Лабораторная работа №4

Вычисление наибольшего общего делителя

- Кюнкриков Даниил Саналович
- студент уч. группы НПИмд-01-24
- Российский университет дружбы народов
- 1132249574@pfur.ru
- https://github.com/DanzanK/2025-2026_math-sec/tree/main

Объект и предмет исследования

- Объект исследования - Алгоритмы вычисления наибольшего общего делителя (НОД)
- Предмет исследования - алгоритмы:
  - Алгоритм Евклида
  - Бинарный алгоритм Евклида
  - Расширенный алгоритм Евклида
  - Расширенный бинарный алгоритм Евклида

Цели и задачи

- Цель работы: Программная реализация алгоритмов вычисления наибольшего общего делителя на языке программирования python
- Задачи: реализовать программно алгоритмы:
  - Алгоритм Евклида
  - Бинарный алгоритм Евклида
  - Расширенный алгоритм Евклида
  - Расширенный бинарный алгоритм Евклида
- продемонстрировать работу алгоритмов

Теоретическая часть

- Наибольший одбщий делитель (ГОД) двух целых чисел a и b - наибольшее целое число d на которое делятся и a и b без отстатка
- Соглано основной лемме Евклида: a = qb + r || / из этого следует, что $\text{НОД}(a, b) = \text{НОД}(b, r)$
- Алгоритм Евклида рекурсивно применяет данную лемму, заменяя пару (a, b) парой $(b, a \mod b)$ пока остаток не равен нулю Последний ненулево остаток является НОД
- Бинарный алгоритм Евклида оптимизирует процесс для вычислительных машин, заменяя дорогостоящую операцию деления на быстрые битовые сдвиги 9деление на 2) и вычитание Он использует следующие свойства:
- $\text{НОД}(a, b) = 2 \cdot \text{НОД}(a/2, b/2)$, если a, b четные
- $\text{НОД}(a, b) = \text{НОД}(a, b/2)$, если a нечетное, b четное

- \(\text{НОД}(a, b) = \text{НОД}(a-b, b)\), если a, b нечетные
- Расширенный алгоритм Евклида не только находит d = (a, b)\(, но и находит пару чисел (x, y), известные как коэффициенты Безу, которые удовлетворяют тождеству:\)\(ax + by = d\)$ Это тождество является представлением НОД в виде линейной комбинации исходных чисел

Процесс выполнения работы

Реализация алгоритма Евклида

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

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

    d = r_prev
    return d

vala = 12345
valb = [24690, 54321, 12541]
for val in valb:
    print(f"GCD({vala}, {val}) = {euclidean_gcd(vala, val)}")
```

Реализация бинарного алгоритма Евклида

```
def binareuc_gcd(a,b):
    g = 1

    while a%2 == 0 and b%2 ==0:
        a //=2
        b //= 2
        g *= 2

    u, v = a, b

    while u != 0:
        while u% 2 ==0:
            u //= 2
        while v% 2 ==0:
            v //= 2
        if u >= v:
            u = u-v
        else:
            v = v-u
```

```
    d = g * v

    return d

vala = 12345
valb = [24690, 54321, 12541]
for val in valb:
    print(f"GCD({vala}, {val}) = {binareuc_gcd(vala, val)}")
```

Реализация расширенного алгоритма Евклида

```
def extendedeuc_gcd(a,b):
    r_prev, r_curr = a,b
    x_prev, x_curr = 1, 0
    y_prev, y_curr = 0, 1

    while r_curr !=0:
        q = r_prev // r_curr
        r_next = r_prev % r_curr
        x_next = x_prev - q * x_curr
        y_next = y_prev - q * y_curr

        r_prev, r_curr = r_curr, r_next
        x_prev, x_curr = x_curr, x_next
        y_prev, y_curr = y_curr, y_next

    d, x, y = r_prev, x_prev, y_prev
    return d, x, y_curr

d1, x1, y1 = extendedeuc_gcd(105,91)
d2, x2, y2 = extendedeuc_gcd(154, d1)
print(f"GDC(105,91) = {d1, x1, y1}")
print(f"final GDC(154,7) = {d2, x2, y2}")
```

Реализация расширенного бинарношго алгоритма Евклида

```
def extendbineuc_gcd(a,b):
    g = 1
    while a% 2 ==0 and b%2 == 0:
        a //=2
        b //= 2
        g *= 2

    u, v = a, b
    A, B = 1, 0
    C, D = 0, 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
            print("v: " + str(v))

        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
            print("u: " + str(u))

        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

dbin,xbin,ybin = extendbineuc_gcd(105, 91)
print(f"GDC(105,91) = {dbin,xbin,ybin}")
```

Результат работы программы

```
GCD(12345, 24690) = 12345
GCD(12345, 54321) = 3
GCD(12345, 12541) = 1
GCD(12345, 24690) = 12345
GCD(12345, 54321) = 3
GCD(12345, 12541) = 1
GDC(105,91) = (7, -6, -15)
final GDC(154,7) = (7, 0, -22)
v: 91
u: 7
u: 7
u: 7
GDC(105,91) = (7, -6, 7)
```

Вывод

В ходе выполнения лабораторной работы были успешно реализованы все четыре алгоритма вычисления НОД