Отчёт по лабораторной работе №5

Дисциплина: Математические основы защиты информации и информационной безопасности

Полиенко Анастасия Николаевна, НПМмд-02-23

Содержание

Список иллюстраций

1 Цель работы

Изучить вероятностные алгоритмы проверки чисел на простоту.

2 Задание

Реализовать четыре теста на определение простоты чисел:

1. Тест Ферма
2. Символ Якоби
3. Тест Соловэя-Штрассена
4. Тест Миллера-Рабина

3 Теоретическое введение

Пусть a - целое число. Числа $\pm 1, \pm a$ называются тривиальными делителями числа a.

Целое число $p \in \mathbb{Z}/\{0\}$ называется простым, если оно не является делителем единицы и не имеет других делителей, кроме тривиальных. В противном случае число $p \in \mathbb{Z}/\{-1, 0, 1\}$ называется составным. Например, числа $\pm 2, \pm 3, \pm 5, \pm 7, \pm 11, \pm 13, \pm 17, \pm 19, \pm 23, \pm 29$ являются простыми.

Пусть $n \in \mathbb{N}, m > 1$. Целые числа a и bназываются сравнимыми по модулю m (обозначается $a \equiv b \; (mod \; m)$) если разность $a - b$ делится на m. Также эта процедура называется нахождением остатка от целочисленного деления a на b.

Проверка чисел на простоту является составной частью алгоритмов генерации простых чисел, применяемых в криптографии с открытым ключом. Алгоритмы проверки на простоту можно разделить на вероятностные и детерминированные.

Детерминированный алгоритм всегда действует по одной и той же схеме и гарантированно решает поставленную задачу (или не дает никакого ответа). *Вероятностный* алгоритм использует генератор случайных чисел и гарантированно точный ответ. Вероятностные алгоритмы в общем случае не менее эффективны, чем детерминированные (если используемый генератор случайных чисел всегда дает набор одних и тех же чисел, зависящих от входных данных, то вероятностный алгоритм становится детерминированным).

Для проверки на простоту числа n вероятностным алгоритмом выбирают случайной число a $(1 < a < n)$ и проверяют условия алгоритма. Если число n не проходит тест по основанию a, то алгоритм выдает результат «Число n составное», и число п 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 с независимых выполнений теста вероятность того, что составное число и будет t раз объявлено простым (вероятность ошибки), не превосходит $\frac{1}{2^t}$.

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

$$a^{p-1} \equiv 1 \ (mod\, p)$$

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

Более подробно см. в [1–6].

4 Выполнение лабораторной работы

1. Реализуем тест Ферма (рис. 4.1).

```
n = int(input()) #>5
a = np.random.randint(2, n-2)
r = a ** (n - 1) % n
if r == 1:
    print("число n. вероятно, простое")
else:
    print("число n составное")
```

Рис. 4.1: Тест Ферма

2. Реализуем алгоритм нахождения символа Якоби (рис. 4.2).

```python
def ka(a):
    k = 0
    while a % 2 == 0:
        k += 1
        a /= 2
    return k, a

def jacobi(n, a):
    g = 1
    while True:
        if a == 0:
            return 0
        if a == 1:
            return g
        k, a_1 = ka(a)
        if k % 2 == 0:
            s = 1
        else:
            if n % 8 == 1 or n % 8 == -1:
                s = 1
            elif n % 8 == 3 or n % 8 == -3:
                s = -1
        if a_1 == 1:
            return g * s
        if n % 4 == 3 and a_1 % 4 == 3:
            s = -s
        a = n % a_1
        n = a_1
        g = g * s

n = int(input()) #>3 нечётное
a = int(input()) #0<a<n
jacobi(n, a)
```

Рис. 4.2: Символ Якоби

3. Реализуем тест Соловэя-Штрассена (рис. 4.3).

```
n = int(input()) #>=5 нечётное
a = np.random.randint(2, n-2)
r = a ** ((n - 1)/2) % n
if r != 1 and r != (n-1):
    print("Число n составное")
else:
    s = jacobi(n, a)
    if r % n == s:
        print("число n составное")
    else:
        print("число n, вероятно, простое")
```

Рис. 4.3: Тест Соловэя-Штрассена

4. Реализуем тест Миллера-Рабина (рис. 4.4).

```
n = int(input()) #>=5 нечётное
s, r = ka(n - 1)
a = np.random.randint(2, n-2)
y = a ** r % n
flag = False
if y != 1 and y != (n - 1):
    j = 1
    while j <= (s - 1) and y != (n - 1):
        y = y ** 2 % n
        if y == 1:
            flag = True
        j += 1
    if y != (n - 1):
        flag = True
if flag:
    print("Число n составное")
else:
    print("Число n, вероятно, простое")
```

Рис. 4.4: Тест Миллера-Рабина

5 Выводы

Изучила вероятностные алгоритмы проверки чисел на простоту.

Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: https://www.gnu.org/software/bash/manual/.
2. Newham C. Learning the bash Shell: Unix Shell Programming. O’Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. O’Reilly Media, 2016. 156 с.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.