Алгоритмы вычисления НОД

Теория, реализация и анализ

Студент: Ван Яо
Группа: НФИмд-01-25
РУДН, 2025

Содержание

1. Введение в НОД

• Определение НОД (Наибольший общий делитель) • Применения: криптография, сокращение дробей, решение диофантовых уравнений • Необходимость эффективных алгоритмов для больших чисел

2. Классический алгоритм Евклида

• **Принцип работы**: Использование деления с остатком
$\gcd(a, b) = \gcd(b, a \bmod b)$

• **Пример кода**:

```
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return abs(a)
```

3. Бинарный алгоритм Евклида

• **Преимущества**: Использование побитовых операций вместо деления с остатком - Если оба числа четные:
$\gcd(a, b) = 2 \cdot \gcd(a/2, b/2)$ - Если одно число четное, другое нечетное: $\gcd(a, b) = \gcd(a/2, b)$ или $\gcd(a, b/2)$ - Если оба числа нечетные: $\gcd(a, b) = \gcd(|a - b|, \min(a, b))$

4. Расширенный алгоритм Евклида

• **Цель**: Найти коэффициенты Безу x и y, такие что: $\gcd(a, b) = a \cdot x + b \cdot y$

• **Ключевая идея**: На каждом шаге поддерживать инвариант:$\gcd(a, b) = a \cdot x + b \cdot y$

• **Пример кода**:

```
def extended_gcd(a, b):
    if b == 0:
        return a, 1, 0
    g, x1, y1 = extended_gcd(b, a % b)
    return g, y1, x1 - (a // b) * y1
```

Выводы - Расширенный алгоритм Евклида позволяет не только найти НОД, но и выразить его как линейную комбинацию исходных чисел. - Алгоритм критически важен в криптографии (например, для вычисления обратного по модулю в RSA). - Реализация требует аккуратной обработки знаков и рекурсивного возврата коэффициентов. - Несмотря на большую вычислительную сложность по сравнению с классическим алгоритмом, расширенная версия незаменима в задачах, требующих коэффициентов Безу.

Спасибо за внимание!