Цель работы

Изучение алгоритмов: Ферма, Соловэя-Штрассена, Миллера-Рабина. Реализация данных алгоритмов программно.

Теоретические сведения

Математической основой современной криптографии является теория чисел. Основным понятием теории чисел, применяющимся в области защиты информации, является простое число. Простым числом p называется натуральное число большее единицы, имеющее только два различных натуральных делителя: единицу и само себя. Изучение простых чисел и их свойств ведёт своё начало с древнейших времён.

На сегодняшний день вопрос простых чисел является основной нерешённой проблемой целочисленной арифметики и исследования в данной области имеют высокую научную ценность. Можно выделить три основных вопроса в данной проблеме простых чисел:

- как построить простое число (актуально при построении огромных простых чисел, используемых в качестве секретных ключей шифрования в некоторых криптографических системах);
- как проверить натуральное число на простоту (поиск наиболее эффективного и точного метода тестирования числа на простоту);
- как получить факторизацию числа (разложение числа на простые множители).

Методы тестирования числа на простоту очень востребованы в криптографических алгоритмах. Поэтому решение именно второго вопроса имеет наибольшее техническое и экономическое значение, так как не существует единого эффективного способа быстрого определения простоты числа и это замедляет работу криптографических алгоритмов. На практике различные методы проверки на простоту натурального числа применяются в разных криптографических алгоритмах, например, в криптографической системе с открытым ключом RSA. Данный криптографический алгоритм основан на том, что факторизация больших натуральных чисел – очень трудная вычислительная операция. Благодаря этому данный алгоритм до сих пор остаётся наиболее эффективным и часто применяемым при шифровании для защиты информации. Но скорость выполнения данного алгоритма шифрования и его криптографическая устойчивость целиком и полностью зависят от выбора пары достаточно больших простых чисел, на основе которых собственно и будет осуществляться процесс шифрования данных.

Также существует множество критериев, по которым классифицируют алгоритмы проверки случайного натурального числа на простоту, но основным является критерий достоверности полученного результата. Согласно данному критерию алгоритмы делятся на *детерминированные* и *вероятностные*.

Особенность *детерминированных* тестов заключается в том, что они гарантированно выдают точный ответ: простое или составное заданное натуральное число. Но главный недостаток существующих *детерминированных* тестов – это огромная вычислительная сложность, и, следовательно, невозможность их применения при больших числах, которые востребованы на практике. Данные тесты рационально применять только на достаточно малых числах.

Вероятностные тесты характеризуются значительно меньшим временем выполнения тестирования числа, поэтому именно такого типа тесты применяются на практике. Но результат, который получается после выполнения теста, является достоверным лишь с некоторой вероятностью, если он положительный (исследуемое число является простым), или полностью достоверным при отрицательном результате (исследуемое число является составным).

Тест Ферма

Французский математик Пьер Ферма в 17 веке выдал закономерность (*Малая теорема Ферма*), которая лежит в основе почти всех методов проверки на простоту:

- 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 $r = a^{n-1}(mod n)$
3. При $r = 1$ результат: «Число n, вероятно, простое». В противном случае результат: «Число n составное».

Тест Соловэя-Штрассена

Роберт Соловей и *Фолькер Штрассен* разработали алгоритм вероятностного тестирования простоты числа, который использует символ Якоби. Определяет числа как составные или вероятно простые.

- 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 $s = (\frac{a}{n})$
5. При $r = s(mod n)$ результат: «Число n, вероятно, простое». В противном случае результат: «Число n составное».

Тест Миллера-Рабина.

Тест Миллера-Рабина — вероятностный полиномиальный тест простоты. Тест Миллера-Рабина позволяет эффективно определять, является ли данное число составным. Однако, с его помощью нельзя строго доказать простоту числа. Тем не менее тест Миллера-Рабина часто используется в криптографии для получения больших случайных простых чисел.

- 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 r - нечетное число
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 действия
    - Положить $j = 1$
    - Если $j \leq s-1$ и $y \neq n-1$ то
        - Положить $y = y^2(mod n)$
        - При $y = 1$ результат: «Число n составное».

        - Положить $j = j + 1$
    - При $y \neq n - 1$ результат: «Число n составное».
5. Результат: «Число n, вероятно, простое».

Выполнение работы

Реализация алгоритмов на языке Python

```
import random

# Тест ферма
# n          - число которое проверяется
# test_count - это количество прогонов
def ferma(n, test_count):
    for i in range(test_count):
        a = random.randint(2, n - 2)
        if (a ** (n - 1) % n != 1):
            return False
    return True

# функция для бинарного эксп
def modulo(base, exponent, mod):
    x = 1
    y = base
    while (exponent > 0):
        if (exponent % 2 == 1):
            x = (x * y) % mod

        y = (y * y) % mod
        exponent = exponent // 2

    return x % mod

# Алгоритм вычисления символа Якоби
def  jacobi(a, n):
    if (a == 0):
        return 0

    ans = 1
    if (a < 0):
        a = -a
        if (n % 4 == 3):
            ans = -ans
    if (a == 1):
        return ans
    while (a):
        if (a < 0):
            a = -a
            if (n % 4 == 3):
                ans = -ans
        while (a % 2 == 0):
            a = a // 2
            if (n % 8 == 3 or n % 8 == 5):
                ans = -ans
        # меняем местами
        a, n = n, a
```

```
        if (a % 4 == 3 and n % 4 == 3):
            ans = -ans

        a = a % n
        if (a > n // 2):
            a = a - n

    if (n == 1):
        return ans

    return 0

# Алгоритм теста Соловэя-Штрассена
def solovoyStrassen(p, test_count):
    if (p < 2):
        return False
    if (p != 2 and p % 2 == 0):
        return False

    for i in range(test_count):
        a = random.randrange(p - 1) + 1
        jacobian = (p +  jacobi(a, p)) % p
        mod = modulo(a, (p - 1) / 2, p)

        if (jacobian == 0 or mod != jacobian):
            return False
    return True

# Алгоритм Миллера Рабина
def miller_rabin(n):
    s = 0
    d = n - 1
    while d % 2 == 0:
        d >>= 1
        s += 1
    assert (2 ** s * d == n - 1)

    def trial_composite(a):
        if pow(a, d, n) == 1:
            return False
        for i in range(s):
            if pow(a, 2 ** i * d, n) == n - 1:
                return False
        return True

    for i in range(8):
        a = random.randrange(2, n)
        if trial_composite(a):
            return False
    return True

def otvet(x,n):
    # если результат был: true - простое, false - сооставное
    if x:
        print(n, "- вероятно простое число");
    else:
        print(n, "- составное число");
    return
```

```
def main():
    n = 1
    while (n < 5) or (n % 2 == 0):
        print("Введите нечетное целое число n>=5")
        n = int(input("n = "))

    print("\nТест Ферма:")
    otvet(ferma(n, 200), n)

    print("\nТест Соловэя-Штрассена:")
    otvet(solovoyStrassen(n, 200),n)

    print("\nТест Миллера-Рабина:")
    otvet(miller_rabin(n),n)
```

Контрольный пример

Ввод [2]:
```
main()
```

```
Введите нечетное целое число n>=5
n = 30327

Тест Ферма:
30327 - составное число

Тест Соловэя-Штрассена:
30327 - составное число

Тест Миллера-Рабина:
30327 - составное число
```

Ввод [3]:
```
main()
```

```
Введите нечетное целое число n>=5
n = 9973

Тест Ферма:
9973 - вероятно простое число

Тест Соловэя-Штрассена:
9973 - вероятно простое число

Тест Миллера-Рабина:
9973 - вероятно простое число
```

Выводы

Изучили алгоритмы Ферма, Соловэя-Штрассена, Миллера-Рабина.

Список литературы

1. Алгоритмы тестирования на простоту и факторизации
2. Проверка простых чисел, защита информации
3. Алгоритм Соловея-Штрассена
4. Тест простоты Миллера-Рабина