Отчет по лабораторной работе №8

Целочисленная арифметика многократной точности

Бурдина Ксения Павловна

19 декабря 2023

Содержание

List of Figures

1 Цель работы

Целью данной работы является освоение целочисленной арифметики многократной точности, которая применяется во многих алгоритмах криптографии.

2 Задание

1. Изучить алгоритмы целочисленной арифметики многократной точности.
2. Программно реализовать представленные алгоритмы: сложение неотрицательных целых чисел; вычитание неотрицательных целых чисел; умножение неотрицательных целых чисел столбиком; быстрый столбик; деление многоразрядных целых чисел.

3 Теоретическое введение

В данной работе рассмотрим алгоритмы для выполнения арифметических операций с большими числами. Будем считать, что число записано в b-ичной системе счисления, b - натуральное число, $b \geq 2$. Натуральное -разрядное число будем записывать в виде: $u = u_1 u_2 ... u_n$.

При работе с большими целыми числами знак такого числа удобно хранить в отдельной переменной. Например, при умножении двух чисел, знак произведения вычисляется отдельно. Квадратные скобки обозначают, что берется целая часть числа [2].

3.1 Алгоритм 1 (сложение неотрицательных целых чисел).

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

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

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

3.2 Алгоритм 2 (вычитание неотрицательных целых чисел).

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

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

- присвоить $j := n, k := 0$ (*k* - заем из старшего разряда);
- присвоить $w_j = (u_j - v_j + k)$ (*mod b*), где w_j - наименьший неотрицательный вычет в данном классе вычетов; $k = \left[\frac{u_j - v_j + k}{b}\right]$;
- присвоить $j := j - 1$. Если $j > 0$, то возвращаемся на шаг 2; если $j = 0$, то результат: w [1].

3.3 Алгоритм 3 (умножение неотрицательных целых чисел столбиком).

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

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

- выполнить присвоения: $w_{m+1} := 0, w_{m+2} := 0, ..., w_{m+n} := 0, j := m$ (*j* перемещается по номерам разрядов числа *v* от младших к старшим);
- если $v_j = 0$, то присвоить $w_j := 0$ и перейти на шаг 6;
- присвоить $i := n, k := 0$ (Значение *i* идет по номерам разрядов числа *u*, *k* отвечает за перенос);
- присвоить $i := u_i * v_j + w_{i+j} := t$ (*mod b*), $k := \frac{t}{b}$, где w_{i+j} - наименьший неотрицательный вычет в данном классе вычетов;
- присвоить $i := i - 1$. Если $i > 0$, то возвращаемся на шаг 4, иначе присвоить $w_j := k$;
- присвоить $j := j - 1$. Если $j > 0$, то вернуться на шаг 2. Если $j = 0$, то результат: *w*.

3.4 Алгоритм 4 (быстрый столбик).

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

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

- присвоить $t := 0$;
- для *s* от 0 до *m+n-1* с шагом 1 выполнить шаги 3 и 4;
- для *i* от 0 до *s* с шагом 1 выполнить присвоение $t := t + u_{n-i} * v_{m-s+i}$;
- присвоить $w_{m+n-s} := t$ (*mod b*), $t := \frac{t}{b}$, где w_{m+n-s} - наименьший неотрицательный вычет по модулю *b*. Результат: *w*.

3.5 Алгоритм 5 (деление многоразрядных целых чисел).

Вход. Числа $u = u_n, \ldots, u_1 u_0$ и $v = v_t, \ldots, v_1 v_0, n \geq t \geq 1, v_t \neq 0$; разрядность чисел соответственно *n* и *t*.

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

- для *j* от 0 до *n-t* присвоить $q_j := 0$;
- пока $u \geq vb^{n-t}$, выполнять: $q_{n-t} := qn - t + 1, u := u - vb^{n-t}$;
- для *i=n, n-1, ..., t+1* выполнять пункты:
  - если $u_i \geq v_t$, то присвоить $q_{i-t-1} := b - 1$, иначе присвоить $qi - t - 1 := \frac{u_i b + u_{i-1}}{v_t}$;
  - пока $q_{i-t-1}(v_t b + v_{t-1}) > u_i b^2 + u_{i-1} b + u_{i-2}$ выполнять $q_{i-t-1} := q_{i-t-1} - 1$;
  - присвоить $u := u - q_{i-t-1} b^{i-t-1} v$;
  - если $u < 0$, то присвоить $u := u + vb^{i-t-1}, q_{i-t-1} := q_{i-t-1} - 1$.
- $r := u$. Результат: *q* и *r*.

4 Ход выполнения лабораторной работы

Для реализации рассмотренных алгоритмов будем использовать среду JupyterLab. Выполним необходимую задачу.

1. Реализация алгоритма 1 - сложение неотрицательных целых чисел:

```
import math
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)
```

```
[6, 9, 1, 3, 4]
```

Figure 4.1: Алгоритм 1

2. Реализация алгоритма 2 - вычитание неотрицательных целых чисел:

```
# 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 -= 1
w.reverse()
print(w)
```

```
[4, 4, 4, 4, 4]
```

Figure 4.2: Алгоритм 2

3. Реализация алгоритма 3 - умножение неотрицательных целых чисел столбиком:

```
# алгоритм 3
u = "123456"
v = "7890"
n = 6
m = 4
w = list()
for i in range(m+n):
    w.append(0)
j = m
```

```
def fun1():
    global v
    global w
    global j
    if j == m:
        j -= 1
    if int(v[j]) == 0:
        w[j] = 0
        fun4()
```

Figure 4.3: Алгоритм 3.1

```
def fun2():
    global k
    global t
    global i
    if i == n:
        i -= 1
    t = int(u[i]) * int(v[j]) + w[i+j] + k
    w[i+j] = t % b
    k = t / b

def fun3():
    global i
    global w
    global j
    global k
    i -= 1
    if i > 0:
        fun2()
    else:
        w[j] = k
```

Figure 4.4: Алгоритм 3.2

```
def fun4():
    global j
    global w
    j -= 1
    if j > 0:
        fun1()
    if j == 0:
        print(w)

fun1()
i = n
k = 0
t = 1
fun2()
fun3()
fun4()
print(w)
```

```
[0, 0, 0, 0, 0, 0, 0.3999999999999986, 4, 0, 0]
```

Figure 4.5: Алгоритм 3.3

4. Реализация алгоритма 4 - быстрый столбик:

```
# 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)
```

```
[8, 3, 1, 4, 0, 2, 0, 5, 0, 0, 0]
```

Figure 4.6: Алгоритм 4

5. Реализация алгоритма 5 - деление многоразрядных целых чисел:

```
# 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)
```

Figure 4.7: Алгоритм 5.1

```
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)
```

```
[0, 2, 9] -39899091
```

Figure 4.8: Алгоритм 5.2

5 Листинг программы

```
import math
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 fun1():
  global v
  global w
  global j
  if j == m:
    j -= 1
  if int(v[j]) == 0:
    w[j] = 0
```

```
    fun4()

def fun2():
  global k
  global t
  global i
  if i == n:
    i -= 1
  t = int(u[i]) * int(v[j]) + w[i+j] + k
  w[i+j] = t % b
  k = t / b

def fun3():
  global i
  global w
  global j
  global k
  i -= 1
  if i > 0:
    fun2()
  else:
    w[j] = k

def fun4():
  global j
  global w
  j -= 1
  if j > 0:
    fun1()
```

```
  if j == 0:
    print(w)

fun1()
i = n
k = 0
t = 1
fun2()
fun3()
fun4()
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)
```

6 Выводы

В ходе работы мы изучили и реализовали алгоритмы целочисленной арифметики многократной точности.

7 Список литературы

1. Фороузан Б. А. Криптография и безопасность сетей. - М.: Интернет-Университет Информационных Технологий : БИНОМ. Лаборатория знаний, 2010. - 784 с. [1]

2. Методические материалы курса [2]