Лабораторная работа №4

Математические основы защиты информации и информационной безопасности

Николаев Дмитрий Иванович, НПМмд-02-24

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить работу алгоритмов вычисления наибольшего общего делителя: алгоритм Евклида, бинарный алгоритм Евклида, расширенный алгоритм Евклида, расширенный бинарный алгоритм Евклида, а также реализовать их программно.

2 Теоретическое введение

2.1 Вычисление наибольшего общего делителя

Пусть числа a и b целые и $b \neq 0$. Разделить a на b с остатком — значит представить a в виде $a = q \cdot b + r$, где $q, r \in \mathbb{Z}$ и $0 \leq r \leq |b|$. Число q называется неполным частным, число r — неполным остатком от деления a на b.

Целое число $d \neq 0$ называется *наибольшим общим делителем* целых чисел $a_1, a_2, \dots, a_k$ (обозначается $d = \text{НОД}(a_1, a_2, \dots, a_k)$), если выполняются следующие условия:

1. каждое из чисел $a_1, a_2, \dots, a_k$ делится на d;
2. если $d_1 \neq 0$ — другой общий делитель чисел $a_1, a_2, \dots, a_k$, то d делится на d_1. Например, $\text{НОД}(12345, 24690) = 12345$, $\text{НОД}(12345, 54321) = 3$, $\text{НОД}(12345, 12541) = 1$.

Ненулевые целые числа a и b называются *ассоциированными* (обозначается $a \sim b$), если a делится на b и b делится на a.

Для любых целых чисел $a_1, a_2, \dots, a_k$ существует наибольший общий делитель d, и его можно представить в виде *линейной комбинации* этих чисел:

$$d = c_1 \cdot a_1 + c_2 \cdot a_2 + \cdots + c_k \cdot a_k, \quad c_i \in \mathbb{Z}.$$

Например, НОД чисел $91, 105, 154$ равен 7. В качестве линейного представления можно взять:

$$7 = 7 \cdot 91 + (-6) \cdot 105 + 0 \cdot 154,$$

$$7 = 4 \cdot 91 + 1 \cdot 105 - 3 \cdot 154.$$

Целые числа $a_1, a_2, \dots, a_k$ называются *взаимно простыми в совокупности*, если НОД$(a_1, a_2, \dots, a_k) = 1$. Целые числа a и b называются *взаимно простыми*, если НОД$(a, b) = 1$.

Целые числа $a_1, a_2, \dots, a_k$ называются *попарно взаимно простыми*, если НОД$(a_i, a_j) = 1$ для всех $1 \leq i \neq j \leq k$.

2.2 Алгоритмы вычисления наибольшего общего делителя

Для вычисления наибольшего общего делителя двух целых чисел применяется способ повторного деления с остатком, называемый **алгоритмом Евклида**.

2.2.1 Алгоритм Евклида

Вход: целые числа a, b; $0 < b \leq a$.
Выход: $d =$ НОД(a, b).

1. Положить $r_0 \leftarrow a$, $r_1 \leftarrow b$, $i \leftarrow 1$.
2. Найти остаток r_{i+1} от деления r_{i-1} на r_i.
3. Если $r_{i+1} = 0$, то положить $d \leftarrow r_i$. В противном случае положить $i \leftarrow i+1$ и вернуться на шаг 2.
4. Результат: d.

2.2.2 Бинарный алгоритм Евклида

Бинарный алгоритм Евклида является более быстрым при реализации на компьютере, поскольку использует двоичное представление чисел a и b. Он основан на следующих свойствах наибольшего общего делителя (считаем, что $0 < b \leq a$):

1. Если оба числа a и b четные, то НОД$(a, b) = 2 \cdot$ НОД$(\frac{a}{2}, \frac{b}{2})$.
2. Если a — нечетное, b — четное, то НОД$(a, b) =$ НОД$(a, \frac{b}{2})$.
3. Если оба числа a и b нечетные и $a > b$, то НОД$(a, b) =$ НОД$(a - b, b)$.
4. Если $a = b$, то НОД$(a, b) = a$.

Вход: целые числа a, b; $0 < b \leq a$.
Выход: $d =$ НОД(a, b).

1. Положить $g \leftarrow 1$.
2. Пока оба числа a и b четные, выполнять $a \leftarrow \frac{a}{2}$, $b \leftarrow \frac{b}{2}$, $g \leftarrow 2g$ до получения хотя бы одного нечетного значения a или b.
3. Положить $u \leftarrow a$, $v \leftarrow b$.
4. Пока $u \neq 0$, выполнять следующие действия:
    1. Пока u четное, полагать $u \leftarrow \frac{u}{2}$.
    2. Пока v четное, полагать $v \leftarrow \frac{v}{2}$.
    3. При $u \geq v$, положить $u \leftarrow u - v$. В противном случае положить $v \leftarrow v - u$.
5. Положить $d \leftarrow v \cdot g$.
6. Результат: d.

2.2.3 Расширенный алгоритм Евклида

Вход: целые числа a, b; $0 < b \leq a$.
Выход: $d =$ НОД(a, b) и такие целые числа x, y, что $ax + by = d$.

1. Положить $r_0 \leftarrow a$, $r_1 \leftarrow b$, $x_0 \leftarrow 1$, $x_1 \leftarrow 0$, $y_0 \leftarrow 0$, $y_1 \leftarrow 1$, $i \leftarrow 1$.
2. Разделить с остатком r_{i-1} на r_i, получив q_i и r_{i+1}: $r_{i-1} = q_i r_i + r_{i+1}$.
3. Если $r_{i+1} = 0$, то положить $d \leftarrow r_i$, $x \leftarrow x_i$, $y \leftarrow y_i$. В противном случае положить $x_{i+1} \leftarrow x_{i-1} - q_i x_i$, $y_{i+1} \leftarrow y_{i-1} - q_i y_i$, $i \leftarrow i + 1$ и вернуться на шаг 2.
4. Результат: d, x, y.

2.2.4 Расширенный бинарный алгоритм Евклида

Вход: целые числа a, b; $0 < b \leq a$.
Выход: $d = \text{НОД}(a, b)$.

1. Положить $g \leftarrow 1$.
2. Пока числа a и b четные, выполнять $a \leftarrow \frac{a}{2}$, $b \leftarrow \frac{b}{2}$, $g \leftarrow 2g$ до получения хотя бы одного нечетного значения a или b.
3. Положить $u \leftarrow a$, $v \leftarrow b$, $A \leftarrow 1$, $B \leftarrow 0$, $C \leftarrow 0$, $D \leftarrow 1$.
4. Пока $u \neq 0$, выполнять следующие действия:
    1. Пока u четное:
        1. Положить $u \leftarrow \frac{u}{2}$.
        2. Если A и B четные, то положить $A \leftarrow \frac{A}{2}$, $B \leftarrow \frac{B}{2}$. В противном случае положить $A \leftarrow \frac{A+b}{2}$, $B \leftarrow \frac{B-a}{2}$.
    2. Пока v четное:
        1. Положить $v \leftarrow \frac{v}{2}$.
        2. Если C и D четные, то положить $C \leftarrow \frac{C}{2}$, $D \leftarrow \frac{D}{2}$. В противном случае положить $C \leftarrow \frac{C+b}{2}$, $D \leftarrow \frac{D-a}{2}$.
    3. Если $u \geq v$, положить $u \leftarrow u - v$, $A \leftarrow A - C$, $B \leftarrow B - D$. В противном случае положить $v \leftarrow v - u$, $C \leftarrow C - A$, $D \leftarrow D - B$.
5. Положить $d \leftarrow v \cdot g$, $x \leftarrow C$, $y \leftarrow D$.
6. Результат: d, x, y.

2.3 Резюме алгоритмов

1. **Алгоритм Евклида**: Это классический алгоритм, который повторяет деление с остатком, пока остаток не станет нулевым. Возвращает последний ненулевой остаток как НОД.
2. **Бинарный алгоритм Евклида**: Использует четность чисел и побитовые сдвиги для ускорения вычислений. Преимущество этого алгоритма заключается в эффективной работе на компьютерах с двоичной арифметикой.
3. **Расширенный алгоритм Евклида**: Помимо нахождения НОД, этот алгоритм вычисляет коэффициенты линейной комбинации $ax + by =$ НОД(a, b).
4. **Расширенный бинарный алгоритм Евклида**: Сочетает подход бинарного алгоритма с расширенным, вычисляя также коэффициенты линейной комбинации, но с использованием более быстрых операций.

3 Выполнение лабораторной работы

Действуя согласно [1], реализуем все описанные алгоритмы на языке Julia.

Программные реализации алгоритма Евклида (Рис.[3.1]), бинарного алгоритма Евклида (Рис.[3.2]), расширенного алгоритма Евклида (Рис.[3.3]) и расширенного бинарного алгоритма Евклида (Рис.[3.4,3.5]) представлены на соответствующих картинках. После чего на начальных данных $a = 91, b = 105$ и с помощью пакета `BenchmarkTools` сравнены алгоритмы нахождения наибольшего общего делителя (Рис.[3.6]), где результаты представлены на Рис.[3.7].

```
1  using BenchmarkTools
2  """Алгоритм Евклида нахождения НОД(a, b)"""
3  function GCD_Euclid(a::Int, b::Int)::Int
4      while b != 0
5          a, b = b, a % b
6      end
7      return a
8  end
```

Рис. 3.1: Код алгоритма Евклида на Julia

```julia
"""Бинарный алгоритм Евклида нахождения НОД(a, b)"""
function GCD_Binary_Euclid(a::Int, b::Int)::Int
    if a == 0 return b end
    if b == 0 return a end
    # Считаем количество делений на 2
    shift = 0
    # Проверка обоих чисел на чётность
    while ((a | b) & 1) == 0
        a >>= 1
        b >>= 1
        shift += 1
    end
    # Проверка первого числа на чётность
    while (a & 1) == 0
        a >>= 1
    end
    while b != 0
        # Проверка 2-го числа на чётность
        while (b & 1) == 0
            b >>= 1
        end
        if a >= b
            a, b = b, a - b
        else
            a, b = a, b - a
        end
    end
    # Умножение на 2 "shift" раз
    # то есть "shift" битовых сдвигов влево
    return a << shift
end
```

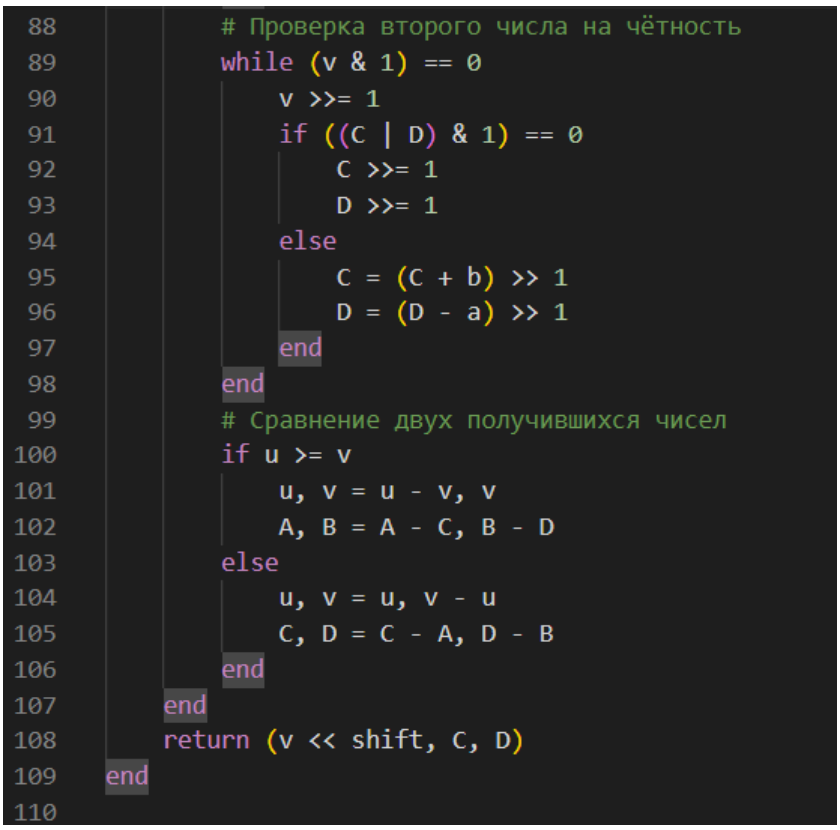
Рис. 3.2: Код бинарного алгоритма Евклида на Julia

```julia
"""Расширенный алгоритм Евклида для нахождения НОД(a,b) и
чисел x и y таких, что выполняется ax + by = НОД(a,b)"""
function GCD_Extended_Euclid(a::Int, b::Int)::Tuple{Int, Int, Int}
    if b == 0
        return (a, 1, 0)
    else
        x0, x1, y0, y1 = 1, 0, 0, 1
        while b != 0
            q = div(a, b)
            a, b = b, a % b
            x0, x1 = x1, x0 - q*x1
            y0, y1 = y1, y0 - q*y1
        end
        return (a, x0, y0)
    end
end
```

Рис. 3.3: Код расширенного алгоритма Евклида на Julia

```julia
"""Расширенный бинарный алгоритм Евклида для нахождения НОД(a,b) и
чисел x и y таких, что выполняется ax + by = НОД(a,b)"""
function GCD_Extended_Binary_Euclid(a::Int, b::Int)::Tuple{Int, Int, Int}
    if b == 0
        return (a, 1, 0)
    end
    if a == 0
        return (b, 0, 1)
    end
    # Считаем число делений на 2
    shift = 0
    while ((a | b) & 1) == 0
        a >>= 1
        b >>= 1
        shift += 1
    end
    u, v, A, B, C, D = a, b, 1, 0, 0, 1
    while u != 0
        # Проверка первого числа на чётность
        while (u & 1) == 0
            u >>= 1
            if ((A | B) & 1) == 0
                A >>= 1
                B >>= 1
            else
                A = (A + b) >> 1
                B = (B - a) >> 1
            end
        end
```

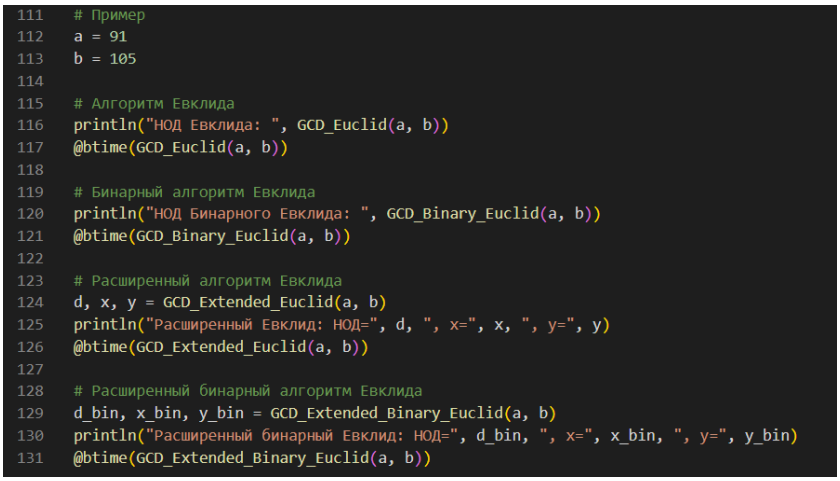
Рис. 3.4: Код расширенного бинарного алгоритма Евклида на Julia (1/2)



```
        # Проверка второго числа на чётность
        while (v & 1) == 0
            v >>= 1
            if ((C | D) & 1) == 0
                C >>= 1
                D >>= 1
            else
                C = (C + b) >> 1
                D = (D - a) >> 1
            end
        end
        # Сравнение двух получившихся чисел
        if u >= v
            u, v = u - v, v
            A, B = A - C, B - D
        else
            u, v = u, v - u
            C, D = C - A, D - B
        end
    end
    return (v << shift, C, D)
end

```

Рис. 3.5: Код расширенного бинарного алгоритма Евклида Евклида на Julia (2/2)



```
# Пример
a = 91
b = 105

# Алгоритм Евклида
println("НОД Евклида: ", GCD_Euclid(a, b))
@btime(GCD_Euclid(a, b))

# Бинарный алгоритм Евклида
println("НОД Бинарного Евклида: ", GCD_Binary_Euclid(a, b))
@btime(GCD_Binary_Euclid(a, b))

# Расширенный алгоритм Евклида
d, x, y = GCD_Extended_Euclid(a, b)
println("Расширенный Евклид: НОД=", d, ", x=", x, ", y=", y)
@btime(GCD_Extended_Euclid(a, b))

# Расширенный бинарный алгоритм Евклида
d_bin, x_bin, y_bin = GCD_Extended_Binary_Euclid(a, b)
println("Расширенный бинарный Евклид: НОД=", d_bin, ", x=", x_bin, ", y=", y_bin)
@btime(GCD_Extended_Binary_Euclid(a, b))
```

Рис. 3.6: Начальные данные для сравнения алгоритмов нахождения НОД на Julia

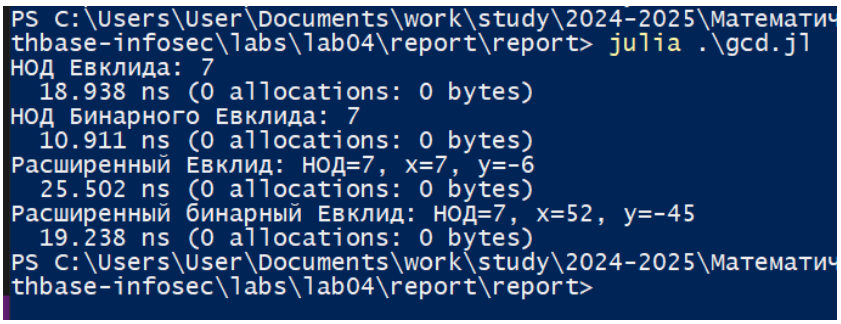


Рис. 3.7: Результат выполнения кода и сравнения алгоритмов нахождения НОД на Julia

4 Выводы

В ходе выполнения лабораторной работы я изучил работу алгоритмов вычисления наибольшего общего делителя: алгоритма Евклида, бинарного алгоритма Евклида, расширенного алгоритма Евклида, расширенного бинарного алгоритма Евклида, а также реализовал их программно.

Список литературы

1. Лабораторная работа № 4. Вычисление наибольшего общего делителя [Электронный ресурс]. Саратовский государственный университет имени Н.Г. Чернышевского, 2024.