Лабораторная работа №4

Цель работы

Освоить на практике реализацию нахождения НОД разнообразными методами Евклида.

Задание

Реализовать с помощью программирования программы нахождения НОД, методами, описанными в задании к лабораторной работе №4.

Теоретическое введение

Эффективный алгоритм для нахождения наибольшего общего делителя двух целых чисел. Алгоритм назван в честь греческого математика Евклида, который впервые описал его в VII и X книгах «Начал». Это один из старейших численных алгоритмов, используемых в наше время. [1].

Аббревиатура НОД расшифровывается как «наибольший общий делитель». Наибольший общий делитель — делитель, который делит без остатка два числа, при этом сам делится без остатка на любой другой делитель исходных двух чисел. То есть это самое большое число, на которое без остатка можно разделить пару чисел, для которых подбирается НОД.[2]

Суть алгоритма заключается в формировании новой пары чисел из меньшего и разницы между большим и меньшим элементом. Процесс, состоящий из арифметических операций, повторяется до тех пор, пока числа не будут равны друг другу. Первоначально инструкция создавалась для натуральных чисел и геометрических величин. Однако в XIX веке ее расширили и стали применять для других объектов: целых чисел Гаусса и многочленов с одной переменной (полиномов).[3].

Оборудование

Лабораторная работа выполнялась дома со следующими характеристиками техники:

– Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.81GHz
– ОС Майкрософт Windows 10
– VirtualBox верс. 6.1.26

Выполнение лабораторной работы

1. Написала программу для простого алгоритма Евклида.

(рис. -@fig:001)

```
d=int(input("Введите 1 число"))
k=int(input("Введите 2 число"))
#d, a - большее число
#k, b - меньшее

if k>d:
  a,b = d,k
else:
  a,b = k,d

c=100

while c!=0:
  c=a%b
  if c!=0:
    a,b=b,c

print("НОД (", d,", ",k,") = ",b)
```

```
Введите 1 число15
Введите 2 число3
НОД ( 15 ,  3 ) =  3
```

Первоначально мы вводим 2 числа для поиска НОД. Мы ищем большое число, если два числа не равны. Далее мы делим большее число на меньшее и остаток от деление записываем в новую переменную(если он есть). Теперь как делимое мы записываем прошлый делитель - a=b, а как делитель новый мы записываем остаток - b=c. И так продолжается, пока при делении двух новых чисел не останется остатка от деления. Когда такое произойдет, мы запишем как наибольший общий делитель (НОД) число, которое было последним без остаточным делителем b, при котором c==0.

2. Реализовала программу для расширенного алгоритма Евклида.

(рис. -@fig:002)

```
a=int(input("Введите 1 число:"))
b=int(input("Введите 2 число:"))

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

print("НОД (", a,", ",b,") = ",d)
print("x : ", x,", ","y : ",y)
```

```
Введите 1 число:12
Введите 2 число:4
НОД ( 12 ,  4 ) =  4
x :  0 ,  y :  1
```

Он отличается от обычного алгоритма тем, что ищет также дополнительные x и y, которые соответствовали бы условию $ax+by=d$, где d - наибольший общий делитель. Это рекурсивный алгоритм, где условием выхода из цикла является b=0, в таком случае функция возвращает d =a, x=1, y=0.

На слайде можно увидеть, как если с найденными числами по алгоритму составить уравнение $ax+by=d$, то мы получим 0+4=4, что соответствует верному решению

3. Реализовала бинарный алгоритм Евклида.

(рис. -@fig:003)

```
A=int(input("Введите 1 число: "))
B=int(input("Введите 2 число: "))
def binary_gcd(A, B):
    k = 1
    while (A != 0) and (B != 0):
        if A > B:q = a // b
        else:q = b // a

        while (A % 2 == 0) and (B % 2 == 0): A /= 2; B /= 2; k *= 2
        while A % 2 == 0: A /= 2
        while B % 2 == 0: B /= 2

        if A >= B: A -= B
        else: B -= A
    return B * k

d=binary_gcd(A, B)

print("НОД (", A,", ",B,") = ",d)
```

```
Введите 1 число:15
Введите 2 число:3
НОД ( 15 ,  3 ) =  3
```

Бинарный алгоритм Евклида также как и обычный ищет НОД методом остатка от деления, но также использует одно из свойств НОД. В данном случае свойство НОД(2*a*,2b)=2НОД(a,b). И даже если одно из чисел нечетное, то можно использовать свойство НОД(a,2*b)=НОД(a,b), так как двойка все равно не попадет в результирующее НОД. И последнее используемое свойство если a>b НОД (a,b)=НОД(a-b,b).

Также этот алгоритм занимает меньше времени на обработку компьютером, так как процессорам проще проделывать операции деления и умножения на 2 из-за двоичного кода.

4.Реализовала бинарный расширенный алгоритм Евклида.

(рис. -@fig:004)

```
def ext_gcd(a, b):
    if a == 0: return 1, 1, b
    if b == 0: return 1, 1, a
    if not a&1 | b&1:
        # оба чётные - x и y не трогаем, gcd удваеваем
        x, y, g = ext_gcd(a>>1, b>>1)
        return x, y, g<<1
    elif not a&1:
        # a - чётное, b - нечётное
        x, y, g = ext_gcd(a>>1, b)
        return (x-b>>1, y+(a>>1), g) if x&1 else (x>>1, y, g)
    elif not b&1:
        # a - нечётное, b - чётное
        x, y, g = ext_gcd(a, b>>1)
        return (x+(b>>1), y-a>>1, g) if y&1 else (x, y>>1, g)
    elif b > a:
        # оба нечётные
        x, y, g = ext_gcd(a, b-a)
        return x-y, y, g
    else:
        # оба нечётные
        x, y, g = ext_gcd(a-b, b)
        return x, y-x, g

a,b = 36,54
x,y,g=ext_gcd(a,b)
print(f"({x})*{a} + ({y})*{b} = {g}") # (-13)*36 + (9)*54 = 18
```

```
(-13)*36 + (9)*54 = 18
```

Этот алгоритм представляет собой рекурсионный бинарный алгоритм, но с дополнительным вычислением x и y, как в пункте 2.

По выводу данной программы, можно сделать вывод, что программа работает верно.

Выводы

Освоила на практике написание 4 разных вариантов реализации кода для нахождения НОД алгоритмами Еклида.

Список литературы

1. Алгоритм Евклида // Wikipedia URL: https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%95%D0%B2%D0%BA%D0%BB%D0%B8%D0%B4%D0%B0 (дата обращения: 24.12.2021).

2. Методы нахождения НОД // ФоксФОРД URL: https://foxford.ru/wiki/matematika/algoritm-evklida (дата обращения: 25.12.2021).
3. НОД Евклида // Наука клубТ URL: https://nauka.club/matematika/algoritm-evklida.html (дата обращения: 27.10.2021).