Отчёт по лабораторной работе №8

Целочисленная арифметика многократной точности

Исмит Шаманта НФИмд-01-22

Содержание

List of Figures

1 Цель работы

Ознакомление с алгоритмами целочисленной арифметики многократной точности, а также их последующая программная реализация.

2 Теоретические сведения

Высокоточная (длинная) арифметика — это операции (базовые арифметические действия, элементарные математические функции и пр.) 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. Для задания чисел такого размера не подходит ни один стандартный целочисленный тип данных современных языков программирования. Представление чисел в формате с плавающей точкой позволяет задать очень большие числа (например, тип long double языка C++ – до 10^{5000}), но не удовлетворяет требованию абсолютной точности, характерному для криптографических приложений. Поэтому большие целые числа представляются в криптографических пакетах в виде последовательности цифр в некоторой системе счисления (обозначим основание системы счисления b): $x = (x_{n-1} x_{n-2} \ldots x_1 x_0)_b$, где $\forall i \in [0, n-1] : 0 \le x_i < b$.

Основание системы счисления b выбирается так, чтобы существовали машинные команды для работы с однозначными и двузначными числами; как правило, b равно 2^8, 2^{16} или 2^{32}.

При работе с большими целыми числами знак такого числа удобно хранить в отдельной переменной. Например, при умножении двух чисел знак произведения вычисляется отдельно.

Далее при описании алгоритмов квадратные скобки означают, что берётся целая часть числа.

2.1 Сложение неотрицательных целых чисел

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

*Выход. Сумма $w = w_0 w_1 \ldots w_n$, 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 $k = [\frac{u_j + v_j + k}{b}]$.
3. Присвоить $j = j - 1$. Если $j > 0$, то возвращаемся на шаг 2; если $j = 0$, то присвоить $w_0 = k$ и результат: w.

2.2 Вычитание неотрицательных целых чисел

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

*Выход. Разность $w = w_0 w_1 \ldots w_n = u - v$.

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

2.3 Умножение неотрицательных целых чисел столбиком

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

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

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.

2.4 Быстрый столбик

*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 $w_{m+n-s} = t \pmod b), t = [\frac{t}{b}]$. Результат: w.

2.5 Деление многоразрядных целых чисел

*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.

3 Выполнение работы

3.1 Реализация алгоритма на языке Python

```
import math
# надо ввести данные сначала
u = "12345"
v = "56789"
b = 10
n = 5
# алгоритм 1
j = n
k = 0

w = list()
for i in range(1, n+1):
    w.append(
        (int(u[n-i]) + int(v[n-i]) + k) % b
    )

    k = (int(u[n-i]) + int(v[n-i]) + k)//b
    j = j - 1
w.reverse()
print(w)
```

```
# алгоритм 2
u = "56789"
v = "12345"

j = n
k = 0
w = list()
for i in range(1, n+1):
    w.append(
        (int(u[n-i]) - int(v[n-i]) + k) % b
    )

    k = (int(u[n-i]) - int(v[n-i]) + k)//b
    j = j - 1
w.reverse()
print(w)

# алгоритм 3
u = "123456"
v = "7890"
n = 6
m = 4

w = list()
for i in range(m+n):
    w.append(0)
j = m
```

```
def step6():
    global j
    global w
    j = j - 1
    if j > 0:
        step2()
    if j == 0:
        print(w)


def step2():
    global v
    global w
    global j
    if j == m:
        j = j-1
    if int(v[j]) == 0:
        w[j] = 0
        step6()


def step4():
    global k
    global t
    global i
    if i == n:
        i = i - 1
    t = int(u[i]) * int(v[j]) + w[i + j] + k
```

```
    w[i + j] = t % b
    k = t / b


def step5():
    global i
    global w
    global j
    global k
    i = i - 1
    if i > 0:
        step4()
    else:
        w[j] = k


step2()
i = n
k = 0
t = 1
step4()
step5()
step6()
print(w)


# алгоритм 4
u4 = "12345"
n = 5
```

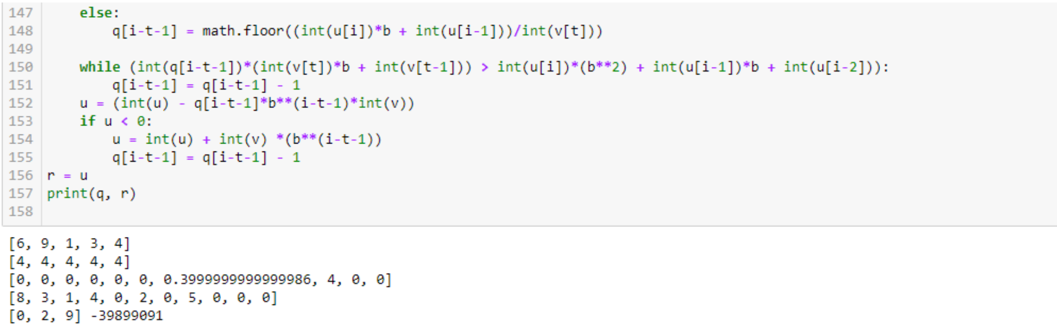
```
v4 = "6789"
m = 4
b = 10
w1 = list()
for i in range(m+n+2):
    w1.append(0)
t1 = 0
for s1 in range(0, m+n):
    for i1 in range(0, s1+1):
        if n-i1>n or m-s1+i1>m or n-i1<0 or m-s1+i1<0 or m-s1+i1-1<0:
            continue
        t1 = t1 + (int(u[n-i1-1]) * int(v[m-s1+i1-1]))
    w1[m+n-s1-1] = t1 % b
    t1 = math.floor(t1/b)
print(w1)


# алгоритм 5
u = "12346789"
n = 7
v = "56789"
t = 4
b = 10
q = list()
for j in range(n-t):
    q.append(0)
r = list()
for j in range(t):
    r.append(0)
```

```
while int(u) >= int(v)*(b**(n-t)):
    q[n-t] = q[n-t] + 1
    u = int(u) - int(v)*(b**(n-t))
u = str(u)
for i in range(n, t+1, -1):
    v = str(v)
    u = str(u)
    if int(u[i]) > int(v[t]):
        q[i-t-1] = b - 1
    else:
        q[i-t-1] = math.floor((int(u[i])*b + int(u[i-1]))/int(v[t]))

    while (int(q[i-t-1])*(int(v[t])*b + int(v[t-1])) > int(u[i])*(b**2) + int(u[i
1])*b + int(u[i-2])):
        q[i-t-1] = q[i-t-1] - 1
    u = (int(u) - q[i-t-1]*b**(i-t-1)*int(v))
    if u < 0:
        u = int(u) + int(v) *(b**(i-t-1))
        q[i-t-1] = q[i-t-1] - 1
r = u
print(q, r)
```

3.2 Контрольный пример



```
    else:
        q[i-t-1] = math.floor((int(u[i])*b + int(u[i-1]))/int(v[t]))

    while (int(q[i-t-1])*(int(v[t])*b + int(v[t-1])) > int(u[i])*(b**2) + int(u[i-1])*b + int(u[i-2])):
        q[i-t-1] = q[i-t-1] - 1
    u = (int(u) - q[i-t-1]*b**(i-t-1)*int(v))
    if u < 0:
        u = int(u) + int(v) *(b**(i-t-1))
        q[i-t-1] = q[i-t-1] - 1
r = u
print(q, r)

[6, 9, 1, 3, 4]
[4, 4, 4, 4, 4]
[0, 0, 0, 0, 0, 0, 0.3999999999999986, 4, 0, 0]
[8, 3, 1, 4, 0, 2, 0, 5, 0, 0, 0]
[0, 2, 9] -39899091
```

Figure 3.1: Работа алгоритма

4 Выводы

Изучили задачу представления больших чисел, познакомились с вычислительными алгоритмами.

Список литературы

1. Длинная арифметика от Microsoft
2. Как оперировать числами, не помещающимися ни в один из числовых типов