Отчёт по лабораторной работе 5. Вероятностные алгоритмы проверки чисел на простоту

Ильин Никита Евгеньевич

Содержание

Список таблиц

Список иллюстраций

1 Цель работы

Цель данной работы - научиться реализовывать алгоритмы проверки чисел на простоту.

2 Задание

1. Реализовать алгоритмы проверки чисел на простоту.

3 Теоретическое введение

Пусть a - целое число. Числа ‡1, ‡a называются тривиальными делителями числа a. Целое число p €Z/{0) называется простым, если оно не является делителем единицы и не имеет других делителей, кроме тривиальных. В противном случае число p €Z/{-1, 0, 1) называется составным. являются простыми. Пусть т є N, т > .1 Целые числа a и в называются сравнимыми по модулю т (обозначается a = b (mod m)) если разность a —6 делится на т. 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 дает не 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, от вероятностный алгоритм становится детерминированным). Для проверки на простоту числа п вероятностным алгоритмом выбирают случайной число a 1(< a < n) и проверяют условия алгоритма. Если число и не проходит тест по основанию a, то алгоритм выдает результат «Число п составное», ичисло идействительно является составным. 19 Если же и проходит тест по основанию a, ничего нельзя сказать о том, действительно ли число п является простым. Последовательно проведя ряд проверок таким тестом для разных а и получив для каждого из них ответ «Число п, вероятно, простое», можно утверждать, что число и является простым с вероятностью, близкой к .1 После t независимых выполнений теста вероятность того, что составное число и будет t раз объявлено простым (вероятность ошибки), ен превосходит з Схема вероятностного алгоритма проверки числа на простоту Выбрать случайное число а, 1 < а < n Число п прошло тест по основанию а Число ,п вероятно, простое Число п не прошло тест по основанию а Число псоставное Условие теста Тест Ферма основан на малой теореме Ферма: для простого числа р и произвольного числа а, 1⩽ а ⩽ р- 1, выполняется сравнение аР-1 = 1(mod р). Следовательно, если для нечетного и существует такое целое а, что 1 ⩽ а < n, НОД(а,n) = 1 и а”-1 ‡ 1(mod n), от число и составное. Отсюда получаем следующий вероятностный алгоритм проверки числа на простоту. 20 имени

1. Алгоритм, реализующий тест Ферма. Вход. Нечетное целое число п ⩾ 5. Выход. «Число п, вероятно, простое» или «Число и составное».
2. Выбрать случайное целое число а, 2 ⩽ а ⩽ n - 2. .2 Вычислить r - а”-1 (mod n).
3. При г = 1 результат: «Число п, вероятно, простое». В противном случае результат: «Число п составное». На шаге 1мы ен рассматривали числа а= 1иа= n- 1, поскольку 11-1 = 1(mod n) для любого целого n и (n - 1)^-1 = (-1)^-1 = 1(mod n) для любого нечетного п. Тест Соловэя-Штрассена. Основан на критерии Эйлера: нечетное число n является простым тогда и только тогда, когда для любого целого числа а, 1 ⩽ а ⩽ n - 1, взаимно простого с n, выполняется сравнение: где (Д) -символ Якоби. Пусть т, п є Z, где п = Р1Р2 .Рr и числа Pi ‡ 2 простые (не обязательно различные). Символ Якоби (*) определяется равенством ()⊠()()
4. Алгоритм вычисления символа Якоби. Вход. Нечетное целое число п ⩾ 3, целое число а, 0 ⩽ а < n. Выход. Символ Якоби (д). .1 Положить д - .1 21
5. При а = 0 результат: 0.

6. При a = 1 результат: д. .4 Представить a ввиде a = 2* a1, где число a, нечетное.

7. При четном k положить s - 1, при нечетном k положить s - 1, если п = ‡1 (mod 8); положить s < -1, если п = ‡3 (mod 8).

8. При a, = 1результат: 9•S. .7Еслиn=3(mod4)иa1=3(mod4),ToS+ .s-

9. Положитьa- n(moda1),n- a1,9- g•sивернутьсянашаг.2 3. Алгоритм, реализующий тест Соловэя-Штрассена. Вход. Нечетное целое число п ⩾ 5. Выход. «Число п, вероятно, простое» или «Число п составное».

10. Выбрать случайное целое число a, 2 ⩽ a < n - 2.

11. Вычислить r - a 2 (mod n). .3 При г ‡ 1 иг ‡ n - 1 результат: «Число п составное».

12. Вычислить символ Якоби ,s - (д).

13. При r s=(mod n) результат: «Число п составное». В противном случае результат: «Число п, вероятно, простое». На сегодняшний день для проверки чисел на простоту чаще всего используется тест Миллера-Рабина, основанный на следующем наблюдении. Пусть число п нечетное и n - 1= 2ог, где г - нечетное. Если и простое, то для любого a ⩾ 2, взаимно простого сп, выполняется условие a₽-1 = 1(mod p).

14. Алгоритм, реализующий тест Миллера-Рабина. Вход. Нечетное целое число п ⩾ 5. Выход. «Число п, вероятно, простое» или «Число п составное». .1 Представить n - 1 в виде n - 1= 25г, где число г нечетное. 2. Выбрать случайное целое число a, 2 ⩽ a < n - 2. 2 ВСКО

Саратовский государственный университет Саратовскийгосударственныйуниверситет ми Чернышевского .3 Вычислить y - a” (mod n). 4. При y ‡ 1 и y ‡ n - 1 выполнить следующие действия. 4.1. Положить j - 1. 4.2. Если j ⩽ s - 1 иу ‡ n - 1, то 4.2.1. Положить y + y7 (mod n). 4.2.2. При y = 1 результат: «Число п составное». 4.2.3. Положить j - j + 1. 4.3.При y ‡ n - 1 результат: «Число п составное». .5 Результат: «Число п, вероятно, простое».

4 Выполнение лабораторной работы

1. Реализуется функция алгоритма теста Ферма. (рис. 4.1).

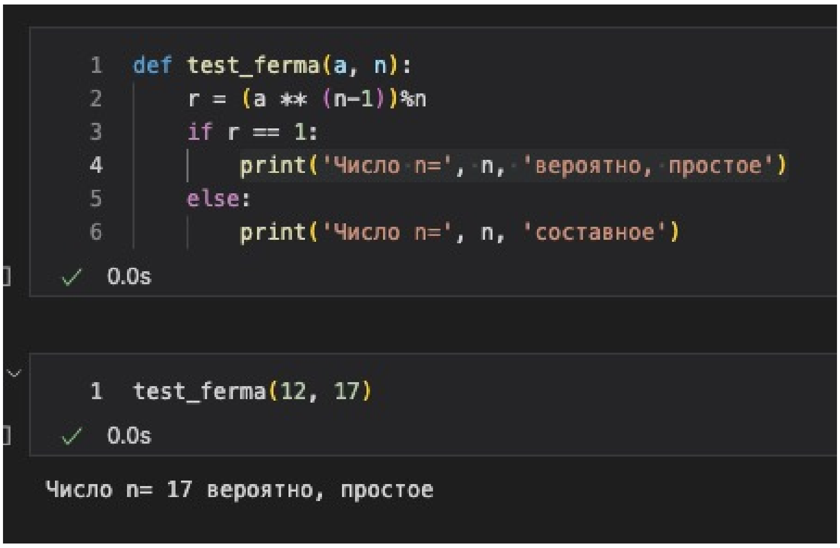

Рис. 4.1: Программная реализация алгоритма теста Ферма.

2. Реализуется функция алгоритма вычисления символа Якоби. (рис. 4.2).

```python

def Jakobi_symbol(a, n):
    g = 1
    while True:
        if a == 0:
            res = 0
            break
        if a == 1:
            res = g
            break
        else:
            k = primefactors(a)[0]
            a1 = primefactors(a)[1]
            if k % 2 == 0:
                s = 1
            if k % 2 != 0:
                if ((n - 1) % 8) == 0 or ((n + 1) % 8) == 0:
                    s = 1
                if ((n - 3) % 8) == 0 or ((n + 3) % 8) == 0:
                    s = -1
        if a1 == 1:
            res = g * s
            break

        if ((n - 3) % 4) == 0 or ((a1 - 3) % 4) == 0:
            s = -s
        a = n % a1
        n = a1
        g = g * s
    return -res
```

✓ 0.0s

Рис. 4.2: Алгоритм вычисления символа Якоби

3. Программная реализация алгоритма Соловэй-Штрассена. (рис. 4.3).

```python
def solovey_strassen(a, n):
    r = (a ** ((n - 1) / 2)) % n
    if r != 1 and r != n - 1:
        print('Число n=', n, 'составное')
    s = Jakobi_symbol(a, n)
    if (r - s) % n != 0:
        print('Число n=', n, 'составное')
    else:
        print('Число n=', n, 'вероятно, простое')

```

```
[6]  ✓ 0.0s
```

```python
solovey_strassen(12, 17)
```

```
[7]  ✓ 0.0s
Число n= 17 вероятно, простое
```

Рис. 4.3: Программная реализация алгоритма Соловэй-Штрассена

4. Программная реализация алгоритма Миллера-Рабина. (рис. 4.4).

```python
def miller_rabin(a, n):
    s = primefactors(n-1)[0]
    r = primefactors(n-1)[1]
    y = (a ** r) % n

    if y != 1 and y != n-1:
        j = 1
        while j <= s-1 and y != n-1:
            y = (y ** 2) % n
            if y == 1:
                return 'Число n=', n, 'составное'
            j += 1
        if y != n - 1:
            return 'Число n=', n, 'составное'
    return 'Число n=', n, 'вероятно, простое'
```

0.0s

```python
miller_rabin(12, 17)
```

0.0s

```
('Число n=', 17, 'вероятно, простое')
```

Рис. 4.4: Программная реализация алгоритма Миллера-Рабина.

5 Выводы

В ходе работы были реализованы алгоритмы проверки чисел на простоту.