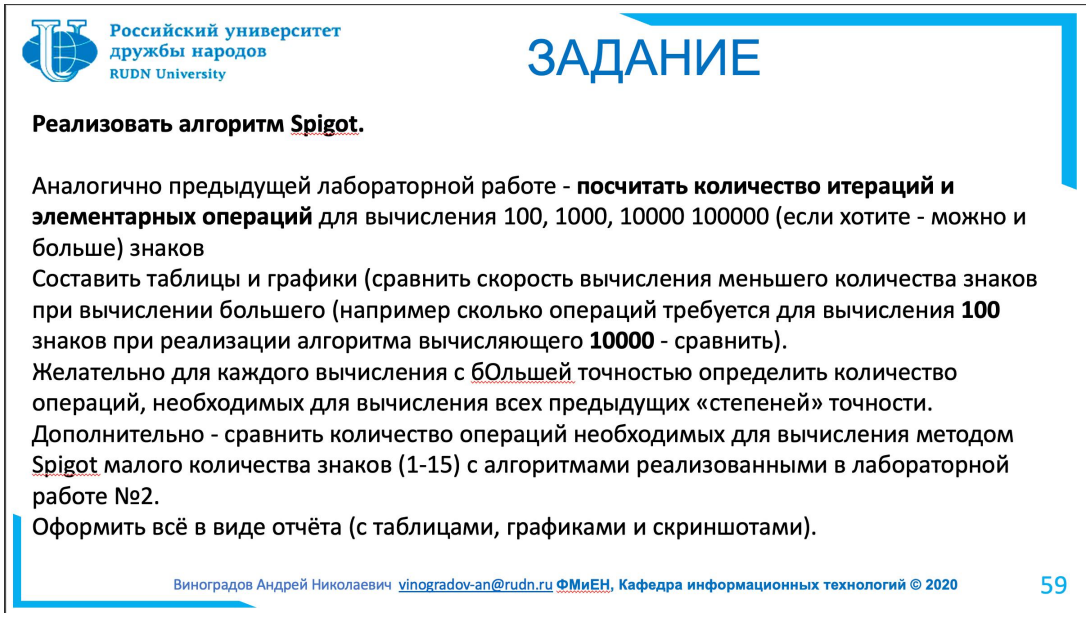
Лабораторная работа №3

Задание:



Выполнение:

1. Подготовка

In [180…
```
!pwd
```

```
/Users/artyem.petrov/dev/university/4-1/it-computer-practice/lab03
```

In [181…
```
!../.venv/bin/pip install matplotlib
```

```
Requirement already satisfied: matplotlib in /Users/artyem.petrov/de
v/university/4-1/it-computer-practice/.venv/lib/python3.13/site-pack
ages (3.10.8)
Requirement already satisfied: contourpy>=1.0.1 in /Users/artyem.pet
rov/dev/university/4-1/it-computer-practice/.venv/lib/python3.13/sit
e-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /Users/artyem.petrov/
dev/university/4-1/it-computer-practice/.venv/lib/python3.13/site-pa
ckages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /Users/artyem.pe
trov/dev/university/4-1/it-computer-practice/.venv/lib/python3.13/si
te-packages (from matplotlib) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /Users/artyem.pe
trov/dev/university/4-1/it-computer-practice/.venv/lib/python3.13/si
te-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in /Users/artyem.petrov/d
ev/university/4-1/it-computer-practice/.venv/lib/python3.13/site-pac
kages (from matplotlib) (2.3.5)
Requirement already satisfied: packaging>=20.0 in /Users/artyem.petr
ov/dev/university/4-1/it-computer-practice/.venv/lib/python3.13/site
-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /Users/artyem.petrov/de
v/university/4-1/it-computer-practice/.venv/lib/python3.13/site-pack
ages (from matplotlib) (12.0.0)
Requirement already satisfied: pyparsing>=3 in /Users/artyem.petrov/
dev/university/4-1/it-computer-practice/.venv/lib/python3.13/site-pa
ckages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in /Users/artye
m.petrov/dev/university/4-1/it-computer-practice/.venv/lib/python3.1
3/site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /Users/artyem.petrov/dev/
university/4-1/it-computer-practice/.venv/lib/python3.13/site-packag
es (from python-dateutil>=2.7->matplotlib) (1.17.0)

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: /Users/artyem.petrov/dev/university/4-1/it-
computer-practice/.venv/bin/python -m pip install --upgrade pip
```

In [182…

```python
import time
import matplotlib.pyplot as plt
import json
import pandas as pd
from typing import Tuple, Dict, List
```

2. Реализуем класс для подсчета операций:

In [183…

```python
class SpigotCounter:
    def __init__(self):
        self.ops = 0
        self.iterations = 0

    def add(self, count=1):
        self.ops += count
```

3. Теперь реализуем саму функцию:

In [184…

```python
def spigot_pi(n_digits: int, counter: SpigotCounter=None) -> Tuple[
    if counter is None:
        counter = SpigotCounter()

    # Длина массива
    L = (10 * n_digits) // 3 + 1
    a = [2] * L
    digits = []

    # Буфер для недействительных цифр
    invalid_count = 0

    for i in range(n_digits):
        counter.iterations += 1

        carry = 0
        # Проходим по массиву с конца к началу
        for j in range(L-1, -1, -1):
            counter.add(2)  # умножение и сложение
            x = a[j] * 10 + carry

            if j > 0:
                denominator = 2*j + 1
                counter.add(2)  # умножение и сложение для denomina
                a[j] = x % denominator
                carry = (x // denominator) * j
                counter.add(3)  # деление, умножение, взятие остатк
            else:
                # Последний элемент (j = 0)
                a[0] = x % 10
                digit = x // 10
                counter.add(2)  # деление и взятие остатка

                # Коррекция цифры
                if digit == 9:
                    invalid_count += 1
                elif digit == 10:
                    digit = 0
                    for k in range(len(digits)-invalid_count, len(d
                        digits[k] = (digits[k] + 1) % 10
                        if digits[k] != 0:
                            break
                    invalid_count = 1
                else:
                    # Сбрасываем недействительные цифры
                    invalid_count = 0

                digits.append(digit)

    # Преобразуем цифры в строку
    result = str(digits[0]) + "."
    for d in digits[1:]:
        result += str(d)
```

```
    return result, counter
```

4. Реализуем функцию для сравнения скорости, точности и кол-ва операция для вычисления числа Pi:

In [185…

```
def compute_and_compare() -> pd.DataFrame:
    precisions = [100, 1000, 10000]
    data = []

    for n in precisions:
        counter = SpigotCounter()
        start_time = time.time()

        pi_str, counter = spigot_pi(n, counter)
        elapsed_time = time.time() - start_time

        data.append({
            'Цифр': n,
            'Операций': counter.ops,
            'Время (сек)': elapsed_time,
            'Итераций': counter.iterations,
            'Операций на цифру': counter.ops / n,
            'Предпросмотр': pi_str[:30] + "..." + pi_str[-30:] if n
        })

    return pd.DataFrame(data)
```

5. Реализуем функцию для сравнения с результатами Лабораторной работы №2

In [186…

```
def compare_with_lab2() -> pd.DataFrame:
    lab2_data = {
        'Мэчин': {
            'ops': [19, 33, 51, 51, 73, 73, 99, 129, 163, 201, 243,
            'time': [1.63338114465804e-05, 1.137499938370714e-05, 1
                     1.6379808272807180e-05, 2.37580862212348e-05,
                     4.095898475497961e-05, 5.5334086901830435e-05,
                     3.75209806238667e-05, 4.241609891913507e-05, 4
                     5.03750852380738e-05, 6.25420707189739e-05, 6.
            'iterations': [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 10,
        },
        'Нилаканта': {
            'ops': [14, 28, 56, 119, 259, 553, 1197, 2576, 5558, 11
                    119644, 257959, 535857],
            'time': [3.91699722968419e-06, 4.91708484417379e-06, 1.
                     4.4791988329051896e-05, 0.0081039340886897644,
                     0.00479249999525683, 0.0211459080666082, 0.036
                     0.176286009017099, 0.846224249876097, 3.945642
                     18.23386294197726, 86.5178642991612, 388.16923
            'iterations': [2, 4, 8, 17, 37, 79, 171, 368, 794, 1710
                           17092, 36850, 76551]
```

```
        }
    }

    data = []

    for prec in range(1, 16):
        counter = SpigotCounter()
        start_time = time.time()
        pi_str, counter = spigot_pi(prec, counter)
        elapsed_time = time.time() - start_time

        data.append({
            'Точность (цифр)': prec,
            'Операции Spigot': counter.ops,
            'Операции Мэчин': lab2_data['Мэчин']['ops'][prec-1],
            'Операции Нилаканта': lab2_data['Нилаканта']['ops'][pre
            'Время Spigot (сек)': elapsed_time,
            'Время Мэчин (сек)': lab2_data['Мэчин']['time'][prec-1]
            'Время Нилаканта (сек)': lab2_data['Нилаканта']['time']
            'Итерации Spigot': counter.iterations,
            'Итерации Мэчин': lab2_data['Мэчин']['iterations'][prec
            'Итерации Нилаканта': lab2_data['Нилаканта']['iteration
        })

    return pd.DataFrame(data)
```

6. Реализуем функция для вывода графиков

In [187…

```
def prepare_data_for_plotting(spigot_big_results: pd.DataFrame, com
    spigot_results_big_formatted = []
    for _, row in spigot_big_results.iterrows():
        spigot_results_big_formatted.append({
            'digits': row['Цифр'],
            'operations': row['Операций'],
            'time': row['Время (сек)']
        })

    comparison_data_formatted = {
        'Spigot': {
            'ops': comparison_df['Операции Spigot'].tolist(),
            'time': comparison_df['Время Spigot (сек)'].tolist()
        },
        'Мэчин': {
            'ops': comparison_df['Операции Мэчин'].tolist(),
            'time': comparison_df['Время Мэчин (сек)'].tolist()
        },
        'Нилаканта': {
            'ops': comparison_df['Операции Нилаканта'].tolist(),
            'time': comparison_df['Время Нилаканта (сек)'].tolist()
        }
    }

    return spigot_results_big_formatted, comparison_data_formatted
```

In [188…

```python
def plot_results(spigot_results_big: pd.DataFrame, comparison_data:
    plt.figure(figsize=(15, 10))

    # Извлекаем данные из comparison_data
    spigot_ops_small = comparison_data['Spigot']['ops']
    spigot_times_small = comparison_data['Spigot']['time']

    machin_ops = comparison_data['Мэчин']['ops']
    machin_times = comparison_data['Мэчин']['time']

    nilakantha_ops = comparison_data['Нилаканта']['ops']
    nilakantha_times = comparison_data['Нилаканта']['time']

    # График 1: Операции для Spigot при разной точности (большие зн
    plt.subplot(2, 2, 1)
    precisions_big = [r['digits'] for r in spigot_results_big]
    ops_big = [r['operations'] for r in spigot_results_big]
    plt.plot(precisions_big, ops_big, 'bo-', linewidth=2, markersiz
    plt.xlabel('Количество цифр')
    plt.ylabel('Арифметических операций')
    plt.title('Spigot: Операции vs Точность (большие значения)')
    plt.grid(True, alpha=0.3)

    # График 2: Время для Spigot (большие значения)
    plt.subplot(2, 2, 2)
    times_big = [r['time'] for r in spigot_results_big]
    plt.plot(precisions_big, times_big, 'ro-', linewidth=2, markers
    plt.xlabel('Количество цифр')
    plt.ylabel('Время (сек)')
    plt.title('Spigot: Время vs Точность (большие значения)')
    plt.grid(True, alpha=0.3)

    # График 3: Сравнение операций всех алгоритмов для малой точнос
    plt.subplot(2, 2, 3)
    x = list(range(1, 16))

    plt.plot(x, spigot_ops_small, 'go-', label='Spigot', linewidth=
    plt.plot(x, machin_ops, 'bo-', label='Мэчин', linewidth=2, mark
    plt.plot(x, nilakantha_ops, 'ro-', label='Нилаканта', linewidth

    plt.xlabel('Точность (количество цифр)')
    plt.ylabel('Арифметических операций')
    plt.title('Сравнение операций (1-15 цифр)')
    plt.legend()
    plt.grid(True, alpha=0.3)

    # График 4: Сравнение времени всех алгоритмов (логарифмическая
    plt.subplot(2, 2, 4)

    plt.semilogy(x, spigot_times_small, 'go-', label='Spigot', line
    plt.semilogy(x, machin_times, 'bo-', label='Мэчин', linewidth=2
    plt.semilogy(x, nilakantha_times, 'ro-', label='Нилаканта', lin

    plt.xlabel('Точность (количество цифр)')
    plt.ylabel('Время выполнения (сек, log scale)')
    plt.title('Сравнение времени (логарифмическая шкала)')
```

```
    plt.legend()
    plt.grid(True, alpha=0.3, which='both')

    plt.tight_layout()
```

7. Анализ операций для вычисления меньшей точности при расчете большей

In [189…

```
def analyze_operations_for_subprecision() -> pd.DataFrame:
    target_digits = [100, 1000, 10000]
    data = []

    for main_digits in target_digits:
        for sub_digits in [10, 50, 100, 500, 1000]:
            if sub_digits <= main_digits:
                total_ops_estimate = 10 * main_digits * main_digits
                ops_for_sub = 10 * sub_digits * sub_digits
                percentage = (ops_for_sub / total_ops_estimate) * 1

                data.append({
                    'Основные цифры': main_digits,
                    'Цифр (подмножество)': sub_digits,
                    'Оценка операций': ops_for_sub,
                    'Процент от полного': percentage
                })

    return pd.DataFrame(data)
```

8. Собираем результаты:

8.1. Вычисление Spigot для разной точности (большие значения)

In [190…

```
spigot_results_big = compute_and_compare()
```

In [191…

```
spigot_results_big.head(20)
```

Out[191…

	Цифр	Операций	Время (сек)	Итераций	Операций на цифру	
0	100	233500	0.016877	100	2335.0	3.14159265358979
1	1000	23335000	0.502436	1000	23335.0	3.1415926535897
2	10000	2333350000	55.744956	10000	233335.0	3.14159265358979

	Цифр	Операций	Время (сек)	Итераций	Операций на цифру	Предпросмотр
0	100	233500	0.021670	100	2335.0	3.14159265358979323846264338327...4062862089986...
1	1000	23335000	0.554927	1000	23335.0	3.14159265358979323846264338327...6120019278766...
2	10000	2333350000	57.743290	10000	233335.0	3.14159265358979323846264338327...6959688159205...

8.2. Сравнение с ЛР2 (малые значения 1-15)

In [192…
```
comparison_data = compare_with_lab2()

comparison_data.head(20)
```

Out[192…

	Точность (цифр)	Операции Spigot	Операции Мэчин	Операции Нилаканта	Время Spigot (сек)	Время Мэчин (сек)	Нила
0	1	25	19	14	0.000002	0.000016	0.0
1	2	92	33	28	0.000003	0.000011	0.0
2	3	222	51	56	0.000004	0.000017	0.0
3	4	380	51	119	0.000007	0.000016	0.0
4	5	580	73	259	0.000009	0.000024	0.0
5	6	864	73	553	0.000014	0.000026	0.0
6	7	1155	99	1197	0.000018	0.000041	0.0
7	8	1488	129	2576	0.000022	0.000055	0.
8	9	1926	163	5558	0.000031	0.000025	0.0
9	10	2350	201	11970	0.000037	0.000038	0.1
10	11	2816	243	25758	0.000045	0.000042	0.8
11	12	3408	243	55559	0.000053	0.000042	3.9
12	13	3965	289	119644	0.000061	0.000050	18.2
13	14	4564	339	257959	0.000072	0.000063	86.5
14	15	5310	339	535857	0.000082	0.000063	388.

	Точность (цифр)	Операции Spigot	Операции Мэчин	Операции Нилаканта	Время Spigot (сек)	Время Мэчин (сек)	Время Нилаканта (сек)	Итерации Spigot	Итерации Мэчин	Итерации Нилаканта
0	1	25	19	14	0.000014	0.000016	0.000004	1	2	2
1	2	92	33	28	0.000016	0.000011	0.000005	2	3	4
2	3	222	51	56	0.000029	0.000017	0.000013	3	4	8
3	4	380	51	119	0.000050	0.000016	0.000045	4	5	17
4	5	580	73	259	0.000074	0.000024	0.008104	5	6	37
5	6	864	73	553	0.000117	0.000026	0.000953	6	7	79
6	7	1155	99	1197	0.000172	0.000041	0.004792	7	8	171
7	8	1488	129	2576	0.000142	0.000055	0.021146	8	9	368
8	9	1926	163	5558	0.000183	0.000025	0.036717	9	10	794
9	10	2350	201	11970	0.000222	0.000038	0.176286	10	11	1710
10	11	2816	243	25758	0.000348	0.000042	0.846224	11	12	3694
11	12	3408	243	55559	0.000452	0.000042	3.945642	12	10	7937
12	13	3965	289	119644	0.000527	0.000050	18.233863	13	11	17092
13	14	4564	339	257959	0.000611	0.000063	86.517864	14	12	36850
14	15	5310	339	535857	0.000710	0.000063	388.169231	15	12	76551

8.3. Анализ операций

In [193…
```
result = analyze_operations_for_subprecision()

result.head(20)
```

Out [193…

	Основные цифры	Цифр (подмножество)	Оценка операций	Процент от полного
0	100	10	1000	1.0000
1	100	50	25000	25.0000
2	100	100	100000	100.0000
3	1000	10	1000	0.0100
4	1000	50	25000	0.2500
5	1000	100	100000	1.0000
6	1000	500	2500000	25.0000
7	1000	1000	10000000	100.0000
8	10000	10	1000	0.0001
9	10000	50	25000	0.0025
10	10000	100	100000	0.0100
11	10000	500	2500000	0.2500
12	10000	1000	10000000	1.0000

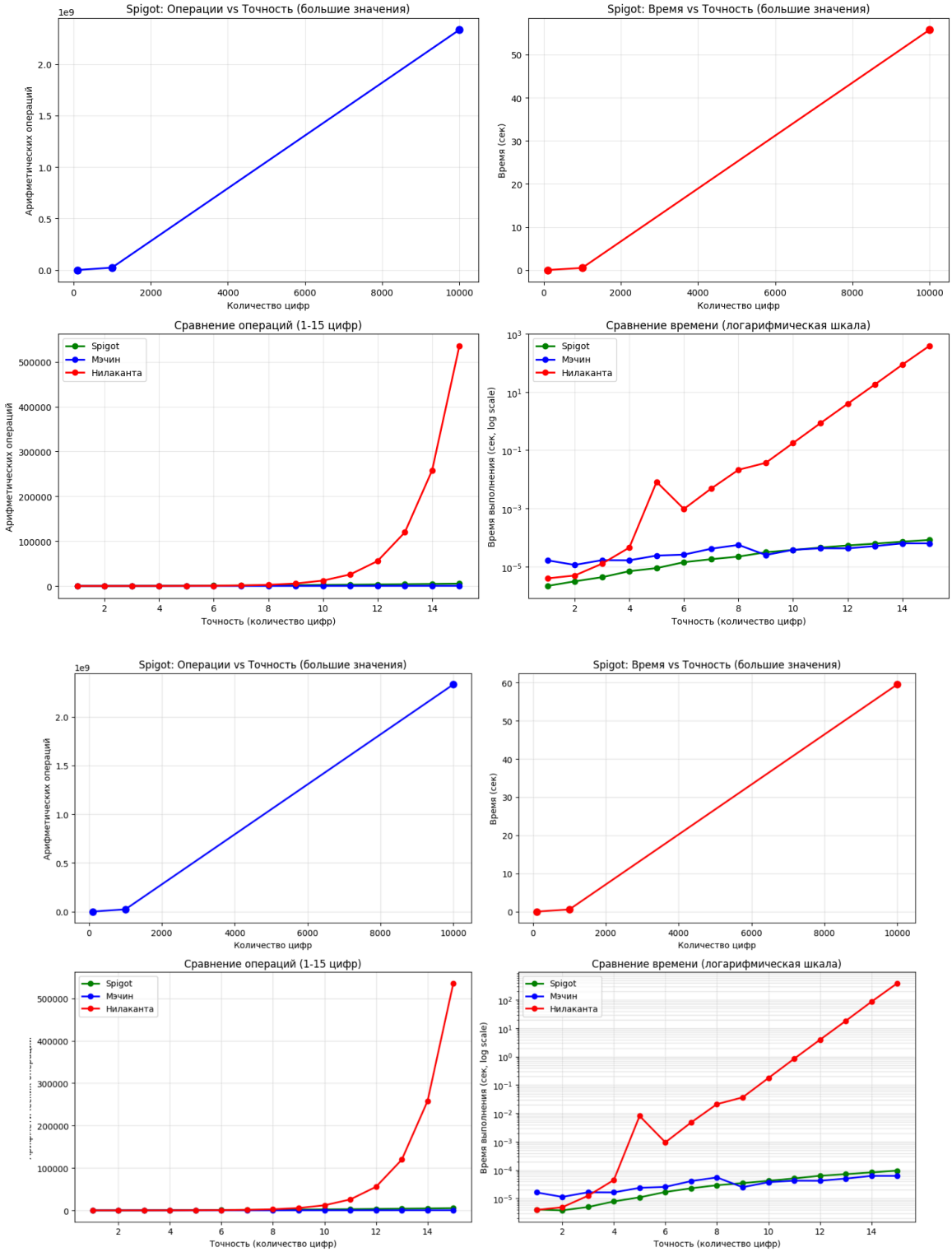
	Основные цифры	Цифр (подмножество)	Оценка операций	Процент от полного
0	100	10	1000	1.0000
1	100	50	25000	25.0000
2	100	100	100000	100.0000
3	1000	10	1000	0.0100
4	1000	50	25000	0.2500
5	1000	100	100000	1.0000
6	1000	500	2500000	25.0000
7	1000	1000	10000000	100.0000
8	10000	10	1000	0.0001
9	10000	50	25000	0.0025
10	10000	100	100000	0.0100
11	10000	500	2500000	0.2500
12	10000	1000	10000000	1.0000

8.4. Графики

In [194…

```
spigot_results_big_formatted, comparison_data_formatted = prepare_d

timestamp = plot_results(spigot_results_big_formatted, comparison_d
```



8.5. Итоговая таблица результатов

In [207…

```
spigot_results_big.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Цифр               3 non-null      int64
 1   Операций           3 non-null      int64
 2   Время (сек)        3 non-null      float64
 3   Итераций           3 non-null      int64
 4   Операций на цифру  3 non-null      float64
 5   Предпросмотр       3 non-null      object
dtypes: float64(2), int64(3), object(1)
memory usage: 276.0+ bytes
```

In [206…

```python
print("Spigot алгоритм (большая точность):")
data = []
for _, r in spigot_results_big.iterrows():
    ops_per_digit = r['Операций'] / r['Цифр']
    data.append(
        {
            'Цифр': r['Цифр'],
            'Операции': f"{r['Операций']:,}",
            'Время (сек)': f"{r['Время (сек)']:.3f}",
            'Итераций': r['Итераций'],
            'Операций на цифру': f"{ops_per_digit:.1f}"
        }
    )

df = pd.DataFrame(data)
df.head(20)
```

```
Spigot алгоритм (большая точность):
```

Out[206…

	Цифр	Операции	Время (сек)	Итераций	Операций на цифру
0	100	233,500	0.017	100	2335.0
1	1000	23,335,000	0.502	1000	23335.0
2	10000	2,333,350,000	55.745	10000	233335.0

	Цифр	Операции	Время (сек)	Итераций	Операций на цифру
0	100	233,500	0.017	100	2335.0
1	1000	23,335,000	0.502	1000	23335.0
2	10000	2,333,350,000	55.745	10000	233335.0

Вывод:

На данной лабораторной работы мы научились реализовывать алгоритм Spigot и сравнили его с алгоритмами из ЛР2