Отчёт по лабораторной работе №6

Разложение чисел на множители

Шевляков Илья Николаевич НФИмд-01-21

Содержание

# Цель работы

Изучение задачи разложения на множители, изучение p-алгоритма Поллрада.

# Теоретические сведения

Разложение на множители — предмет непрерывного исследования в прошлом; и такие же исследования, вероятно, продолжатся в будущем. Разложение на множители играет очень важную роль в безопасности некоторых криптосистем с открытым ключом.

Согласно Основной теореме арифметики любое положительное целое число больше единицы может быть уникально записано в следующей главной форме разложения на множители, где — простые числа и — положительные целые числа.

Поиск эффективных алгоритмов для разложения на множители больших составных чисел ведется давно. К сожалению, совершенный алгоритм для этого пока не найден. Хотя есть несколько алгоритмов, которые могут разложить число на множители, ни один не способен провести разложение достаточно больших чисел в разумное время. Позже мы увидим, что это хорошо для криптографии, потому что современные криптографические системы полагаются на этот факт. В этой секции мы даем несколько простых алгоритмов, которые проводят разложение составного числа. Цель состоит в том, чтобы сделать процесс разложения на множители менее трудоёмким.

В 1974 г. Джон Поллард разработал метод, который находит разложение числа на простые числа. Метод основан на условии, что не имеет сомножителя, большего, чем заранее определенное значение , называемое границей. Алгоритм Полларда показывает, что в этом случае

Сложность. Заметим, что этот метод требует сделать операций возведения в степень . Есть быстрый алгоритм возведения в степень, который выполняет это за операций. Метод также использует вычисления НОД, который требует операций. Мы можем сказать, что сложность — так или иначе больше, чем или , где — число битов в . Другая проблема – этот алгоритм может заканчиваться сигналом об ошибке. Вероятность успеха очень мала, если имеет значение, не очень близкое к величине .

## p-алгоритм Поллрада

* Вход. Число , начальное значение , функция , обладающая сжимающими свойствами.
* Выход. Нетривиальный делитель числа .

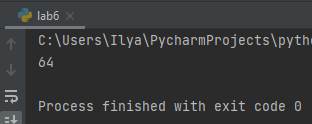
1. Положить
2. Вычислить
3. Найти
4. Если , то положить и результат: . При результат: ДЕЛИТЕЛЬ НЕ НАЙДЕН. При вернуться на шаг 2.

# Выполнение работы

## Реализация алгоритма на языке Python

from math import gcd  
  
ag = 1  
bg = 1  
  
  
def func(x, n):  
 return (x \* x + 5) % n  
  
  
def func\_2(n, a, b, d):  
 a = func(a, n) % n  
 b = func(func(b, n), n) % n  
 d = gcd(a - b, n)  
 if 1 < d < n:  
 p = d  
 print(p)  
 exit()  
 if d == n:  
 print('Not found')  
 if d == 1:  
 global ag  
 ag = b  
 func\_2(n, a, b, d)  
  
  
def lab6():  
 n = 121  
 c = 1  
 a = c  
 b = c  
 a = func(a, n) % n  
 b = func(a, n) % n  
 d = gcd(a - b, n)  
 if 1 < d < n:  
 p = d  
 print(p)  
 exit()  
 if d == n:  
 pass  
 if d == 1:  
 func\_2(n, a, b, d)  
  
  
lab6()

## Контрольный пример



Работа алгоритма

# Выводы

Изучили задачу разложения на множители и p-алгоритм Поллрада.

# Список литературы

1. [Алгоритмы тестирования на простоту и факторизации](https://intuit.ru/studies/courses/13837/1234/lecture/31191)
2. [P-метод Полларда](https://ru.wikipedia.org/wiki/P%E2%88%921-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4_%D0%9F%D0%BE%D0%BB%D0%BB%D0%B0%D1%80%D0%B4%D0%B0#:~:text=%2D%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%20%D0%9F%D0%BE%D0%BB%D0%BB%D0%B0%D1%80%D0%B4%D0%B0%20%E2%80%94%20%D0%BE%D0%B4%D0%B8%D0%BD%20%D0%B8%D0%B7%20%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%B2%20%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8%20%D1%86%D0%B5%D0%BB%D1%8B%D1%85%20%D1%87%D0%B8%D1%81%D0%B5%D0%BB.&text=%D0%92%20%D1%81%D0%BE%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D1%85%20%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%D1%85%20%D1%81%D1%82%D0%B0%D1%80%D0%B0%D1%8E%D1%82%D1%81%D1%8F%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D1%8C,%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2%20%D0%B8%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%20%D0%B2%20%D1%86%D0%B5%D0%BB%D0%BE%D0%BC.)