

Отчёт по лабораторной работе 8

Целочисленная арифметика многократной точности

Гаджиев Нурсултан Тофик оглы НФИ-01-22

Содержание

1	Цель работы	5
2	Теоретические сведения	6
2.1	Сложение неотрицательных целых чисел	7
2.2	Вычитание неотрицательных целых чисел	7
2.3	Умножение неотрицательных целых чисел столбиком	7
2.4	Быстрый столбик	8
2.5	Деление многозначных целых чисел	8
3	Выполнение лабораторной работы	10
4	Выводы	14
5	Список литературы	15

List of Tables

List of Figures

3.1	Начальные данные	10
3.2	Алгоритм Сложение неотрицательных целых чисел	10
3.3	Алгоритм вычитания неотрицательных целых чисел	11
3.4	Алгоритм умножения неотрицательных целых чисел столбиком первая часть	11
3.5	Алгоритм умножения неотрицательных целых чисел столбиком вторая часть	12
3.6	Алгоритм быстрого столбика	12
3.7	Алгоритм деления многоразрядных целых чисел	13
3.8	Алгоритм деления многоразрядных целых чисел	13
3.9	Результат алгоритмов	13

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): $x = (x_{n-1}x_{n-2} \dots x_1x_0)_b$, где $\forall i \in [0, n-1] : 0 \leq x_i < b$.

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 $u = u_1 u_2 \dots u_n$ и $v = v_1 v_2 \dots v_n$; разрядность чисел n ; основание системы счисления b .

*Выход. Сумма $w = w_0 w_1 \dots w_n$, 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 $k = \left\lfloor \frac{u_j + v_j + k}{b} \right\rfloor$.
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 $u = u_1 u_2 \dots u_n$ и $v = v_1 v_2 \dots v_n, u > v$; разрядность чисел n ; основание системы счисления b .

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

1. Присвоить $j = n, k = 0$ (k - заём из старшего разряда).
2. Присвоить $w_j = (u_j - v_j + k) \pmod{b}$; $k = \left\lfloor \frac{u_j - v_j + k}{b} \right\rfloor$.
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 $u = u_1 u_2 \dots u_n, v = v_1 v_2 \dots v_m$; основание системы счисления b .

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

1. 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 $t = u_i \cdot v_j + w_{i+j} + k, w_{i+j} = t \pmod{b}, k = \lfloor \frac{t}{b} \rfloor$.
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 $w_{m+n-s} = t \pmod{b}, t = \lfloor \frac{t}{b} \rfloor$. 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 $u = u + v b^{i-t-1}$, $q_{i-t-1} = q_{i-t-1} - 1$.

4. $r = u$. Результат: q и r .

3 Выполнение лабораторной работы

1. Написал блок данных (рис. 3.1)

```
2 # надо ввести данные сначала
3 u = "12345"
4 v = "56789"
5 b = 10
6 n = 5
```

Figure 3.1: Начальные данные

2. Написал алгоритм сложения неотрицательных целых чисел (рис. 3.2)

```
9 # алгоритм 1
10 j = n
11 k = 0
12
13 w = []
14 ▼ for i in range(1, n + 1):
15     w.append((int(u[n - i]) + int(v[n - i]) + k) % b)
16
17     k = (int(u[n - i]) + int(v[n - i]) + k) // b
18     j = j - 1
19 w.reverse()
20 print(w)
21
```

Figure 3.2: Алгоритм Сложение неотрицательных целых чисел

3. Написал алгоритм вычитания неотрицательных целых чисел (рис. 3.3)

```

22 # алгоритм 2
23 u = "56789"
24 v = "12345"
25
26 j = n
27 k = 0
28 w = []
29 for i in range(1, n + 1):
30     w.append((int(u[n - i]) - int(v[n - i]) + k) % b)
31
32     k = (int(u[n - i]) - int(v[n - i]) + k) // b
33     j = j - 1
34 w.reverse()
35 print(w)
36

```

Figure 3.3: Алгоритм вычитания неотрицательных целых чисел

4. Написал алгоритм умножения неотрицательных целых чисел столбиком(рис. 3.4)(рис. 3.5)

```

37 # алгоритм 3
38 u = "123456"
39 v = "7890"
40 n = 6
41 m = 4
42
43 w = []
44 for i in range(m + n):
45     w.append(0)
46     j = m
47
48
49 def step6():
50     global j
51     global w
52     j = j - 1
53     if j > 0:
54         step2()
55     if j == 0:
56         print(w)
57
58
59 def step2():
60     global v
61     global w
62     global j
63     if j == m:
64         j = j - 1
65     if int(v[j]) == 0:
66         w[j] = 0
67         step6()

```

Figure 3.4: Алгоритм умножения неотрицательных целых чисел столбиком первая часть

```

69
70 ▼ def step4():
71     global k
72     global t
73     global i
74 ▼     if i == n:
75         i = i - 1
76         t = int(u[i]) * int(v[j]) + w[i + j] + k
77         w[i + j] = t % b
78         k = t / b
79
80
81 ▼ def step5():
82     global i
83     global w
84     global j
85     global k
86     i = i - 1
87 ▼     if i > 0:
88         step4()
89 ▼     else:
90         w[j] = k
91
92
93     step2()
94     i = n
95     k = 0
96     t = 1
97     step4()
98     step5()
99     step6()
100 print(w)

```

Figure 3.5: Алгоритм умножения неотрицательных целых чисел столбиком вторая часть

5. Написал алгоритм быстрого столбика (рис. 3.6)

```

101
102 # алгоритм 4
103 u4 = "12345"
104 n = 5
105 v4 = "6789"
106 m = 4
107 b = 10
108 w1 = []
109 ▼ for i in range(m + n + 2):
110     w1.append(0)
111     t1 = 0
112 ▼ for s1 in range(0, m + n):
113 ▼     for i1 in range(0, s1 + 1):
114 ▼         if n - i1 > n or m - s1 + i1 > m or n - i1 < 0 or m - s1 + i1 < 0 or m - s1 + i1 - 1 < 0:
115             continue
116         t1 = t1 + (int(u[n - i1 - 1]) * int(v[m - s1 + i1 - 1]))
117     w1[m + n - s1 - 1] = t1 % b
118     t1 = math.floor(t1 / b)
119 print(w1)
120

```

Figure 3.6: Алгоритм быстрого столбика

6. Написал алгоритм деления многоразрядных целых чисел (рис. 3.7)(рис. 3.8)

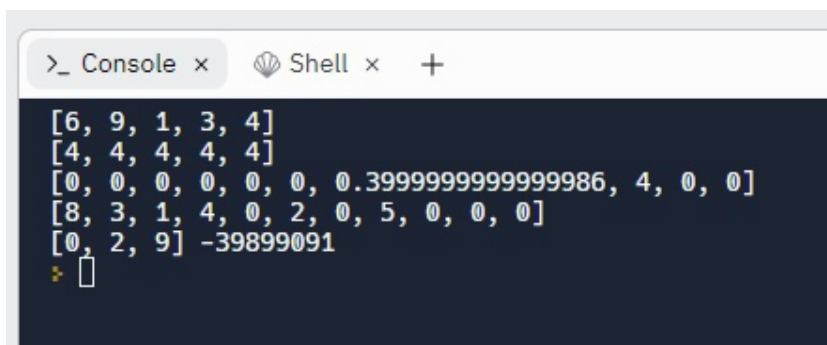
```
121 # алгоритм 5
122 u = "12346789"
123 n = 7
124 v = "56789"
125 t = 4
126 b = 10
127 q = []
128 for j in range(n - t):
129     q.append(0)
130 r = []
131 for j in range(t):
132     r.append(0)
133
134 while int(u) >= int(v) * (b**(n - t)):
135     q[n - t] = q[n - t] + 1
136     u = int(u) - int(v) * (b**(n - t))
137     u = str(u)
138 for i in range(n, t + 1, -1):
139     v = str(v)
140     u = str(u)
141     if int(u[i]) > int(v[t]):
142         q[i - t - 1] = b - 1
143     else:
144         q[i - t - 1] = math.floor((int(u[i]) * b + int(u[i - 1])) / int(v[t]))
145
```

Figure 3.7: Алгоритм деления многоразрядных целых чисел

```
145
146 while (int(q[i - t - 1]) * (int(v[t]) * b + int(v[t - 1])) > int(u[i]) *
147     (b**2) + int(u[i - 1]) * b + int(u[i - 2]))):
148     q[i - t - 1] = q[i - t - 1] - 1
149     u = (int(u) - q[i - t - 1] * b**(i - t - 1) * int(v))
150     if u < 0:
151         u = int(u) + int(v) * (b**(i - t - 1))
152         q[i - t - 1] = q[i - t - 1] - 1
153 r = u
154 print(q, r)
```

Figure 3.8: Алгоритм деления многоразрядных целых чисел

7. Получил результат (рис. 3.9)



```
>_ Console x Shell x +
[6, 9, 1, 3, 4]
[4, 4, 4, 4, 4]
[0, 0, 0, 0, 0, 0, 0.39999999999999986, 4, 0, 0]
[8, 3, 1, 4, 0, 2, 0, 5, 0, 0, 0]
[0, 2, 9] -39899091
```

Figure 3.9: Результат алгоритмов

4 Выводы

Изучал задачу представления больших чисел, познакомились с вычислительными алгоритмами и реализовали их.

5 Список литературы

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