Федеральное государственное автономное образовательное учреждение высшего профессионального образования Национальный исследовательский университет «Высшая школа экономики»

Московский институт электроники и математики им. А. Н. Тихонова

Образовательная программа «Компьютерная безопасность»

**ОТЧЕТ**

по дисциплине «Теоретико-числовые методы в криптографии»

Программная реализация метода Брента для факторизации чисел

Выполнил: студент группы СКБ-171 Сарахан Владислав Сергеевич

Проверил: Доцент

Нестеренко А. Ю.

МОСКВА – 2021

**ВСТУПЛЕНИЕ**

Алгоритм Брента – алгоритм факторизации чисел, работающий для определения небольших делителей. Разработан Ричардом Брентом в 1980 году. Является улучшенной версией Ро-алгоритма Полларда, так как позволяет уменьшить количество выполняемых операций примерно на 25 % [1]. Время работы данного алгоритма зависит от *p*, где *p* – это делитель факторизуемого числа.

**ЗАДАЧА**

Реализовать алгоритм Брента в sage, сравнить результаты и время поиска делителя для разных случаев факторизуемых чисел.

**АЛГОРИТМ БРЕНТА**

Данный алгоритм является вероятностным алгоритмом факторизации чисел. Для алгоритма Брента число шагов поиска делителя является случайной величиной и зависит от генератора псевдослучайных чисел в алгоритме. При этом если алгоритм не сумел определить делители факторизуемого числа, то нельзя точно сказать, является ли это число простым или составным [2].

Пусть *m* – это число, которое мы хотим факторизовать. Для данного алгоритма сначала зафиксируем некоторый многочлен (deg > 1). Рассмотрим конечное отображение кольца вычетов для функции : .

Для произвольного элемента данного кольца рассмотрим последовательность элементов, задаваемых соотношением:

Так как в данном кольце конечное число элементов, то найдутся такие элементы, что выполняется равенство:

(1)

Величину *t* будем называть длиной цикла, а минимальное число *b*, при котором этот цикл начинается, будем называть длиной подхода [3].

Рассмотрим простой делитель числа *m* (предположим, что он существует) – число *p*.Тогда из сравнения (1) следует сравнение:

То есть данный многочлен порождает последовательность в кольце вычетов которая зациклится и будет выполнено условие:

Тем самым получаем, что НОД(*m*,)=*p* (при условии, что )

Для поиска таких индексов Брент предложил использовать свой метод для поиска циклов в последовательностях. Брент модернизировал метод Полларда-Флойда [4]. Брент предложил искать совпадения в последовательности с помощью следующей проверки:

При этом для всех индексов n выполняется:

Простой делитель p определяется условием:

*1<*НОД(*m*,)*<m*

Алгоритм Брента:

1.Берем многочлен

2.Берем случайный вычет в кольце . Присвоить значения *z=0, k=0,n=0,t=1*

3.Вычислить . Присвоить *n=n+1*

4.Если *n=t*, то присвоить значения *z=x, k=k+1, t=2t*

5.Присвоить *p=НОД(m,z-x)*

6.Если *1<p<m*, то *p* – делитель числа m, алгоритм закончен

7.Если *k>c* – то неудача, алгоритм завершен, делителей нет. Иначе вернуться к пункту 3

**ПРОГРАММНАЯ РЕАЛИЗАЦИЯ**

Код программы прикреплен в отдельном файле и загружен по ссылке https://github.com/vssarahan/brent\_alg. В программе я реализовал метод Брента с помощью функции *fact\_brent.*

Реализация полностью совпадает с реализацией алгоритма сверху при использовании некоторых функций Sage: *gcd(z-x,m)* – нахождение НОД двух чисел

Функция fact\_brent принимает на вход число *m* – факторизуемое число - и количество шагов алгоритма *c* и выводит 1, если алгоритм считает, что данное число простое, иначе выводит один из его делителей. При этом в ходе программы я генерирую два случайных числа для алгоритма – число *x* в кольце вычета и число *d* для многочлена в этом же кольце с помощью *randint*.

Так как алгоритм Брента выдает нам просто один делитель, его тоже необходимо проверить на наличие других делителей, для чего и нужна функция *factorization\_by\_brent*, позволяющая разложить число до конца на простые делители. В ней я использовал функцию sage *is\_prime* для определения простоты числа.

**РЕЗУЛЬТАТЫ**

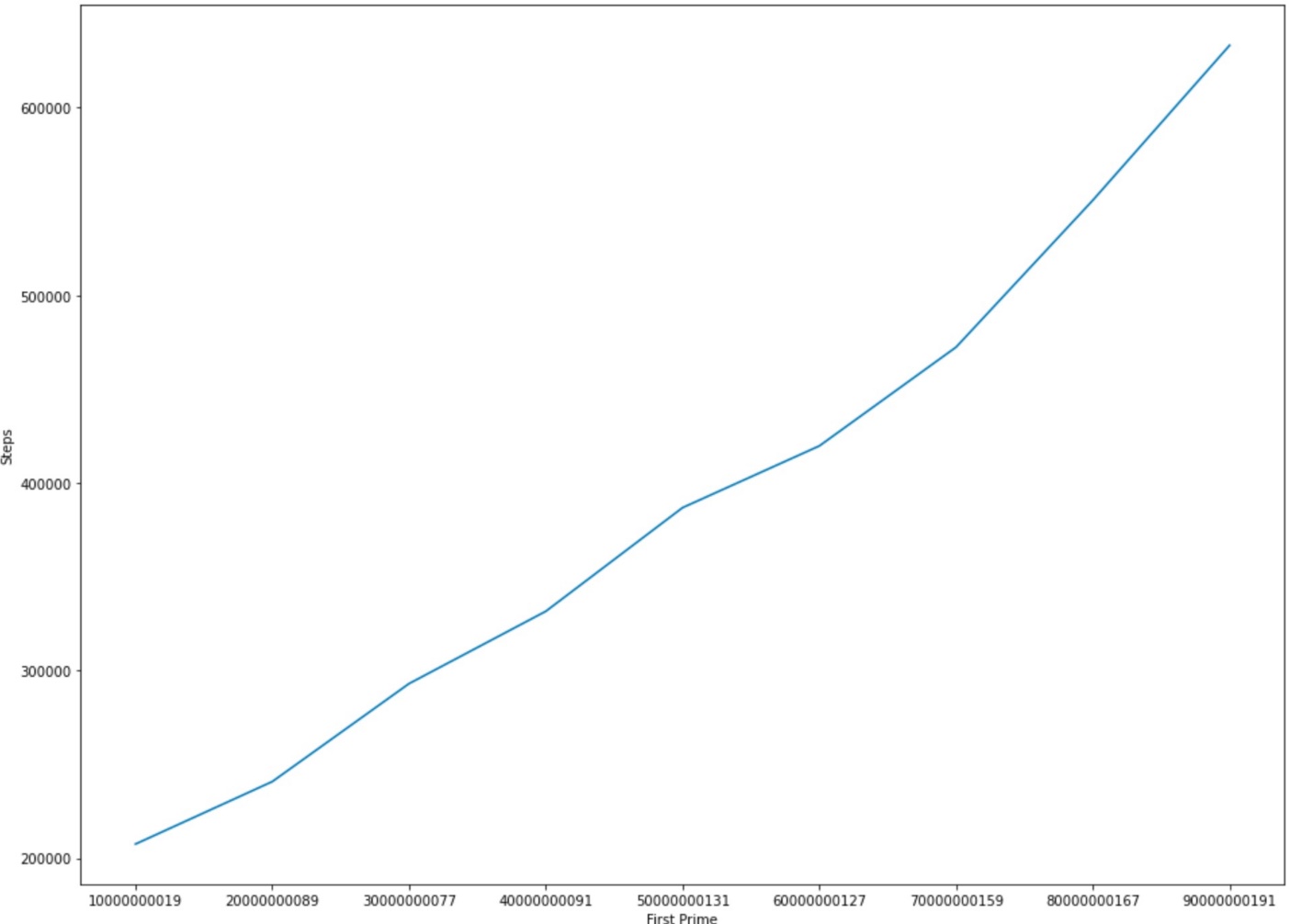
Пример работы алгоритма

Изображение выглядит как текст

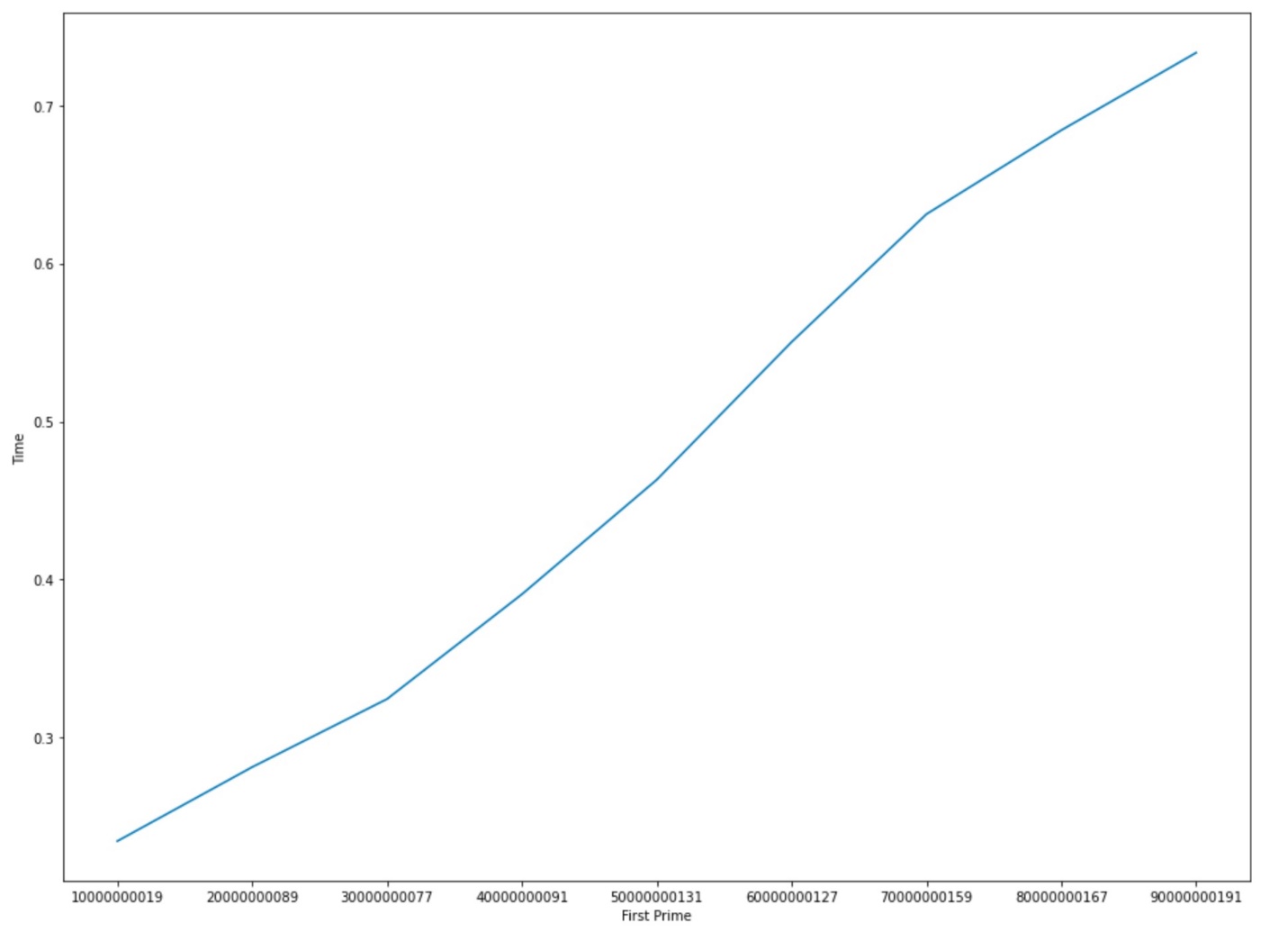
Автоматически созданное описание

|  |  |  |
| --- | --- | --- |
| Первый делитель | Время поиска | Количество шагов |
| **10000000019** | 0.234375 | 207523 |
| **20000000089** | 0.28125 | 240816 |
| **30000000077** | 0.324332 | 293041 |
| **40000000091** | 0.390625 | 331595 |
| **50000000131** | 0.463125 | 386830 |
| **60000000127** | 0.5503125 | 419741 |
| **70000000159** | 0.63125 | 472406 |
| **80000000167** | 0.684375 | 550718 |
| **90000000191** | 0.7334375 | 633290 |

|  |  |  |
| --- | --- | --- |
| Первый делитель | Время поиска | Количество шагов |
| **100000007** | 0.015625 | 15577 |
| **1000000007** | 0.0625 | 53110 |
| **10000000019** | 0.234375 | 154497 |
| **100000000003** | 0.73203125 | 425365 |
| **1000000000039** | 2.334375 | 834738 |
| **10000000000037** | 7.33125 | 5058057 |
| **100000000000031** | 24.1426125 | 13422637 |
| **1000000000000037** | 76.338 | 19482185 |
| **10000000000000061** | 243.765625 | 103024179 |

**Зависимость количества шагов от первого полученного делителя числа**  


**Зависимость времени работы алгоритма от первого полученного делителя числа**



**ВЫВОД**

В данной работе я реализовал метод Брента для факторизации чисел и получил и сравнил зависимости времени работы алгоритма и количества шагов от первого полученного делителя в алгоритме

**ИСТОЧНИКИ**

1.**Reisel**, 2012, Selected Areas in Cryptography. Prime Numbers and Computer Methods for Factorization. 2nd ed.

2**. А.Ю. Нестеренко** ТЕОРЕТИКО-ЧИСЛОВЫЕ МЕТОДЫ В КРИПТОГРАФИИ Москва 2012

3. **А.Ю. Нестеренко** Алгоритмы поиска длин циклов в последовательностях и их приложения

4. **Pollard**, 1975, A Monte Carlo method for factorization, с. 176.