ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ»

Факультет физико-математических и естественных наук

Кафедра теории вероятностей и кибербезопасности

Лабораторная работа № 8

Целочисленная арифметика многократной точности

Студент:	Ван Яо
Группа:	НФИмд-01-25

МОСКВА

2025 г.

Цель работы

1. Изучить теоретические основы арифметических операций с целыми числами многократной точности.
2. Реализовать программно алгоритмы сложения, вычитания, умножения и деления больших целых чисел.
3. Провести тестирование реализованных алгоритмов на различных примерах. ## Теоретическая часть

1.Представление больших целых чисел

Большие целые числа, превосходящие стандартные типы данных компьютера, представляются в виде массивов цифр в b-ичной системе счисления, где $b \geq 2$ — основание системы. Число u записывается как:

$$u = u_1 u_2 \cdots u_n$$

где u_i — цифры числа, u_1 — старший разряд, u_n — младший разряд.

Для удобства реализации знак числа хранится отдельно, а все арифметические операции выполняются над абсолютными значениями чисел.

2. Алгоритмы арифметических операций

2.1 Алгоритм сложения (Алгоритм 1) **Вход:** Два неотрицательных числа $u = u_1 u_2 ... u_n$ и $v = v_1 v_2 ... v_n$, основание системы счисления b.

Выход: Сумма $w = w_0 w_1 ... w_n$, где w_0 — цифра переноса (0 или 1).

Алгоритм:

1. Инициализация: $j := n$, $k := 0$ (j — индекс разряда, k — перенос).
2. Вычисление: $w_j := (u_j + v_j + k) \mod b$, $k := \lfloor (u_j + v_j + k)/b \rfloor$.
3. Уменьшение j на 1. Если $j > 0$, вернуться на шаг 2.
4. Присвоить $w_0 := k$.

2.2 Алгоритм вычитания (Алгоритм 2) **Вход:** Два неотрицательных числа $u = u_1u_2...u_n$ и $v = v_1v_2...v_n$, где $u \geq v$.

Выход: Разность $w = w_1w_2...w_n = u - v$.

Алгоритм:

1. Инициализация: $j := n$, $k := 0$ (k — заём).
2. Вычисление: $w_j := (u_j - v_j + k) \mod b$, $k := \lfloor (u_j - v_j + k)/b \rfloor$.
3. Уменьшение j на 1. Если $j > 0$, вернуться на шаг 2.

2.3 Алгоритм умножения столбиком (Алгоритм 3) **Вход:** Числа $u = u_1u_2...u_n$, $v = v_1v_2...v_m$, основание системы b.

Выход: Произведение $w = w_1w_2...w_{m+n}$.

Алгоритм:

1. Инициализация: установить $w_{m+1} := 0, ..., w_{m+n} := 0$, $j := m$.
2. Если $v_j = 0$, установить $w_j := 0$, перейти к шагу 6.
3. Установить $i := n$, $k := 0$.
4. Вычислить $t := u_i \cdot v_j + w_{i+j} + k$, $w_{i+j} := t \mod b$, $k := \lfloor t/b \rfloor$.
5. Уменьшить i на 1. Если $i > 0$, вернуться на шаг 4, иначе $w_j := k$.
6. Уменьшить j на 1. Если $j > 0$, вернуться на шаг 2.

2.4 Алгоритм быстрого умножения (Алгоритм 4) **Вход:** Числа $u = u_1u_2...u_n$, $v = v_1v_2...v_m$, основание системы b.

Выход: Произведение $w = w_1w_2...w_{m+n}$.

Алгоритм:

1. Установить $t := 0$.
2. Для s от 0 до $m + n - 1$:
   - Для i от 0 до s: $t := t + u_{n-i} \cdot v_{m-s+i}$.
   - $w_{m+n-s} := t \mod b$, $t := \lfloor t/b \rfloor$.

2.5 Алгоритм деления (Алгоритм 5) **Вход:** Числа $u = u_n...u_1u_0$, $v = v_t...v_1v_0$, где $n \geq t \geq 1$, $v_t \neq 0$.

Выход: Частное $q = q_{n-t}...q_0$, остаток $r = r_t...r_0$.

Алгоритм:

1. Инициализация: для j от 0 до $n - t$ установить $q_j := 0$.
2. Пока $u \geq v \cdot b^{n-t}$, выполнять: $q_{n-t} := q_{n-t} + 1$, $u := u - v \cdot b^{n-t}$.

3. Для $i = n, n-1, ..., t+1$:
   - Если $u_i \geq v_t$, то $q_{i-t-1} := b-1$, иначе $q_{i-t-1} := \lfloor (u_i \cdot b + u_{i-1})/v_t \rfloor$.
   - Пока $q_{i-t-1} \cdot (v_t \cdot b + v_{t-1}) > u_i \cdot b^2 + u_{i-1} \cdot b + u_{i-2}$, уменьшать q_{i-t-1} на 1.
   - $u := u - q_{i-t-1} \cdot b^{i-t-1} \cdot v$.
   - Если $u < 0$, то $u := u + v \cdot b^{i-t-1}$, $q_{i-t-1} := q_{i-t-1} - 1$.
4. Результат: $r := u$.

Практическая реализация

Пример кода

1.Вспомогательные функции

2.Алгоритм вычитания

3. Алгоритм вычитания

4. Алгоритм умножения столбиком

5. Алгоритм быстрого умножения

6. Алгоритм деления

```python
def compare(u_digits, v_digits):

    while len(u_digits) > 1 and u_digits[-1] == 0:
        u_digits.pop()
    while len(v_digits) > 1 and v_digits[-1] == 0:
        v_digits.pop()
    if len(u_digits) != len(v_digits):
        return len(u_digits) - len(v_digits)
    for i in range(len(u_digits)-1, -1, -1):
        if u_digits[i] != v_digits[i]:
            return u_digits[i] - v_digits[i]
    return 0
```

```python
def divide(u_str, v_str, base=10):
    if v_str == "0":
        raise ValueError("Division by zero")
    if u_str == "0":
        return "0", "0"

    u = str_to_digits(u_str, base)[::-1]
    v = str_to_digits(v_str, base)[::-1]


    while len(u) > 1 and u[0] == 0:
        u.pop(0)
    while len(v) > 1 and v[0] == 0:
        v.pop(0)

    n = len(u)
    t = len(v)
    q = [0] * (n - t + 1)
    r = u[:]

    v_shifted = v + [0] * (n - t)


    def ge(a, b):
        # a, b
        a = a[:]
        b = b[:]
        while len(a) > 1 and a[0] == 0: a.pop(0)
        while len(b) > 1 and b[0] == 0: b.pop(0)
        if len(a) != len(b):
            return len(a) > len(b)
        return a >= b


    def sub_highfirst(a, b):
        # a >= b
        a = a[:]
        b = b[:]

        b = [0]*(len(a)-len(b)) + b
        borrow = 0
        res = [0]*len(a)
        for i in range(len(a)-1, -1, -1):
            val = a[i] - borrow - b[i]
            if val < 0:
                val += base
                borrow = 1
            else:
                borrow = 0
            res[i] = val
        return res


    for i in range(n - t + 1):

        q_i = 0
        v_current = v + [0] * (n - t - i)  # v * base^(n-t-i)
        while ge(r, v_current):
            r = sub_highfirst(r, v_current)
            q_i += 1
        q[i] = q_i

        while len(r) > 1 and r[0] == 0:
            r.pop(0)


    while len(q) > 1 and q[0] == 0:
        q.pop(0)
    while len(r) > 1 and r[0] == 0:
        r.pop(0)

    q_str = ''.join(str(x) for x in q)
    r_str = ''.join(str(x) for x in r)
    return q_str, r_str
```

тестирование

Выводы

1. **Теоретические знания:** В ходе лабораторной работы изучены теоретические основы арифметических операций с целыми числами многократной точности. Освоены различные алгоритмы для выполнения операций сложения, вычитания, умножения и деления больших чисел, представленных в b-ичной системе счисления.
2. **Практические навыки:** Успешно реализованы все пять алгоритмов на языке Python:
    - Алгоритм 1: сложение неотрицательных целых чисел
    - Алгоритм 2: вычитание неотрицательных целых чисел
    - Алгоритм 3: умножение неотрицательных целых чисел столбиком
    - Алгоритм 4: быстрый метод умножения
    - Алгоритм 5: деление многоразрядных целых чисел
3. **Результаты тестирования:** Проведено тестирование всех реализованных алгоритмов на различных примерах. Все алгоритмы работают корректно и выдают ожидаемые результаты. Программа успешно обрабатывает как простые, так и сложные случаи, включая операции с переносами, займами и различными основаниями систем счисления.
4. **Применение и значимость:** Полученные навыки имеют важное практическое значение в криптографии, компьютерной алгебре и других областях, где требуется работа с числами, превышающими стандартные типы данных. Алгоритмы многократной точности являются фундаментальными для реализации современных криптографических протоколов и систем компьютерной алгебры.