

Отчёт по лабораторной работе №8

Ван Яо

Содержание

Цель работы	5
Ход лабораторной работы	6
Теоретические основы	6
1.Представление больших целых чисел	6
2. Алгоритмы арифметических операций	6
Практическая реализация	9
Пример кода	9
Выводы	16
Литература	17
Приложения	18

Список таблиц

Список иллюстраций

1	Вспомогательные функции для работы с большими числами	9
2	Реализация алгоритма сложения	9
3	Реализация алгоритма вычитания	10
4	Реализация умножения столбиком	11
5	Реализация быстрого умножения	12
6	Основная часть алгоритма деления	13
7	Дополнительные функции для деления	14
8	Тестирование всех арифметических операций	15

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.

Ход лабораторной работы

Теоретические основы

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:

$$u = u_1 u_2 \dots u_n$$

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 $u = u_1 u_2 \dots u_n$ и $v = v_1 v_2 \dots v_n$, основание системы счисления b .

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

Алгоритм:

1. Инициализация: $j := n, k := 0$ (j — индекс разряда, k — перенос).
2. Вычисление: $w_j := (u_j + v_j + k) \bmod b, k := \lfloor (u_j + v_j + k)/b \rfloor$.

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 $u = u_1 u_2 \dots u_n$ и $v = v_1 v_2 \dots v_n$, где $u \geq v$.

Выход: Разность $w = w_1 w_2 \dots w_n = u - v$.

Алгоритм:

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

2.3 Алгоритм умножения столбиком (Алгоритм 3)

Вход: Числа $u = u_1 u_2 \dots u_n, v = v_1 v_2 \dots v_m$, основание системы b .

Выход: Произведение $w = w_1 w_2 \dots w_{m+n}$.

Алгоритм:

1. Инициализация: установить $w_{m+1} := 0, \dots, w_{m+n} := 0, j := m$.
2. Если $v_j = 0$, установить $w_j := 0$, перейти к шагу 6.
3. Установить $i := n, k := 0$.
4. Вычислить $t := u_i \cdot v_j + w_{i+j} + k, w_{i+j} := t \bmod b, k := \lfloor t/b \rfloor$.
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 $u = u_1 u_2 \dots u_n, v = v_1 v_2 \dots v_m$, основание системы b .

Выход: Произведение $w = w_1 w_2 \dots w_{m+n}$.

Алгоритм:

1. Установить $t := 0$.

2. Для s от 0 до $m + n - 1$:

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

2.5 Алгоритм деления (Алгоритм 5)

Вход: Числа $u = u_n \dots u_1 u_0, v = v_t \dots v_1 v_0$, где $n \geq t \geq 1, v_t \neq 0$.

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

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 $i = n, n - 1, \dots, t + 1$:
 - 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

```
def str_to_digits(s, base=10):  
    return [int(ch) for ch in s[::-1]]  
  
def digits_to_str(digits, base=10):  
    if not digits:  
        return "0"  
  
    while len(digits) > 1 and digits[-1] == 0:  
        digits.pop()  
    return ''.join(str(d) for d in digits[::-1])
```

Рис. 1: Вспомогательные функции для работы с большими числами

2.Алгоритм вычитания

```
def add(u_str, v_str, base=10):  
    u = str_to_digits(u_str or "0", base)  
    v = str_to_digits(v_str or "0", base)  
    n = max(len(u), len(v))  
    w = []  
    carry = 0  
    for i in range(n):  
        digit_u = u[i] if i < len(u) else 0  
        digit_v = v[i] if i < len(v) else 0  
        total = digit_u + digit_v + carry  
        w.append(total % base)  
        carry = total // base  
    if carry:  
        w.append(carry)  
    return digits_to_str(w, base)
```

Рис. 2: Реализация алгоритма сложения

3. Алгоритм вычитания

```
def subtract(u_str, v_str, base=10):
    if u_str == v_str:
        return "0"
    u = str_to_digits(u_str, base)
    v = str_to_digits(v_str, base)

    w = []
    borrow = 0
    for i in range(len(u)):
        digit_u = u[i] - borrow
        digit_v = v[i] if i < len(v) else 0
        borrow = 0
        if digit_u < digit_v:
            digit_u += base
            borrow = 1
        w.append(digit_u - digit_v)
    return digits_to_str(w, base)
```

Рис. 3: Реализация алгоритма вычитания

4. Алгоритм умножения столбиком

```
def multiply_schoolbook(u_str, v_str, base=10):
    if u_str == "0" or v_str == "0":
        return "0"
    u = str_to_digits(u_str, base)
    v = str_to_digits(v_str, base)

    w = [0] * (len(u) + len(v))

    for j in range(len(v)):
        if v[j] == 0:
            continue
        carry = 0
        for i in range(len(u)):
            temp = u[i] * v[j] + w[i + j] + carry
            w[i + j] = temp % base
            carry = temp // base
        if carry:
            w[len(u) + j] += carry
    return digits_to_str(w, base)
```

Рис. 4: Реализация умножения столбиком

5. Алгоритм быстрого умножения

```
def multiply_fast(u_str, v_str, base=10):
    if u_str == "0" or v_str == "0":
        return "0"
    u = str_to_digits(u_str, base)
    v = str_to_digits(v_str, base)
    m, n = len(u), len(v)
    w = [0] * (m + n)
    total_carries = 0

    for s in range(m + n - 1):
        total = 0

        for i in range(max(0, s - n + 1), min(s + 1, m)):
            j = s - i
            if j < n:
                total += u[i] * v[j]
        total += w[s]
        w[s] = total % base
        carry = total // base
        if carry:
            if s + 1 >= len(w):
                w.append(0)
            w[s + 1] += carry

    i = 0
    while i < len(w):
        if w[i] >= base:
            carry = w[i] // base
            w[i] %= base
            if i + 1 >= len(w):
                w.append(0)
            w[i + 1] += carry
        i += 1

    return digits_to_str(w, base)
```

Рис. 5: Реализация быстрого умножения

6. Алгоритм деления

```
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
```

Рис. 6: Основная часть алгоритма деления

```

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

```

Рис. 7: Дополнительные функции для деления

тестирование

```
if __name__ == "__main__":  
  
    print("123 + 456 =", add("123", "456"))  
  
    print("456 - 123 =", subtract("456", "123"))  
  
    print("123 * 456 =", multiply_schoolbook("123", "456"))  
  
    q, r = divide("56088", "123")  
    print("56088 ÷ 123 = ", q, "Остаток", r)
```

```
123 + 456 = 579  
456 - 123 = 333  
123 * 456 = 56088  
56088 ÷ 123 = 456 Остаток 0
```

Рис. 8: Тестирование всех арифметических операций

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.

Литература

1. Кнут Д. Искусство программирования. Том 2. Получисленные алгоритмы. М.: Вильямс, 2007.
2. Python Software Foundation. Официальная документация Python. <https://docs.python.org/3/>

Приложения

Полный исходный код программ доступен в репозитории GitHub:
<https://github.com/wangyao200036/cryptography-labs/tree/main/lab8>