

# **Отчёт по лабораторной работе №4**

**Алгоритм Евклида**

Ван Цзыкунь

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Теоретические сведения</b>	<b>5</b>
2.1	Наибольший общий делитель . . . . .	5
2.2	Алгоритм Евклида . . . . .	5
2.3	Бинарный алгоритм Евклида . . . . .	6
2.4	Расширенный алгоритм Евклида . . . . .	7
<b>3</b>	<b>Выполнение работы</b>	<b>8</b>
3.1	Реализация алгоритмов на языке Python . . . . .	8
3.2	Контрольный пример . . . . .	11
<b>4</b>	<b>Выводы</b>	<b>12</b>
	<b>Список литературы</b>	<b>13</b>

# List of Figures

3.1	Работа алгоритмов . . . . .	11
-----	-----------------------------	----

# 1 Цель работы

Изучение алгоритма Евклида нахождения НОД и его вариаций.

## 2 Теоретические сведения

### 2.1 Наибольший общий делитель

Наибольший общий делитель (НОД) – это число, которое делит без остатка два числа и делится само без остатка на любой другой делитель данных двух чисел. Проще говоря, это самое большое число, на которое можно без остатка разделить два числа, для которых ищется НОД.

### 2.2 Алгоритм Евклида

При работе с большими составными числами их разложение на простые множители, как правило, неизвестно. Но для многих прикладных задач теории чисел поиск разложения числа на множители является важной, часто встречающейся практической задачей. В теории чисел существует сравнительно быстрый способ вычисления НОД двух чисел, который называется алгоритмом Евклида.

Алгоритм Евклида

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

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

4. Результат:  $d$ .

Пример: Найти НОД для 30 и 18.

$$30 / 18 = 1 \text{ (остаток 12)}$$

$$18 / 12 = 1 \text{ (остаток 6)}$$

$$12 / 6 = 2 \text{ (остаток 0)}$$

Конец: НОД – это делитель 6.

## 2.3 Бинарный алгоритм Евклида

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

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

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

## 2.4 Расширенный алгоритм Евклида

Расширенный алгоритм Евклида находит наибольший общий делитель  $d$  чисел  $a$  и  $b$  и его линейное представление, т. е. целые числа  $x$  и  $y$ , для которых  $ax + by = d$ , и не требует «возврата», как в рассмотренном примере. Пусть  $d$  – НОД для  $a$  и  $b$ , т. е.  $d = (a, b)$ , где  $a > b$ . Тогда существуют такие целые числа  $x$  и  $y$ , что  $d = ax + by$ . Иными словами, НОД двух чисел можно представить в виде линейной комбинации этих чисел с целыми коэффициентами

- Вход. Целые числа  $a, b$ ;  $0 < b \leq a$ .
  - Выход:  $d = \text{НОД}(a, b)$ ; такие целые числа  $x, y$ , что  $ax + by = d$ .
1. Положить  $r_0 = a, r_1 = b, x_0 = 1, x_1 = 0, y_0 = 0, y_1 = 1, i = 1$
  2. Разделить с остатком  $r_{i-1}$  на  $r_i$ :  $r_{i-1} = q_i * r_i + r_{i+1}$
  3. Если  $r_{i+1} = 0$ , то положить  $d = r_i, x = x_i, y = y_i$ . В противном случае положить  $x_{i+1} = (x_{i-1}) - q_i * x_i, y_{i+1} = (y_{i-1}) - q_i * y_i, i = i + 1$  и вернуться на шаг 2.
  4. Результат:  $d, x, y$ .

## 3 Выполнение работы

### 3.1 Реализация алгоритмов на языке Python

```
def euklid_simply(a, b):
```

```
    while a!=0 and b!=0:
```

```
        if a>=b:
```

```
            a %= b
```

```
        else:
```

```
            b %=a
```

```
    return a or b
```

```
def euklid_extended(a, b):
```

```
    if a == 0:
```

```
        return (b, 0, 1)
```

```
    else:
```

```
        div, x, y = euklid_extended(b%a, a)
```

```
    return (div, y-(b//a)*x, x)
```

```
def euklid_binary(a, b):
```

```
    g = 1
```

```
    while(a%2 == 0 and b%2 == 0):
```

```
        a = a/2
```

```
        b = b/2
```



```

        g = 2*g
    u,v = a,b
    while u != 0:
        if u%2 == 0:
            u = u/2
        if v%2 == 0:
            v = v/2
        if u>=v:
            u = u-v
        else:
            v = v-u
    d = g*v
    return d

```

```

def euklid_bin_extended(a, b):
    g = 1
    while(a%2 == 0 and b%2 == 0):
        a = a/2
        b = b/2
        g = 2*g
    u = a
    v = b
    A = 1
    B = 0
    C = 0
    D = 1
    while u!=0:
        if u%2 == 0:
            u = u/2

```

```

    if A%2 == 0 and B%2 == 0:
        A = A/2
        B = B/2
    else:
        A = (A+b)/2
        B = (B-a)/2
if v%2 == 0:
    v = v/2
    if C%2 == 0 and D%2 == 0:
        C = C/2
        D = D/2
    else:
        C = (C+b)/2
        D = (D-a)/2
if u>=v:
    u = u-v
    A = A-C
    B = B-D
else:
    v = v-u
    C = C-A
    D = D-B
d = g*v
x = C
y = D
return (d, x, y)

```

## 3.2 Контрольный пример

```
In [2]: 1 main()
Введите числа a999
Введите число b99
Вызываем функцию Евклида
9
А теперь можно вызвать функцию расширенного
(9, 1, -10)
А теперь функция бинарного Евклида
9.0
А теперь функция расширенного бинарного Евклида
(9.0, 12.0, -121.0)
```

Figure 3.1: Работа алгоритмов

## 4 Выводы

Изучили алгоритм Евклида нахождения НОД.

# Список литературы

1. ВЫЧИСЛЕНИЕ НАИБОЛЬШЕГО ОБЩЕГО ДЕЛИТЕЛЯ
2. В очередной раз о НОД