Отчет по лабораторной работе по предмету Математические основы защиты информации и информационной безопасности

Лабораторная работа №4. Вычисление наибольшего общего делителя

Никита Андреевич Топонен

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Цель работы — изучить алгоритмы нахождения наибольшего общего делителя.

2 Задание

- Реализовать алгоритмы нахождения наибольшего общего делителя:
    1. Алгоритм Евклида;
    2. Бинарный алгоритм Евклида;
    3. Расширенный алгоритм Евклида;
    4. Расширенный бинарный алгоритм Евклида.

3 Теоретическое введение

Наибольшим общим делителем двух чисел a и b называется наибольшее число, на которое a и b делятся без остатка. Для записи может использоваться аббревиатура НОД. Для двух чисел можно записать вот так: НОД (a, b).

Например, для 4 и 16 НОД будет 4. Как мы к этому пришли:

1. Зафиксируем все делители четырех: 4, 2, 1.
2. Все делители шестнадцати: 16, 8, 4 и 1.
3. Выбираем общие: это 4, 2, 1. Самое большое общее число: 4.

Взаимно простые числа — это натуральные числа, у которых только один общий делитель — единица. Их НОД равен 1.

4 Выполнение лабораторной работы

4.1 Алгоритм Евклида

В рамках данной лабораторной работы я реализовал алгоритм Евклида на языке Java. Ниже приведен код:

```
private static long GCD(long a, long b) {
    if (b <= 0 || b > a) {
        throw new RuntimeException("Cannot calculate GCD if b <= 0 or b > a")
    }

    List<Long> r = new ArrayList<>(List.of(a, b));

    while(r.get(r.size() - 1) != 0) {
        long rPrev = r.get(r.size() - 2);
        long rCurr = r.get(r.size() - 1);
        r.add(rPrev % rCurr);
    }

    return r.get(r.size() - 2);
}
```

4.2 Бинарный алгоритм Евклида

В рамках данной лабораторной работы я реализовал бинарный алгоритм Евклида на языке Java. Ниже приведен код:

```
    private static long binaryGCD(long a, long b) {
        if (b <= 0 || b > a) {
            throw new RuntimeException("Cannot calculate GCD if b <= 0 or b > a")
        }

        long g = 1;

        while (isEven(a) && isEven(b)) {
            a = a / 2;
            b = b / 2;
            g = g * 2;
        }

        long u = a;
        long v = b;

        while (u != 0) {
            if (isEven(u)) {
                u = u / 2;
            }

            if (isEven(v)) {
                v = v / 2;
            }
```

```
            if (u >= v) {
                u = u - v;
            } else {
                v = v - u;
            }
        }

        return g * v;
    }
```

4.3 Расширенный алгоритм Евклида

В рамках данной лабораторной работы я реализовал расширенный алгоритм Евклида на языке Java. Ниже приведен код:

```
    private static long extendedGCD(long a, long b) {
        if (b <= 0 || b > a) {
            throw new RuntimeException("Cannot calculate GCD if b <= 0 or b > a")
        }

        List<Long> r = new ArrayList<>(List.of(a, b));
        List<Long> x = new ArrayList<>(List.of(1L, 0L));
        List<Long> y = new ArrayList<>(List.of(0L, 1L));
        long q;

        while (r.get(r.size() - 1) != 0) {
            long rPrev = r.get(r.size() - 2);
            long rCurr = r.get(r.size() - 1);
            long rNext = rPrev % rCurr;
            r.add(rNext);
```

```
            q = rPrev / rCurr;
            if (rNext != 0) {
                long xPrev = x.get(x.size() - 2);
                long xCurr = x.get(x.size() - 1);
                x.add(xPrev - q * xCurr);
                long yPrev = y.get(y.size() - 2);
                long yCurr = y.get(y.size() - 1);
                y.add(yPrev - q * yCurr);

            }
        }

        return r.get(r.size() - 2);
    }
```

4.4 Расширенный бинарный алгоритм Евклида

В рамках данной лабораторной работы я реализовал расширенный бинарный алгоритм Евклида на языке Java. Ниже приведен код:

```
    private static long extendedBinaryGCD(long a, long b) {
        if (b <= 0 || b > a) {
            throw new RuntimeException("Cannot calculate GCD if b <= 0 or b > a")
        }

        long g = 1;

        while (isEven(a) && isEven(b)) {
            a = a / 2;
            b = b / 2;
```

```
    g = 2 * g;
}

long u = a;
long v = b;
long A = 1;
long B = 0;
long C = 0;
long D = 1;

while (u != 0) {
    while (isEven(u)) {
        u = u / 2;
        if (isEven(A) && isEven(B)) {
            A = A / 2;
            B = B / 2;
        } else {
            A = (A + b) / 2;
            B = (B - a) / 2;
        }
    }
    while (isEven(v)) {
        v = v / 2;
        if (isEven(C) && isEven(D)) {
            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;
        }
    }

    return g * v;
}
```

4.5 Проверка реализации

Для проверки выполнил нахождение НОД(105, 91) всеми алгоритмами. В итоге получил одинаковые ответы.

```
public static void main(String[] args) {
    long a = 105L;
    long b = 91L;

    System.out.println("GCD: " + GCD(a, b));
    System.out.println("binaryGCD: " + binaryGCD(a, b));
    System.out.println("extendedGCD: " + extendedGCD(a, b));
    System.out.println("extendedBinaryGCD: " + extendedBinaryGCD(a, b));
}
```

Результаты выполнения программы на иллюстрации (рис. 4.1).

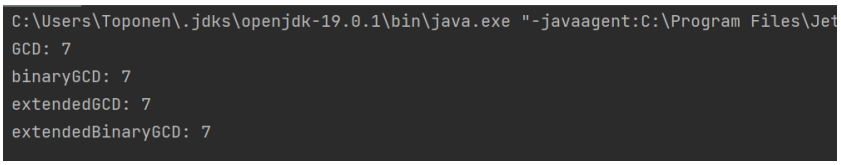

Рис. 4.1: Результаты проверки

5 Выводы

В рамках данной лабораторной работы я изучил алгоритмы нахождения наибольшего общего делителя. Также реализовал алгоритмы на языке Java.

Список литературы