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 ПРАКТИЧЕСКОЙ РАБОТЕ №2
дисциплины «Алгоритмизация»

Выполнила:
Беседина Инга Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

_____
(подпись)

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

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Ход работы

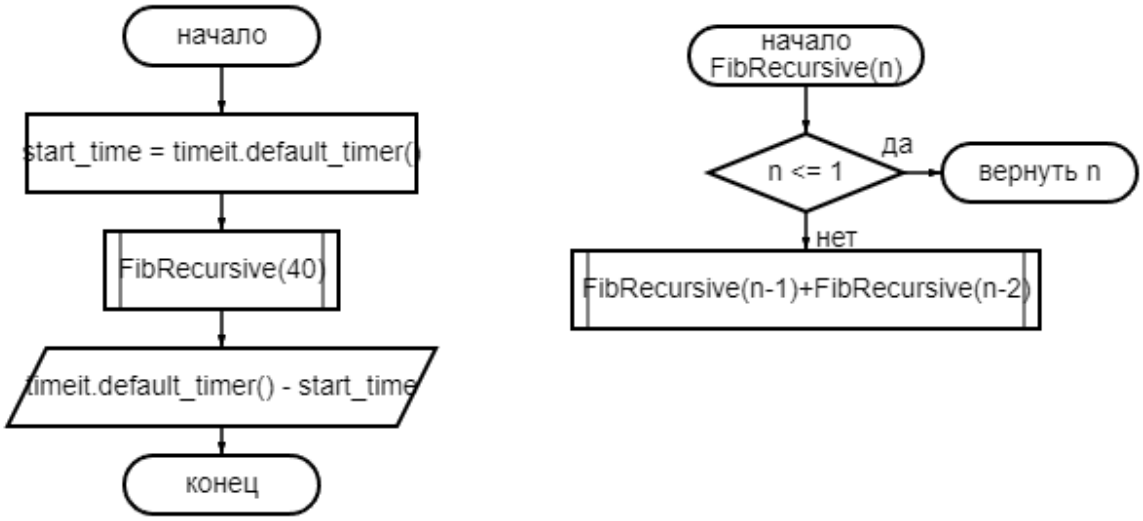

Рисунок 1. Блок схема наивного алгоритма

```
import timeit
def FibRecursive(n):
        if n <= 1:
                return n
        else:
                return FibRecursive(n-1)+FibRecursive(n-2)

start_time = timeit.default_timer()
FibRecursive(40)
print(timeit.default_timer() - start_time)
```

Рисунок 2. Наивный вариант решения

```
>>>
========== RESTART: F:\Алгоритмизация\ПР_2\Новый текстовый документ.py =========
1.2199999999962241e-05
>>>
========== RESTART: F:\Алгоритмизация\ПР_2\Новый текстовый документ.py =========
7.930000000000437e-05
>>>
========== RESTART: F:\Алгоритмизация\ПР_2\Новый текстовый документ.py =========
0.001420300000000041
>>>
========== RESTART: F:\Алгоритмизация\ПР_2\Новый текстовый документ.py =========
0.008001499999999995
>>>
========== RESTART: F:\Алгоритмизация\ПР_2\Новый текстовый документ.py =========
0.062165499999999985
>>>
========== RESTART: F:\Алгоритмизация\ПР_2\Новый текстовый документ.py =========
0.3078981999999999
>>>
========== RESTART: F:\Алгоритмизация\ПР_2\Новый текстовый документ.py =========
2.634258
>>>
========== RESTART: F:\Алгоритмизация\ПР_2\Новый текстовый документ.py =========
28.6774507
>>>
```

Рисунок 3. Время работы алгоритма

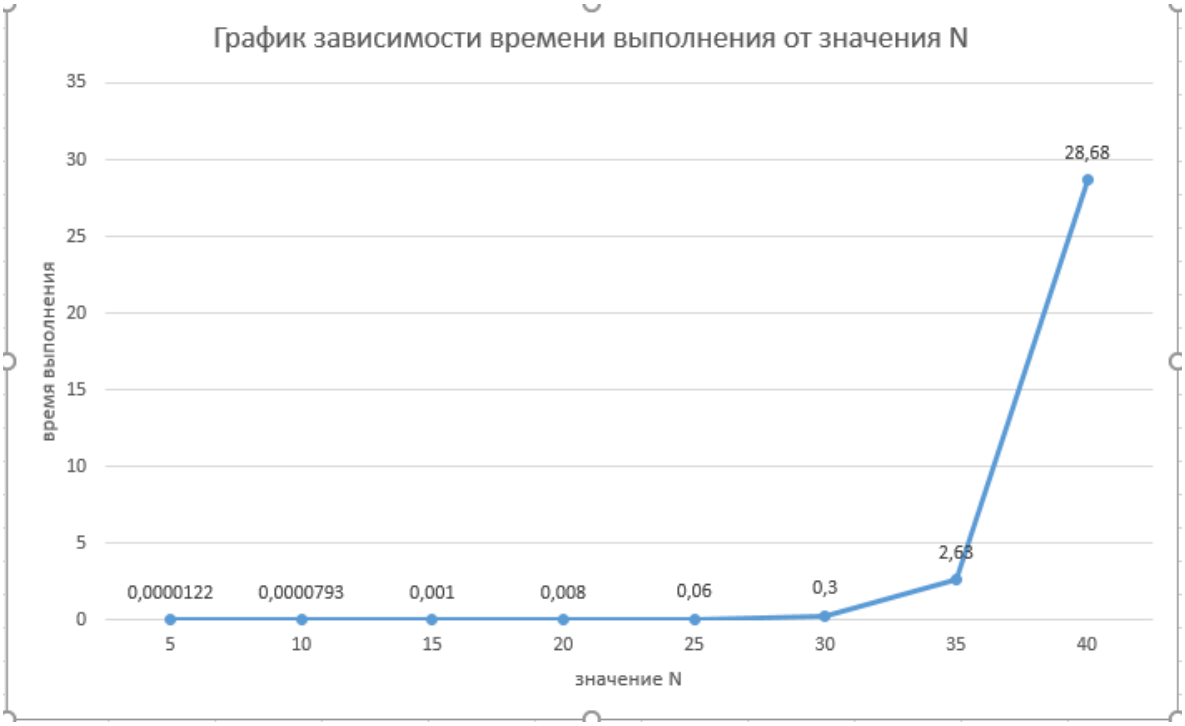

Рисунок 4. График зависимости времени выполнения от значения N

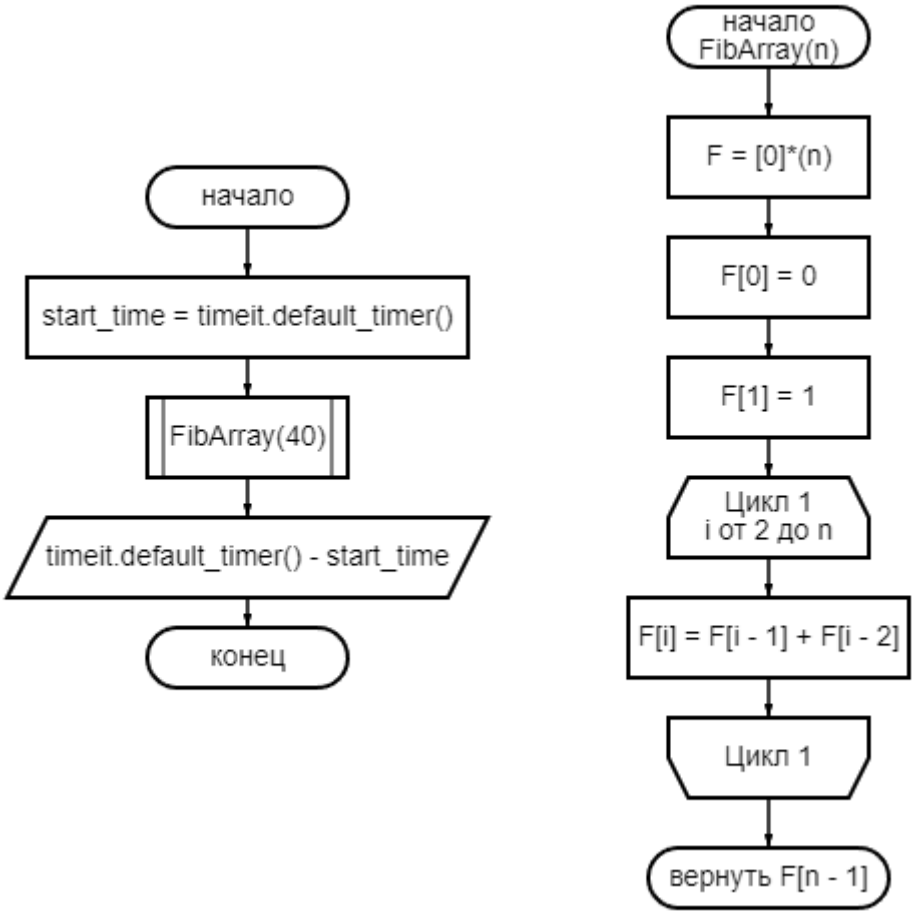

Рисунок 5. Блок схема оптимального алгоритма

```python
import timeit

def FibArray(n):
    F = [0]*(n)
    F[0] = 0
    F[1] = 1
    for i in range(2, n):
        F[i] = F[i - 1] + F[i - 2]
    return F[n - 1]


start_time = timeit.default_timer()
FibArray(40)
print(timeit.default_timer() - start_time)
```

Рисунок 6. Оптимальный вариант решения

```
>>>
================== RESTART: F:\Алгоритмизация\ПР_2\FibAlg2.py ==================
1.4999999999987246e-05
>>>
================== RESTART: F:\Алгоритмизация\ПР_2\FibAlg2.py ==================
2.6299999999979118e-05
>>>
================== RESTART: F:\Алгоритмизация\ПР_2\FibAlg2.py ==================
2.049999999997887e-05
>>>
================== RESTART: F:\Алгоритмизация\ПР_2\FibAlg2.py ==================
2.3300000000003873e-05
>>>
================== RESTART: F:\Алгоритмизация\ПР_2\FibAlg2.py ==================
2.599999999997049e-05
>>>
================== RESTART: F:\Алгоритмизация\ПР_2\FibAlg2.py ==================
2.7500000000013625e-05
>>>
================== RESTART: F:\Алгоритмизация\ПР_2\FibAlg2.py ==================
2.9700000000021376e-05
>>>
================== RESTART: F:\Алгоритмизация\ПР_2\FibAlg2.py ==================
3.179999999997074e-05
>>>
```

Рисунок 7. Время работы алгоритма

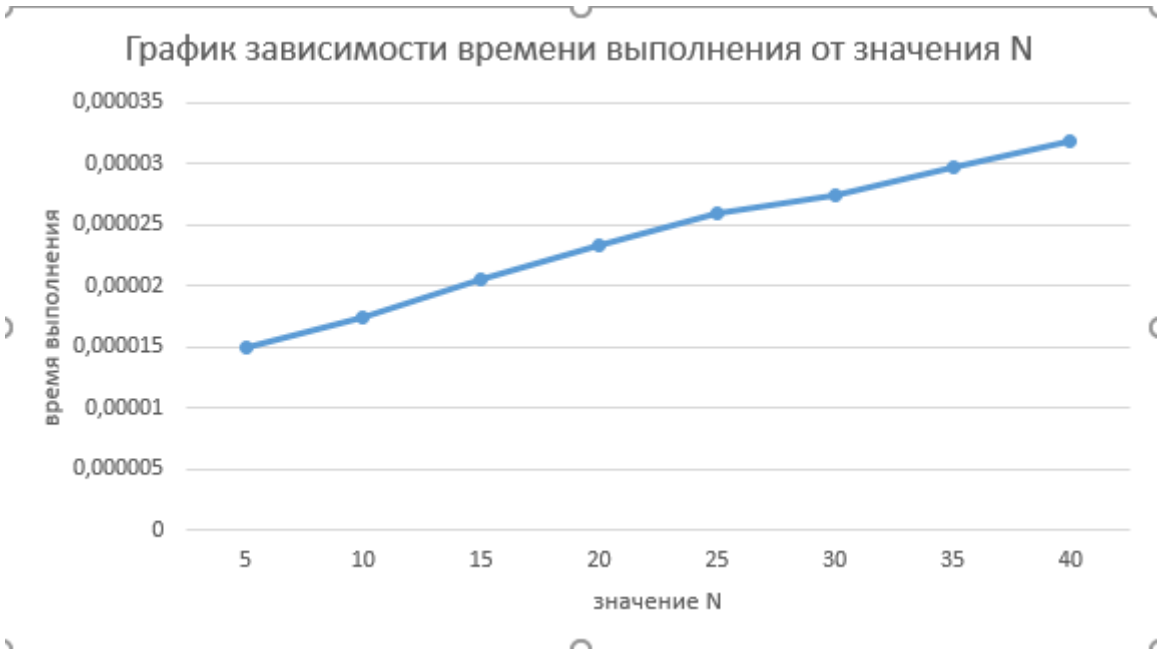

Рисунок 8. График зависимости времени выполнения от значения N

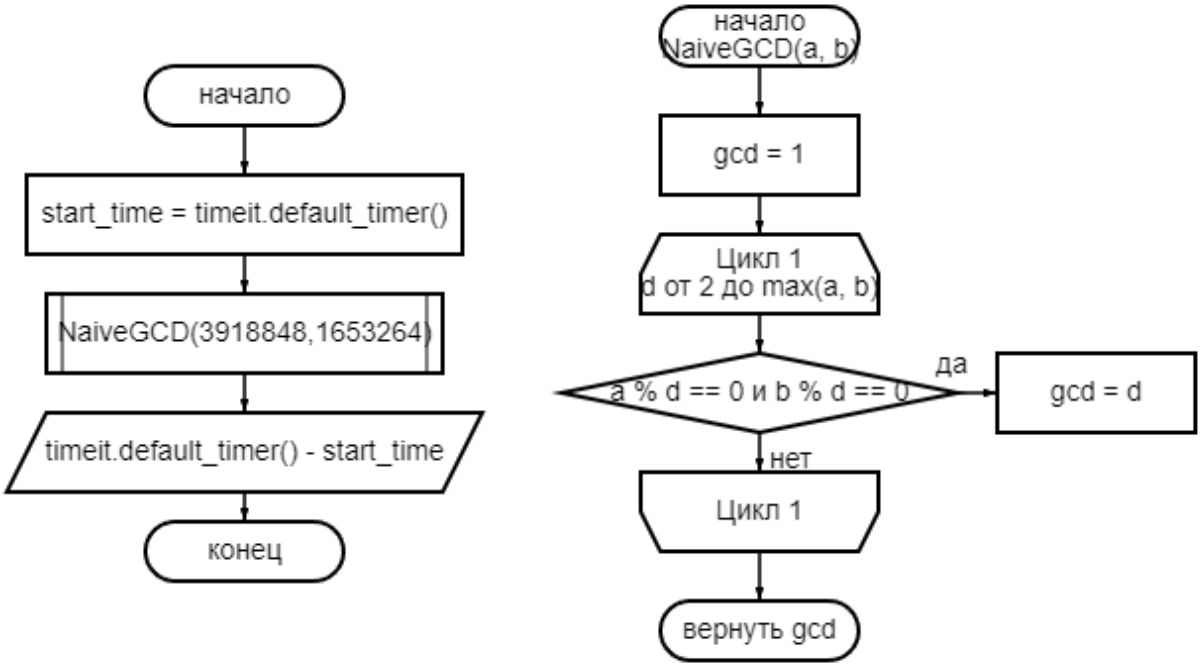

Рисунок 9. Блок схема наивного алгоритма

```python
import timeit

def NaiveGCD(a, b):
    gcd = 1
    for d in range(2, max(a,b)):
        if a % d == 0 and b % d == 0:
            gcd = d
    return gcd

start_time = timeit.default_timer()
print(NaiveGCD(3918848,1653264))
print(timeit.default_timer() - start_time)
```

Рисунок 10. Наивный вариант поиска НОД

```
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: F:\Алгоритмизация\ПР_2\GCDAlg1.py ==================
61232
0.2499
>>>
```

Рисунок 11. Время работы алгоритма

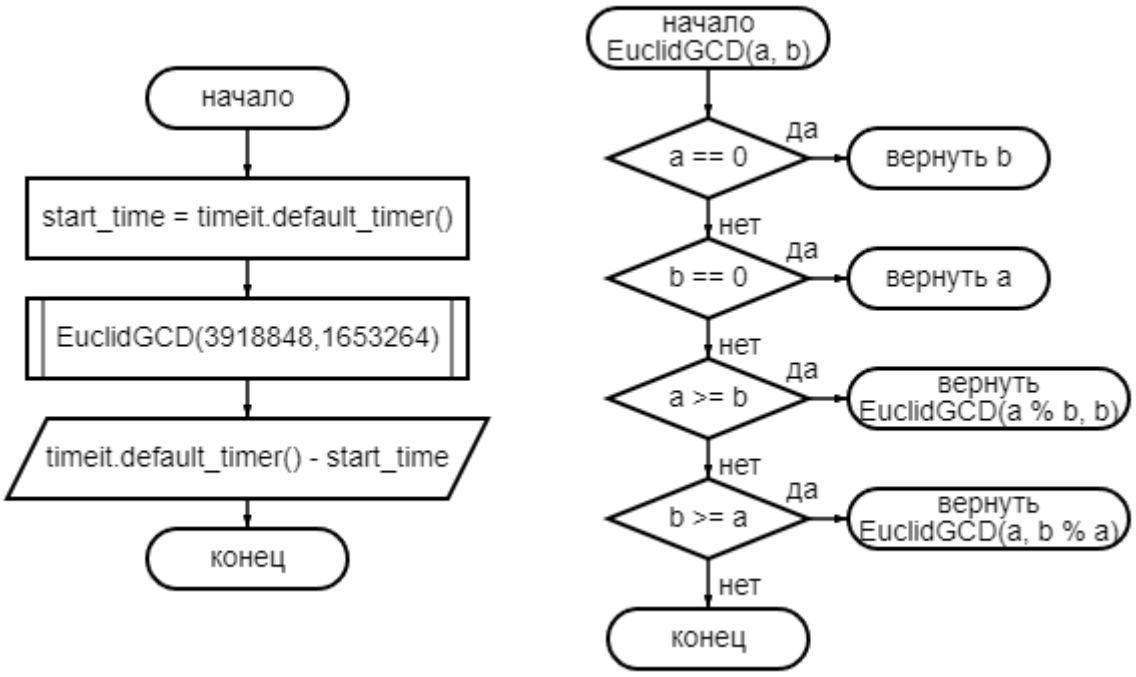

Рисунок 12. Блок схема оптимального алгоритма

```
import timeit

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)

start_time = timeit.default_timer()
print(EuclidGCD(3918848, 1653264))
print(timeit.default_timer() - start_time)
```

Рисунок 13. Оптимальный вариант поиска НОД

```
>>>
================== RESTART: F:\Алгоритмизация\ПР_2\GCDAlg2.py ==================
61232
0.014185400000000015
>>>
```

Рисунок 14. Время работы алгоритма

Вывод: в ходе выполнения практической работы было проведено сравнение времени работы наивного и оптимального алгоритма для поиска n-го числа Фибоначчи и наивного и оптимального алгоритма поиска НОД.