степень может уменьшиться. Если все коэффициенты делятся на p, получается нулевой многочлен, и корнями являются все p классов. По составному модулю ограничение «не более степени» также неприменимо: $x^2 - 1$ имеет четыре корня по модулю 8.

Разложим $m = p_1^{e_1} \cdots p_r^{e_r}$ на степени различных простых. Тогда

$$f(x) \equiv 0 \pmod m \quad \Leftrightarrow \quad f(x) \equiv 0 \pmod{p_i^{e_i}} \quad \text{для всех } i.$$

Причина: попарно взаимно простые простые степени одновременно делят число тогда и только тогда, когда его делит их произведение. По каждому локальному модулю находят полное множество корней. Затем для каждой комбинации локальных корней решают систему КТО. Если число локальных корней равно r_i, всего получится $\prod_i r_i$ корней по модулю m.

Решения только по модулям p_i не заменяют решения по $p_i^{e_i}$. Например, 0 — корень $x^2 - 3$ по модулю 3, но корней по модулю 9 у этого многочлена нет.

2.2 Один шаг поднятия

Зафиксируем простое p и целое $k \geq 1$. Пусть a — представитель корня по модулю p^k, причём $0 \leq a < p^k$. Все классы по модулю p^{k+1}, переходящие в a при уменьшении модуля, имеют вид

$$b = a + tp^k, \qquad t = 0,1, \dots, p-1.$$

Никаких других кандидатов нет: разность $b - a$ должна быть кратна p^k, а указанные p значений дают ровно все различные продолжения этого класса.

Вывод формулы. Для целочисленного многочлена из бинома Ньютона следует тождество $f(a + h) = f(a) + hf'(a) + h^2Q(a, h)$, где Q также имеет целые коэффициенты. Подставим $h = tp^k$. Так как $2k \geq k + 1$, последний член делится на p^{k+1}, и потому

$$f(a + tp^k) \equiv f(a) + tp^k f'(a) \pmod{p^{k+1}}.$$

По предположению $p^k \mid f(a)$, поэтому $c = f(a)/p^k$ — целое число. Условие для следующего корня сводится к линейному сравнению

$$c + tf'(a) \equiv 0 \pmod{p}.$$

Деление на p^k здесь выполняется в целых числах вместе с уменьшением модуля. Это не умножение на обратный к p^k: такого обратного по модулю p^{k+1} нет. В вычислениях достаточно хранить остатки c и $d = f'(a)$ по модулю p.

2.3 Три возможных случая

Таблица 1. Продолжения одного корня с уровня p^k на уровень p^{k+1}.

Условие по модулю p	Допустимые t	Число продолжений
$d \neq 0$	$t \equiv -cd^{-1} \pmod{p}$	1
$d = 0, c = 0$	Все $0,1, \dots, p-1$	p
$d = 0, c \neq 0$	Нет	0

Простой корень — корень, для которого $f'(a) \not\equiv 0 \pmod{p}$. Он имеет единственное поднятие на каждый следующий уровень. Действительно, у любого потомка $b \equiv a \pmod{p}$, поэтому $f'(b) \equiv f'(a) \not\equiv 0 \pmod{p}$. По индукции на каждом шаге применим первый случай. Это конечная форма леммы Гензеля о простом корне.

Особый, или кратный, корень удовлетворяет $f'(a) \equiv 0 \pmod{p}$. Он может дать p продолжений либо не дать ни одного. Даже если на одном шаге ветвление произошло, на следующем некоторые ветви могут оборваться. Поэтому следует обрабатывать все текущие корни, а не выбирать одну удачную цепочку.

Вывод действует и для $p = 2$: в нём нет деления на 2! или другие факториалы. У многочлена $x^2 - A$ производная $2a$ чётна, поэтому необходимо проверять оба особых случая. Базовая лемма о простом корне и её доказательство изложены также в конспекте Keith Conrad, Hensel's Lemma, §2. Здесь используется только арифметика конечных модулей; введение p-адических чисел не требуется.

2.4 Почему алгоритм находит все корни

Обозначим через R_k множество представителей корней в диапазоне $0 \leq a < p^k$. Сначала R_1 получают перебором $0, \dots, p-1$. Для каждого $a \in R_k$ решают линейное сравнение относительно t и объединяют все продолжения в R_{k+1}.

Любой корень по модулю p^{k+1} при сокращении модуля даёт корень из R_k, а затем единственным образом записывается как $a + tp^k$ с указанными границами. Следовательно, ни один корень не пропущен. Разные родители не могут иметь одного потомка: остаток потомка по модулю p^k однозначен. Если R_k пусто, все следующие множества также пусты.

3 Разобранные примеры

3.1 Единственное поднятие каждого простого корня

Решим $x^2 \equiv 2 \pmod{343}$, где $343 = 7^3$. По модулю 7 корни равны 3 и 4; производная $2x$ в обоих случаях ненулевая.

Для $a = 3$ на первом шаге имеем $c = (3^2 - 2)/7 = 1$ и $d = 6$. Из $1 + 6t \equiv 0 \pmod 7$ получаем $t = 1$, поэтому новый корень равен $3 + 7 = 10$. Далее $c = (10^2 - 2)/49 = 2$ и $d \equiv 20 \equiv 6 \pmod 7$. Следовательно, $t \equiv -2 \cdot 6 \equiv 2 \pmod 7$ и следующий корень равен $10 + 2 \cdot 49 = 108$.

Таблица 2. Все шаги поднятия; c и d приведены к остаткам по модулю 7.

k	a	c	d	t	$a + t7^k$
1	3	1	6	1	10
1	4	2	1	5	39
2	10	2	6	2	108
2	39	3	1	4	235

Ответ: $R_1 = \{3,4\}$, $R_2 = \{10,39\}$, $R_3 = \{108,235\}$. Проверка: $108^2 - 2 = 34 \cdot 343$, $235^2 - 2 = 161 \cdot 343$. Других корней нет, поскольку оба базовых корня простые и имеют ровно по одной цепочке.

3.2 Ветвление с последующим обрывом

Решим $x^2 \equiv 9 \pmod{27}$. По модулю 3 имеется только корень $a = 0$. На шаге к модулю 9 получаем $c = -9/3 = -3 \equiv 0 \pmod 3$, $d = 0$. Подходят все $t = 0,1,2$, поэтому $R_2 = \{0,3,6\}$.

Теперь отдельно рассматриваем каждого родителя по модулю 9. Для $a = 0$ имеем $c = -9/9 = -1 \equiv 2 \pmod 3$, $d = 0$: ветвь обрывается. Для $a = 3$ имеем $c = 0$; для $a = 6$ — $c = 27/9 = 3 \equiv 0 \pmod 3$. В обоих случаях $d = 0$, поэтому каждый корень даёт три продолжения:

$$3 \mapsto \{3,12,21\}, \qquad 6 \mapsto \{6,15,24\}.$$

Итак, по модулю 27 получаем шесть корней: 3,6,12,15,21,24. Число корней здесь больше степени многочлена, поскольку кольцо вычетов по модулю 27 не является полем.

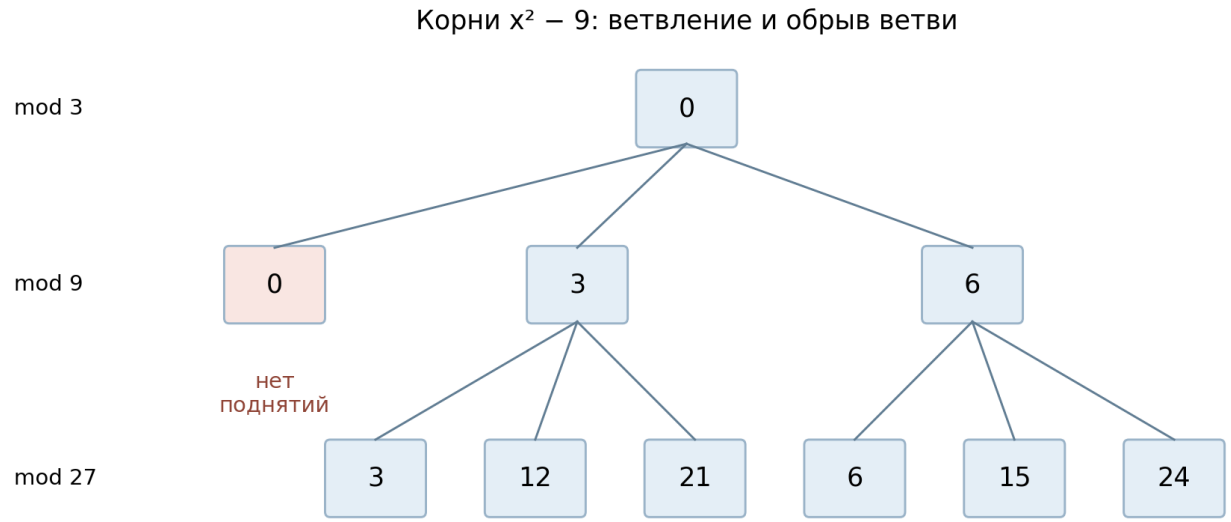


Рисунок 1. Корень 0 по модулю 9 не продолжается; корни 3 и 6 дают по три потомка.

3.3 Корень без продолжений

Для $x^2 \equiv 3 \pmod 9$ базовый корень по модулю 3 равен 0. Однако $c = (0^2 - 3)/3 = -1 \not\equiv 0 \pmod 3$, а $d = 0$. Продолжений нет. Прямая проверка подтверждает: квадрат кратного 3 делится на 9 и не может давать остаток 3.

3.4 Кубическое сравнение

Рассмотрим $f(x) = x^3 - 2$ по модулю $125 = 5^3$. По модулю 5 единственный корень — 3. Производная $f'(3) = 27 \equiv 2 \pmod 5$ ненулевая. При $k = 1$ имеем $c = 25/5 = 5 \equiv 0$, поэтому $t = 0$ и корень по модулю 25 остаётся равным 3.

При $k = 2$ получаем $c = 25/25 = 1$ и $d = 2$. Из $1 + 2t \equiv 0 \pmod 5$ следует $t = 2$, значит, $x = 3 + 2 \cdot 25 = 53$. Проверка: $53^3 - 2 = 148875 = 1191 \cdot 125$. Корень единственный.

3.5 Степень двойки и составной модуль

Найдём все корни $x^2 \equiv 1 \pmod{72}$. Разложение $72 = 8 \cdot 9$ приводит к двум локальным задачам. По степеням двойки получаем $\{1\}$ по модулю 2, $\{1,3\}$ по модулю 4 и $\{1,3,5,7\}$ по модулю 8. По модулю 9 корни равны 1 и 8. Следовательно, КТО должна дать $4 \cdot 2 = 8$ корней.

Для пары $a \pmod 8$, $b \pmod 9$ полагаем $x = a + 8t$. Поскольку $8^{-1} \equiv 8 \pmod 9$, имеем $t \equiv 8(b - a) \pmod 9$. Например, пара $a = 3$, $b = 8$ даёт $t \equiv 40 \equiv 4 \pmod 9$, откуда $x = 35$.

После обработки всех восьми пар и сортировки получаем

$$x \in \{1,17,19,35,37,53,55,71\} \pmod{72}.$$

Поднятие корней $x^2 - 1$ от модуля 8 к модулю 16 дополнительно показывает, что число корней не обязано расти: 1 и 7 дают по два продолжения, а 3 и 5 не продолжаются. По модулю 16 остаются четыре корня 1,7,9,15.

4 Программные примеры до индивидуальных заданий

Перенесите следующие блоки в один файл в указанном порядке. Коэффициенты задаются **от свободного члена к старшему**: кортеж `(-2, 0, 1)` означает $x^2 - 2$, а `(-2, 0, 0, 1)` — $x^3 - 2$. Нулевые промежуточные коэффициенты пропускать нельзя.

4.1 Значение, производная и проверка простого модуля

Схема Горнера вычисляет значение без отдельного построения каждой степени. Производная константы представляется пустым кортежем; функция `value` правильно возвращает для него 0. Проверка простоты использует только целочисленные действия.

```
def value(coeffs, x):
    result = 0
    for c in reversed(coeffs):
        result = result * x + c
    return result


def derivative(coeffs):
    return tuple(i * coeffs[i] for i in range(1, len(coeffs)))


def is_prime(p):
    if p < 2:
        return False
    from math import isqrt
    return all(p % d for d in range(2, isqrt(p) + 1))
```

4.2 Обработка всех родителей на одном шаге

Функция `lift_step` вызывается с простым p, целым $k \geq 1$ и полным списком представителей корней в диапазоне $[0, p^k)$. Проверку простоты выполняет внешняя функция из следующего блока. Выражение `fa // modulus` используется после проверки делимости, поэтому частное точное даже для отрицательного `fa`.

```
def lift_step(coeffs, p, k, roots):
    modulus = p**k
    deriv = derivative(coeffs)
    children = []
    for a in roots:
        fa = value(coeffs, a)
        if fa % modulus:
            raise ValueError("В списке есть число, не являющееся корнем")
        c, d = (fa // modulus) % p, value(deriv, a) % p
        if d != 0:
            digits = [(-c * pow(d, -1, p)) % p]
        else:
            digits = range(p) if c == 0 else []
        children.extend(a + t * modulus for t in digits)
    return sorted(set(children))
```

4.3 Полные множества по всем степеням простого

Результат — список уровней: элемент с индексом 0 содержит корни по модулю p, с индексом 1 — по модулю p^2 и так далее. Пустой уровень сохраняется в результате, чтобы обрыв был явно виден.

```
def prime_power_roots(coeffs, p, e):
    if not is_prime(p) or e < 1:
        raise ValueError("Нужны простое p и целое e >= 1")
    roots = [a for a in range(p) if value(coeffs, a) % p == 0]
    levels = [roots]
    for k in range(1, e):
        roots = lift_step(coeffs, p, k, roots)
        levels.append(roots)
    return levels
```

4.4 Разложение небольшого модуля

Пробное деление возвращает пары `(p, e)` для разложения $m = \prod p^e$. Этот простой алгоритм предназначен для учебных модулей; его трудоёмкость не следует переносить на большие криптографические числа. Во время цикла уже найденные делители полностью удаляются, поэтому каждый добавленный делитель прост.

```
def factor_small(m):
    if m < 2:
        raise ValueError("Нужен модуль m >= 2")
    factors, d = [], 2
    while d * d <= m:
        e = 0
        while m % d == 0:
            m //= d
            e += 1
        if e:
            factors.append((d, e))
        d += 1
    if m > 1:
        factors.append((m, 1))
    return factors
```

4.5 КТО для множеств корней

Для взаимно простых m, n каждая пара $a \pmod m$, $b \pmod n$ даёт единственный класс $a + m((b - a)m^{-1} \bmod n)$ по модулю mn. Функция `combine_sets` перебирает все пары, нормализует и сортирует результат. Начальная пара «множество `[0]`, модуль 1» означает отсутствие ограничений. Если один из наборов пуст, итоговый набор тоже пуст.

```
def combine_sets(left, m, right, n):
    from math import gcd
    if m < 1 or n < 2 or gcd(m, n) != 1:
        raise ValueError("Нужны m >= 1, n >= 2 и НОД(m, n) = 1")
    u = pow(m, -1, n)
    return sorted({(a + m * ((b - a) * u % n)) % (m * n)
                   for a in left for b in right})


def roots_mod(coeffs, m):
    result, modulus = [0], 1
    for p, e in factor_small(m):
        local = prime_power_roots(coeffs, p, e)[-1]
        result = combine_sets(result, modulus, local, p**e)
        modulus *= p**e
    return result
```

4.6 Запуск примеров и ожидаемый результат

```
print("Простой корень:", prime_power_roots((-2, 0, 1), 7, 3))
print("Ветвление:", prime_power_roots((-9, 0, 1), 3, 3))
print("Обрыв:", prime_power_roots((-3, 0, 1), 3, 2))
print("Кубическое:", prime_power_roots((-2, 0, 0, 1), 5, 3))
print("Модуль 72:", roots_mod((-1, 0, 1), 72))
```

Ожидаемый вывод

```
Простой корень: [[3, 4], [10, 39], [108, 235]]
Ветвление: [[0], [0, 3, 6], [3, 6, 12, 15, 21, 24]]
Обрыв: [[0], []]
Кубическое: [[3], [3], [53]]
Модуль 72: [1, 17, 19, 35, 37, 53, 55, 71]
```

4.7 Независимая проверка

Проверка каждого выданного числа подтверждает только корректность найденных корней. Чтобы подтвердить **полноту**, для малых модулей сравните весь результат со списком, полученным перебором всех представителей. Ниже многочлен вычисляется другим способом — через сумму степеней, что помогает обнаружить ошибку в схеме Горнера или порядке коэффициентов.

```
def brute_roots(coeffs, m):
    # Независимое вычисление через степени, без схемы Горнера.
    return [x for x in range(m)
            if sum(c * x**i for i, c in enumerate(coeffs)) % m == 0]


for coeffs, m in [((-2, 0, 1), 343), ((-9, 0, 1), 27),
                  ((-3, 0, 1), 9), ((-2, 0, 0, 1), 125),
                  ((-1, 0, 1), 72), ((0,), 12), ((1,), 12)]:
    assert roots_mod(coeffs, m) == brute_roots(coeffs, m)
print("Проверки пройдены")
```

Успешный запуск заканчивается строкой `Проверки пройдены`. Для варианта из таблицы замените кортеж коэффициентов и модуль на собственные. Перебор по большому составному модулю применяют здесь как контроль, а основным способом решения остаются поднятие и КТО.

5 Индивидуальные задания

В таблице 3 заданы $f(x)$, нечётное простое p, показатель s и число B. Для каждого варианта положите

$$g(x) = x^2 - p^2 B, \qquad N = 2^s p^3.$$

Задание 1. Базовые корни. Выпишите коэффициенты f для программы и найдите все корни $f(x) \equiv 0 \pmod{p}$ прямой подстановкой. Вычислите $f'(a) \bmod p$ для каждого корня и определите его тип. В данных вариантах все корни f по нечётному простому модулю являются простыми; это нужно подтвердить вычислением.

Задание 2. Поднятие простых корней. Найдите полные множества R_1, R_2, R_3 для f по модулям p, p^2, p^3. Для наименьшего базового корня оформите вручную оба шага таблицей с колонками $k, a, c \bmod p, d \bmod p, t, b$. Остальные цепочки вычислите программно и объясните, почему они не могут ветвиться или оборваться.

Задание 3. Особые корни. Для g найдите корни по модулям p, p^2, p^3. Вручную покажите первый шаг и правило отбора продолжающихся родителей на втором. Постройте дерево либо таблицу «родитель — потомки». Укажите, какие ветви обрываются и сколько корней остаётся. Для длинных списков допустима полная параметрическая запись с точными границами параметров.

Задание 4. Составной модуль. Для исходного f, а не для g, найдите корни по всем уровням $2, 4, \ldots, 2^s$. Объедините конечное множество с корнями по модулю p^3 и получите все решения по модулю N. Если оба локальных набора непусты, вручную восстановите корень для пары из наименьших локальных представителей; остальные пары обработайте программой. Если набор пуст, укажите первый пустой уровень и докажите неразрешимость по N.

Задание 5. Полнота и проверка. Сопоставьте каждый локальный набор и полный ответ по N с независимым перебором. Проверьте диапазон, отсутствие повторов и формулу числа корней. Объясните на примере задания 3, почему нельзя сохранять только одного потомка особого корня. Приложите исходный код и фактический вывод.

6 Варианты исходных данных

Таблица 3. Во всех вариантах нечётный локальный модуль равен p^3.

№	$f(x)$	p	s	B
1	$x^2 - 49$	3	4	1
2	$x^2 - 49$	5	3	2
3	$x^2 - 65$	7	4	2
4	$x^2 - 25$	3	3	2
5	$x^2 - 19$	5	4	0
6	$x^2 - 113$	7	3	3
7	$x^2 - 97$	3	4	0
8	$x^2 - 81$	5	3	1
9	$x^2 - 177$	7	4	0
10	$x^2 - 43$	3	3	1
11	$x^3 - x + 280$	5	4	2
12	$x^3 - 2x + 344$	7	3	1
13	$x^3 - 2x + 155$	3	4	2
14	$x^3 - x + 406$	5	3	0
15	$x^3 - x + 227$	7	4	3
16	$x^3 - 2x + 196$	3	3	0
17	$x^3 + 17$	5	4	4
18	$x^3 - 2x + 512$	7	3	0
19	$x^3 - 2x + 227$	3	4	1
20	$x^3 + 134$	5	3	2

Все 20 вариантов содержат одинаковые пять заданий. Пустое множество решений в отдельном задании является допустимым результатом, который требуется обосновать.

7 Требования к отчёту и оценивание

Отчёт должен содержать номер варианта, исходные многочлены и модули; необходимые теоретические формулы; ручную таблицу двух шагов для f; описание всех ветвей для g; наборы корней по степеням двойки; полное решение по N; код, результат независимой проверки и короткий вывод о полноте.

Все корни перечисляйте в возрастающем порядке в диапазоне $[0, m)$. Если используете параметрическое семейство, отдельно укажите модуль и границы всех параметров. В выводе сравните простые и особые корни по характеру продолжения.

Таблица 4. Критерии оценивания.

Составляющая	Баллы
Базовые корни и производные для f	15
Полное поднятие f и ручная цепочка	20
Все ветви для g и объяснение обрывов	20
Корни по 2^s, КТО и полный ответ по N	20
Независимая проверка и воспроизводимый код	15
Оформление и ответы на защите	10
Всего	100

8 Типичные ошибки

1. Решение только по модулю p вместо p^3. Исправление: выписать каждый уровень и проверить делимость на его полный модуль.
2. Обращение нулевой производной. Исправление: сначала проверить $d \bmod p$, затем выбрать один из трёх случаев.
3. Замена точного частного $f(a)/p^k$ остатком $f(a) \bmod p$. Последний равен нулю у любого текущего корня и не содержит нужной информации.
4. Сохранение одного потомка при ветвлении. Исправление: обработать все $t = 0, \dots, p-1$, когда $c = d = 0$.
5. Одна система КТО вместо всех комбинаций корней. Исправление: использовать декартово произведение локальных наборов.
6. Проверка только подстановкой найденных корней. Исправление: дополнительно обосновать полноту и сравнить полные множества с перебором.

9 Контрольные вопросы

1. Почему многочлен можно вычислять на классах вычетов?
2. Когда число корней ограничено степенью многочлена?
3. Зачем разлагать составной модуль именно на простые степени?
4. Почему все кандидаты имеют вид $a + tp^k$ с $0 \le t < p$?
5. Как из бинома Ньютона выводится формула поднятия?
6. Почему частное $f(a)/p^k$ целое и что означает его сокращение?
7. Как вычислить единственное продолжение при ненулевой производной?
8. Когда особый корень имеет p продолжений, а когда ни одного?
9. Почему простой корень остаётся простым на всех следующих уровнях?
10. Что меняется при $p = 2$ и применима ли общая формула?
11. Может ли число корней уменьшиться при увеличении степени модуля?
12. Почему разные родители не могут иметь одного общего потомка?
13. Как КТО связывает количество локальных и глобальных корней?
14. Чем проверка корректности найденных корней отличается от проверки полноты?

Литература

1. Мартынов Л. М. **Алгебра и теория чисел для криптографии**. 2020. §6.4: производная и корни многочленов; §7.2–7.3: линейные сравнения, системы и переход к сравнениям по простому модулю.
2. Trappe W., Washington L. C. **Introduction to Cryptography with Coding Theory**. 2nd ed. 2006. §3.4: китайская теорема об остатках.
3. Conrad K. **Hensel's Lemma**. Авторский конспект, §2, теорема 2.1 и её доказательство. Полный текст. Дата обращения: 16.09.2026. Дополнительное чтение о простом корне.

Методика, числовые примеры, код и варианты подготовлены для данной работы. Ответы преподавателю оформлены отдельным документом.