

Лабораторная работа 4

Серенко Данил Сергеевич, НФИмд-01-23

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

ПРЕЗЕНТАЦИЯ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

дисциплина: Математические основы защиты информации и информационной безопасности

Преподаватель: Кулябов Дмитрий Сергеевич

Студент: Серенко Данил Сергеевич

Группа: НФИмд-01-23

МОСКВА

2023 г.

Прагматика выполнения лабораторной работы

Требуется реализовать:

1. Алгоритм Евклида
2. Бинарный алгоритм Евклида
3. Расширенный алгоритм Евклида
4. Расширенный бинарный алгоритм Евклида

Цель работы

Освоить на практике алгоритмы вычисления НОД.

Выполнение лабораторной работы

1. Алгоритм Евклида

1. Берёт два числа a и b , где $a > b$
2. Повторяет деление a на b , заменяя a значением b и b остатком от деления, пока b не станет равным 0.
3. Последнее ненулевое значение a будет НОД.

```
def euclidean_gcd(a, b):
    r0, r1, i = a, b, 1
    while True:
        ri_next = r0 % r1
        if ri_next == 0:
            d = r1
            return d
        r0, r1, i = r1, ri_next, i + 1
```

gdc_eucl

2. Бинарный алгоритм Евклида

1. Если оба числа четные, делит оба числа на 2 и удваивает итоговый НОД
2. Если только одно из чисел четное, делит только его на 2.
3. Из большего числа вычитает меньшее.
4. Повторяет процесс, пока числа не станут равными. Это число становится НОД, умноженным на полученный ранее множитель.

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

    u, v = a, b
    while u != 0:
        while u % 2 == 0:
            u //= 2
        while v % 2 == 0:
            v //= 2
        if u >= v:
            u -= v
        else:
            v -= u
    return g * v
```

gdc_binary

3. Расширенный алгоритм Евклида

1. Кроме нахождения НОД, алгоритм находит такие числа x и y , что $ax+by=\text{НОД}(a,b)$.
2. Начинается с базовых коэф.: $x_0 = 1, y_0 = 0$ (для a) и $x_1 = 0, y_1 = 1$ (для b).
3. При каждом шаге обновляются значения коэффициентов, используя остаток и частное от деления.

```
def extended_gcd(a, b):  
    r0, r1, x0, x1, y0, y1, i = a, b, 1, 0, 0, 1, 1  
    while True:  
        q, ri_next = divmod(r0, r1)  
        if ri_next == 0:  
            return r1, x1, y1  
        x_next = x0 - q * x1  
        y_next = y0 - q * y1  
        r0, r1, x0, x1, y0, y1, i = r1, ri_next, x1, x_next, y1, y_next, i + 1
```

gdc_extended

4. Расширенный бинарный алгоритм Евклида

1. Как и обычный бинарный алгоритм, но также отслеживает коэффициенты x и y .
2. Когда числа делятся на 2, коэффициенты корректируются соответствующим образом.
3. Когда из одного числа вычитается другое, соответствующие коэффициенты также вычитаются.

```

43 def extended_binary_gcd(a, b):
44     g = 1
45     while a % 2 == 0 and b % 2 == 0:
46         a, b, g = a // 2, b // 2, 2 * g
47
48     u, v, A, B, C, D = a, b, 1, 0, 0, 1
49     while u != 0:
50         while u % 2 == 0:
51             u //= 2
52             if A % 2 == 0 and B % 2 == 0:
53                 A, B = A // 2, B // 2
54             else:
55                 A, B = (A + b) // 2, (B - a) // 2
56
57         while v % 2 == 0:
58             v //= 2
59             if C % 2 == 0 and D % 2 == 0:
60                 C, D = C // 2, D // 2
61             else:
62                 C, D = (C + b) // 2, (D - a) // 2
63
64         if u >= v:
65             u, A, B = u - v, A - C, B - D
66         else:
67             v, C, D = v - u, C - A, D - B
68
69     return g * v, C, D
70

```

gdc_extended_binary

Результат работы программы

```

a, b = 56, 98

print("Using Euclidean Algorithm:", euclidean_gcd(a, b))
print("Using Binary Euclidean Algorithm:", binary_gcd(a, b))
print("Using Extended Euclidean Algorithm:", extended_gcd(a, b)[0]) # Just the gcd, not x and y
print("Using Extended Binary Euclidean Algorithm:", extended_binary_gcd(a, b)[0]) # Just the gcd, not x and y
print("Using Python's built-in math.gcd:", math.gcd(a, b))

```

output

Выводы

В результате выполнения работы я освоил на практике применение алгоритмов нахождения НОД.