МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Проверка чисел на простоту

ОТЧЁТ
ПО ДИСЦИПЛИНЕ
«ТЕОРЕТИКО-ЧИСЛОВЫЕ МЕТОДЫ В КРИПТОГРАФИИ»

студента 5 курса 531 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Сенокосова Владислава Владимировича

Преподаватель, профессор _____ В.А. Молчанов
подпись, дата

Саратов 2024

Содержание

1 Цель работы и порядок выполнения

Цель работы — изучение основных методов проверки простоты чисел и их программная реализация.

Порядок выполнения работы

1. Рассмотреть тест Ферма проверки чисел на простоту и привести его программную реализацию.

2. Рассмотреть тест Соловея-Штрассена проверки чисел на простоту и привести его программную реализацию.

3. Рассмотреть тест Миллера-Рабина и привести его программную реализацию.

2 Алгоритм проверки чисел на простоту Ферма

В основе любого теста простоты чаще всего лежит некоторый критерий простоты числа N, состоящий из конечной серии условий простоты. Алгоритмы, основанные на проверке части условий критерия простоты, принято называть вероятностными тестами простоты. Таким образом получаем следующее определение.

Вероятностный тест проверки числа на простоту — это алгоритм, который проверяет, является ли заданное число n простым, но не с абсолютной точностью, а с определённой вероятностью. Такой тест может иногда ошибаться, заявляя составное число простым (ложно положительный результат), но обычно вероятность ошибки можно сделать сколь угодно малой.

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

Следовательно, если для нечетного n существует такое целое a, что $1 \leq a \leq p - 1$, НОД(a, n) = 1 и $a^{n-1} \equiv 1\ (mod\ n)$, то число n составное. Отсюда получаем следующий вероятностный алгоритм проверки числа на простоту.

Алгоритм на основе теста Ферма:

Вход: нечетное целое число $n \geq 5$ и k – количество раундов проверки.

Выход: «число n, вероятно, простое» или «число n составное».

1. Выбрать случайное целое число a, $2 \leq a \leq n - 2$,
2. Вычислить $r = a^{n-1}\ (mod\ n)$,
3. Если $r \neq 1$, завершить алгоритм с результатом «число n составное» в противном случае с результат: «число n, вероятно, простое».

Сложность алгоритма теста Ферма: $O(\log^3 n)$

Определение. Число n называется псевдопростым по основанию a, если для чисел a и n выполняется сравнение $a^{n-1} \equiv 1\ (mod\ n)$. Нетрудно видеть,

что N является псевдопростым по основанию a в том и только в том случае, когда $(a, N) = 1$ и порядок элемента a в группе Z_n делит число $N - 1$. Кроме того, заметим, что псевдопростое по основанию a число не обязательно является простым.

Утверждение. Пусть N нечетное число. Тогда выполняются утверждения:

а) множество всех $a \in Z_N$, относительно которых N является псевдопростым, образует подгруппу в Z_N;

б) если N не является псевдопростым хотя бы по одному основанию a, то N не является псевдопростым относительно по крайней мере половины чисел из Z_n.

Доказательство: Утверждение а) доказывается с помощью критерия быть подгруппой в конечной группе. Утверждение б) следует из пункта а) и теоремы Лагранжа о порядке подгруппы конечной группы. Действительно, если

$$H_N = \{a \in Z_N |\, a^{N-1} = 1 \ (mod\ N)\}$$

то по пункту а) $H_N < Z_N$. По условию пункта б) $H_N \neq Z_N$. По теореме Лагранжа $|H_N|$ делит $|Z_N|$. Учитывая все сказанное, получаем $\frac{|H_N|}{|Z_N|} \leq \frac{1}{2}$.

Введем понятие вероятности успеха в алгоритме.

Пусть N — составное число. Тогда под вероятностью успеха будем понимать вероятность события, состоящего в том, что алгоритм выдаст ответ: «N — составное». Эта вероятность, очевидно, равна:

$$P_0 = 1 - \frac{|H_N|}{N - 1}$$

Для повышения эффективности проверки числа на простоту и пользуют раунды. В реализованном ниже алгоритме и далее будет использоваться k – в качестве количества раундов.

Псевдокод реализованного алгоритма Ферма:

Функция $ferma(n, k)$:

Вход: n — число для проверки

k — количество раундов проверки

Выход: $True$ — если число вероятно простое

$False$ — если число составное

Если $n < 1$, вернуть $False$

Если $n \leq 3$, вернуть $True$ (так как числа 2 и 3 простые)

Для i от 1 до k (выполняем k раундов проверки):

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

Вычислить $r = a^{n-1} \, mod \, n$

Если $r \neq 1$, вернуть $False$ (число составное)

Если все раунды завершены успешно, вернуть $True$ (число вероятно простое)

Сложность реализованного алгоритма теста Ферма: $O(k \log^3 n)$

3 Алгоритм проверки чисел на простоту Соловея-Штрассена

Тест Соловея-Штрассена также является вероятностным тестом простоты. Он основан на сравнении значений символа Якоби и вычисления модульного возведения в степень. В его основе лежит критерий Эйлера.

Критерий Эйлера. Нечетное число n является простым тогда и только тогда, когда для любого a выполняется свойство: $E_n(a) = (a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right)(mod\ n))$. n −простое число тогда и только тогда, когда $E_n^+ = Z_n^*$, где $E_n^+ = \{a \in \{1, 2, \dots, n-1\} | E_n(a)\}$.

Алгоритм на основе теста Соловея-Штрассена:

Вход: нечетное целое число $n \geq 5$ и k – количество раундов проверки. Выход: «число n, вероятно, простое» или «число n составное».

1. Выбрать случайное целое число $a, 2 \leq a \leq n - 2$;
2. Вычислить $r = a^{\frac{n-1}{2}}\ (mod\ n)$, завершить алгоритм с результатом «число n составное».
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 $r = s\ (mod\ n)$ результат: «Число n составное». В противном случае результат: «Число n, вероятность, простое».

Определение: Число n называется эйлеровым псевдопростым по основанию a, если для чисел a, n выполняется сравнение $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right)(mod\ n)$. Заметим, что эйлерово псевдопростое по основанию a число n не обязательно является простым.

Утверждение. Пусть n нечетное число. Тогда выполняются утверждения:

а) множество всех $a \in Z_n$, относительно которых n является эйлеровым псевдопростым, образует подгруппу в Z_n;

б) если n – составное число, то n не является псевдопростым эйлеровым числом относительно по крайней мере половины чисел из Z_n.

Доказательство: Утверждение а) доказывается с по мощью критерия быть подгруппой в конечной группе и свойств символа Якоби. Утверждение б) следует из пункта а) и теоремы Лагранжа о порядке подгруппы конечной группы. Действительно, если

$$K_n = \{a \in Z_n |\, a^{\frac{n-1}{2}} = \left(\frac{a}{n}\right) (mod\ n)\}$$

то по пункту а) $K_n < Z_n$. По условию пункта б) $K_n \neq Z_n$. По теореме Лагранжа $|K_n|$ делит $|Z_n|$. Учитывая все сказанное, получаем $\frac{|K_n|}{|Z_n|} \leq \frac{1}{2}$.

Пусть N — составное. Из пункта б) утверждения следует, что вероятность успеха в тесте Соловея–Штрассена оценивается следующим образом:

$$P_0 = 1 - \frac{|K_n|}{n-1} \geq 1 - \frac{|K_n|}{|Z_n|} \geq \frac{1}{2}$$

Таким образом, вероятность успеха теста не менее $\frac{1}{2}$.

Сложность алгоритма теста Соловея-Штрассена: $O(\log^3 n)$

Псевдокод реализованного алгоритма Соловея-Штрассена:

Функция $strassen(k, n)$:

Вход: n — число для проверки k — количество раундов проверки

Выход: $True$ — если число вероятно простое $False$ — если число составное

Если $n < 1$, вернуть $False$

Если $n \leq 3$, вернуть $True$

Если $n == 2$, вернуть $True$ (так как 2 — простое число)

Если n — четное, вернуть $False$ (так как четные числа, кроме 2, не могут быть простыми)

Для i от 1 до k (выполняем k раундов проверки):

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

Если НОД$(a, n) > 1$, вернуть $False$ (число составное)

Вычислить символ Якоби $j = jacobi(a, n)$

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

Если $j \mod n \neq r$, вернуть $False$ (число составное)

Если все раунды завершены успешно, вернуть $True$ (число вероятно простое)

Сложность реализованного алгоритма теста Соловея-Штрассена: $O(klog^3 n)$ где k – количество раундов

4 Алгоритм проверки чисел на простоту Миллера-Рабина

Тест Миллера-Рабина — вероятностный алгоритм для проверки простоты числа. Он эффективен для больших чисел, но может иногда ошибаться, заявляя составное число простым (ложноположительный результат). Однако вероятность ошибки контролируется количеством раундов проверки.

Критерий Миллера. Пусть n нечётное и $n - 1 = 2^s t$ для нечетного t. Тогда n является простым тогда и только тогда, когда для любого a $a \in Z_n^*$ выполняется свойство: $M_n(a) = \{a^t \equiv 1\ (mod\ n) \vee (\exists\ 0 \leq k < s)\ (2^{2^k t} \equiv 1\ (mod\ n))\}$. То есть, n −простое число тогда и только тогда, когда $M_n^+ = Z_n^*$, где $M_n^+ = \{a \in \{1, 2, \dots, n-1\} |\ M_n(a)\}$.

Алгоритм Миллера-Рабина:

Вход: нечетное целое число $n \geq 5$ и k – количество раундов проверки.

Выход: «число n, вероятно, простое» или «число n составное».

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

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

Вычислить $r = 2^r \pmod n$,

Если $y = 1$ или $y = n - 1$, то перейти на следующую итерацию главного цикла,

Выполнить $s - 1$ раз:

Вычислить $y = y^2\ (mod\ n)$

Если $y = 1$, завершить алгоритм с результатом «число n составное»,

Если $y \neq n - 1$, перейти к завершению с результатом «число n составное»

Завершить алгоритм с результатом «число n, вероятно, простое».

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

Определение. Число N псвдопростое по основанию a называется сильно псевдопростым по основанию a, если выполняется одно из условий:

1) Либо $a^u = 1\ (mod\ N)$

2) Либо найдется $k \in \{0, \ldots, t-1\}$ такое, что

$$a^{2^k u} = -1 \ (mod\ N)$$

М. Рабин доказал, что в случае составного N вероятность правильного ответа в тесте (т. е. ответа «N — составное») не менее $\frac{3}{4}$.

Пусть A_N — множество всех $a \in Z_N$, относительно которых N является сильно псевдопростым. Тогда вероятность P_0 успеха в тесте Миллера–Рабина может быть оценена следующим образом:

$$P_0 = 1 - \frac{|A_N|}{N-1} \geq 1 - \frac{|A_N|}{|Z_N|} = 1 - \frac{|A_N|}{phi(N)}$$

где $phi(N) -$ функция Эйлера для числа N.

Псевдокод реализованного алгоритма Соловея-Штрассена:

Функция $\boldsymbol{divider(n)}$:

Вход: n — число для разложения

Выход: s — количество степеней 2, q — нечетный множитель, такой что $n - 1 = 2^s * q$

Инициализировать $s = 0$

Установить $q = n - 1$

Пока q делится на 2:

Увеличить s на 1

Разделить q на 2

Вернуть s и q

Функция $\boldsymbol{check(a, s, q, n)}$:

Вход:

a — случайное число

s — количество степеней 2 в разложении $n - 1$

q — нечетный множитель в разложении $n - 1$

n — число для проверки

Выход: $True$ — если условие выполняется, $False$ — если нет

Для i от 0 до $s - 1$:

Если $a\text{^}(2^i * q) \equiv n - 1 \ (mod\ n)$, вернуть $True$

Вернуть $False$

Функция $robin\_miller(n, k)$:

Вход:

n — число для проверки

k — количество раундов проверки

Выход:

$True$ — если число вероятно простое

$False$ — если число составное

Если $n < 4$, вернуть $True$ (так как 2 и 3 — простые числа)

Если n четное, вернуть $False$ (так как четные числа, кроме 2, не могут быть простыми)

Для i от 1 до k (выполняем k раундов проверки):

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

Вызвать делитель (n), чтобы получить s и q, где $n - 1 = 2^s * q$

Если НОД$(n, a) \neq 1$, вернуть $False$ (n делится на a)

Если $a^q \equiv 1 \ (mod\ n)$ или $check(a, s, q, n)$ возвращает $True$: продолжить к следующей итерации

Иначе вернуть $False$ (число составное)

Если все раунды завершены успешно, вернуть $True$ (число вероятно простое)

Сложность реализованного алгоритма теста Миллера-Рабина: $O(klog^3\ n)$ где k – количество раундов

5 Тестирование алгоритмов проверки числа на простоту

Программа принимает на вход одно число n и количество раундов k для алгоритмов. Все три алгоритма отрабатывают одинаковое число раундов k, на одном и том же вводном числе и последовательно выводят свой результат. Предварительно у пользователя есть возможность указать какой тест необходимо использовать, либо вывести результаты сразу всех тестов и сравнить полученные результаты.

Тест Ферма. См.Рис.1, 2:

```
Введите тип теста проверки числа на простоту:

1 - Тест Ферма
2 - Тест Соловея-Штрассена
3 - Тест Робина-Миллера
4 - Протестировать все тесты

:>1
Введите число n, которые хотите проверить на простоту
n = 21
Введите число k равное количеству раундов проверки
k = 10
Тест Ферма
Число 21 Составное
```

Рисунок 1 – Число 21 распознано алгоритмом Ферма с числом раундов 10 как составное

```
Введите тип теста проверки числа на простоту:

1 - Тест Ферма
2 - Тест Соловея-Штрассена
3 - Тест Робина-Миллера
4 - Протестировать все тесты

:>1
Введите число n, которые хотите проверить на простоту
n = 23
Введите число k равное количеству раундов проверки
k = 10
Тест Ферма
Число 23 Вероятно простое
```

Рисунок 2 – Число 23 распознано алгоритмом Ферма с числом раундов 10 как простое

Тест Соловея-Штрассена. См.Рис.3, 4:

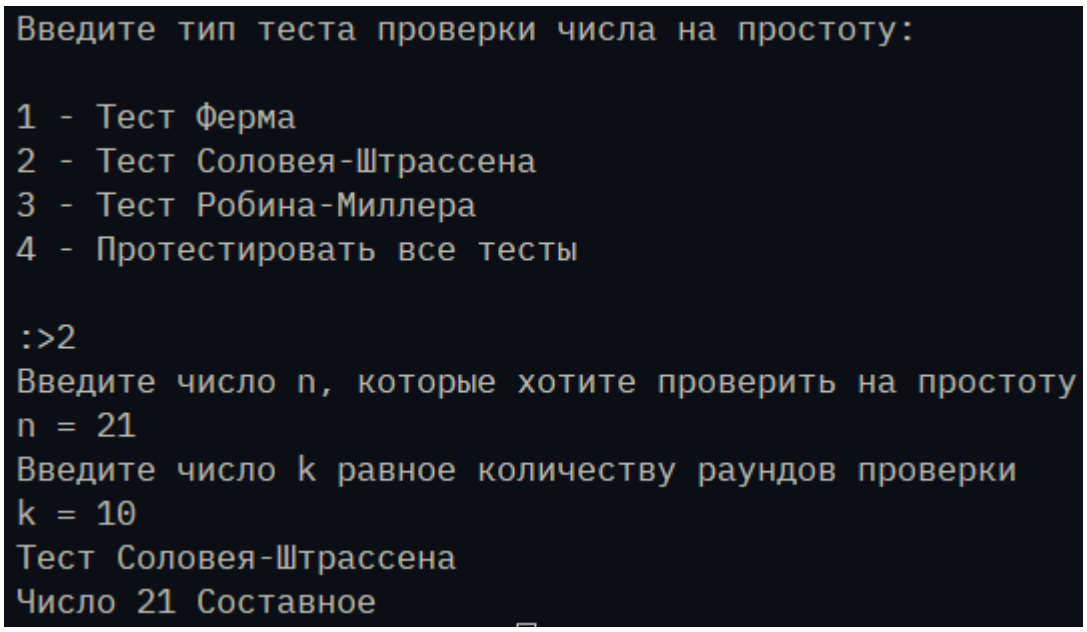

Рисунок 3 – Число 21 распознано алгоритмом Соловея-Штрассена с числом раундов 10 как составное

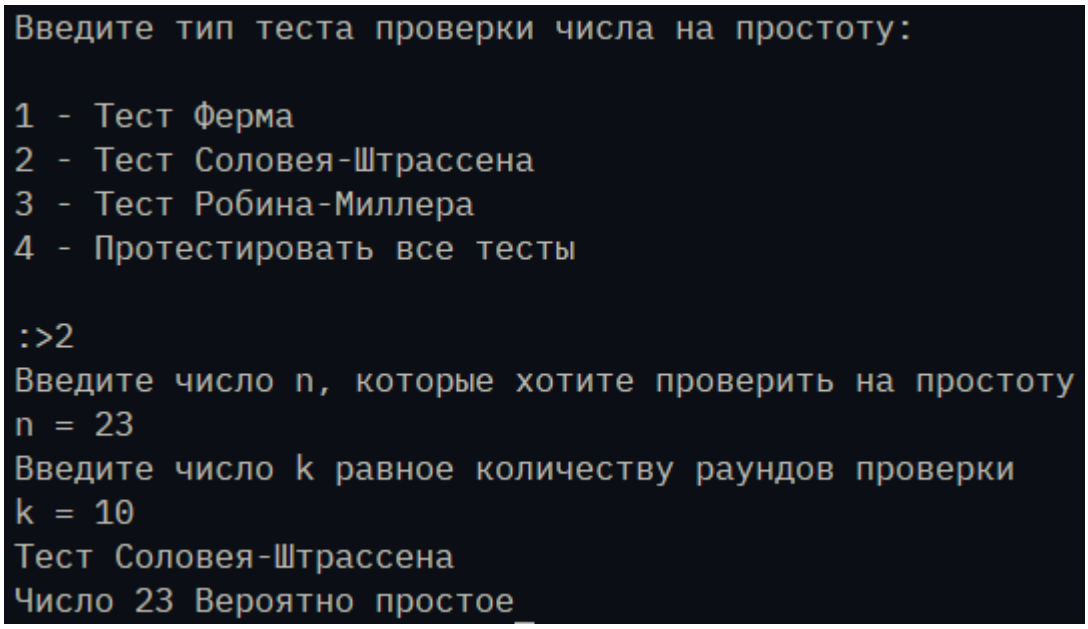

Рисунок 4 – Число 23 распознано алгоритмом Соловея-Штрассена с числом раундов 10 как простое

Тест Робина-Миллера. См.Рис.5, 6:

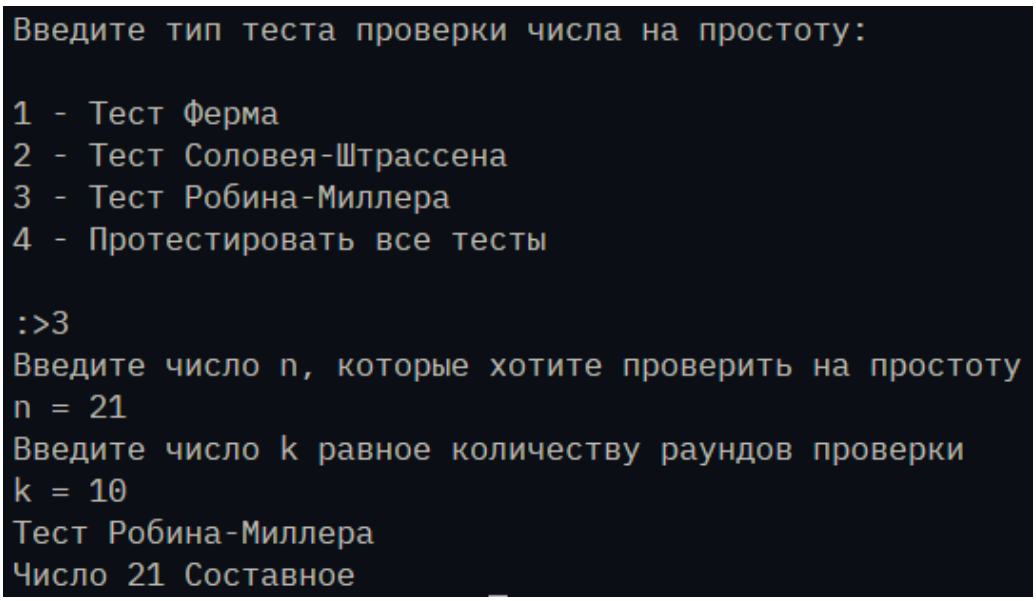

Рисунок 5 – Число 21 распознано алгоритмом Робина-Миллера с числом раундов 10 как составное

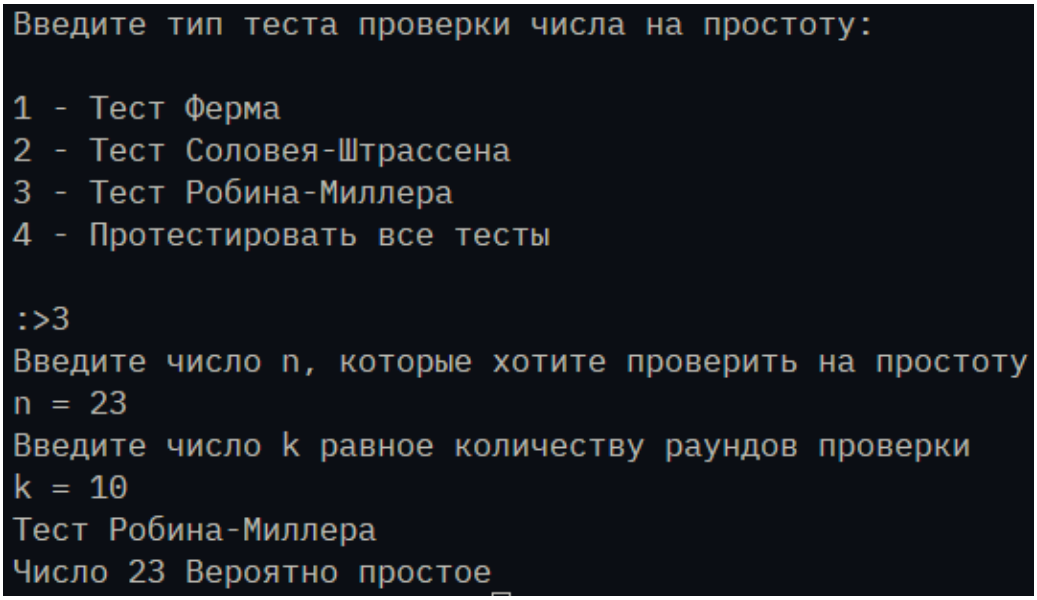

Рисунок 6 – Число 23 распознано алгоритмом Соловея-Штрассена с числом раундов 10 как простое

Тестирование сразу всех алгоритмов на больших числах. См.Рис.7, 8, 9:

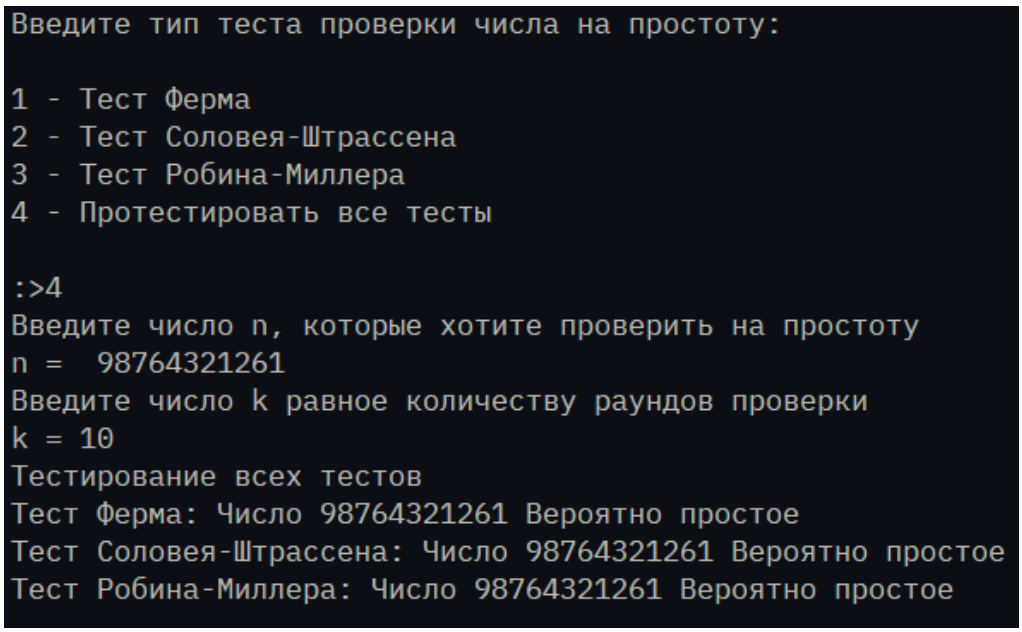

Рисунок 7 – Число 98764321261 распознано как простое с числом раундов 10

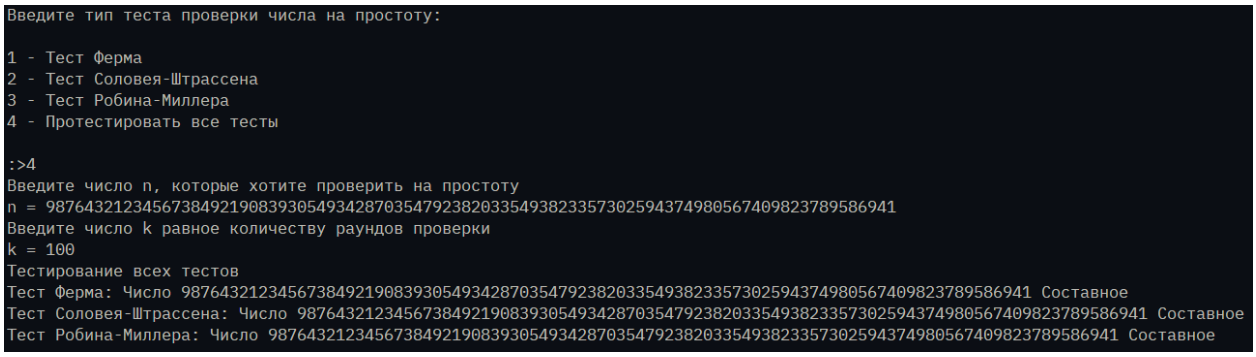

Рисунок 8 – Число
98764321234567384921908393054934287035479238203354938233573025943749805674098
23789586941
распознано как простое с числом раундов 100

ЗАКЛЮЧЕНИЕ

В ходе работы была изучена и реализована программная проверка простоты чисел с использованием трех различных алгоритмов: теста Ферма, теста Соловея-Штрассена и теста Миллера-Рабина. Каждый из этих методов обладает своими преимуществами и недостатками, что делает их применимыми в различных сценариях.

Тест Ферма является простым и быстрым, однако может давать ложные срабатывания для некоторых составных чисел. Тест Соловея-Штрассена более надежен и позволяет значительно уменьшить количество ошибок, связанных с ложными срабатываниями. Тест Миллера-Рабина представляет собой универсальный и эффективный алгоритм, который идеально подходит для проверки больших чисел на простоту.

Все алгоритмы были успешно реализованы на языке Python, что позволило провести тестирование и сопоставление их производительности. Результаты тестов подтвердили, что каждый из методов имеет свои области применения в зависимости от требуемой точности и скорости проверки. Таким образом, полученные результаты подчеркивают важность выбора подходящего метода проверки простоты чисел в зависимости от конкретных задач и условий.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Глухов М. М. и др. Введение в теоретико-числовые методы криптографии: учеб. пособие - Москва : Лань, 2011.

2. Маховенко Е.Б. Теоретико-числовые методы в криптографии. М.: Гелиос АРВ, 2006.

3. Черемушкин, А. В. Лекции по арифметическим алгоритмам в криптографии. - Москва : МЦНМО, 2002.

4. Панкратова И.А. Теоретико-числовые методы в криптографии. Томск, 2009.

5. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии. М.:МЦНМО, 2003.

6. Венбо Мао. Современная криптография: теория и практика. М.:Вильямс, 2005.

ПРИЛОЖЕНИЕ А

Реализация программ проверки числа на простоту

```
import random

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

# Тест Ферма
def ferma(n, k):
    if n < 1:
        return False
    if n <= 3:
        return True
    for _ in range(k):
        a = random.randint(2, n - 2)
        r = pow(a, n - 1, n)
        if r != 1: return False
    return True

# Функция для вычисления символа Якоби
def jacobi(a, n):
    assert n > 0 and n % 2 == 1
    a = a % n
    t = 1
    while a != 0:
        while a % 2 == 0:
            a //= 2
            r = n % 8
            if r == 3 or r == 5:
                t = -t
        a, n = n, a
        if a % 4 == 3 and n % 4 == 3:
            t = -t
        a = a % n
    return t if n == 1 else 0

# Тест Соловея-Штрассена
def strassen(k, n):
    if n < 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0:
        return False

    for _ in range(k):
```

```
        a = random.randint(2, n - 1)
        if gcd(a, n) > 1:
            return False
        if jacobi(a, n) % n != pow(a, (n - 1) // 2, n):
            return False
    return True

# n - 1 = 2^Sq
def divider(n):
    s = 0
    q = n - 1
    while q % 2 == 0:
        s += 1
        q //= 2
    return s, q

# a^((2^k)q) ≡ -1 (mod m) =>  -1 ≡ m - 1 (mod m)
def check(a, s, q, n):
    for i in range(s):
        if pow(a, pow(2, i) * q, n) == n - 1:
            return True
    return False

# Тест Робина-Миллера
def robin_miller(n, k=100):
    if n < 4:
        return True
    if n % 2 == 0:
        return False
    else:
        for _ in range(k):
            a = random.randint(2, n - 1)
            s, q = divider(n)
            if n % a != 0 and (pow(a, q, n) == 1 or check(a, s, q,
n)):
                    continue
            else:
                return False
        return True

if __name__ == "__main__":
    type_ = """Введите тип теста проверки числа на простоту: \n
1 - Тест Ферма
2 - Тест Соловея-Штрассена
3 - Тест Робина-Миллера
4 - Протестировать все тесты\n"""
    print(type_)
    param = None
    while param not in ["1", "2", "3", "4"]:
        param = input(":>")
```

```
    n, k = "",""
    while not (n.isdigit() and k.isdigit()):
        n = input("Введите число n, которые хотите проверить на
простоту\nn = ")
        k = input("Введите число k равное количеству раундов
проверки\nk = ")
    match param:
        case "1":
            print("Тест Ферма")
            res = ferma(int(n), int(k))
            s = "Вероятно простое" if res else "Составное"
            print(f"Число {n} {s}")
        case "2":
            print("Тест Соловея-Штрассена")
            res = strassen(int(k), int(n))
            s = "Вероятно простое" if res else "Составное"
            print(f"Число {n} {s}")
        case "3":
            print("Тест Робина-Миллера")
            res = robin_miller(int(n), int(k))
            s = "Вероятно простое" if res else "Составное"
            print(f"Число {n} {s}")
        case "4":
            print("Тестирование всех тестов")
            res1 = ferma(int(n), int(k))
            res2 = strassen(int(k), int(n))
            res3 = robin_miller(int(n), int(k))
            s1 = "Вероятно простое" if res1 else "Составное"
            s2 = "Вероятно простое" if res2 else "Составное"
            s3 = "Вероятно простое" if res3 else "Составное"
            print(f"Тест Ферма: Число {n} {s1}\nТест Соловея-
Штрассена: Число {n} {s2}\nТест Робина-Миллера: Число {n} {s2}\n")
```