

Отчёт по лабораторной работе №5

Ван Яо

Содержание

Цель работы	5
Ход лабораторной работы	6
Теоретические основы	6
Определение	6
Алгоритмы реализации	6
1. Тест Ферма	6
2. Тест Соловэя-Шторассена	7
3. Тест Миллера-Рабина	7
Практическая реализация	8
Реализация теста Ферма	8
Реализация теста Соловэя-Штрассена	9
Реализация теста Миллера-Рабина	12
Функциональное тестирование	13
Выводы	14
Литература	15
Приложения	16

Список таблиц

Список иллюстраций

Цель работы

1. изучить теоретические основы вероятностных алгоритмов проверки чисел на простоту
2. реализовать программно три вероятностных теста
 - тест ферма
 - тест Соловэя-Шторассена
 - тест Миллера-Рабина
3. провести сравнительный анализ эффективности и точности алгоритмов

Ход лабораторной работы

Теоретические основы

Определение

Простое число - натуральное число, имеющее ровно два делителя: единицу и само себя **Составное число** - натуральное число, имеющее более двух делителей **Вероятностный алгоритм** - алгоритм, использующий генератор случайных чисел и дающий не гарантированно точный ответ **Детерминированный алгоритм** - алгоритм, всегда действующий по одной схеме и гарантированно решающий поставленную задачу

Алгоритмы реализации

1. Тест Ферма

Основана на малой теореме Ферма: для простого числа p и произвольного числа a , $1 \leq a \leq p - 1$ выполняется сравнение:

$$a^{p-1} \equiv 1 \pmod{p}$$

Алгоритм:

1. Выбрать случайное целое число $a, 2 \leq a \leq n - 2$
2. вычислить $r \leftarrow a^{n-1} \pmod{n}$
3. Если $r = 1$, результат: Число n , вероятно, простое, иначе Число n составное

2. Тест Соловэя-Шторассена

Основная на Критерии Эйлера и вычислении символов Якоби

Алгоритм вычислении символов Якоби:

1. при $a = 0$, результат: 0
2. при $a = 1$, результат: g
3. представить a в виде $a = 2^k a_1$, где a_1 нечетное
4. определить s в зависимости от k и $n \pmod{8}$
5. выполнить рекурсивные вычисления

Алгоритм:

1. выбрать случайное целое число $a, 2 \leq a \leq n - 2$
2. вычислять $r \leftarrow a^{\frac{n-1}{2}} \pmod{n}$
3. если $r \neq 1$ и $r \neq n - 1$, результат: Число составное
4. вычислять символ якоби $s \leftarrow \begin{pmatrix} a \\ n \end{pmatrix}$
5. если $r \not\equiv s \pmod{n}$, результат :число n составное, иначе число n , вероятно простое

3. Тест Миллера-Рабина

Алгоритм:

1. представить $n - 1$ в виде $n - 1 = 2^s r$, где r нечетное
2. выбрать случайное целое число $a, 2 \leq a < n - 2$
3. вычислять $y \leftarrow a^r \pmod{n}$
4. если $y \neq 1$ и $y \neq n - 1$, выполнить итерации возведения в квадрат
5. если $y = 1$, результат :число n составное
6. если после всех итераций $y \neq n - 1$, результат :число n составное, иначе число n , вероятно простое

Практическая реализация

Реализация теста Ферма

```
def fermat_test(n,k=5):
    if n==2 or n==3:
        return "probably prime"
    if n<=1 or n%2==0:
        return "composite"

    for _ in range(k):
        a=random.randint(2,n-2)

        if gcd(a,n)!=1:
            return "composite"

        if pow(a,n-1,n)!=1:
            return "composite"

    return "probably prime"
```

Реализация теста Соловэя-Штрассена

```
def jacobi_symbol(a,n):
    if n%2==0 or n<3:
        raise ValueError("n должно быть нечетным и n>=3")

    g=1
    while True:
        if a==0:
            return 0

        if a==1:
            return g

        k=0
        a1=a
        while a1%2==0:
            k+=1
            a1//=2

        s=1
        if k%2==1:
            r8=n%8
            if r8==3 or r8==5:
                s=-1

        if a1==1:
            return g*s

        if n%4==3 and a1%4==3:
            s=-s

        a=n%a1
        n=a1
        g=g*s
```

```
def solovay_strassen_test(n,k=5):
    if n==2:
        return "probably prime"

    if n<=1 or n%2==0:
        return "composite"

    for _ in range(k):
        a=random.randint(2,n-2)

        if gcd(a,n)!=1:
            return "composite"

        exponent=(n-1)//2
        r=pow(a,exponent,n)

        if r!=1 and r!=n-1:
            return "composite"

        s=jacobi_symbol(a,n)

        if s== -1:
            s_mod=n-1

        else:
            s_mod=s

        if r!=s_mod%n:
            return "composite"

    return "probably prime "
```

Реализация теста Миллера-Рабина

```
def miller_rabin_test(n,k=5):
    if n==2 or n==3:
        return "probably prime"

    if n<=1 or n%2==0:
        return "composite"

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

    for _ in range(k):
        a=random.randint(2,n-2)

        y=pow(a,r,n)

        if y!=1 and y!=n-1:
            j=1
            while j<=s-1 and y!=n-1:
                y=pow(y,2,n)
                if y==1:
                    return "composite"
                j+=1
            if y!=n-1:
                return "composite"

    return "probably prime"
```

Функциональное тестирование

Число для проверки	Ожидаемый результат	Тест Ферма	Тест Соловэя-Штассена	Тест Миллера-Рабина
17	простое	√	√	√
25	составное	√	√	√
97	составное	√	√	√
561	составное	×	√	√
1105	составное	×	√	√

Выводы

1. Теоретические знания:

Изучены математические основы вероятностных тестов простоты, включая малую теорему Ферма, критерий Эйлера и их применение в криптографии.

2. Практические навыки:

Реализованы три вероятностных алгоритма проверки чисел на простоту. Тест Ферма показал наличие кармайкловых чисел (например, 561), которые проходят тест, но являются составными.

3. Аналитические способности:

Тест Миллера-Рабина показал наибольшую надежность среди рассмотренных алгоритмов, правильно идентифицируя кармайкловы числа.

Литература

1. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии. М.: МЦНМО, 2003.
2. Python Software Foundation. Официальная документация Python. <https://docs.python.org/3/>

Приложения

Полный исходный код программ доступен в репозитории GitHub:

<https://github.com/wangyao200036/cryptography-labs/tree/main/lab5>