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 ∈ 𝑍 \ {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 ≡ 1(𝑚𝑜𝑑 𝑛) выполняется при любом 𝑎, 1 ≤ 𝑎 ≤ 𝑛 − 1, взаимно простом с 𝑛, называются числами Кармайкла.

Пусть , где и числа , простые (не обязательно различные). Символ Якоби определяется равенством: = … .

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 , 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 проходит тест по основанию , ничего нельзя сказать о том, действительно ли число является простым. Последовательно проведя ряд проверок таким тестом для разных 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 .

**Теорема**. Для нечетного составного числа справедливы следующие утверждения.

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

2. Если число псевдопростое по основаниям и , то псевдопростое по основаниям и .

3. Если число не является псевдопростым хотя бы по одному основанию , то является псевдопростым не более чем по основаниям, где – функция Эйлера.

**Теорема. Критерия Корселта.**

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

1) свободно от квадратов

2) для каждого простого делителя числа число делится на .

**3. Результаты работы.**

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

Согласно малой теореме Ферма для простого числа и произвольного целого числа , , выполняется сравнение .

Следовательно, если для нечетного существует такое целое , что , и , то число составное.

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

Теорема. Критерий Эйлера

Нечетное число является простым тогда и только тогда, когда для любого целого числа , , взаимно простого с , выполняется сравнение

Критерий Эйлера лежит в основе вероятностного алгоритма Соловея-Штрассена.

Определение.

Пусть число нечетное составное и число произвольное целое, взаимно простое с , . Число называется *эйлеровым псевдопростым* по основанию , если выполняется сравнение , т.е. если для числа алгоритм Соловея-Штрассена выдает результат «Число , вероятно, простое».

Теорема

Для нечетного составного числа справедливы следующие утверждения.

1. Если число эйлерово псевдопростое по основанию и не является таковым по основанию , то оно не эйлерово псевдопростое по основанию .

2. Если число эйлерово псевдопростое по основаниям и , то псевдопростое по основаниям .

3. Если число не является эйлеровым псевдопростым хотя бы по одному основанию , то является эйлеровым псевдопростым не более чем по основаниям, где – функция Эйлера.

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

Вероятность того, что тест Соловэя-Штрассена объявит нечетное составное число n простым, меньше чем .

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

Определение

Пусть число нечетное простое, , где – нечетное, и – произвольное целое число, , взаимно простое с . Число называется сильно псевдопростым по основанию , если .

Вероятность того, что тест Миллера-Рабина объявит нечетное составное число , не являющееся степенью простого числа, простым, меньше чем . Для большинства нечетных составных чисел оснований, по которым является сильно псевдопростым, на самом деле гораздо меньше чем .

**Псевдокоды рассмотренных алгоритмов**

Тест Ферма.

*Вход*. Нечетное целое число .

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

1. Выбрать случайное целое число .

2. Вычислить

3. При результат: «Число , вероятно, простое». В противном случае результат: «Число составное».

Тест Соловея-Штрассена.

*Вход.* Нечетное целое число .

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

1. Выбрать случайное целое число .

2. Вычислить .

3. При результат: «Число n составное».

4. Вычислить символ Якоби .

5. При результат: «Число n составное». В противном случае результат: «Число , вероятно, простое».

Тест Миллера-Рабина.

*Вход*. Нечетное целое число .

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

1. Представить в виде , где число нечетное.

2. Выбрать случайное целое число .

3. Вычислить .

4. При и выполнить следующие действия.

4.1. Положить .

4.2. Если , то

4.2.1. Положить .

4.2.2. При результат: «Число составное».

4.2.3. Положить .

4.3. При результат: «Число составное».

5. Результат: «Число , 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: при умножении в столбик.

Сложность алгоритма теста Соловэя-Штрассена определеяется сложностью вычисления символа Якоби и равна: .

Сложность алгоритма теста Миллера-Рабина: .

**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 |

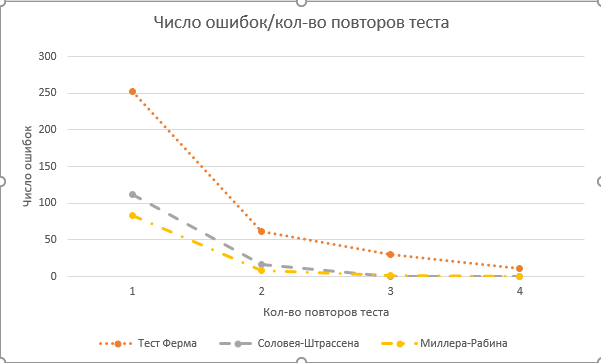
При использовании алгоритмов в силу вероятностного характера проводимого тестирования возможно ошибочное определение составных чисел как простых по двум причинам:

1) число повторов теста слишком мало

2) тест Ферма применен к числам Кармайкла

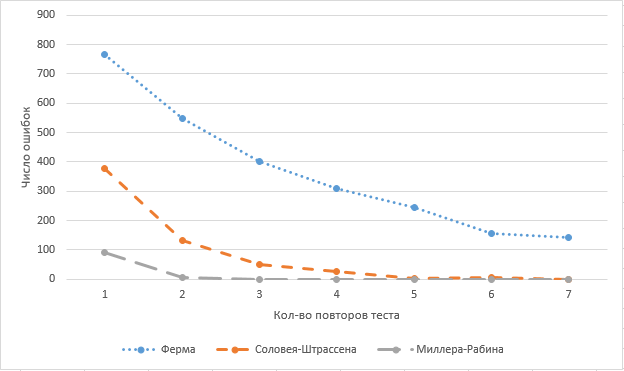
Для исследования зависимости числа ошибок, сделанных каждым тестом (для числа, не являющимся числом Кармайкла), от числа повторов использовалось число 3277. Число повторов – 1000.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |



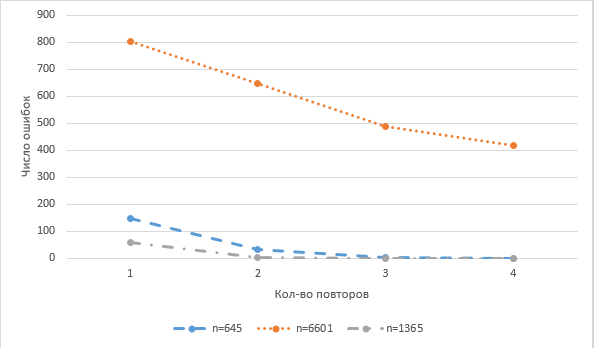
Для исследования зависимости числа ошибок, сделанных при тестировании числа Кармайкла, от числа проходов использовалось число 1729:

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |



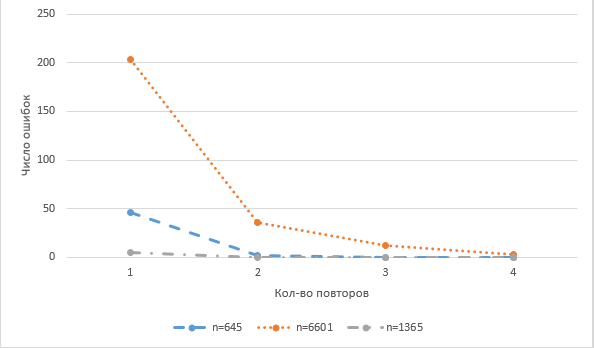
Рассмотрим теперь, как влияет выбор числа на каждый из алгоритмов.

|  |  |  |  |
| --- | --- | --- | --- |
| Число повторов для теста Ферма | Число ошибок | | |
|  |  |  |
| 1 | 149 | 802 | 58 |
| 2 | 31 | 648 | 2 |
| 3 | 4 | 488 | 0 |
| 4 | 0 | 418 | 0 |



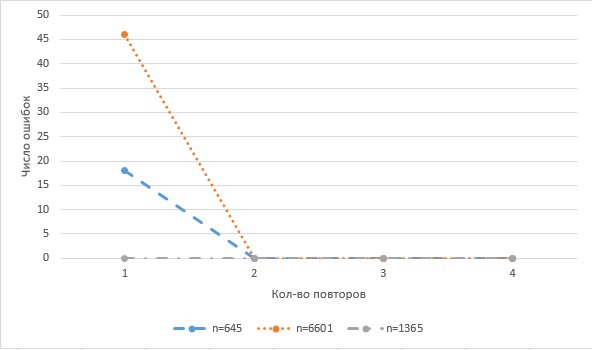
Лучший результат получился для сильно составного числа . Для чисел, в каноническое разложение которых входят большие простые множители (), тест Ферма работает хуже. Худший результат для данного вида теста получается для чисел Кармайкла.

|  |  |  |  |
| --- | --- | --- | --- |
| Число повторов для теста Соловея-Штрассена | Число ошибок | | |
|  |  |  |
| 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 | | |
|  |  |  |
| 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