Лабораторная Работа No 6

Математические Основы Защиты Информации и Информационной Безопасности

Хосе Фернандо Леон Атупанья | НФИмд-01-24

Содержание

1. Цель работы

Ознакомиться с алгоритмом разложения чисел на множители. И написать код, соответствующий этому процессу (лабораторная работа 6).

2. Выполнение лабораторной работы

Алгоритм, реализующий p-метод Полладла

В этой отчете следующий код реализует p-метод Полларда для целочисленной факторизации. Этот алгоритм определяет нетривиальный множитель заданного целого числа n, используя псевдослучайную функцию f(x) со сжимающими свойствами. Ниже приведена подробная реализация в Julia с последующим объяснением.

```
using Random
using Printf
```

Функция GCD:

Вспомогательная функция gcd(a, b) определена для вычисления наибольшего общего делителя двух целых чисел с использованием евклидова алгоритма.

```
# Function to compute the GCD (Greatest Common Divisor)
function gcd(a, b)
    while b != 0
        (a, b) = (b, a % b)
    end
    return abs(a)
end
```

Функция p-метода Полларда:

Основная функция pollards_p_method(n, c, f) принимает в качестве входных данных: n: число для разложения на множители. c: Начальное значение для алгоритма f: Псевдослучайная функция сжатия.

```
# Function to implement Pollard's p-method
function pollards_p_method(n::Int, c::Int, f::Function)
    # Step 1: Initialize a and b
    a = c
    b = c
```

Затем введенная строка преобразуется в целое число с помощью синтаксического анализа (Int, n) и сохраняется в num1. Мы вызываем нашу функцию с аргументом num1, чтобы получить результат.

```
    while true
        # Step 2: Update a and b using the function f
        a = f(a) % n
        b = f(f(b) % n) % n
```

Алгоритм итеративно вычисляет обновления для переменных a и b, используя функцию f(x). Переменная b обновляется дважды за итерацию, чтобы обеспечить необходимое расхождение между a и b.

Итеративный цикл:

На каждой итерации вычисляется наибольший общий делитель (НОД) Ia−bI и n: Если 1< d < n, алгоритм возвращает d как нетривиальный множитель n. Если d=n, это означает, что коэффициент не найден, и алгоритм завершает работу. Если d=1, процесс продолжается.

```
        # Step 3: Compute d = GCD(a - b, n)
        d = gcd(abs(a - b), n)

        # Step 4: Check conditions for termination
        if d > 1 && d < n
            return d # Non-trivial divisor found
        elseif d == n
            return "No divisor found"
        end
        # If d == 1, continue the loop
    end
end
```

Код протестирован с помощью: n=1359331, c=1 и f(x)=x ^ 2 + 5. Алгоритм успешно идентифицирует число 1181 как нетривиальный делитель числа 1359331.

```
# Example parameters
n = 1359331
c = 1
f(x) = (x^2 + 5)
```

```julia
# Run the algorithm
result = pollards_p_method(n, c, f)

# Print the result
if typeof(result) == Int
    println("A non-trivial divisor of $n is $result.")
else
    println(result)
end
```

После запуска кода с параметрами примера будет получен следующий результат:

OUTPUT:

```
A non-trivial divisor of 1359331 is 1181.
```

3. Выводы

В этом упражнении p-метод Полларда был реализован в Julia для разложения целых чисел на множители. Алгоритм успешно продемонстрировал свою способность находить нетривиальные делители составных чисел, используя псевдослучайные итеративные обновления и свойства наибольшего общего делителя. Используя пример с n=1359331, алгоритм определил 1181 как нетривиальный фактор, подтверждающий его эффективность.

Реализация демонстрирует эффективность p-метода Полларда в сценариях, где традиционные методы факторизации могут быть дорогостоящими с точки зрения вычислений. Использование в методе простых арифметических операций и модульных сокращений делает его интуитивно понятным и вычислительно эффективным для целых чисел среднего размера.

Это упражнение подчеркивает практическую полезность алгоритмов теории чисел в вычислительной математике, криптографии и решении задач. Кроме того, оно демонстрирует простоту реализации передовых математических методов в Julia, подчеркивая пригодность языка для решения математических и алгоритмических задач.