Лабораторная работа 8

Целочисленная арифметика многократной точности

Пологов Владислав Александрович

2022 Москва

RUDN University, Moscow, Russian Federation

Цель работы

Цель работы

Реализовать программно следующие алгоритмы:

1. Сложение неотрицательных целых чисел;
2. Вычитание неотрицательных целых чисел;
3. Умножение неотрицательных целых чисел столбиком;
4. Быстрый столбки;
5. Деление многоразрядных целых чисел.

Описание реализации

Описание реализации

Для реализации алгоритмов использовались средства языка Python.

Реализация

Алгоритм, реализующий сложение неотрицательных целых чисел

На вход будут подаваться два неотрицательных числа u и v разрядностью n с основанием системы счисления b. На выходе получим сумму w = w_0 w_1 w_2…, w_0 - цифра переноса, которая всегда равна 0 либо 1. Алгоритм представлен на рисунке 1. (рис. -fig. 1) Код, реализующий данный алгоритм, представлен на рисунке 2. (рис. -fig. 2)

Алгоритм, реализующий сложение неотрицательных целых чисел

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

Figure 1: Алгоритм, реализующий сложение неотрицательных целых чисел

Код, реализующий алгоритм сложения неотрицательных целых чисел

```
def first_alg():
    u = '234'
    v = '156'
    b = 10
    n = 3

    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('Result of first algorthm:', w)
```

Алгоритм, реализующий вычитание неотрицательных целых чисел

На вход будут подаваться два неотрицательных числа u и v разрядностью n с основанием системы счисления b. На выходе получим разность w = w_0 w_1 w_2… = u - v. Алгоритм представлен на рисунке 3. (рис. -fig. 3) Код, реализующий данный алгоритм, представлен на рисунке 4. (рис. -fig. 4)

Алгоритм, реализующий вычитание неотрицательных целых чисел

1. Присвоить $j := n$, $k \coloneqq 0$ (k – заем из старшего разряда).

31

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

Figure 3: Алгоритм, реализующий вычитание неотрицательных целых чисел

Код, реализующий алгоритм вычитания неотрицательных целых чисел

```python
def second_alg():
    u = '456'
    v = '123'
    b = 10
    n = 3

    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('Result of second algorthm:', w)
```

Алгоритм, реализующий умножение неотрицательных целых чисел столбиком

На вход будут подаваться два неотрицательных числа u и v с основанием системы счисления b. На выходе получим произведение w = w_0 w_1 w_2… = u * v. Алгоритм представлен на рисунке 5. (рис. -fig. 5) Код, реализующий данный алгоритм, представлен на рисунке 6. (рис. -fig. 6)

Алгоритм, реализующий умножение неотрицательных целых чисел столбиком

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 $t := u_i \cdot v_j + w_{i+j} + k$, $w_{i+j} := t \ (mod\ b)$, $k := \frac{t}{b}$, где w_{i+j} – наименьший неотрицательный вычет в данном классе вычетов.
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.

Figure 5: Алгоритм, реализующий умножение неотрицательных целых чисел

Код, реализующий алгоритм умножения неотрицательных целых чисел столбиком

```python
u = '1234'
v = '89'
n = 4
m = 2

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

def s_6():    ...

def s_2():...

def s_4():...

def s_5():...

s_2()
i = n
k = 0
t = 1
s_4()
s_5()
```

Алгоритм быстрый столбик

На вход будут подаваться два неотрицательных числа u и v с основанием системы счисления b. На выходе получим произведение w = w_0 w_1 w_2… = u * v. Алгоритм представлен на рисунке 7. (рис. -fig. 7) Код, реализующий данный алгоритм, представлен на рисунке 8. (рис. -fig. 8)

Алгоритм быстрый столбик

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 $w_{m+n-s} := t \ (mod\ b)$, $t := \frac{t}{b}$, где w_{m+n-s} – наименьший неотрицательный вычет по модулю b. Результат: w.

Figure 7: Алгоритм, реализующий метод умножения “быстрый столбик”

Код, реализующий алгоритм быстрый столбик

```
def th_alg():
    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(u4[n-i1-1]) * int(v4[m-s1+i1-1]))
        w1[m+n-s1-1] = t1 % b
        t1 = math.floor(t1/b)
    print('Result of th algorthm:', w1)
```

Figure 8: Код, реализующий алгоритм быстрый столбик

Алгоритм деления многоразрядных целых чисел

На вход будут подаваться два неотрицательных числа u и v разрядностью n и t соответственно. На выходе получим частное q и остаток r. Алгоритм представлен на рисунке 9. (рис. -fig. 9) Код, реализующий данный алгоритм, представлен на рисунке 10. (рис. -fig. 10)

1. Для j от 0 до $n-t$ присвоить $q_j \coloneqq 0$.
2. Пока $u \geq vb^{n-t}$, выполнять: $q_{n-t} \coloneqq q_{n-t} + 1$, $u \coloneqq u - vb^{n-t}$.
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 $q_{i-t-1} \coloneqq b-1$, иначе присвоить $q_{i-t-1} \coloneqq \frac{u_i b + u_{i-1}}{v_t}$.

   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 $q_{i-t-1} \coloneqq q_{i-t-1} - 1$.

   3.3 присвоить $u \coloneqq u - q_{i-t-1} b^{i-t-1} v$.

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

Figure 9: Алгоритм, реализующий алгоритм деления многоразрядных целых чисел

Код, реализующий алгоритм деления многоразрядных целых чисел

```python
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('Result of fifth algorthm:', q, r)
```

Вывод

Вывод

- Реализованы следующие алгоритмы:
    1. Сложение неотрицательных целых чисел;
    2. Вычитание неотрицательных целых чисел;
    3. Умножение неотрицательных целых чисел столбиком;
    4. Быстрый столбки;
    5. Деление многоразрядных целых чисел.