Отчёт по лабораторной работе №4

Ван Яо

Содержание

Список таблиц

Список иллюстраций

Цель работы

1. Изучить математические понятия и свойства наибольшего общего делителя (НОД)
2. Реализовать четыре алгоритма вычисления НОД:
    - Классический алгоритм Евклида
    - Бинарный алгоритм Евклида
    - Расширенный алгоритм Евклида
    - Расширенный бинарный алгоритм Евклида
3. Провести сравнительный анализ производительности алгоритмов

Ход лабораторной работы

Теоретические основы

Определение НОД

Наибольший общий делитель целых чисел $a_1, a_2, ..., a_k$ обозначается НОД$(a_1, a_2, ..., a_k)$ и удовлетворяет условиям:

1. Каждое из чисел $a_1, a_2, ..., a_k$ делится на d
2. Если $d_1 \neq 0$ — другой общий делитель, то d делится на d_1

Основные свойства

- НОД$(a, b) =$ НОД(b, a)
- НОД$(a, b) =$ НОД$(a, b \mod a)$
- НОД$(a, b) =$ НОД$(a, b - ka)$ для любого $k \in \mathbb{Z}$

Алгоритмы реализации

1. Классический алгоритм Евклида

Алгоритм: - Вход: a, b (где $0 < b \leq a$) - Выход: $d =$ НОД(a, b)

1. Пока $b \neq 0$, выполнять:
   $a, b = b, a \mod b$
2. Вернуть $|a|$

Этот алгоритм основан на свойстве:
$\text{НОД}(a, b) = \text{НОД}(b, a \mod b)$

2. Бинарный алгоритм Евклида

Использует битовые операции вместо деления с остатком. Основан на следующих свойствах:

- Если оба числа чётные:
  $\text{НОД}(a, b) = 2 \cdot \text{НОД}\left(\frac{a}{2}, \frac{b}{2}\right)$
- Если a нечётное, b чётное:
  $\text{НОД}(a, b) = \text{НОД}\left(a, \frac{b}{2}\right)$
- Если оба нечётные:
  $\text{НОД}(a, b) = \text{НОД}(|a - b|, \min(a, b))$

Алгоритм эффективен благодаря замене деления побитовыми сдвигами.

3. Расширенный алгоритм Евклида

Находит не только $\text{НОД}(a, b)$, но и целые коэффициенты x, y, такие что:

$$ax + by = \text{НОД}(a, b)$$

Используется в криптографии (например, для вычисления обратных по модулю).

4. Расширенный бинарный алгоритм Евклида

Сочетает преимущества бинарного метода и расширенного алгоритма. Позволяет находить коэффициенты x, y с использованием побитовых операций, что ускоряет работу на больших числах. 1. **Основное тождество Безу с учётом бинарных преобразований:**

$$\gcd(a, b) = \gcd\left(\frac{a}{2}, b\right) \quad \text{при чётном } a \text{ и нечётном } b$$

Это позволяет заменить деление с остатком на побитовый сдвиг (`a >> 1`), что значительно ускоряет выполнение.

2. **Обновление коэффициентов при рекурсивном возврате:**

   Если на шаге рекурсии получено решение:

$$\gcd(a, b) = a \cdot x' + b \cdot y'$$

   и был применён сдвиг $ a →a/2 $, то при возврате коэффициенты обновляются как:

$$x = 2x', \quad y = y'$$

   (с коррекцией знака в зависимости от чётности и соотношения a и b).

Эти формулы лежат в основе эффективной реализации расширенного бинарного алгоритма, сочетающего скорость бинарного метода и функциональность расширенного алгоритма.

Практическая реализация

Пример кода

```python
def gcd_euclidean(a,b):
    a,b=abs(a),abs(b)

    while b != 0:
        a,b = b,a%b

    return a
```

Рис. 1: Классический алгоритм Евклида

```python
def gcd_binary(a,b):
    if a == 0:
        return abs(b)
    if b == 0:
        return abs(a)

    g = 0
    while ((a|b) & 1) == 0:
        a >>= 1
        b >>= 1
        g += 1

    while (a & 1) == 0:
        a >>= 1

    while b != 0:
        while (b & 1) == 0:
            b >>= 1
        if a > b:
            a,b = b,a
        b = b-a
    return a << g
```

```python
def extended_gcd(a,b):
    a,b=abs(a),abs(b)

    x1,y1 = 1,0
    x2,y2 = 0,1

    while b != 0:
        q =a // b
        r = a % b
        a,b = b,r

        x1,x2 = x2,x1-q*x2
        y1,y2 = y2,y1-q*y2
    return a, x1,y1
```

Функциональное тестирование

Тестовый пример	Ожидаемый результат	Фактический результат	Статус
НОД(12345, 24690)	12345	12345	√
НОД(12345, 54321)	3	3	√
НОД(12345, 12541)	1	1	√

Тестовый пример	Ожидаемый результат	Фактический результат	Статус
НОД(91, 105, 154)	7	7	√

Выводы

1. **Теоретические знания:**

   Углублено изучены математические основы НОД, включая его определение, свойства и связь с линейными комбинациями.

2. **Практические навыки:**

   Реализованы и протестированы четыре алгоритма вычисления НОД. Выявлены ошибки в расширенной версии, связанные с нормализацией коэффициентов.

3. **Аналитические способности:**

   Проведён детальный сравнительный анализ по времени выполнения. Бинарные алгоритмы показали преимущество на больших числах.

4. **Профессиональные компетенции:**

   Улучшены навыки отладки, тестирования и оптимизации алгоритмов, особенно важные в криптографии и системном программировании.

5. **Рекомендации:**

   Для высокопроизводительных систем рекомендуется использовать **бинарный алгоритм Евклида** или его расширенную версию. При необходимости нахождения коэффициентов — использовать корректно реализованный **расширенный алгоритм с нормализацией знаков**.

Литература

1. Кнут Д. Искусство программирования. Том 1. Основные алгоритмы. М.: Вильямс, 2006.
2. Python Software Foundation. Официальная документация Python. https://docs.python.org/3/

Приложения

Полный исходный код программ доступен в репозитории GitHub: https://github.com/wangyao200036/cryptography-labs/tree/main/lab4