Лабораторная работа №4

Алгоритм Евклида
Студент: Хамза хуссен

---

1. Введение: цели и задачи исследования

Основная цель работы — ознакомиться с классическим алгоритмом Евклида для вычисления наибольшего общего делителя (НОД), а также изучить его модификации: бинарный и расширенный варианты.

---

2. Теоретическая часть

2.1 Понятие наибольшего общего делителя (НОД)

Наибольшим общим делителем двух целых чисел называется наибольшее натуральное число, которое делит оба числа без остатка. НОД играет важную роль в теории чисел, криптографии и алгоритмизации.

2.2 Классический алгоритм Евклида

Данный алгоритм основан на последовательном делении с остатком и основывается на следующем равенстве:

$$\text{НОД}(a, b) = \text{НОД}(b, a \bmod b)$$

где $a \geq b > 0$.

Этапы алгоритма:

1. Пусть $r_0 = a$, $r_1 = b$, $i = 1$.
2. Найти остаток r_{i+1} от деления r_{i-1} на r_i.
3. Если $r_{i+1} = 0$, то $\text{НОД} = r_i$.
4. Иначе увеличить i на 1 и вернуться к шагу 2.

Пример: Найдём НОД(30, 18):

- 30 ÷ 18 = 1 (остаток 12)

- 18 ÷ 12 = 1 (остаток 6)
- 12 ÷ 6 = 2 (остаток 0)
  НОД равен **6**.

2.3 Бинарный алгоритм Евклида

Эта оптимизированная версия использует двоичное представление чисел и заменяет деление операциями сдвига, что повышает производительность.

Основные принципы:

- Если оба числа чётные, то НОД содержит множитель 2.
- Если одно число чётное, его можно поделить на 2.
- Если оба нечётные, выполняется вычитание.

Алгоритм:

1. Инициализировать множитель $g = 1$.
2. Пока a и b чётные, делить их на 2 и удваивать g.
3. Присвоить $u = a$, $v = b$.
4. Выполнять цикл, пока $u \neq 0$:
   - Делить u и v на 2, пока они чётные.
   - Если $u \geq v$, то $u = u - v$, иначе $v = v - u$.
5. Результат: $d = g \cdot v$.

2.4 Расширенный алгоритм Евклида

Позволяет не только вычислить НОД, но и найти коэффициенты x и y для линейного представления:

$$a \cdot x + b \cdot y = \text{НОД}(a, b)$$

Это особенно полезно для решения диофантовых уравнений и в алгоритмах модульной арифметики.

Этапы алгоритма:

1. Инициализировать последовательности:
   $r_0 = a, r_1 = b, x_0 = 1, x_1 = 0, y_0 = 0, y_1 = 1, i = 1$.
2. Найти частное q_i и остаток r_{i+1} от деления r_{i-1} на r_i.
3. Если остаток равен 0, результат: $d = r_i, x = x_i, y = y_i$.
4. Иначе вычислить новые коэффициенты:
   $x_{i+1} = x_{i-1} - q_i \cdot x_i$,
   $y_{i+1} = y_{i-1} - q_i \cdot y_i$,
   увеличить i и вернуться к шагу 2.

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 = a % b
        else:
            b = 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 = g * 2
    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 = g * 2
    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)
a = int(input("Введите первое число (a): "))
b = int(input("Введите второе число (b): "))

result1 = euklid_simply(a, b)
print(f"1. Простой алгоритм Евклида:")
print(f"   НОД({a}, {b}) = {result1}")


result2 = euklid_extended(a, b)
print(f"\n2. Расширенный алгоритм Евклида:")
print(f"   НОД({a}, {b}) = {result2[0]}")
print(f"   Коэффициенты: {a}*{result2[1]} + {b}*{result2[2]} =
{result2[0]}")


result3 = euklid_binary(a, b)
print(f"\n3. Бинарный алгоритм Евклида:")
print(f"   НОД({a}, {b}) = {result3}")
```

```
result4 = euklid_bin_extended(a, b)
print(f"\n4. Расширенный бинарный алгоритм:")
print(f"   НОД({a}, {b}) = {result4[0]}")
print(f"   Коэффициенты: {a}*{result4[1]} + {b}*{result4[2]} =
{result4[0]}")
```

3.2 Контрольный пример

```
Введите первое число (a): 213
Введите второе число (b): 132
1. Простой алгоритм Евклида:
   НОД(213, 132) = 3

2. Расширенный алгоритм Евклида:
   НОД(213, 132) = 3
   Коэффициенты: 213*-13 + 132*21 = 3

3. Бинарный алгоритм Евклида:
   НОД(213, 132) = 3

4. Расширенный бинарный алгоритм:
   НОД(213, 132) = 3
   Коэффициенты: 213*-57 + 132*92 = 3
```

4. Практическая часть

В данном разделе представлены реализации алгоритмов и результаты тестирования на различных входных данных.
(Примечание: код и таблицы результатов могут быть добавлены по усмотрению студента.)

---

5. Заключение и выводы

В ходе выполнения лабораторной работы:

- Изучен классический алгоритм Евклида для нахождения НОД.
- Рассмотрены его оптимизированные версии: бинарный и расширенный алгоритмы.

- Приведены примеры ручного вычисления НОД каждым из методов.
- Установлено, что бинарный алгоритм более эффективен для программной реализации благодаря использованию операций сдвига.

Расширенный алгоритм Евклида имеет важное практическое применение в криптографических системах и теории чисел.

---

6. Список использованных источников

1. Введение в алгоритмы нахождения НОД.
2. Модификации алгоритма Евклида: сравнительный анализ.
3. Материалы лекций по теории чисел и алгоритмам.