Лабораторная работа 5

Вероятностные алгоритмы проверки чисел на простоту

Пологов Владислав Александрович

2022 Москва

RUDN University, Moscow, Russian Federation

Цель работы

Цель работы

Реализовать алгоритмы проверки числа на простоту:

1. Алгоритм, реализующий тест Ферма;
2. Алгоритм вычисления символа Якоби;
3. Алгоритм, реализующий тест Соловэя-Штрассена;
4. Алгоритм, реализующий тест Миллера-Рабина.

Описание реализации

Описание реализации

Для реализации алгоритмов использовались средства языка Python.

Реализация

Алгоритм, реализующий тест Ферма

На вход мы подаём нечётное целое число n >= 5. На выходе получаем результат работы алгоритма и суждение о том, является ли число вероятно простым или составным. Алгоиртм, реализующий тест Ферма и его реализация на Python приведёны на рисунке 1. (рис. -fig. 1)

Алгоритм, реализующий тест Ферма

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

```
def test_ferm(n):
    flag = True
    for i in range(10):
        a = randint(2, n - 2)
        r = (a**(n-1)) % n
        if r != 1:
            flag = False
    if flag == True:
        return('Число n, вероятно, простое')
    else:
```

Алгоритм вычисления символа Якоби

Для реализации алгоритма вычисления символа Якоби использовалась дополнительная переменная g.Символ Якоби практически никогда не вычисляют по определению. Чаще всего для вычисления используют свойства символа Якоби, главным образом — квадратичный закон взаимности. Ключевое используемое при вычислении свойство символа Якоби — квадратичный закон взаимности. Благодаря ему алгоритм похож на алгоритм Евклида нахождения наибольшего общего делителя двух чисел, в котором тоже аргументы на каждом шаге меняются местами. Аналогично алгоритму Евклида, при перестановке аргументов больший заменяется на остаток от деления на меньший. Это возможно благодаря периодичности символа Якоби. Однако, поскольку символ Якоби определён только при условии нечётности второго аргумента, то до перестановки

Алгоритм вычисления символа Якоби

1 *(проверка взаимной простоты)*. Если НОД (a, b)≠1, выход из алгоритма с ответом **0**.

2 *(инициализация)*. r:=1

3 *(переход к положительным числам)*.
Если a<0 **то**
 a:=-a
 Если b mod 4 = 3 **то** r:=-r
Конец если

4 *(избавление от чётности)*. t:=0
Цикл ПОКА a – чётное
 t:=t+1
 a:=a/2
Конец цикла
Если t – нечётное, **то**
 Если b mod 8 = 3 или 5, **то** r:=-r.
Конец если

5 *(квадратичный закон взаимности)*. **Если** a mod 4 = b mod 4 = 3, **то** r:=-r.
c:=a; a:=b mod c; b:=c.

6 *(выход из алгоритма?)*. Если a≠0, то идти на шаг 4, иначе выйти из алгоритма с ответом **r**.

```python
def jacobi_symbol(a, b):
    even = lambda x: x%2==0  # lambda функция для проверки числа на четность
    if math.gcd(a,b)!=1: return 0  # Функция math.gcd() возвращает наибольший общий

    r = 1
    if a < 0:
        a = -a
        if b % 4 == 3:
            r = -r

    while True:
        t = 0
        while even(a):
            t += 1
            a //= 2

        if not even(t):
            if b%8 in (3,5):
                r = -r

        if a%4 == b%4 == 3:
            r =- r

        c = a
        a = b % c
        b = c

        if a==0: return r
```

Алгоритм, реализующий тест Соловэя-Штрассена

Для его вычисления понадобится вызывать функцию нахождения символа Якоби. На вход мы подаём нечётное целое число n >= 5. На выходе получаем результат работы алгоритма и суждение о том, является ли число вероятно простым или составным. Алгоритм, реализующий тест Соловэя-Штрассена и его реализация на Python представлены на рисунке 3. (рис. -fig. 3)

Алгоритм, реализующий тест Соловэя-Штрассена

Вход. Нечетное целое число $n \geq 5$.

Выход. «Число n, вероятно, простое» или «Число n составное».

1. Выбрать случайное целое число a, $2 \leq a < n - 2$.
2. Вычислить $r \leftarrow a^{\frac{n-1}{2}} (mod\ n)$.
3. При $r \neq 1$ и $r \neq n - 1$ результат: «Число n составное».
4. Вычислить *символ Якоби* $s \leftarrow \left(\frac{a}{n}\right)$.
5. При $r \equiv s\ (mod\ n)$ результат: «Число n составное». В противном случае результат: «Число n, вероятно, простое».


```
def sol_shtassen(n):
    flag = True
    for i in range(10):
        a = randint(2, n - 1)
        r = (a**((n-1)/2)) % 2
        if r != 1 and r != n - 1:
            flag = False
        jac = jacobi_symbol(a, n)
        if r == jac % n:
            flag = False
        else:
            flag = True
    if flag == True:
        return('Число n, вероятно, простое')
    else:
        return('Число n  составное')
```

Алгоритм, реализующий тест Миллера-Рабина

На вход мы подаём нечётное целое число n >= 5. На выходе получаем результат работы алгоритма и суждение о том, является ли число вероятно простым или составным.

Алгоритм, реализующий тест Миллера-Рабина представлен на рисунке 4. (рис. -fig. 4)

Код алгоритма, реализующего тест Миллера-Рабина, представлен на рисунке 5. (рис. -fig. 5)

Алгоритм, реализующий тест Миллера-Рабина

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 число r нечетное.
2. Выбрать случайное целое число a, $2 \leq a < n-2$.
3. Вычислить $y \leftarrow a^r (mod\ n)$.
4. При $y \neq 1$ и $y \neq n-1$ выполнить следующие действия.

   4.1. Положить $j \leftarrow 1$.

   4.2. Если $j \leq s-1$ и $y \neq n-1$, то

      4.2.1. Положить $y \leftarrow y^2\ (mod\ n)$.

      4.2.2. При $y = 1$ результат: «Число n составное».

      4.2.3. Положить $j \leftarrow j+1$.

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

Алгоритм, реализующий тест Миллера-Рабина

```python
def miller_robin(n):
    flag = True
    even = lambda x: x%2==0
    r = n - 1
    s = 0
    while even(r):
        s += 1
        r //= 2

    for i in range(10):
        a = randint(2, n - 2)

        y = (a ** r) % n
        if y != 1 and y != n - 1:
            j = 1
            if j <= s - 1 and y != n - 1:
                y = (y ** 2) % n
                if y == 1:
                    flag = False
                    break
                j += 1
            if y != n - 1:
                flag = False
                break
        flag = True
    if flag == True:
        return('Число n, вероятно, простое')
    else:
        return('Число n  составное')
```

Вывод

Вывод

- Реализовали следующие алгоритмы для проверки чисел на простоту:
    1. Алгоритм, реализующий тест Ферма;
    2. Алгоритм вычисления символа Якоби;
    3. Алгоритм, реализующий тест Соловэя-Штрассена;
    4. Алгоритм, реализующий тест Миллера-Рабина.