Отчет по лабораторной работе №8

Целочисленная арифметика многократной точности

Бурдина Ксения Павловна

19 декабря 2023

Содержание

# 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, . Натуральное -разрядное число будем записывать в виде: .

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]](https://esystem.rudn.ru/pluginfile.php/2089897/mod_folder/content/0/mathsec_lection12-message-integrity-authentication.pdf?forcedownload=1).

## 3.1 Алгоритм 1 (сложение неотрицательных целых чисел).

*Вход*. Два неотрицательных числа и ; разрядность чисел *n*; основание системы счисления *b*.

*Выход*. Сумма , где - цифра переноса - всегда равная 0 либо 1.

* присвоить (*j* идет по разрядам, *k* следит за переносом);
* присвоить (*mod* *b*), где - наименьший неотрицательный вычет в данном классе вычетов; ;
* присвоить . Если , то возвращаемся на шаг 2; если , то присвоить и результат: .

## 3.2 Алгоритм 2 (вычитание неотрицательных целых чисел).

*Вход*. Два неотрицательных числа и , ; разрядность чисел *n*; основание системы счисления *b*.

*Выход*. Разность .

* присвоить (*k* - заем из старшего разряда);
* присвоить (*mod* *b*), где - наименьший неотрицательный вычет в данном классе вычетов; ;
* присвоить . Если , то возвращаемся на шаг 2; если , то результат: [[1]](https://intuit.ru/studies/courses/552/408/lecture/9350).

## 3.3 Алгоритм 3 (умножение неотрицательных целых чисел столбиком).

*Вход*. Числа и ; основание системы счисления *b*.

*Выход*. Произведение .

* выполнить присвоения: , , …, , (*j* перемещается по номерам разрядов числа *v* от младших к старшим);
* если , то присвоить и перейти на шаг 6;
* присвоить , (Значение *i* идет по номерам разрядов числа *u*, *k* отвечает за перенос);
* присвоить (*mod b*), , где - наименьший неотрицательный вычет в данном классе вычетов;
* присвоить . Если , то возвращаемся на шаг 4, иначе присвоить ;
* присвоить . Если , то вернуться на шаг 2. Если , то результат: *w*.

## 3.4 Алгоритм 4 (быстрый столбик).

*Вход*. Числа и ; основание системы счисления *b*.

*Выход*. Произведение .

* присвоить ;
* 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 ;
* присвоить (*mod b*), , где - наименьший неотрицательный вычет по модулю *b*. Результат: *w*.

## 3.5 Алгоритм 5 (деление многоразрядных целых чисел).

*Вход*. Числа и , , ; разрядность чисел соответственно *n* и *t*.

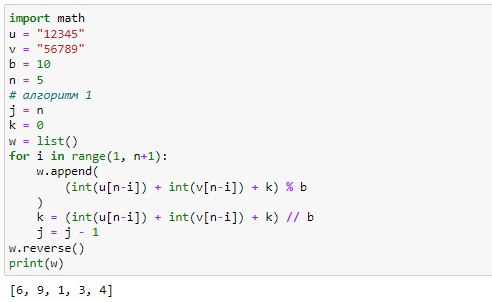
*Выход*. Частное , остаток .

* для *j* от 0 до *n-t* присвоить ;
* пока , выполнять: , ;
* для *i=n, n-1, …, t+1* выполнять пункты:
  + если , то присвоить , иначе присвоить ;
  + пока выполнять ;
  + присвоить ;
  + если , то присвоить , .
* . Результат: *q* и *r*.

# 4 Ход выполнения лабораторной работы

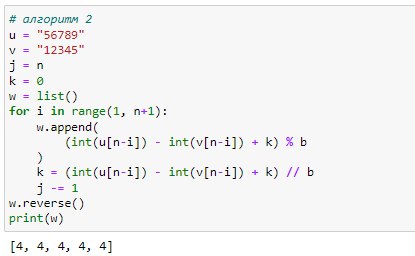
Для реализации рассмотренных алгоритмов будем использовать среду JupyterLab. Выполним необходимую задачу.

1. Реализация алгоритма 1 - сложение неотрицательных целых чисел:



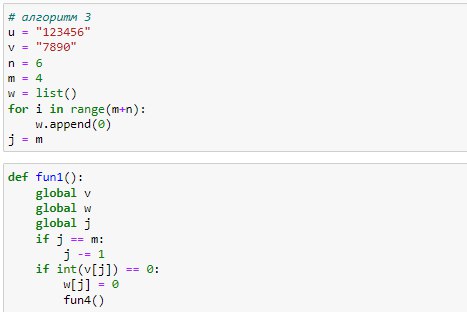
Алгоритм 1

1. Реализация алгоритма 2 - вычитание неотрицательных целых чисел:

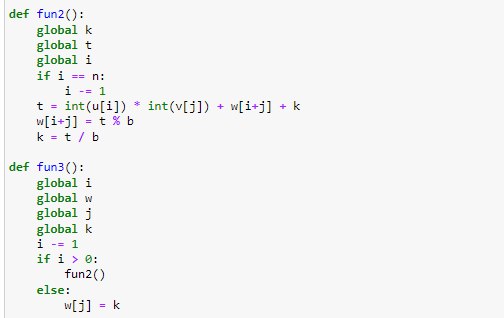


Алгоритм 2

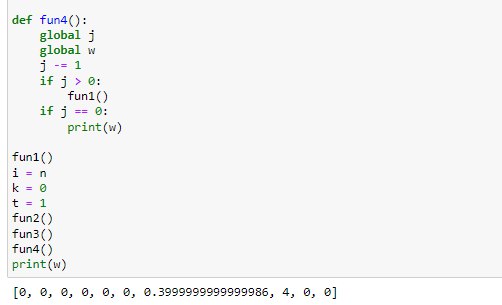
1. Реализация алгоритма 3 - умножение неотрицательных целых чисел столбиком:



Алгоритм 3.1

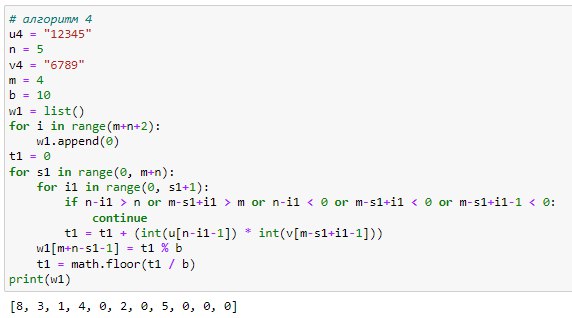


Алгоритм 3.2



Алгоритм 3.3

1. Реализация алгоритма 4 - быстрый столбик:

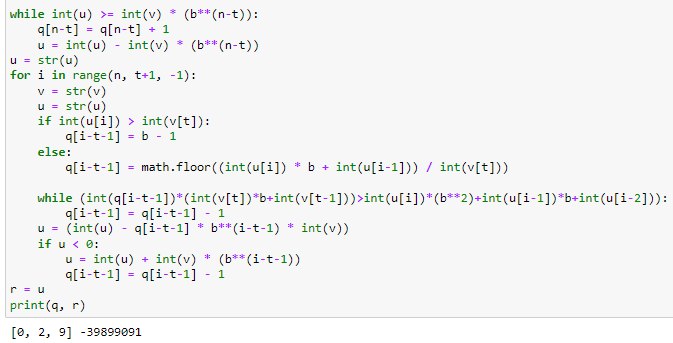


Алгоритм 4

1. Реализация алгоритма 5 - деление многоразрядных целых чисел:



Алгоритм 5.1



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]](https://intuit.ru/studies/courses/552/408/lecture/9350)
2. Методические материалы курса [[2]](https://esystem.rudn.ru/pluginfile.php/2089897/mod_folder/content/0/mathsec_lection12-message-integrity-authentication.pdf?forcedownload=1)