**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**
**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

# 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: Кулябов Дмитрий Сергеевич,
д-р.ф.-м.н., проф.

Москва 2021

# Содержание

# List of Figures

# 1 Цель работы

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

# 2 Задание

1. Реализовать алгоритм Евклида;
2. Реализовать бинарный алгоритм Евклида;
3. Реализовать расширенный алгоритм Евклида;
4. 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) [1].

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 [2]. 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].

## 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

1. **Реализация алгоритма Евклида**

    1. Задам функцию *euclid()*, в которую буду передавать два числа. По алгоритму Евклида найду НОД и передам его как результат выполнения функции.
    2. Вызову функцию для чисел 12345 и 24690. Алгоритм верно находит НОД = 12345.

```
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 4.1: Алгоритм Евклида

2. **Реализация бинарного алгоритма Евклида**

    1. Задам функцию *binary_euclid()*, в которую буду передавать два числа. По бинарному алгоритму Евклида найду НОД и передам его как результат выполнения функции.
    2. Вызову функцию для чисел 12345 и 24690. Алгоритм верно находит НОД = 12345.


Figure 4.2: Бинарный алгоритм Евклида

3. **Реализация расширенного алгоритма Евклида**

    1. Задам функцию *extended_euclid()*, в которую буду передавать два числа. По расширенному алгоритму Евклида найду НОД, коэффициенты *x* и *y*, затем передам их как результат выполнения функции.
    2. Вызову функцию для чисел 12345 и 24690. Алгоритм верно находит НОД = 12345, *x = 1* и *y = 0*: 12345x1 + 24690x0 = 12345.


Figure 4.3: Расширенный алгоритм Евклида

4. **Реализация расширенного бинарного алгоритма Евклида**

    1. Задам функцию *ext_bi_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.
    2. Вызову функцию для чисел 12345 и 24690. Алгоритм верно находит НОД = 12345, *x = 1* и *y = 0*: 12345x1 + 24690x0 = 12345.

```
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
        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 4.4: Расширенный бинарный алгоритм Евклида (2)

# 5 Выводы

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

# Список литературы


1. Алгоритм Евклида [Электронный ресурс]. Википедия, 2019. URL: https://ru.wikipedia.org/wiki/Алгоритм_Евклида.
2. Бинарный алгоритм Евклида [Электронный ресурс]. Википедия, 2015. URL: https://ru.wikipedia.org/wiki/Бинарный_алгоритм_вычисления_НОД.
3. Расширенный алгоритм Евклида [Электронный ресурс]. НОУ ИНТУИТ, 2015. URL: https://intuit.ru/studies/courses/552/408/lecture/9351?page=3.