Отчёт по лабораторной работе №6

Ван Яо

Содержание

Список таблиц

Список иллюстраций

Цель работы

1. Изучить теоретические основы алгоритмов разложения чисел на множители
2. Реализовать программно два метода
   - ρ - метод Полларда
   - метод квадратов Ферма
3. Провести тестирование алгоритмов на примерах и проанализировать их эффективность

Ход лабораторной работы

Теоретические основы

1. ρ - метод Полларда

Метод основан на поиске цикла в последовательности, гегерируемой функцией $f(x)$,например $f(x) = x^2 + c \mod n$.Если разность двух элементов последовательности имеет нетривиальный общий делитель с n,то он и является искомым делителем

Алгоритм:

1. Выбрать начальное значение c и функцию f
2. Использовать два указателя a и b, двигая их с разной скоростью
3. На каждой итерации вычислить $d = \text{НОД}(|a - b|, n)$
4. Если $1 < d < n$, тоd - нетривиальный делитель

2. Метод квадратов Ферма

Метод основан на представлении числа n в виде разности квадратов:

$$n = s^2 - t^2 = (s + t)(s - t)$$

Алгоритм:

1. Найти наименьшее $s \geq \lceil \sqrt{n} \rceil$
2. Проверить, является ли $s^2 - n$ полным квадратои
3. Если да, то $p = s - t$,$q = s + t$

Практическая реализация

Реализация ρ -метода Полларда

```
def pollards_rho(n,c=1,f=None,max_iterations=1000000):
    if n%2==0:
        return 2
    if f is None:
        f = lambda x:(x*x+1)%n

    a=c
    b=c

    for i in range(max_iterations):
        a=f(a)
        b=f(f(b))
        d=gcd(abs(a-b),n)

        if 1<d<n:
            return d
        if d==n:
            return None

    return None
```

Реализация метода квадратов Ферма

```
def fermat_factorizaton(n,max_iterations=1000000):
    if n%2==0:
        return 2

    s=math.isqrt(n)
    if s*s==n:
        return s

    s+=1
    for i in range(max_iterations):
        t2=s*s-n
        t=math.isqrt(t2)

        if t*t==t2:
            p=s-t
            if p>1 and n%p==0:
                return p
            q=s+t
            if q<n and n%q==0:
                return q
            return p

        s+=1

        if s>(n+1)/2:
            break
    return None
```

Функциональное тестирование

ρ -метод Полларда

```
n=1359331
n2=10403
```

```
factor=pollards_rho(n,c=1,f=lambda x:(x*x+5)%n)
factor2=pollards_rho(n2,c=1,f=lambda x:(x*x+5)%n2)
```

```
if factor:
    print(f"{factor}")
    print(f"{n//factor}")
```

```
1181
1151
```

```
if factor2:
    print(f"{factor2}")
    print(f"{n2//factor2}")
```

```
103
101
```

Метод квадратов Ферма

```
factor=fermat_factorization(n)
```

```
factor2=fermat_factorization(n2)
```

```
if factor:
    print(f"{factor}")
    print(f"{n//factor}")
```

```
1151
1181
```

```
if factor2:
    print(f"{factor2}")
    print(f"{n2//factor2}")
```

```
101
103
```

Сравнительный анализ

Метод	Преимущества	Недостатки
ρ-метод Полларда	Быстрый на малых делителях	Может не сойтись для некоторых чисел
Метод квадратов Ферма	Простота реализации	Медленный, если делители далеки от $\sqrt{n}$

Выводы

1. **Теоретические знания:**

   Изучены два классических алгоритма факторизации: ρρ-метод Полларда и метод квадратов Ферма, их математические основы и области применения.

2. **Практические навыки:**

   Успешно реализованы оба алгоритма на Python. Проведено тестирование на составных числах, показавшее корректность работы методов.

3. **Аналитические способности:**

   Сравнительный анализ показал, что ρρ-метод Полларда эффективен для чисел с малыми делителями, а метод Ферма удобен для чисел, близких к квадрату целого числа.

Литература

1. Pollard J.M. A Monte Carlo method for factorization. BIT Numerical Mathematics, 1975.
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/lab6