Отчёт по лабораторной работе №8

Целочисленная арифметика многократной точности

Студент: Гонсалес Ананина Луис Антонио, 1032175329
Группа: НФИмд-02-21
Преподаватель: Кулябов Дмитрий Сергеевич,
д-р.ф.-м.н., проф.

Москва 2021

Содержание

List of Figures

1 Цель работы

Цель данной лабораторной работы- изучить теорию и реализовать рассмотренные алгоритмы программно.

2 Теоретические сведения

Целочисленная арифметика многократной точности

Мы считаем, что числа записаны в b-ичной системе счисления, где b — фиксированное натуральное число, b 2. При этом натуральное число, записываемое не более чем n цифрами в b-ичной системе счисления, мы обозначаем u1 *. . . u**n* (допуская, что несколько старших разрядов u1, . . . , *u**k* могут равняться нулю). Основание b не всегда равно 2; иногда оно соответствует размеру машинного слова, отведенному под запись обычных целых чисел. В этом случае мы работаем с массивом, содержащим большое целое число[1].

При работе с большими целыми числами удобно хранить знак такого числа в отдельной ячейке или переменной. Если мы хотим, например, перемножить два числа, то знак произведения мы вычисляем отдельно.

Алгоритм A (сложение неотрицательных целых чисел).

Для двух неотрицательных чисел u1 *. . . u**n* и v1 *. . . v**n* вычисляется их сумма w0 *. . . w**n; *при этом* w*0 — цифра переноса — всегда равна 0 или 1.

Алгоритм S (вычитание неотрицательных целых чисел).

По двум n-разрядным неотрицательным целым числам u= u1 . . . u**n**v = v1 *. . . v**n* 0 вычисляется их разность w = w1 *. . . w**n* = u − v.

Замечание: Для того, чтобы в общем случае установить, что u1 *. . . u**n* v1 . . . v**n**, **надо пройти по цифрам, вычисляя** u**j** *− v**j. *Это простая проверка; с ее помощью находится знак разности* u − v* в общем случае.

Алгоритм M (умножение неотрицательных целых чисел столбиком).

Для чисел u = u1 *. . . u**n* и v = v1 *. . . v**m* в системе счисления с основанием

b мы находим их произведение $w = uv = w1$ *. . . w**m+n*.

Алгоритм FM («быстрый столбик»)

1	шаг. $t := 0$.
2	**шаг.** (цикл) Для s от 0 до $m + n - 1$ с шагом 1 выполнить шаги 3 и
3	**шаг.** Для i от 0 до s с шагом 1 выполнить присвоение t := t + u$n-i \cdot$ v$m-s+i$.
4	**шаг.** Присвоить *w**m+n−s* := $t \pmod b$ — наименьший неотрицательный вычет по модулю b (опять-таки, это не деление, а чтение записи памяти, если $b = 2$ или b — размер машинного слова); $t := [t/b]$.

3 Выполнение работы

In [1]:
```
import math
# надо ввести данные сначала
u = "12345"
v = "56789"
b = 10
n = 5
```

In [2]:
```
# 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 3.1: Алгоритм1

In [3]:
```
# 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)
```

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

In [4]:
```
# 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
```

Figure 3.2: Алгоритм2

```
    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)
```

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

Figure 3.3: Алгоритм3

In [5]:
```
# 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]
```

In [6]:
```
# 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]):
```

Figure 3.4: Алгоритм4

```
]: # 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)

[0, 2, 9] -39899091

]:
```

Figure 3.5: Алгоритм5

4 Выводы

В итоге в данной лабораторной работы я изучил теорию и реализовал рассмотренные алгоритмы программно.

Список литературы

1. Дискретное логарифмирование в конечном поле [Электронный ресурс]. Википедия, 2021. URL: https://studfile.net/preview/2439346/page:35/.