

Лабораторная работа №8: Презентация.

Целочисленная арифметика многократной точности.

Евдокимов Максим Михайлович. Группа - НФИмд-01-24.¹

5 октября, 2024, Москва, Россия

¹Российский Университет Дружбы Народов

Цели и задачи работы

Реализовать все рассмотренные алгоритмы программно.

1. сложение неотрицательных целых чисел.
2. вычитание неотрицательных целых чисел.
3. умножение неотрицательных целых чисел.
4. быстрый столбик.
5. деление многоразрядных целых чисел.

Теоретическое введение

Сложение неотрицательных целых чисел:

Алгоритм сложения двух многоразрядных неотрицательных целых чисел, представленных в виде массивов цифр.

Алгоритм:

1. Инициализировать перенос нулем.
2. Для каждой цифры с наименьшего разряда:
3. Сложить соответствующие цифры двух чисел и перенос.
4. Записать младшую цифру результата в текущий разряд.
5. Обновить перенос.
6. Если после последнего разряда остался перенос, добавить его в результат.

Вычитание неотрицательных целых чисел:

Алгоритм вычитания двух многоразрядных неотрицательных целых чисел, представленных в виде массивов цифр, при условии, что первое число больше или равно второму.

Алгоритм:

1. Инициализировать заем нулем.
2. Для каждой цифры с наименьшего разряда:
3. Вычесть из соответствующей цифры первого числа цифру второго числа и заем.
4. Если результат отрицательный, добавить 10 и установить заем в 1.
5. Записать результат в текущий разряд.
6. Удалить ведущие нули из результата.

Умножение неотрицательных целых чисел:

Алгоритм умножения двух многоразрядных неотрицательных целых чисел, представленных в виде массивов цифр.

Алгоритм:

1. Инициализировать результат нулем.
2. Для каждой цифры второго числа с наименьшего разряда:
3. Умножить первое число на эту цифру.
4. Сдвинуть результат влево на соответствующее количество разрядов.
5. Сложить результат с текущим результатом.

Быстрый столбик - алгоритм Карацубы:

Алгоритм умножения двух многоразрядных неотрицательных целых чисел, основанный на принципе “разделяй и властвуй” и позволяющий уменьшить количество умножений по сравнению с классическим алгоритмом.

Алгоритм:

1. Разделить каждое число на две равные части.
2. Вычислить три промежуточных произведения:
3. Произведение первых частей чисел.
4. Произведение вторых частей чисел.
5. Произведение сумм первых и вторых частей чисел.
6. Использовать промежуточные произведения для вычисления конечного результата.

Деление многоразрядных целых чисел:

Алгоритм деления двух многоразрядных неотрицательных целых чисел, представленных в виде массивов цифр, с получением частного и остатка.

Алгоритм:

1. Инициализировать частное и остаток нулями.
2. Пока делимое больше или равно делителю:
3. Оценить максимально возможное значение очередной цифры частного.
4. Умножить делитель на эту цифру и вычесть результат из делимого.
5. Добавить цифру к частному.
6. Остаток равен текущему значению делимого.

- Все алгоритмы предполагают, что числа представлены в системе счисления с основанием 10.
- Алгоритмы могут быть адаптированы для работы с другими системами счисления.
- Алгоритм Карацубы имеет сложность $O(n^{\log_2 3})$, что лучше, чем $O(n^2)$ для классического алгоритма умножения.

Ход работы

Задание 1

```
1  # сложение неотрицательных целых чисел
2  function add_big_integers(a::Vector{Int}, b::Vector{Int})
3      lenA, lenB = length(a), length(b)
4      lenC, k = max(lenA, lenB), 0
5      result = zeros{Int, 1}, lenC
6
7      # Дополняем меньшее число нулями слева
8      if lenA < lenC
9          a = vcat(zeros{Int, 1}, lenC - lenA, a)
10     elseif lenB < lenC
11         b = vcat(zeros{Int, 1}, lenC - lenB, b)
12     end
13
14     for i in lenC:-1:1
15         sum = a[i] + b[i] + k
16         result[i] = sum % 10
17         k = sum ÷ 10
18     end
19     if k > 0
20         return [k; result]
21     else
22         return result
23     end
24 end

add_big_integers (generic function with 1 method)
```

Рис. 1: Сумма целых неотрицательных чисел

Задание 2

```
1  # Вычитание неотрицательных целых чисел
2  function subtract_big_integers(a::Vector{Int}, b::Vector{Int})
3      lenA, lenB = length(a), length(b)
4      lenC, k = max(lenA, lenB), 0
5      result = zeros{Int, 1}(lenC)
6      if lenA < lenB || (lenA == lenB && a < b)
7          a, b = b, a
8          negative_result = true
9      else
10         negative_result = false
11     end
12
13     # Дополняем меньшее число нулями слева
14     if lenA < lenC
15         a = vcat(zeros{Int, 1}(lenC - lenA), a)
16     elseif lenB < lenC
17         b = vcat(zeros{Int, 1}(lenC - lenB), b)
18     end
19
20     for i in length(a):-1:1
21         a_digit = a[i]
22         b_digit = b[i]
23         diff = a_digit - b_digit - k
24         if diff < 0
25             diff += 10
26             k = 1
27         else
28             k = 0
29         end
30         result[i] = diff
31     end
32
33     while result[1] == 0 && length(result) > 1
34         popfirst!(result)
35     end
36     return result
37 end
```

subtract_big_integers (generic function with 1 method)

Рис. 2: Разница целых неотрицательных чисел

Задание 3

```
1  # умножение неотрицательных целых чисел
2  function multiply_big_integers(a::Vector{Int}, b::Vector{Int})
3      lenA, lenB = length(a), length(b)
4      result = zeros{Int}(lenA+lenB+1)
5      lenC = max(lenA, lenB)
6
7      for i in lenA:-1:1
8          for j in lenB:-1:1
9              product = a[i] * b[j]
10             sum = result[i+j+1] + product
11             result[i+j+1] = sum % 10
12             result[i+j] += sum ÷ 10
13         end
14     end
15
16     # Убираем ведущие нули
17     while length(result) > 1 && result[1] == 0
18         result = result[2:end]
19     end
20     return result
21 end
```

multiply_big_integers (generic function with 1 method)

Рис. 3: Умножение целых неотрицательных чисел

Задание 4

```
1 # Базовый способ для неотрицательных целых чисел
2 function karatsuba_multiply(a::Vector{Int}, b::Vector{Int})
3     if length(a) <= 1 || length(b) <= 1
4         return [a[1] * b[1]]
5     end
6     lenA, lenB, lenC = length(a), length(b), length(a)+length(b)
7     m = max(lenA, lenB) + 2
8     high1, low1 = split_vector(a, m, lenA)
9     high2, low2 = split_vector(b, m, lenB)
10
11     z0 = multiply_big_integers(low1, low2) # a_0 * b_0
12     z1 = multiply_big_integers(low1, high2) # a_0 * b_1
13     z2 = multiply_big_integers(high1, low2) # a_1 * b_0
14     z3 = multiply_big_integers(high1, high2) # a_1 * b_1
15
16     e10, e11, e12 = vcat(z3, zeros(Int, m*2)), vcat(zeros(Int, m), add_big_integers(z1, z2), zeros(Int, m)), vcat(zeros(Int, m*2), z0)
17     z = add_big_integers(add_big_integers(e10, e12), e11)
18     if z[1] == 0
19         return z[2:end]
20     end
21     return z
22 end
23
24 # Функция для разделения вектора на две части
25 function split_vector(v::Vector{Int}, m::Int, l::Int)
26     high = v[1:l-m]
27     low = v[l-m+1:end]
28     return high, low
29 end
30
31 split_vector(generic function with 2 methods)
```

Рис. 4: алгоритм Карацуба для целых неотрицательных чисел

```
1  # деление многоразрядных целых чисел
2  function divide_big_integers(a::Vector{Int}, b::Vector{Int})
3      function vector_to_bigint(v::Vector{Int})
4          result = BigInt(0)
5          for digit in v
6              result = result * 10 + digit
7          end
8          return result
9      end
10
11     a_big = vector_to_bigint(a)
12     b_big = vector_to_bigint(b)
13
14     result = a_big ÷ b_big
15     remainder = a_big % b_big
16     result_digits = reverse(digits(result))
17
18     fractional_part = Vector{Int}()
19     for _ in 1:10
20         remainder *= 10
21         digit = remainder ÷ b_big
22         remainder %= b_big
23         push!(fractional_part, digit)
24     end
25
26     return result_digits, fractional_part
27 end

divide_big_integers (generic function with 1 method)
```

Рис. 5: Деление целых неотрицательных чисел

```
1 # Наши значения  
2 numb1, numb2 = rand(1:9, 12), rand(1:9, 12)  
  
([4, 1, 3, 8, 1, 2, 7, 1, 4, 9, 5, 1], [3, 7, 1, 7, 6, 5, 6, 8, 2, 2, 7, 8])
```

Рис. 6: Сгенерированные значения

```
1 sum = add_big_integers(num1, num2)
2 println("Сумма: ", sum)
3
4 diff = subtract_big_integers(num1, num2)
5 println("Разность: ", diff)
6
7 product = multiply_big_integers(num1, num2)
8 println("Произведение: ", product)
9
10 karatsuba_product = karatsuba_multiply(num1, num2)
11 println("Произведение (быстрый столбик): ", karatsuba_product)
12
13 quotient, remainder = divide_big_integers(num1, num2)
14 println("Целая часть: ", quotient)
15 println("Остаток: ", remainder)
```

Сумма: [7, 8, 5, 5, 7, 8, 3, 9, 7, 2, 2, 9]
Разность: [4, 2, 0, 4, 7, 0, 3, 2, 6, 7, 3]
Произведение: [1, 5, 3, 8, 4, 1, 3, 6, 6, 3, 0, 9, 0, 7, 0, 0, 4, 6, 3, 3, 8, 3, 7, 8]
Произведение (быстрый столбик): [1, 5, 3, 8, 4, 1, 3, 6, 6, 3, 0, 9, 0, 7, 0, 0, 4, 6, 3, 3, 8, 3, 7, 8]
Целая часть: [1]
Остаток: [1, 1, 3, 1, 0, 0, 8, 9, 8, 4]

Рис. 7: Результаты применения всех алгоритмов к векторам

Выводы по проделанной работе

В ходе выполнения лабораторной работы были изучены способы работы с алгоритмами целочисленной арифметики многократной точности, а также на их основе реализованы функции суммирования, разности, умножения, быстрого столбика и деления для целых неотрицательных чисел в виде векторов цифр.