Отчёт по лабораторной работе №8.

Целочисленная арифметика многократной точности

Коне Сирики

29 декабря 2024

Российский университет дружбы народов, Москва, Россия

Объединённый институт ядерных исследований, Дубна, Россия

Информация

Докладчик

- Коне Сирики
- Студент физмат
- профессор кафедры прикладной информатики и теории вероятностей
- Российский университет дружбы народов
- konesirisil@yandex.ru
- https://github.com/skone19

Цели и задачи работы

Целью данной лабораторной работы является ознакомление с алгоритмами целочисленной арифметики многократной точности, а также их последующая программная реализация.

Задачи: Рассмотреть и реализовать на ЯП Python:

1) Алгоритм сложения неотрицательных целых чисел;

2) Алгоритм вычитания неотрицательных целых чисел;

3-4) Алгоритм умножения неотрицательных целых чисел столбиком и быстрым столбиком;

5) Алгоритм деления многоразрядных целых чисел.

Теоретическое введение

Высокоточная (длинная) арифметика

— это операции над числами большой разрядности (многоразрядными числами), т.е. числами, разрядность которых превышает длину машинного слова универсальных процессоров общего назначения (более 128 бит).

В современных асимметричных криптосистемах в качестве ключей, как правило, используются целые числа длиной 1000 и более битов. Они представляются в виде последовательности цифр в некоторой системе счисления: $x = (x_{n-1}x_{n-2}\dots x_1x_0)_b$, где $\forall i \in [0, n-1] : 0 \leq x_i < b$.

$$k$$

$$\begin{array}{r}
u_1\; u_2\; u_3\; \dots\; u_{n-1}\; u_n \\
0\;\; 0\;\; v_1\; \dots\; v_{m-1}\; v_m \\
\hline
w_0\, w_1\, w_2\; \dots\; w_{p-1}\; w_p
\end{array}
\begin{cases} n \\ n+m \\ n-m \end{cases}$$

Сложение неотрицательных целых чисел

Вход. Два неотрицательных числа $u = u_1 u_2 \dots u_n$ и $v = v_1 v_2 \dots v_n$; разрядность чисел n; основание системы счисления b.

Выход. Сумма $w = w_0 w_1 \dots w_n$, где w_0 - цифра переноса, всегда равная 0 либо 1.

1. Присвоить $j := n, k := 0$ (*j идет по разрядам, k следит за переносом*).
2. Присвоить $w_j = (u_j + v_j + k) \pmod b$, где $k = \left[\frac{u_j+v_j+k}{b}\right]$.
3. Присвоить $j := j-1$. Если $j > 0$, то возвращаемся на шаг 2; если $j = 0$, то присвоить $w_0 := k$ и результат: w.

Вычитание неотрицательных целых чисел

Вход. Два неотрицательных числа $u = u_1u_2 \dots u_n$ и $v = v_1v_2 \dots v_n$, $u > v$; разрядность чисел n; основание системы счисления b.

Выход. Разность $w = w_0w_1 \dots w_n = u - v$.

1. Присвоить $j := n$, $k := 0$ (k – заём из старшего разряда).
2. Присвоить $w_j = (u_j - v_j + k) \pmod b$; $k = \left[\frac{u_j - v_j + k}{b}\right]$.
3. Присвоить $j := j - 1$. Если $j > 0$, то возвращаемся на шаг 2; если $j = 0$, то результат: w.

Рис. 3: Алгоритм вычитания неотрицательных целых чисел

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

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

1. Выполнить присвоения: $w_{m+1} := 0, w_{m+2} := 0, \dots, w_{m+n} := 0$, $j := m$ (*j перемещается по номерам разрядов числа v от младших к старшим*).
2. Если $v_j = 0$, то присвоить $w_j := 0$ и перейти на шаг 6.
3. Присвоить $i := n$, $k := 0$ (*значение i идет по номерам разрядов числа u, k отвечает за перенос*).
4. Присвоить $t := u_i \cdot v_j + w_{i+j} + k$, $w_{i+j} := t \pmod b$, $k := [\frac{t}{b}]$.
5. Присвоить $i := i - 1$. Если $i > 0$, то возвращаемся на шаг 4, иначе присвоить $w_j := k$.
6. Присвоить $j := j - 1$. Если $j > 0$, то вернуться на шаг 2. Если $j = 0$, то результат: w.

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

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

1. Присвоить $t := 0$.
2. Для s от 0 до $m + n - 1$ с шагом 1 выполнить шаги 3 и 4.
3. Для i от 0 до s с шагом 1 выполнить присвоение $t := t + u_{n-i} \cdot v_{m-s+i}$.
4. Присвоить $w_{m+n-s} := t \pmod{b}$, $t := \left[\frac{t}{b}\right]$. Результат: w.

Рис. 5: Алгоритм умножения неотрицательных целых чисел быстрым столбиком

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

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

1. Для j от 0 до $n - t$ присвоить $q_j := 0$.
2. Пока $u \geq vb^{n-t}$, выполнять: $q_{n-t} := q_{n-t} + 1$, $u := u - vb^{n-t}$.
3. Для $i = n, n-1, \dots, t+1$ выполнять пункты 3.1 – 3.4:

   3.1. если $u_i \geq v_t$, то присвоить $q_{i-t-1} := b - 1$, иначе присвоить $q_{i-t-1} := \frac{u_i b + u_{i-1}}{v_t}$.

   3.2. пока $q_{i-t-1}(v_t b + v_{t-1}) > u_i b^2 + u_{i-1} b + u_{i-2}$ выполнять $q_{i-t-1} := q_{i-t-1} - 1$.

   3.3. присвоить $u := u - q_{i-t-1} b^{i-t-1} v$.

   3.4. если $u < 0$, то присвоить $u := u + vb^{i-t-1}$, $q_{i-t-1} := q_{i-t-1} - 1$.
4. $r := u$. Результат: q и r.

Ход выполнения и результаты

Подготовка

```
import math

str2num = {chr(letter_ord) : (letter_ord - ord("A") + 10)
                for letter_ord in range(ord("A"), ord("Z") + 1)}
for digit in "0123456789":
    str2num[digit] = int(digit)
num2str = {value : key for (key, value) in str2num.items()}

def fill0(u, n, array = False):
    result = [0] * (n - len(u))
    if array:
        result.extend(u)
        return result
    return "".join([str(i) for i in result]) + u
```

Сложение неотрицательных целых чисел. Реализация

```python
def addition(u_str, v_str, b):
    u = [str2num[letter] for letter in u_str]
    v = [str2num[letter] for letter in v_str]
    if len(u) != len(v): # если разрядности чисел не совпадают..
        if len(u) < len(v): u = fill0(u, len(v), True)
        else: v = fill0(v, len(u), True)
    n = len(u); k = 0 # шаг 1
    w = [] # сумма
    for j in range(n - 1, -1, -1):                 #
        w.append(((u[j] + v[j] + k) % b))          # шаг 2-3
        k = math.floor((u[j] + v[j] + k) / b)      #
    w.append(k); w.reverse() # шаг 3
    return "".join([num2str[digit] for digit in w])
```

```
[12]
print(addition("321", "1567", 10))
print(addition("01101", "11011", 2))
print(addition("B081", "4ACD", 16))
```

```
01888
101000
0FB4E
```

Рис. 7: Примеры нахождения сумм пар чисел в разных системах счисления

```python
def subtraction(u_str, v_str, b):
    u = [str2num[letter] for letter in u_str]
    v = [str2num[letter] for letter in v_str]
    if len(u) != len(v):
        if len(u) < len(v): u = fill0(u, len(v), True)
        else: v = fill0(v, len(u), True)
    elif u < v:
        return "u должно быть больше v"
    n = len(u); w = []; k = 0 # шаг 1
    for j in range(n - 1, -1, -1):              #
        w.append(((u[j] - v[j] + k) % b))       # шаг 2-3
        k = math.floor((u[j] - v[j] + k) / b)  #
    w.reverse()
    return "".join([num2str[digit] for digit in w])
```

Вычитание неотрицательных целых чисел. Результаты

```
[14]
    print(subtraction("789", "111", 10))
    print(subtraction("11001", "01011", 2))
    print(subtraction("F630", "1AAA", 16))

... 678
    01110
    DB86
```

Рис. 8: Примеры нахождения разностей пар чисел в разных системах счисления

Умножение неотрицательных целых чисел столбиком. Реализация

```python
def multiply_column(u_str, v_str, b):
    u = [str2num[letter] for letter in u_str]
    v = [str2num[letter] for letter in v_str]
    n = len(u); m = len(v)
    w = [0] * (m + n)      # шаг 1
    for j in range(m - 1, -1, -1):
        if v[j] != 0:
            k = 0          # шаг 3
            for i in range(n - 1, -1, -1):          #
                t = u[i] * v[j] + w[i + j + 1] + k  # шаг 4
                w[i + j + 1] = t % b                #
                k = math.floor(t / b)               #
            w[j] = k                                # шаг 5
    return "".join([num2str[digit] for digit in w])  # шаг 6
```

Умножение неотрицательных целых чисел столбиком. Результаты

```
print(multiply_column("777", "1234", 10))
print(multiply_column("1101", "110001100", 2))
print(multiply_column("FD76", "3AE01A", 16))
[16]
...
0958818
1010000011100
3A4A9CFDFC
```

Рис. 9: Примеры нахождения произведения пар чисел в разных системах счисления

```python
def multiply_quick(u_str, v_str, b):
    u = [str2num[letter] for letter in u_str]
    v = [str2num[letter] for letter in v_str]
    n = len(u); m = len(v)
    w = [0] * (m + n)
    t = 0 # шаг 1
    for s in range(0, m + n): # шаг 2
        for i in range(0, s + 1):                              #
            if (0 <= n - i - 1 < n) and (0 <= m - s + i - 1 < m): # шаг 3
                t = t + u[n - i - 1] * v[m - s + i - 1]        #
        w[m + n - s - 1] = t % b #
        t = math.floor(t / b)    # шаг 4
    return "".join([num2str[digit] for digit in w])
```

Быстрый столбик. Результаты

```
print(multiply_quick("777", "1234", 10))
print(multiply_quick("1101", "110001100", 2))
print(multiply_quick("FD76", "3AE01A", 16))
```

[18]

```
0958818
1010000011100
3A4A9CFDFC
```

Рис. 10: Примеры нахождения произведения пар чисел быстрым столбиком в разных системах счисления

```python
def to10(u_str, b, array = False):
    u_array = u_str if array else [str2num[letter] for letter in u_str]
    u = 0
    for i in range(len(u_array)): u += (b**i)*u_array[len(u_array)-i-1]
    return u

def to_b(number, b, n = 1):
    (q, r) = (math.floor(number / b), number % b); w = num2str[r]
    while q >= b: (q, r) = (math.floor(q / b), q % b); w = w+num2str[r]
    if q != 0: w = w + num2str[q]
    while len(w) < n: w = w + "0"
    return w[::-1]

def trim_zero(a):
    while a[0] == '0' and len(a) > 1: a = a[1:]
```

```python
def division(u_str, v_str, b):
    u = u_str; v = v_str
    n = len(u) - 1; t = len(v) - 1 # разрядности чисел
    if v[0] == 0 or not (n >= t >= 1):
        return "Некорректные входные данные"
    q = [0] * (n - t + 1) # шаг 1
    while to10(u, b) >= to10(v, b) * (b ** (n - t)): #
        q[n - t] = q[n - t] + 1                        #
        a = to_b(b ** (n - t), b)                      # шаг 2
        a = multiply_column(v, a, b)                   #
        u = subtraction(u, a, b)                       #
        if len(u) > len(u_str):                # сохраняем начальную
            u = u[1:] if u[0] == '0' else u    # разрядность числа
    u = [str2num[letter] for letter in u]
```

Деление многоразрядных целых чисел. Реализация

```
for i in range(n, t, -1): # шаг 3
    if u[n - i] >= v[0]: q[i - t - 1] = b - 1 # шаг 3.1
    else: q[i-t-1] = math.floor((u[n-i] * b + u[n-i+1]) / v[0])
    while q[i-t-1] * (v[0] * b + v[1]) > u[n-i] * (b**2) +
                                       + u[n-i+1] * b + u[n-i+2]:
        q[i - t - 1] = q[i - t - 1] - 1
    u_10 = to10(u, b, True); v_10 = to10(v, b, True) # шаг 3.3
    a = v_10 * q[i-t-1] * (b**(i-t-1)); u_10 -= a   ##
    if u_10 < 0:                                     # шаг
        u_10 = u_10 + v_10 * (b ** (i - t - 1))      # 3.4
        q[i - t - 1] = q[i - t - 1] - 1             ##
    u = to_b(u_10, b, n + 1); u = [str2num[letter] for letter in u]
(q, r) = ("".join([num2str[digit] for digit in q]),
          "".join([num2str[digit] for digit in u]))
```

```
print(division("1000", "15", 10))
print(division("1111010111", "10010", 2)) # 983 / 18 = (54, 11)
print(division("76870", "232", 16)) # 485,488 / 562 = (863, 482)
[21]
...  ('66', '10')
     ('110110', '1011')
     ('35F', '1E2')
```

Рис. 11: Примеры нахождения частных и остатков от деления пар чисел в разных системах счисления

Заключение

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: было проведено краткое знакомство с алгоритмами целочисленной арифметики многократной точности (сложение неотрицательных целых чисел, вычитание неотрицательных целых чисел, умножение неотрицательных целых чисел столбиком и быстрым столбиком, деление многоразрядных целых чисел), после чего все пять алгоритмов были успешно реализованы на языке программирования **Python**.

Спасибо за внимание