Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

# Лабораторная работа №4

студента 5 курса 531 группы
специальности 10.05.01 «Компьютерная безопасность»
факультета компьютерных наук и информационных технологий
Енца Михаила Владимировича

Преподаватель
профессор _____ В. А. Молчанов
подпись, дата

Заведующий кафедрой
д.ф.-м.н., доцент _____ М. Б. Абросимов
подпись, дата

Саратов 2018

**1. Постановка задачи.**

Изучение основных методов проверки простоты чисел и их программная реализация.

**2. Теоретические сведения по рассмотренным темам с их обоснованием.**

Целое число $n \in Z \setminus \{0, 1\}$ называется простым, если оно не имеет других делителей, кроме 1 и себя самого. В противном случае, число называется составным.

Нечётные составные числа $n$, для которых сравнение $a\ n{-}1 \equiv 1(mod\ n)$ выполняется при любом $a$, $1 \leq a \leq n - 1$, взаимно простом с $n$, называются числами Кармайкла.

Пусть $m, n \in Z$, где $n = p_1 p_2 \ldots p_r$ и числа $p_i > 2, \forall\ i = 1, r$ простые (не обязательно различные). Символ Якоби $\left(\frac{m}{n}\right)$ определяется равенством: $\left(\frac{m}{n}\right) = \left(\frac{m}{p1}\right)\left(\frac{m}{p2}\right) \ldots \left(\frac{m}{pr}\right)$.

Вероятностный алгоритм проверки числа на простоту использует генератор случайных чисел и дает не гарантированно точный ответ. Вероятностные алгоритмы в общем случае не менее эффективны, чем детерминированные.

Для того чтобы проверить вероятностным алгоритмом, является ли целое число $n$ простым, выбирают случайное число $a, 1 < a < n$, и проверяют условие алгоритма. Если число $n$ не проходит тест по основанию $a$, то алгоритм выдает результат «Число $n$ составное», и число $n$ действительно является составным.

Если же $n$ проходит тест по основанию $a$, ничего нельзя сказать о том, действительно ли число $n$ является простым. Последовательно проведя ряд проверок таким тестом для разных $a$ и получив для каждого из них ответ «Число $n$, вероятно, простое», можно утверждать, что число $n$ является простым с вероятностью, близкой к 1. После $t$ независимых выполнений теста

вероятность того, что составное число $n$ будет $t$ раз объявлено простым, не превосходит $\frac{1}{2^t}$.

**Теорема**. Для нечетного составного числа $n > 0$ справедливы следующие утверждения.

1. Число $n$ является псевдопростым по основанию $a$ тогда и только тогда, когда $n-1$ делится на порядок числа $a$ по модулю $n$.

2. Если число $n$ псевдопростое по основаниям $a$ и $b$, то $n$ псевдопростое по основаниям $ab(mod\ n), ab^{-1}(mod\ n)$ и $a^{-1}b(mod\ n)$.

3. Если число $n$ не является псевдопростым хотя бы по одному основанию $a$, то $n$ является псевдопростым не более чем по $\frac{\varphi(n)}{2}$ основаниям, где $\varphi$ – функция Эйлера.

**Теорема. Критерия Корселта.**

Нечетное составное число n является числом Кармайкла тогда и только тогда, когда:

1) $n$ свободно от квадратов

2) для каждого простого делителя $p$ числа $n$ число $n-1$ делится на $p-1$.

**3. Результаты работы.**

**Описание алгоритма теста Ферма проверки чисел на простоту.**

Согласно малой теореме Ферма для простого числа $p$ и произвольного целого числа $a$, $1 \le a \le p-1$, выполняется сравнение $a^{p-1} \equiv 1(mod\ p)$.

Следовательно, если для нечетного $n$ существует такое целое $a$, что $1 \le a < n$, НОД$(a,n) = 1$ и $a^{n-1} \equiv 1(mod\ n)$, то число $n$ составное.

**Описание алгоритма теста Словея-Штрассена проверки чисел на простоту.**

Теорема. Критерий Эйлера

Нечетное число $n$ является простым тогда и только тогда, когда для любого целого числа $a$, $1 \le a \le n-1$, взаимно простого с $n$, выполняется сравнение $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right)(mod\ n)$.

Критерий Эйлера лежит в основе вероятностного алгоритма Соловея-Штрассена.

Определение.

Пусть число $n$ нечетное составное и число $a$ произвольное целое, взаимно простое с $n$, $2 \leq a \leq n-1$. Число $n$ называется *эйлеровым псевдопростым* по основанию $a$, если выполняется сравнение $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right)(mod\ n)$, т.е. если для числа $n$ алгоритм Соловея-Штрассена выдает результат «Число $n$, вероятно, простое».

Теорема

Для нечетного составного числа $n$ справедливы следующие утверждения.

1. Если число $n$ эйлерово псевдопростое по основанию $a$ и не является таковым по основанию $b$, то оно не эйлерово псевдопростое по основанию $ab(mod\ n)$.

2. Если число $n$ эйлерово псевдопростое по основаниям $a$ и $b$, то $n$ псевдопростое по основаниям $ab(mod\ n), ab^{-1}(mod\ n)$ и $a^{-1}b(mod\ n)$.

3. Если число $b$ не является эйлеровым псевдопростым хотя бы по одному основанию $a$, то $n$ является эйлеровым псевдопростым не более чем по $\frac{\varphi(n)}{2}$ основаниям, где $\varphi$ – функция Эйлера.

4. Если число $n$ является эйлеровым псевдопростым по основанияю $a$, то оно является псевдопростым по основанию $a$.

Вероятность того, что тест Соловэя-Штрассена объявит нечетное составное число n простым, меньше чем $\frac{1}{2}$.

**Описание алгоритма теста Миллера-Рабина проверки чисел на простоту**

Определение

Пусть число $n$ нечетное простое, $n-1=2^s r$, где $r$ – нечетное, и $a$ – произвольное целое число, $1 \leq a \leq n-1$, взаимно простое с $n$. Число $n$ называется сильно псевдопростым по основанию $a$, если $a^r \equiv -1(mod\ n)$.

Вероятность того, что тест Миллера-Рабина объявит нечетное составное число $n$, не являющееся степенью простого числа, простым, меньше чем $\frac{1}{4}$. Для большинства нечетных составных чисел $n$ оснований, по которым $n$ является сильно псевдопростым, на самом деле гораздо меньше чем $\frac{\varphi(n)}{4}$.

**Псевдокоды рассмотренных алгоритмов**

Тест Ферма.

*Вход*. Нечетное целое число $n \geq 5$.

*Выход*. «Число $n$, вероятно, простое». В противном случае результат: «Число $n$ составное».

1. Выбрать случайное целое число $a, 2 \leq a \leq n-2$.

2. Вычислить $r = a^{n-1}(mod\ n)$

3. При $r = 1$ результат: «Число $n$, вероятно, простое». В противном случае результат: «Число $n$ составное».

Тест Соловея-Штрассена.

*Вход*. Нечетное целое число $n \geq 5$.

*Выход*. «Число $n$, вероятно, простое» или «Число $n$ составное».

1. Выбрать случайное целое число $a, 2 \leq a \leq n-2$.

2. Вычислить $r = a^{\frac{n-1}{2}}(mod\ n)$.

3. При $r \neq 1$ и $r \neq n-1$ результат: «Число n составное».

4. Вычислить символ Якоби $s = \left(\frac{a}{n}\right)$.

5. При $r \equiv s\ (mod\ n)$ результат: «Число n составное». В противном случае результат: «Число $n$, вероятно, простое».

Тест Миллера-Рабина.

*Вход*. Нечетное целое число $n \geq 5$.

*Выход*. «Число $n$, вероятно, простое» или «Число $n$ составное».

1. Представить $n-1$ в виде $n-1=2^s r$, где число $r$ нечетное.

2. Выбрать случайное целое число $a, 2 \leq a \leq n-2$.

3. Вычислить $y = a^r (mod\ n)$.

4. При $y \neq 1$ и $y \neq n-1$ выполнить следующие действия.

4.1. Положить $j = 1$.

4.2. Если $j \leq s-1$ и $y \neq n-1$, то

4.2.1. Положить $y = y^2 (mod\ n)$.

4.2.2. При $y = 1$ результат: «Число $n$ составное».

4.2.3. Положить $j = j + 1$.

4.3. При $y \neq n-1$ результат: «Число $n$ составное».

5. Результат: «Число $n$, вероятно, простое».

**Коды программ, реализующей рассмотренные алгоритмы**

Программа реализована на языке Python (версия интерпретатора 3.6).

Модуль программы *prime.py*:

```
import random
from time import time

import sympy

from utils import jacobi_symbol

def fermat_primality(n, K=5):
    for i in range(K):
        a = random.randint(2, n - 2)
        if pow(a, (n - 1), n) != 1:
            return False
    return True


def solovay_strassen(n, K=10):
    if n == 2: return True
    if not n & 1: return False
    for k in range(K):
        a = random.randrange(2, n - 2)
        r = pow(a, (n - 1) // 2, n)
        if r != 1 and r != n - 1:
            return False
        s = jacobi_symbol(a, n) % n
        if r != s:
            return False
    return True
```

```
def miller_rabin(n, K=10):
    if n == 2 or n == 3:
        return True

    if n < 2 or n % 2 == 0:
        return False

    s, t = 0, n - 1
    while t % 2 == 0:
        t //= 2
        s += 1

    for k in range(K):
        a = random.randrange(2, n - 2)
        x = pow(a, t, n)
        if x == 1 or x == n - 1:
            continue
        for i in range(1, s):
            x = (x * x) % n
            if x == 1:
                return False
            if x == n - 1:
                break
        if x != n - 1:
            return False
    return True


def test(n=3277, K=3):
    fermat = 0
    solo = 0
    milrab = 0
    for i in range(1000):
        if fermat_primality(n, K):
            fermat += 1
        if solovay_strassen(n, K):
            solo += 1
        if miller_rabin(n, K):
            milrab += 1
    print(fermat, solo, milrab)


def prime_error_test(f_test, n, K):
    err = 0
    g_res = sympy.isprime(n)
    for i in range(1000):
        if f_test(n, K) != g_res:
            err += 1
    return err
```

```
def speed_test(f_test, n, K, cnt):
    start = time()
    for i in range(cnt):
        f_test(n, K)
    return time() - start


def main():
    K = 1
    test(3277, K)
    test(1729, K)
    N = 104087
    print(speed_test(fermat_primality, N, K, 200000))
    print(speed_test(solovay_strassen, N, K, 200000))
    print(speed_test(miller_rabin, N, K, 200000))

    trys = list(range(1, 6))
    for k in trys:
        print('Fermat err k={}: '.format(k), prime_error_test(fermat_primality, N, k))

    for k in trys:
        print('Solovey err k={}: '.format(k), prime_error_test(solovay_strassen, N, k))

    for k in trys:
        print('Miller err k={}: '.format(k), prime_error_test(miller_rabin, N, k))


if __name__ == '__main__':
    main()
```

**Оценки сложности рассмотренных алгоритмов**

Сложность алгоритма теста Ферма: $O(log^3 n)$ при умножении в столбик.

Сложность алгоритма теста Соловэя-Штрассена определеяется сложностью вычисления символа Якоби и равна: $O(log^3 a)$.

Сложность алгоритма теста Миллера-Рабина: $O((log a)^3)$.

**Результаты тестирования программ**

Пример работы алгоритмов Евклида

| Тестируемое число | Число повторов теста | Результат | | |
|---|---|---|---|---|
| | | Тест Ферма | Тест Словея-Штрассена | Тест Миллера-Рабина |

| | | | | |
|---|---|---|---|---|
| 113 | 30 | Простое | Простое | Простое |
| 12673 | 1 | Простое | Составное | Составное |

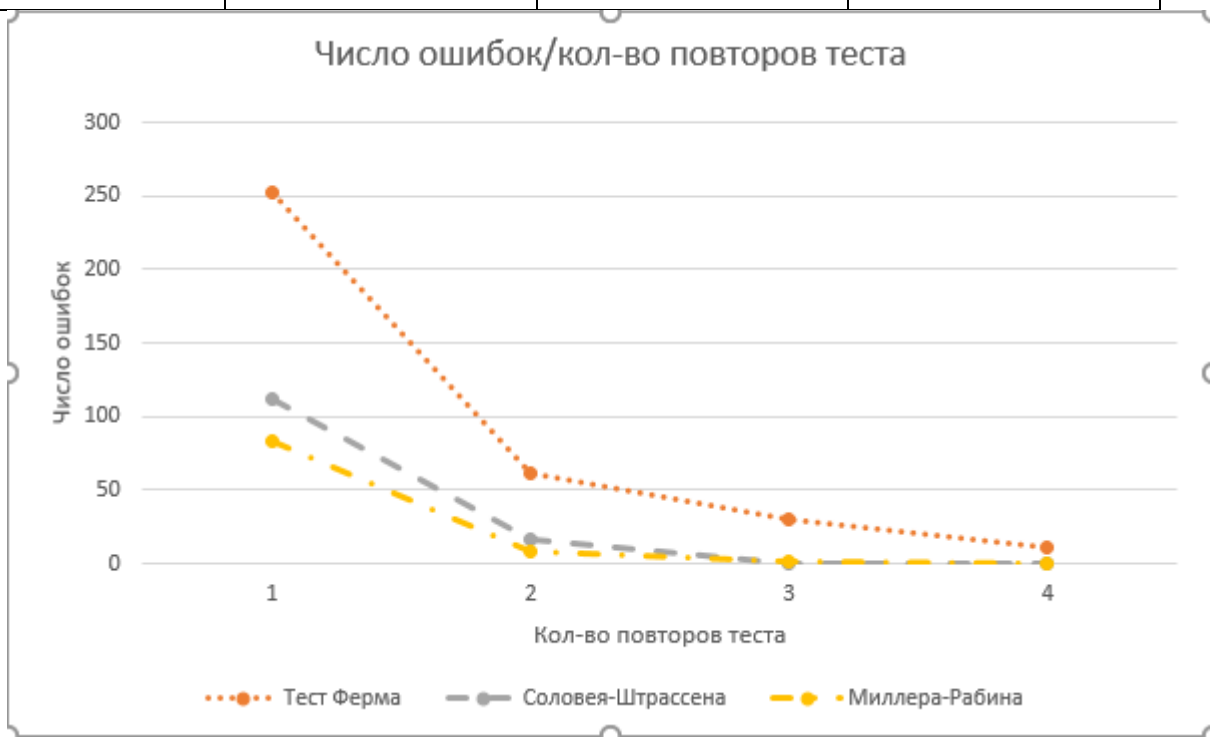
При использовании алгоритмов в силу вероятностного характера проводимого тестирования возможно ошибочное определение составных чисел как простых по двум причинам:

1) число повторов теста слишком мало

2) тест Ферма применен к числам Кармайкла

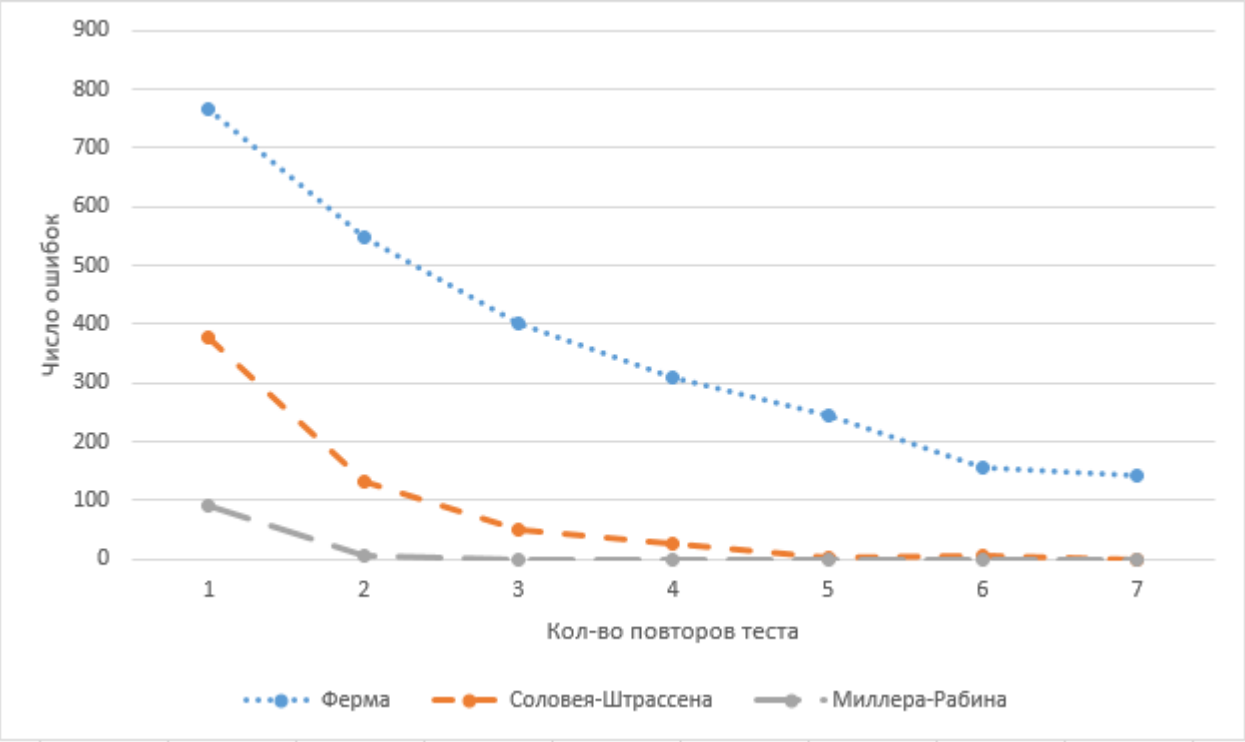
Для исследования зависимости числа ошибок, сделанных каждым тестом (для числа, не являющимся числом Кармайкла), от числа повторов использовалось число 3277. Число повторов – 1000.

| Число повторов | Тест Ферма | Тест Соловея-Штрассена | Тест Миллера-Рабина |
|---|---|---|---|
| 1 | 253 | 112 | 83 |
| 2 | 62 | 16 | 9 |
| 3 | 30 | 0 | 1 |
| 4 | 11 | 0 | 0 |



Для исследования зависимости числа ошибок, сделанных при тестировании числа Кармайкла, от числа проходов использовалось число 1729:

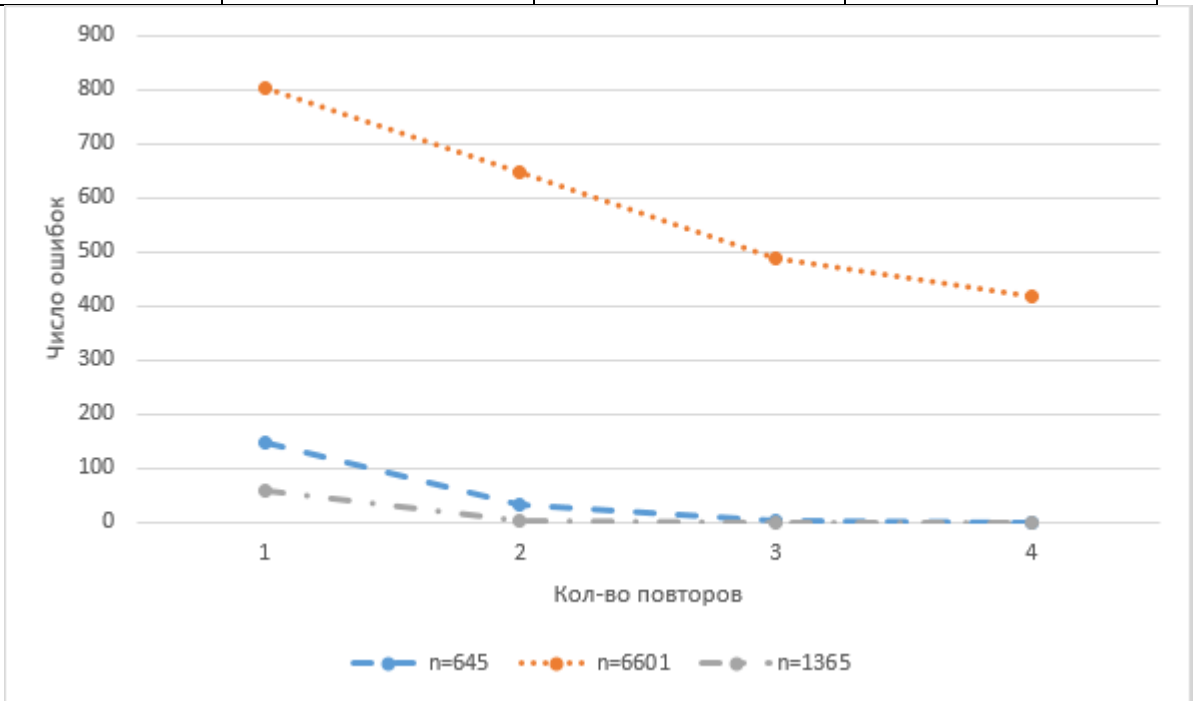
| Число повторов | Тест Ферма | Тест Соловея-Штрассена | Тест Миллера-Рабина |
|---|---|---|---|
| 1 | 766 | 376 | 93 |
| 2 | 547 | 133 | 7 |
| 3 | 402 | 52 | 0 |
| 4 | 308 | 26 | 1 |
| 5 | 246 | 3 | 0 |
| 6 | 157 | 5 | 0 |
| 7 | 144 | 0 | 0 |




Рассмотрим теперь, как влияет выбор числа на каждый из алгоритмов.

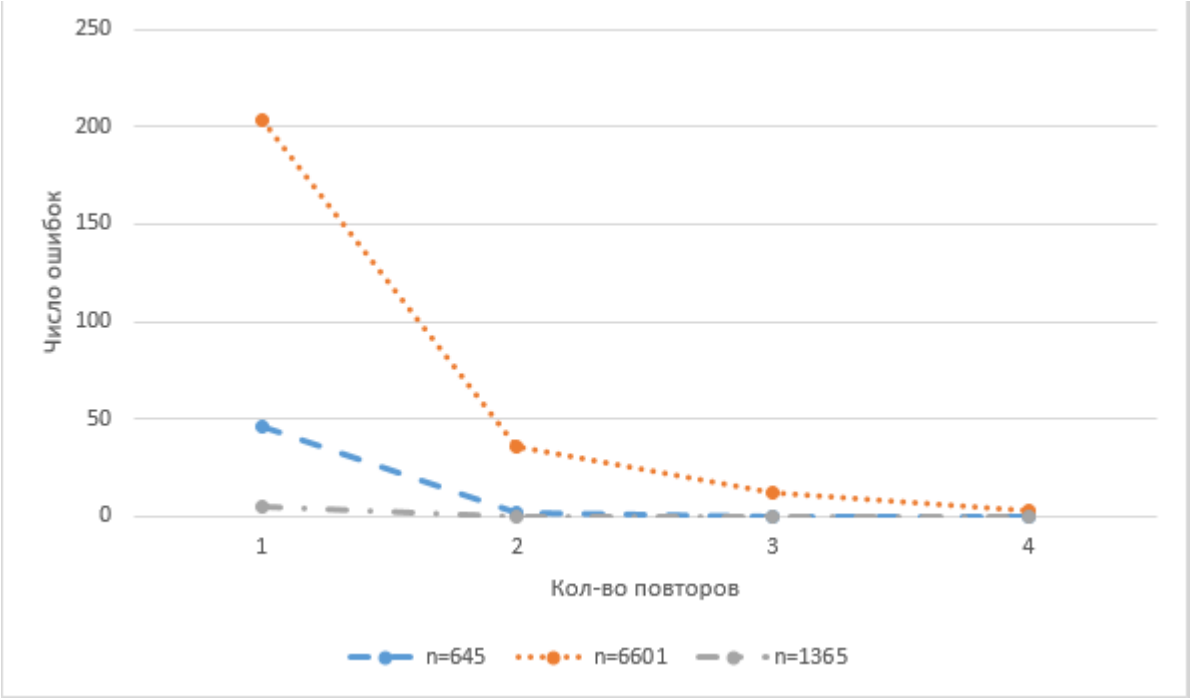
| Число повторов для теста Ферма | Число ошибок | | |
|---|---|---|---|
| | $n = 645$ | $n = 6601$ | $n = 1365$ |
| 1 | 149 | 802 | 58 |
| 2 | 31 | 648 | 2 |
| 3 | 4 | 488 | 0 |

| | | | |
|---|---|---|---|
| 4 | 0 | 418 | 0 |



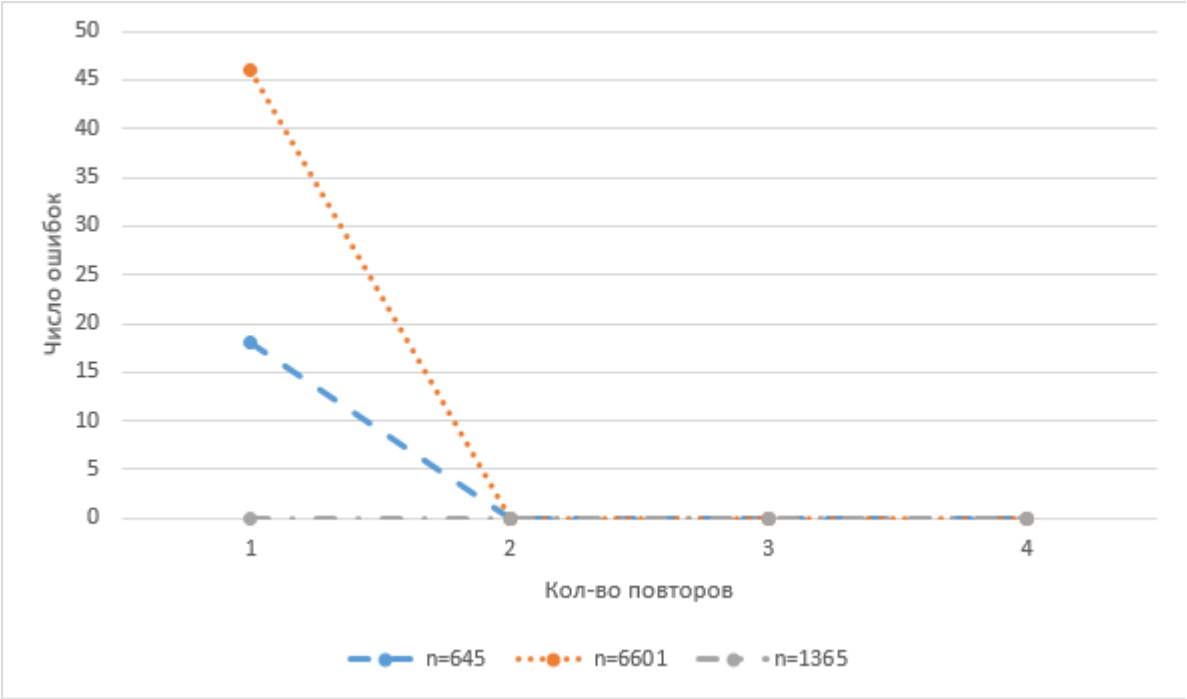

Лучший результат получился для сильно составного числа $1365 = 3 * 5 * 7 * 13$. Для чисел, в каноническое разложение которых входят большие простые множители ($645 = 3 * 5 * 43$ и $2147 = 19 * 113$), тест Ферма работает хуже. Худший результат для данного вида теста получается для чисел Кармайкла.

| Число повторов для теста Соловея-Штрассена | Число ошибок | | |
|---|---|---|---|
| | $n = 645$ | $n = 6601$ | $n = 1365$ |
| 1 | 46 | 203 | 5 |
| 2 | 2 | 36 | 0 |
| 3 | 0 | 12 | 0 |
| 4 | 0 | 3 | 0 |




Для чисел, являющихся сильно составными, или в разложение которых входят большие простые множители, тест Соловея-Штрассена почти не даёт сбоев. Худший результат получается, когда на вход подаётся число Кармайкла, однако с увеличением количества повторов число ошибок теста уменьшается.

| Число повторов для теста Миллера-Рабина | Число ошибок | | |
|---|---|---|---|
| | $n = 645$ | $n = 6601$ | $n = 1365$ |
| 1 | 18 | 46 | 0 |
| 2 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 |




Тест Миллера-Рабина является устойчивым ко всем видам чисел. В частности, для чисел Кармайкла худший результат получается, когда количество повторов теста равно 1. В остальных случаях тест почти не даёт сбоев.

**Результаты тестирования алгоритмов на быстродействие**

Для получения объективной оценки быстродействия алгоритмов нужно применить их к простым числам. Для числа 104087 и 200000 повторов теста получились следующие результаты: время выполнения теста Ферма – 3.9, Соловея-Штрассена – 9.23, Миллера-Рабина – 4.09.

Худший результат по времени показал тест Соловея-Штрассена, но по достоверности он превосходит тест Ферма. Тест Миллера-Рабина хоть и является самым достоверным, однако по скорости не превосходит тест Ферма. Тест Ферма неэффективен по сравнению с тестом Миллера-Рабина, так как не все числа выдерживают проверки на простоту, в особенности числа Кармайкла