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»

Выполнил:
Середа Кирилл Витальевич
1 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Ход выполнения заданий

1) Написал наивный алгоритм для вычисления чисел Фибоначи

```
def F1(n):
    if n == 0: return 0
    elif n == 1: return 1
    else: return (F1(n-1)+F1(n-2))
```

Рисунок 1 – функция для вычисление чисел Фибоначи

1	0,00004
2	0,000011
3	0,000013
4	0,000014
5	0,000012
6	0,000016
7	0,000014
8	0,000016
9	0,000019
10	0,000023
12	0,000064
13	0,000085
14	0,000146
15	0,000228
16	0,000381
17	0,000558
18	0,000912
19	0,001463
20	0,002331
21	0,003774
22	0,006134
23	0,009902
24	0,01633
25	0,026814
26	0,041943
27	0,068198
28	0,111004
29	0,178888
30	0,287629
31	0,467096
32	0,769157
33	1,259859
34	1,975385
35	3,254543

Рисунок 2 – таблица времени выполнения программы

2) Написал оптимизированный алгоритм для вычисления чисел Фибоначи.

```python
def F2(n, dic = {0: 0, 1: 1}):
    if n not in dic:
        dic[n] = F2(n-1, dic) + F2(n-2, dic)
    return dic[n]
```

Рисунок 3 – оптимизированная функция для вычисления чисел Фибоначчи

2	1,63406E-07
3	1,64594E-07
4	1,63474E-07
5	1,62482E-07
6	1,63187E-07
7	1,6272E-07
8	1,63541E-07
9	1,61973E-07
10	1,62579E-07
11	1,6368E-07
12	1,62125E-07
13	1,63501E-07
14	1,63501E-07
15	1,64481E-07
16	1,64493E-07
17	1,62788E-07
18	1,6387E-07
19	1,63648E-07
20	1,6349E-07
21	1,62077E-07
22	1,62414E-07
23	1,62414E-07
24	1,63568E-07
25	1,63537E-07
26	1,62327E-07
27	1,6537E-07
28	1,64149E-07
29	1,63773E-07
30	1,64243E-07
31	1,62166E-07
32	1,63743E-07
33	1,62712E-07
34	1,64472E-07
35	1,65089E-07

Рисунок 4 – таблица времени выполнения программы

3) Составил графики число-от-времени для обоих алгоритмов

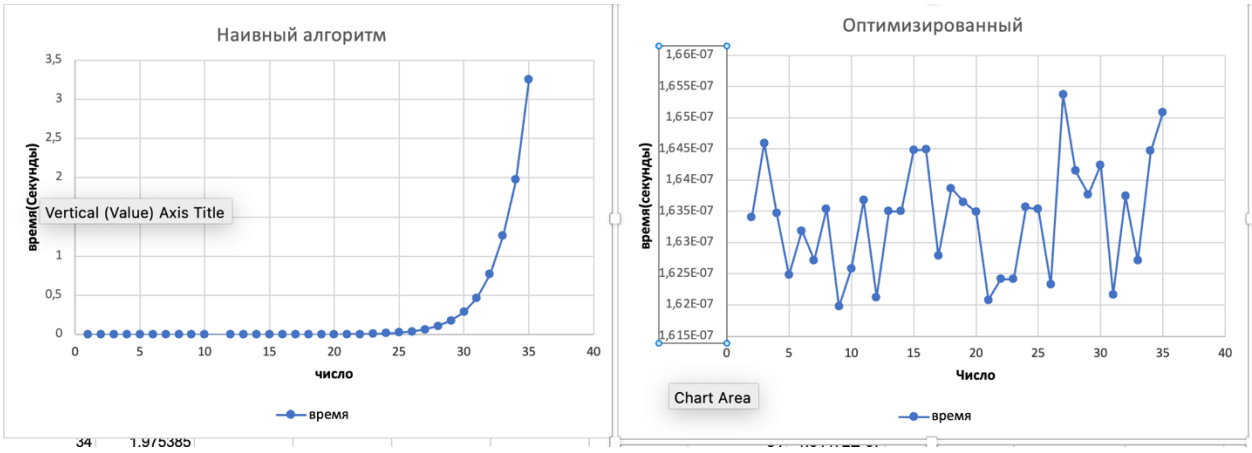
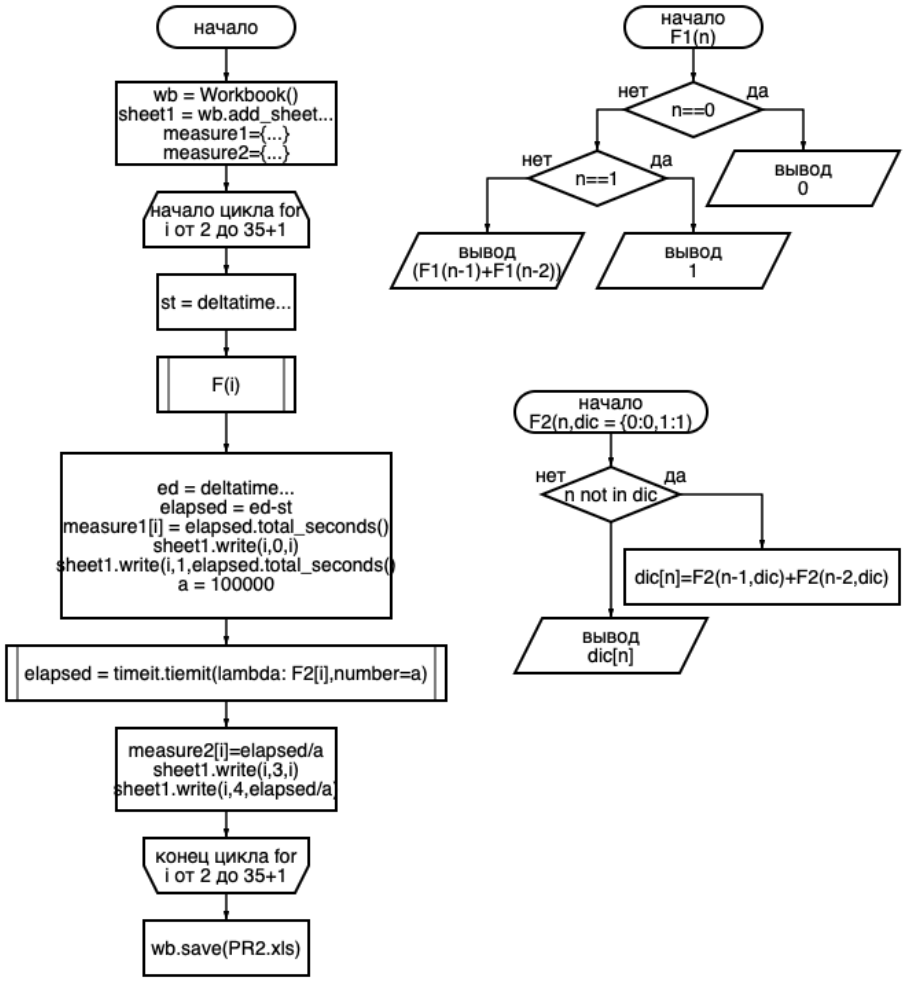


Рисунок 5 – Графики n от t

Блок-схема всей программы:



Полный код программы:

```
import datetime
import timeit
import xlwt
from xlwt import Workbook

wb = Workbook()
sheet1 = wb.add_sheet("Sheet 1")

def F1(n):
    if n == 0: return 0
    elif n == 1: return 1
    else: return (F1(n-1)+F1(n-2))

def F2(n, dic = {0: 0, 1: 1}):
    if n not in dic:
        dic[n] = F2(n-1, dic) + F2(n-2, dic)
    return dic[n]



measure1 = {0:0,1:0}
measure2 = {0:0,1:0}
for i in range(2,35+1):
    st = datetime.datetime.now()
    F1(i)
    ed = datetime.datetime.now()
    elapsed = ed-st
    measure1[i] = elapsed.total_seconds()

    sheet1.write(i,0,i)
    sheet1.write(i,1,elapsed.total_seconds())

    a = 1000000
    elapsed = timeit.timeit(lambda: F2(i),number=a)
    measure2[i] = elapsed/a

    sheet1.write(i,3,i)
    sheet1.write(i,4,elapsed/a)

print(measure1)
print(measure2)

wb.save("PR2.xls")
```

4) Написал наивный алгоритм для вычисления НОД

```
def F3(a, b):
    gcd = 1
    for d in range(2, min(a, b) + 1):
        if a % d == 0 and b % d == 0:
            gcd = d
    return gcd
```

Рисунок 6 – Наивный алгоритм для НОД

39188480	16532640	1,061057
39188480	16532641	1,041733
39188480	16532642	1,040301
39188480	16532643	1,027923
39188480	16532644	1,024364
39188481	16532640	1,049828
39188481	16532641	1,039831
39188481	16532642	1,057756
39188481	16532643	1,255462
39188481	16532644	1,036372
39188482	16532640	1,083608
39188482	16532641	1,046537
39188482	16532642	1,034786
39188482	16532643	1,027817
39188482	16532644	1,057753
39188483	16532640	1,026003
39188483	16532641	1,024656
39188483	16532642	1,02511
39188483	16532643	1,025623
39188483	16532644	1,023565
39188484	16532640	1,027935
39188484	16532641	1,030881
39188484	16532642	1,022797
39188484	16532643	1,028588
39188484	16532644	1,089026

Рисунок 7 – таблица времени выполнения

5) Написал оптимизированную функцию

```
def F4(a, b):
    while b:
        a, b = b, a % b
    return a
```

Рисунок 8 – Оптимизированный алгоритм

39188480	16532640	3,6643E-07
39188480	16532641	5,9903E-07
39188480	16532642	5,2208E-07
39188480	16532643	6,3117E-07
39188480	16532644	5,9386E-07
39188481	16532640	4,7548E-07
39188481	16532641	5,2276E-07
39188481	16532642	9,3214E-07
39188481	16532643	5,2043E-07
39188481	16532644	6,2655E-07
39188482	16532640	5,6612E-07
39188482	16532641	5,2123E-07
39188482	16532642	5,6917E-07
39188482	16532643	5,974E-07
39188482	16532644	6,6256E-07
39188483	16532640	5,5606E-07
39188483	16532641	5,634E-07
39188483	16532642	6,6114E-07
39188483	16532643	7,0539E-07
39188483	16532644	7,3444E-07
39188484	16532640	4,4507E-07
39188484	16532641	5,2467E-07
39188484	16532642	5,6454E-07
39188484	16532643	6,6108E-07
39188484	16532644	8,1048E-07

Рисунок 9 – Таблица времени выполнения

6) Составил графики

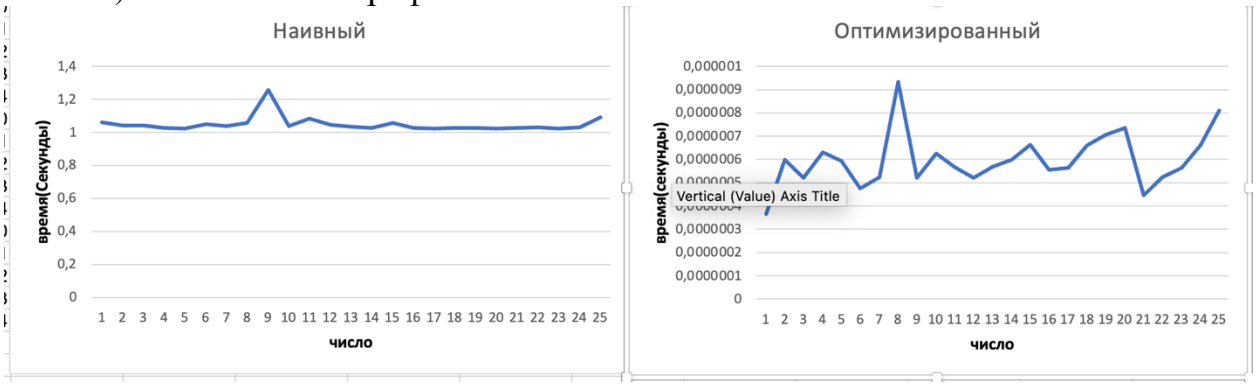


Рисунок 10 – графики времени

Код программы:

```
import datetime
import timeit
import xlwt
```

```
from xlwt import Workbook

wb = Workbook()
sheet1 = wb.add_sheet("Sheet 1")

#region
def F1(n):
    if n == 0: return 0
    elif n == 1: return 1
    else: return (F1(n-1)+F1(n-2))

def F2(n, dic = {0: 0, 1: 1}):
    if n not in dic:
        dic[n] = F2(n-1, dic) + F2(n-2, dic)
    return dic[n]
#endregion

def F3(a, b):
    gcd = 1
    for d in range(2, min(a, b) + 1):
        if a % d == 0 and b % d == 0:
            gcd = d
    return gcd

def F4(a, b):
    while b:
        a, b = b, a % b
    return a

#region
#measure1 = {0:0,1:0}
#measure2 = {0:0,1:0}
#endregion
count = 0

for i in range(39188480,39188480+5):
    for j in range(16532640,16532640+5):
        print(count)
        st = datetime.datetime.now()
        g = F3(i,j)
        ed = datetime.datetime.now()
        elapsed = ed-st
        #measure1[i] = elapsed.total_seconds()

        sheet1.write(count,0,i)
        sheet1.write(count, 1, j)
        sheet1.write(count,2,elapsed.total_seconds())

        a = 1000000
        elapsed = timeit.timeit(lambda: F4(i,j),number=a)
        #measure2[i] = elapsed/a

        sheet1.write(count,4,i)
        sheet1.write(count,5,j)
        sheet1.write(count,6,elapsed/a)
        count+=1

wb.save("PR2.2.xls")
```

Блок-схема:

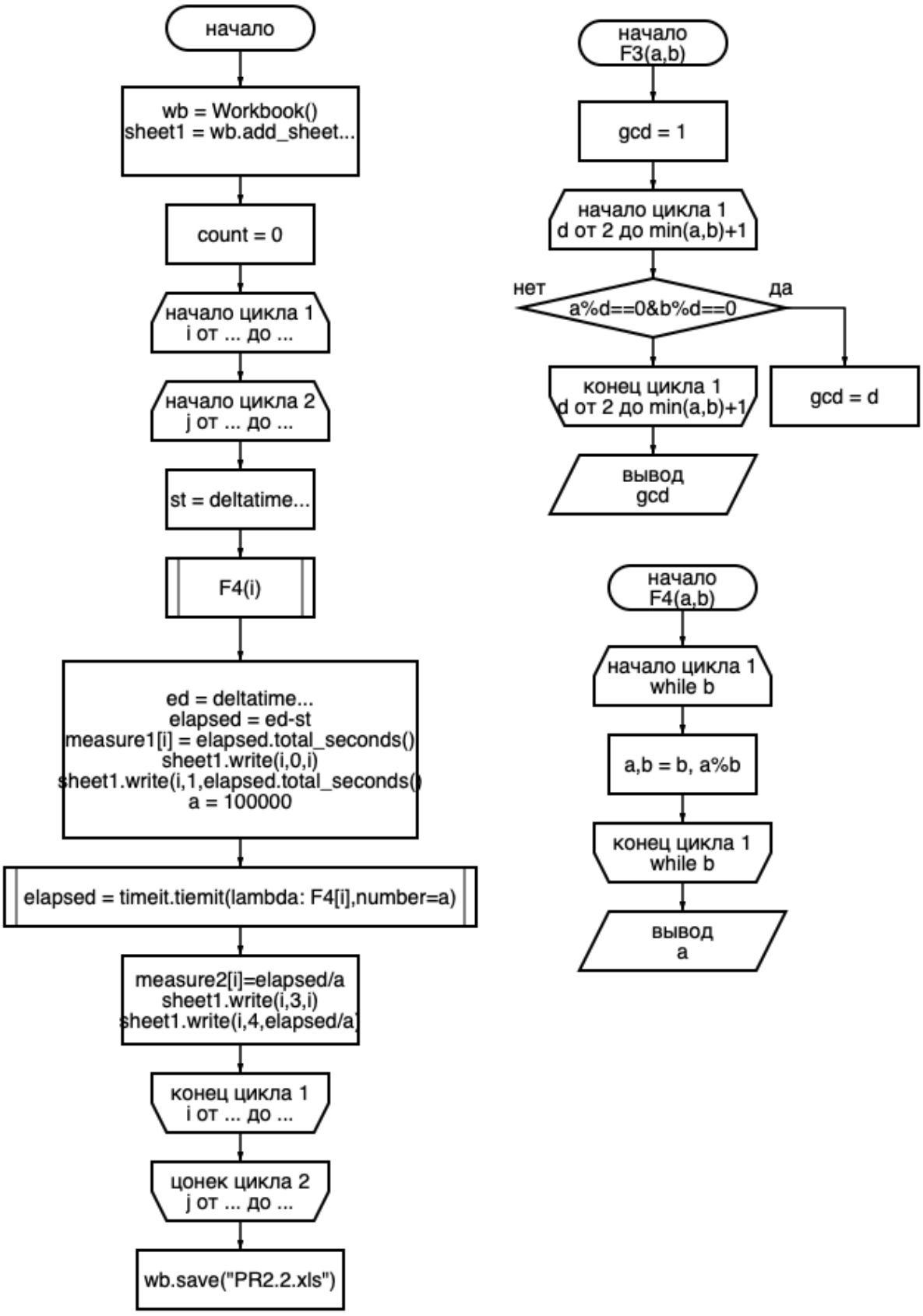


Рисунок 11 – блок схема для НОД

Вывод:

1) Можно сделать вывод что скорость выполнения первого алгоритма напрямую зависит от числа Фибоначчи, в то время как скорость выполнения второго алгоритма не зависит от величины числа и находится в рамках погрешности, что подтверждается графиком.

2) Можно сделать вывод, что время работы алгоритма НОД в обоих случаях находится в пределах погрешности, однако во втором случае скорость выполнения значительно выше в сравнении с первым вариантом.