Лабораторная работа 5

Вероятностные алгоритмы проверки чисел на простоту

Греков Максим Сергеевич

2022 Москва

RUDN University, Moscow, Russian Federation

Цель работы

Цель работы

- Ознакомиться с определением простых чисел
- Изучить свойства простых чисел и подходы к их обнаружению
- Реализовать вероятностные алгоритмы проверки чисел на простоту

Описание

Простое число

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

Целое число p называется простым, если оно не является делителем единицы и не имеет других делителей, кроме тривиальных.

В противном случае число p называется составным.

Например, числа ± 2, ± 3, ± 5, ± 7,± 11,± 13,± 17,± 19,± 23,± 29 являются простыми.

Проверка на простоту

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.

Типы алгоритмов

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

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 случайное 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 n действительно является составным.

Количество тестов

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.

Рассмотрим такие вероятностные алгоритмы как тест Ферма (рис. 1), Соловэя-Штрассена (рис. 3) (а также алгоритм вычисления символа Якоби (рис. 2)), Миллера-Рабина (рис. 4), и выполним с их помощью проверки (рис. 5).

Алгоритмы

Тест Ферма

```python
def fermat(n: int) -> bool:
    a = randint(2, n-2)
    r = (a**(n-1)) % n
    return r == 1
```

Figure 1: Тест Ферма

Вычисление символа Якоби

```python
def jacobi(a: int, b: int):
    def even(x): return x % 2 == 0
    if math.gcd(a, b) != 1:
        return 0

    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
```

Figure 2: Вычисление символа Якоби

Тест Соловэя-Штрассена

```python
def solovay_strassen(n: int) -> bool:
    a = randint(2, n-1)
    if math.gcd(a, n) > 1:
        return False
    if (a**((n-1)//2) - jacobi(a, n)) % n != 0:
        return False
    return True
```

Figure 3: Тест Соловэя-Штрассена

Тест Миллера-Рабина

```python
def miller_rabin(n: int) -> bool:
    def even(x): return x % 2 == 0
    r = n-1
    s = 0
    while even(r):
        s += 1
        r //= 2

    a = randint(2, n-3)

    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 False
            j += 1
        if y != n-1:
            return False
    return True
```

Figure 4: Тест Миллера-Рабина

Результаты

Результаты

```python
def simplicity(method, n: int, steps=10) -> bool:
    for i in range(steps):
        if not method(n):
            return False
    return True


def main():
    methods = [fermat, solovay_strassen, miller_rabin]

    for method in methods:
        print(method.__name__)
        for i in range(5, 150):
            if simplicity(method, i):
                print(i, end=' ')
        print('\n')
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ JUPYTER

```
fermat
5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149

solovay_strassen
5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149

miller_rabin
5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149
```

Figure 5: Результаты

Выводы

Выводы

- Ознакомились с определением простых чисел
- Изучили свойства простых чисел и подходы к их обнаружению
- Реализовали вероятностные алгоритмы проверки чисел на простоту