Лабораторная работа №4

Вычисление наибольшего общего делителя

Каймакджыоглу Мерич Дорук

2025-10-25

Содержание I

Раздел 1

1. Информация

1.1 Докладчик

- **Каймакджыоглу Мерич Дорук**

1.1 Докладчик

- **Каймакджыоглу Мерич Дорук**
- Студент, кафедра Математического моделирования и искусственного интеллекта (ММиИИ)

1.1 Докладчик

- **Каймакджыоглу Мерич Дорук**
- Студент, кафедра Математического моделирования и искусственного интеллекта (ММиИИ)
- Российский университет дружбы народов

1.1 Докладчик

- **Каймакджыоглу Мерич Дорук**
- Студент, кафедра Математического моделирования и искусственного интеллекта (ММиИИ)
- Российский университет дружбы народов
- 1032245391@pfur.ru

1.1 Докладчик

- **Каймакджыоглу Мерич Дорук**
- Студент, кафедра Математического моделирования и искусственного интеллекта (ММиИИ)
- Российский университет дружбы народов
- 1032245391@pfur.ru
- https://github.com/dorukme123

Раздел 2

2. Вводная часть

2.1 Актуальность

- Алгоритмы вычисления НОД являются фундаментальными в вычислительной теории чисел и криптографии.

2.1 Актуальность

- Алгоритмы вычисления НОД являются фундаментальными в вычислительной теории чисел и криптографии.
- **Расширенный алгоритм Евклида** — это не просто теоретическое упражнение; это основной промышленный инструмент для нахождения **модульных обратных элементов**.

2.1 Актуальность

- Алгоритмы вычисления НОД являются фундаментальными в вычислительной теории чисел и криптографии.
- **Расширенный алгоритм Евклида** — это не просто теоретическое упражнение; это основной промышленный инструмент для нахождения **модульных обратных элементов**.
- Без этого алгоритма невозможно было бы вычисление закрытого ключа d в криптосистеме **RSA** (где $d \equiv e^{-1} \pmod{\phi(N)}$).

2.1 Актуальность

- Алгоритмы вычисления НОД являются фундаментальными в вычислительной теории чисел и криптографии.
- **Расширенный алгоритм Евклида** — это не просто теоретическое упражнение; это основной промышленный инструмент для нахождения **модульных обратных элементов**.
- Без этого алгоритма невозможно было бы вычисление закрытого ключа d в криптосистеме **RSA** (где $d \equiv e^{-1} \pmod{\phi(N)}$).
- Бинарные алгоритмы демонстрируют, как теоретико-числовые задачи оптимизируются для реальных аппаратных реализаций.

2.2 Объект и предмет исследования

- **Объект:** Алгоритмы вычислительной теории чисел, применяемые в криптографии.

2.2 Объект и предмет исследования

- **Объект:** Алгоритмы вычислительной теории чисел, применяемые в криптографии.
- **Предмет:** Алгоритм Евклида, бинарный алгоритм Евклида, расширенный алгоритм Евклида и расширенный бинарный алгоритм Евклида для вычисления НОД.

2.3 Цели и задачи

- **Цель:** Изучить и программно реализовать четыре различных алгоритма для нахождения НОД.

2.3 Цели и задачи

- **Цель:** Изучить и программно реализовать четыре различных алгоритма для нахождения НОД.
- **Задачи:**

2.3 Цели и задачи

- **Цель:** Изучить и программно реализовать четыре различных алгоритма для нахождения НОД.
- **Задачи:**
  - Реализовать классический алгоритм Евклида, основанный на делении с остатком.

2.3 Цели и задачи

- **Цель:** Изучить и программно реализовать четыре различных алгоритма для нахождения НОД.
- **Задачи:**
  - Реализовать классический алгоритм Евклида, основанный на делении с остатком.
  - Реализовать бинарный алгоритм, использующий сдвиги и вычитания.

2.3 Цели и задачи

- **Цель:** Изучить и программно реализовать четыре различных алгоритма для нахождения НОД.
- **Задачи:**
  - Реализовать классический алгоритм Евклида, основанный на делении с остатком.
  - Реализовать бинарный алгоритм, использующий сдвиги и вычитания.
  - Реализовать расширенный алгоритм Евклида для нахождения коэффициентов Безу (x, y).

2.3 Цели и задачи

- **Цель:** Изучить и программно реализовать четыре различных алгоритма для нахождения НОД.
- **Задачи:**
  - Реализовать классический алгоритм Евклида, основанный на делении с остатком.
  - Реализовать бинарный алгоритм, использующий сдвиги и вычитания.
  - Реализовать расширенный алгоритм Евклида для нахождения коэффициентов Безу (x, y).
  - Проверить корректность работы всех реализованных функций.

2.4 Материалы и методы

- **Язык программирования:** Python.

2.4 Материалы и методы

- **Язык программирования:** Python.
- **Алгоритмы:** Четыре алгоритма НОД, описанные в методических указаниях.

2.4 Материалы и методы

- **Язык программирования:** Python.
- **Алгоритмы:** Четыре алгоритма НОД, описанные в методических указаниях.
- **Математический аппарат:** Теория чисел, деление с остатком, тождество Безу (линейное представление НОД).

Раздел 3

3. Теоретические сведения & Реализация

3.1 Ключевые алгоритмы

- **1. Алгоритм Евклида:**

3.1 Ключевые алгоритмы

- **1. Алгоритм Евклида:**
  - **Принцип:** $\text{НОД}(a, b) = \text{НОД}(b, a \mod b)$.

3.1 Ключевые алгоритмы

- **1. Алгоритм Евклида:**
  - **Принцип:** НОД$(a, b) =$ НОД$(b, a \mod b)$.
  - **Завершение:** Когда $r = 0$, предыдущий r является НОД.

3.1 Ключевые алгоритмы

- **1. Алгоритм Евклида:**
  - **Принцип:** НОД$(a, b) =$ НОД$(b, a \mod b)$.
  - **Завершение:** Когда $r = 0$, предыдущий r является НОД.
- **2. Бинарный алгоритм Евклида:**

3.1 Ключевые алгоритмы

- **1. Алгоритм Евклида:**
  - **Принцип:** НОД$(a, b) =$ НОД$(b, a \mod b)$.
  - **Завершение:** Когда $r = 0$, предыдущий r является НОД.
- **2. Бинарный алгоритм Евклида:**
  - **Принцип:** Использует свойства четности.

3.1 Ключевые алгоритмы

- **1. Алгоритм Евклида:**
  - **Принцип:** НОД$(a, b) =$ НОД$(b, a \mod b)$.
  - **Завершение:** Когда $r = 0$, предыдущий r является НОД.
- **2. Бинарный алгоритм Евклида:**
  - **Принцип:** Использует свойства четности.
  - НОД$(a, b) = 2 \cdot$ НОД$(a/2, b/2)$

3.1 Ключевые алгоритмы

- **1. Алгоритм Евклида:**
  - **Принцип:** НОД$(a, b) =$ НОД$(b, a \mod b)$.
  - **Завершение:** Когда $r = 0$, предыдущий r является НОД.
- **2. Бинарный алгоритм Евклида:**
  - **Принцип:** Использует свойства четности.
  - НОД$(a, b) = 2 \cdot$ НОД$(a/2, b/2)$
  - НОД$(a, b) =$ НОД$(a - b, b)$

3.1 Ключевые алгоритмы

- **1. Алгоритм Евклида:**
  - **Принцип:** НОД$(a, b) =$ НОД$(b, a \mod b)$.
  - **Завершение:** Когда $r = 0$, предыдущий r является НОД.
- **2. Бинарный алгоритм Евклида:**
  - **Принцип:** Использует свойства четности.
  - НОД$(a, b) = 2 \cdot$ НОД$(a/2, b/2)$
  - НОД$(a, b) =$ НОД$(a - b, b)$
  - **Преимущество:** Заменяет дорогое деление (%) на быстрый сдвиг (») и вычитание.

3.1 Ключевые алгоритмы

- **1. Алгоритм Евклида:**
  - **Принцип:** НОД$(a, b) =$ НОД$(b, a \mod b)$.
  - **Завершение:** Когда $r = 0$, предыдущий r является НОД.
- **2. Бинарный алгоритм Евклида:**
  - **Принцип:** Использует свойства четности.
  - НОД$(a, b) = 2 \cdot$ НОД$(a/2, b/2)$
  - НОД$(a, b) =$ НОД$(a - b, b)$
  - **Преимущество:** Заменяет дорогое деление (%) на быстрый сдвиг (») и вычитание.
- **3. Расширенный алгоритм Евклида:**

3.1 Ключевые алгоритмы

- **1. Алгоритм Евклида:**
  - **Принцип:** НОД$(a, b) =$ НОД$(b, a \mod b)$.
  - **Завершение:** Когда $r = 0$, предыдущий r является НОД.
- **2. Бинарный алгоритм Евклида:**
  - **Принцип:** Использует свойства четности.
  - НОД$(a, b) = 2 \cdot$ НОД$(a/2, b/2)$
  - НОД$(a, b) =$ НОД$(a - b, b)$
  - **Преимущество:** Заменяет дорогое деление (%) на быстрый сдвиг (») и вычитание.
- **3. Расширенный алгоритм Евклида:**
  - **Цель:** Найти d, x, y, для которых $ax + by = d$.

3.1 Ключевые алгоритмы

- **1. Алгоритм Евклида:**
  - **Принцип:** НОД$(a, b) =$ НОД$(b, a \mod b)$.
  - **Завершение:** Когда $r = 0$, предыдущий r является НОД.
- **2. Бинарный алгоритм Евклида:**
  - **Принцип:** Использует свойства четности.
  - НОД$(a, b) = 2 \cdot$ НОД$(a/2, b/2)$
  - НОД$(a, b) =$ НОД$(a - b, b)$
  - **Преимущество:** Заменяет дорогое деление (%) на быстрый сдвиг (») и вычитание.
- **3. Расширенный алгоритм Евклида:**
  - **Цель:** Найти d, x, y, для которых $ax + by = d$.
  - **Метод:** Поддерживает x_i и y_i на каждом шаге i, выражая остаток r_i как линейную комбинацию a и b.

3.2 Демонстрация

1. Классический алгоритм Евклида

```
def euclidean_gcd(a, b):
    r_prev, r_curr = a, b
    # prev = a, curr = b

    while r_curr != 0:
        r_next = r_prev % r_curr
        r_prev = r_curr
        r_curr = r_next

    d = r_prev

    return d
```

3.2 Демонстрация

1. Классический алгоритм Евклида

```
def euclidean_gcd(a, b):
    r_prev, r_curr = a, b
    # prev = a, curr = b

    while r_curr != 0:
        r_next = r_prev % r_curr
        r_prev = r_curr
        r_curr = r_next

    d = r_prev

    return d
```

3.2 Демонстрация

1. Классический алгоритм Евклида

```
def euclidean_gcd(a, b):
    r_prev, r_curr = a, b
    # prev = a, curr = b

    while r_curr != 0:
        r_next = r_prev % r_curr
        r_prev = r_curr
        r_curr = r_next

    d = r_prev

    return d
```

3.2 Демонстрация

1. Классический алгоритм Евклида

```
def euclidean_gcd(a, b):
    r_prev, r_curr = a, b
    # prev = a, curr = b

    while r_curr != 0:
        r_next = r_prev % r_curr
        r_prev = r_curr
        r_curr = r_next

    d = r_prev

    return d
```

3.3 Выводы

- Задачи выполнены: Все четыре алгоритма НОД были успешно реализованы на Python в соответствии с заданием.

3.3 Выводы

- Задачи выполнены: Все четыре алгоритма НОД были успешно реализованы на Python в соответствии с заданием.
- Навыки: Получен практический опыт реализации фундаментальных алгоритмов теории чисел, лежащих в основе современной криптографии.