Лабораторная работа №4

Математические основы защиты информации и информационной безопасности

Николаев Дмитрий Иванович, НПМмд-02-24

29 сентября 2024

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

Прагматика выполнения

Прагматика выполнения

- Освоение алгоритмов вычисления наибольшего общего делителя — алгоритма Евклида, бинарного алгоритма Евклида, расширенного алгоритма Евклида, расширенного бинарного алгоритма Евклида

Цели

Цели

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

Задачи

Задачи

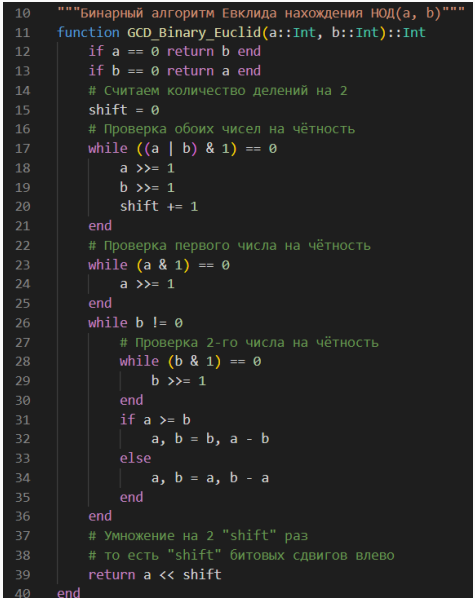
1. Освоить и реализовать алгоритм Евклида на языке Julia;
2. Освоить и реализовать бинарный алгоритм Евклида на языке Julia;
3. Освоить и реализовать расширенный алгоритм Евклида на языке Julia;
4. Освоить и реализовать расширенный бинарный алгоритм Евклида на языке Julia.

Выполнение работы

```julia
using BenchmarkTools
"""Алгоритм Евклида нахождения НОД(a, b)"""
function GCD_Euclid(a::Int, b::Int)::Int
    while b != 0
        a, b = b, a % b
    end
    return a
end
```

Рис. 1: Код алгоритма Евклида на Julia

Бинарный алгоритм Евклида



```
"""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
```

Расширенный алгоритм Евклида

```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: Код расширенного алгоритма Евклида на Julia

Расширенный бинарный алгоритм Евклида (1/2)

```
"""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
```

Расширенный бинарный алгоритм Евклида (2/2)

```
88          # Проверка второго числа на чётность
89          while (v & 1) == 0
90              v >>= 1
91              if ((C | D) & 1) == 0
92                  C >>= 1
93                  D >>= 1
94              else
95                  C = (C + b) >> 1
96                  D = (D - a) >> 1
97              end
98          end
99          # Сравнение двух получившихся чисел
100         if u >= v
101             u, v = u - v, v
102             A, B = A - C, B - D
103         else
104             u, v = u, v - u
105             C, D = C - A, D - B
106         end
107     end
108     return (v << shift, C, D)
109 end
```

Начальные данные

```julia
# 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. 6: Начальные данные для сравнения алгоритмов нахождения НОД на Julia

Результат выполнения кода и сравнения алгоритмов

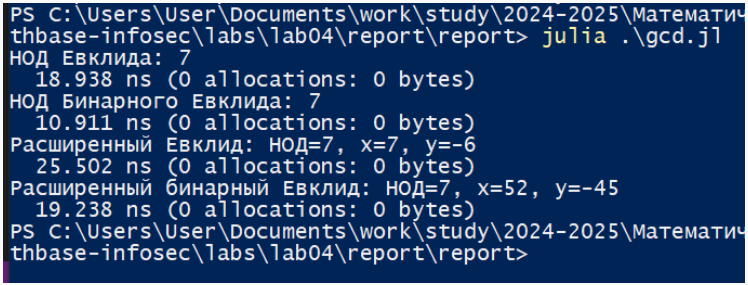


Рис. 7: Результат выполнения кода и сравнения алгоритмов нахождения НОД на Julia

Результаты

Результаты

По результатам 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.