Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
дисциплины «Алгоритмизация»
Вариант 29

Выполнил:
Саенко Андрей Максимович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», заочная
форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., канд. технических
наук, доцент кафедры
инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Порядок выполнения работы:

Задание 1. Алгоритмы поиска чисел Фибоначчи

Код программы:

```
import timeit
import matplotlib.pyplot as plt
def FibRecursive(n):
    if n<=1:
        return n
    else:
        return FibRecursive(n-1)+FibRecursive(n-2)
def FibArray(n):
    F=[0 for i in range(0,n+1)]
    F[0],F[1]=0,1
    for i in range(2,n+1):
        F[i]=F[i-1]+F[i-2]
    return F[n]
time_fib_recursive =(timeit.timeit(lambda: FibRecursive(10),
number=100000))/100000
time_fib_array = (timeit.timeit(lambda: FibArray(10), number=100000))/100000
FR=FibRecursive(10)
FA=FibArray(10)
print("FibRecursive(10) =", FR, ",время выполнения:", time_fib_recursive)
print("FibArray(10) =",FA, ",время выполнения:", time_fib_array)
if time_fib_recursive < time_fib_array:
    print("Функция FibRecursive быстрее")
else:
    print("Функция FibArray быстрее")
x=[1,2,3,4,5,6,7,8,9,10]
y1=[]
y2=[]
c=10000
for i in range(1,11):
    time_fib_recursive = timeit.timeit(lambda: FibRecursive(i), number=c)
    time_fib_array = timeit.timeit(lambda: FibArray(i), number=c)
    y1.append(time_fib_recursive/c)
    y2.append(time_fib_array/c)
plt.figure(1)
plt.plot(x, y1, "--",x,y2,"-")
plt.legend(['FibRecursive(n)', 'FibArray(n)'])
plt.xlabel("Значение n")
plt.ylabel("Время выполнения")
plt.show()
```

Результат работы программы:

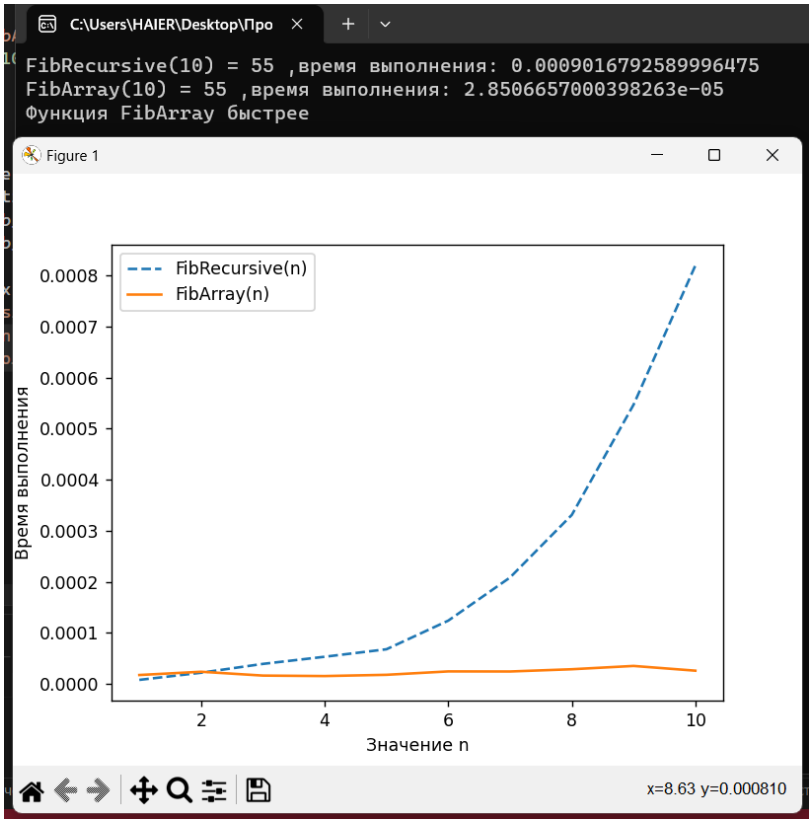

Рисунок 1 – Результат работы программы

Задание 2. Алгоритмы поиска НОД

Код программы:

```
import timeit
import matplotlib.pyplot as plt
def EuclidGCD(a,b):
    if a==0:
        return b
    if b==0:
        return a
    if a>=b:
        return EuclidGCD(a % b,b)
    if b>=a:
        return EuclidGCD(a,b%a)
def GCD(a,b):
    gcd=1
    for d in range(2,max(a,b)):
        if a%d==0 and b%d==0:
            gcd=d
    return gcd
time_gcd=(timeit.timeit(lambda: GCD(3918848,1653264), number=10))/10
time_euclid_gcd=(timeit.timeit(lambda: EuclidGCD(3918848,1653264), number=10))/10
gcd=GCD(3918848,1653264)
euclidGCD=EuclidGCD(3918848,1653264)
print("GCD(3918848,1653264) =", gcd, ",время выполнения:", time_gcd)
print("EuclidGCD(3918848,1653264) =",euclidGCD, ",время выполнения:",
time_euclid_gcd)
if time_gcd > time_euclid_gcd:
    print("Функция EuclidGCD быстрее, чем GCD")
else:
    print("Функция GCD быстрее, чем EuclidGCD")
x2=[]
```

```
y1n=[]
y2n=[]
for i in range(1200000,1400000,83129):
    time_gcd = timeit.timeit(lambda: GCD(i,1320300), number=20)
    time_euclid_gcd = timeit.timeit(lambda: EuclidGCD(i,1320300), number=20)
    x2.append(i)
    y1n.append(time_gcd/20)
    y2n.append(time_euclid_gcd/20)
plt.figure(1)
plt.plot(x2, y1n, "--",x2,y2n,"-")
plt.legend(['GCD(a,b)', 'Euclid_GCD(a,b)'])
plt.xlabel("Значение a")
plt.ylabel("Время выполнения")
plt.show()
```

Результат работы программы:

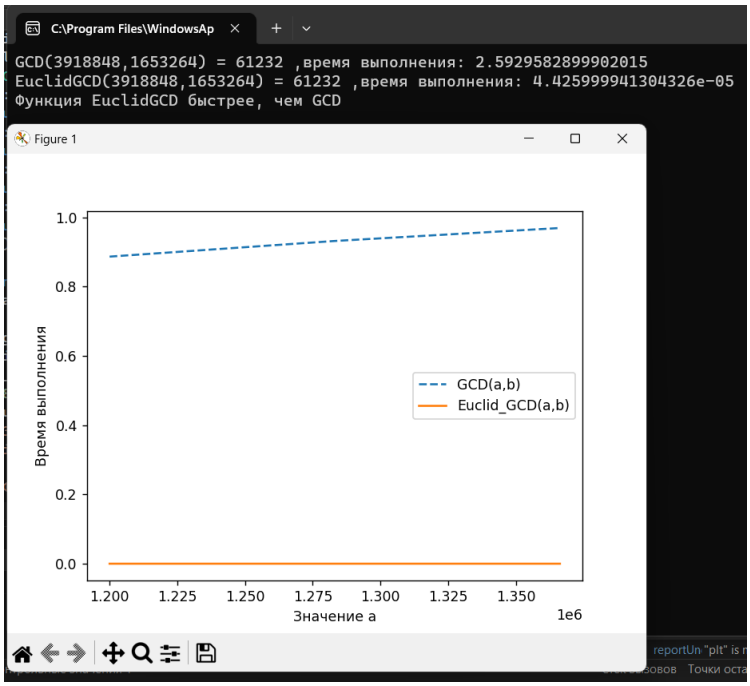

Рисунок 2 – Результат работы программы

Вывод

В ходе выполнения работы на языке Python были реализованы наивные и оптимизированные алгоритмы поиска чисел Фибоначчи и НОД.