Отчёт по лабораторной работе №4. Вычисление наибольшего общего делителя

Дисциплина: Математические основы защиты информации и информационной безопасности

Студент: Агеева Анастасия Сергеевна, 1032212304
Группа: НФИмд-02-21
Преподаватель: д-р.ф.-м.н., проф. Кулябов Дмитрий Сергеевич

4 декабря, 2021, Москва

Прагматика

Прагматика данной лабораторной работы

- В рамках дисциплины “Математические основы защиты информации и информационной безопасности” нам необходимо изучить ее разделы. Данная лабораторная работа входит в раздел “Основы блочного шифрования”.
- Данная работа необходима для более глубоко и детального понимания работы алгоритмов шифрования.

Цель

Цель выполнения данной лабораторной работы

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

Задачи

Задачи выполнения данной лабораторной работы

1. Реализовать алгоритм Евклида;
2. Реализовать бинарный алгоритм Евклида;
3. Реализовать расширенный алгоритм Евклида;
4. Реализовать расширенный бинарный алгоритм Евклида.

Результаты выполнения данной лабораторной работы

Алгоритм Евклида

In [1]:
```
def euclid(a, b):
    while a != 0 and b != 0:
        if a >= b:
            a %= b
        else:
            b %= a
    return a or b
```

In [2]:
```
nod = euclid(12345, 24690)
nod
```

Out[2]: 12345

Figure 1: Алгоритм Евклида

Бинарный алгоритм Евклида

In [3]:
```
def binary_euclid(a, b):
    if a == 0:
        return b
    if b == 0:
        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
```

In [4]:
```
nod = binary_euclid(12345, 24690)
nod
```

Out[4]: 12345

Figure 2: Бинарный алгоритм Евклида

Расширенный алгоритм Евклида

In [5]:
```
def extended_euclid(a, b):
    if a == 0:
        y = 0
        x = 1
        return b, y, x
    else:
        d, x, y = extended_euclid(b%a, a)
    return d, y - (b//a)*x, x
```

In [6]:
```
nod = extended_euclid(12345, 24690)
nod
```

Out[6]:
```
(12345, 1, 0)
```

Figure 3: Расширенный алгоритм Евклида

Расширенный бинарный алгоритм Евклида (1)

In [7]:

```
def ext_bi_euclid(a, b):
    if a < b:
        a, b = b, a
    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
        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
```

Figure 4: Расширенный бинарный алгоритм Евклида (1)

Расширенный бинарный алгоритм Евклида (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
```

In [8]:
```
nod = ext_bi_euclid(12345, 24690)
nod
```

Out[8]: (12345, 0, 1)

Figure 5: Расширенный бинарный алгоритм Евклида (2)

Выводы

- Исходя из теоретических сведений, программы выполнены без ошибок, чему свидетельствуют полученные результаты.
- В ходе данной лабораторной работы я реализовала четыре алгоритма нахождения НОД.