Лабораторная работа №4

Алгоримы вычисления наибольшего общего делителя

Дугаева Светлана Анатольевна, НФИмд-02-22

Содержание

1 Цель работы **5**

2 Задание **6**

3 Теоретическое введение **7**
- 3.1 Алгоритм Евклида 7
- 3.2 Бинарный алгоритм Евклида 7
- 3.3 Расширенный алгоритм Евклида 8
- 3.4 Расширенный бинарный алгоритм Евклида 8

4 Выполнение лабораторной работы **9**
- 4.1 Реализация алгоритма Евклида 9
- 4.2 Реализация бинарного алгоритма Евклида 10
- 4.3 Реализация расширенного алгоритма Евклида 10
- 4.4 Реализация расширенного бинарного алгоритма Евклида 11

5 Выводы **13**

Список иллюстраций

4.1 Алгоритм Евклида 9
4.2 Бинарный алгоритм Евклида 10
4.3 Расширенный алгоритм Евклида 11
4.4 Расширенный бинарный алгоритм Евклида (2) 12

Список таблиц

1 Цель работы

Цель данной лабораторной работы изучение нахождения наибольшего общего делителя при помощи алгоритма Евклида и его адаптаций.

2 Задание

Заданием является:

- Реализовать алгоритм Евклида;
- Реализовать бинарный алгоритм Евклида;
- Реализовать расширенный алгоритм Евклида;

- Реализовать расширенный бинарный алгоритм Евклида.

3 Теоретическое введение

3.1 Алгоритм Евклида

Алгори́тм Евкли́да — эффективный алгоритм для нахождения наибольшего общего делителя двух целых чисел (или общей меры двух отрезков).

В самом простом случае алгоритм Евклида применяется к паре положительных целых чисел и формирует новую пару, которая состоит из меньшего числа и разницы между большим и меньшим числом. Процесс повторяется, пока числа не станут равными. Найденное число и есть наибольший общий делитель исходной пары.

Для данного алгоритма существует множество теоретических и практических применений. В частности, он является основой для криптографического алгоритма с открытым ключом RSA, широко распространённого в электронной коммерции. Также алгоритм используется при решении линейных диофантовых уравнений, при построении непрерывных дробей, в методе Штурма. Алгоритм Евклида является основным инструментом для доказательства теорем в современной теории чисел, например таких как теорема Лагранжа о сумме четырёх квадратов и основная теорема арифметики.

3.2 Бинарный алгоритм Евклида

Бинарный алгоритм Евклида — метод нахождения наибольшего общего делителя двух целых чисел. Данный алгоритм “быстрее” обычного алгоритма

Евклида, т.к. вместо медленных операций деления и умножения используются сдвиги.

Он основан на использовании следующих свойств НОД:

- НОД(2m, 2n) = 2 НОД(m, n),
- НОД(2m, 2n+1) = НОД(m, 2n+1),
- НОД(-m, n) = НОД(m, n).

3.3 Расширенный алгоритм Евклида

Расширенный алгоритм Евклида — это алгоритм определения коэффициентов, позволяющих выразить наибольший общий делитель числовой пары через эти два числа, т.е. вычислить d = *НОД (a, b)* и в то же самое время вычислить значения x и y, такие что $ax + by = d$.

3.4 Расширенный бинарный алгоритм Евклида

Расширенный бинарный алгоритм Евклида является, как очевидно из названия, квинтэссенцией расширенного и бинарного алгоритмов. Таким образом, при вычислении НОД используются сдвиги, как в бинарном алгоритме, и при этом на выходе можно получить значения коэффициентов x и y, как в расширенном алгоритме.

4 Выполнение лабораторной работы

Для реализации шифров мы будем использовать Python, так как его синтаксис позволяет быстро реализовать необходимые нам алгоритмы.

4.1 Реализация алгоритма Евклида

Задам функцию *euclid()*, в которую буду передавать два числа. По алгоритму Евклида найду НОД и передам его как результат выполнения функции.

Вызову функцию для чисел 15625 и 125. Алгоритм верно находит НОД = 125.

```
def euclid (a, b):
    while a!= 0 and b != 0:
        if a>b:
            a %= b
        else:
            b %= a
    return a or b
```

```
euclid (15625, 125)
```

```
125
```

Рис. 4.1: Алгоритм Евклида

4.2 Реализация бинарного алгоритма Евклида

Задам функцию *binary_euclid()*, в которую буду передавать два числа. По бинарному алгоритму Евклида найду НОД и передам его как результат выполнения функции.

Вызову функцию для чисел 15625 и 125. Алгоритм верно находит НОД = 125.

```python
def binary_euclid (a, b):
    if a == b:
        return a
    g = 0
    while (a | b) & 1 == 0:
        g += 1
        a >>= 1
        b >>= 1
    while a & 1 == 0:
        a >>= 1
    while b != 0:
        while b & 1 == 0:
            b >>= 1
        if a > b:
            a, b = b, a
        b -= a
    return a << g
```

```python
binary_euclid (15625, 125)
```

```
125
```

Рис. 4.2: Бинарный алгоритм Евклида

4.3 Реализация расширенного алгоритма Евклида

Задам функцию *ext_euclid()*, в которую буду передавать два числа. По расширенному алгоритму Евклида найду НОД, коэффициенты *x* и *y*, затем передам их как результат выполнения функции.

Вызову функцию для чисел 15625 и 125. Алгоритм верно находит НОД = 125, *x* = *0* и *y* = *1*: 15625x0 + 125x1 = 125.

```
def ext_euclid (a, b):
    if a == 0:
        y = 0
        x = 1
        return b, y, x
    else:
        d, x, y = ext_euclid(b%a, a)
    return d, y - (b//a)*x, x
```

```
ext_euclid(15625, 125)
```

```
(125, 0, 1)
```

Рис. 4.3: Расширенный алгоритм Евклида

4.4 Реализация расширенного бинарного алгоритма Евклида

Задам функцию *ext_bin_euclid()*, в которую буду передавать два числа. По расширенному бинарному алгоритму Евклида найду НОД, коэффициенты *x* и *y*, затем передам их как результат выполнения функции.

Вызову функцию для чисел 15625 и 125. Алгоритм верно находит НОД = 125, *x* = *0* и *y* = *1*: 15625x0 + 125x1 = 125.

```
def ext_bin_euclid (a, b):
    g = 1
    while (a % 2 == 0) and (b % 2 == 0):
        a /= 2
        b /= 2
        g *= 2
    u = a
    v = b
    A = 1
    B = 0
    C = 0
    D = 1
    while u != 0:
        while u % 2 == 0:
            u /= 2
            if (A % 2  == 0) and (B % 2 == 0):
                A /= 2
                B /= 2
            else:
                A = (A + b)/2
                B = (B - a)/2
```

```python
        while v % 2 == 0:
            v /= 2
            if (C % 2  == 0) and (D % 2 == 0):
                C /= 2
                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
```

```python
ext_bin_euclid (15625, 125)
```

```
(125, 0, 1)
```

Рис. 4.4: Расширенный бинарный алгоритм Евклида (2)

5 Выводы

В ходе данной лабораторной работы я реализовала четыре алгоритма нахождения НОД.