**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ»**

**Факультет безопасности информационных технологий**

**Дисциплина:**

**“Алгоритмы и структуры данных”**

**Реферат на тему**

***“*Метод квадратичного решета разложения целых чисел на множители, метод сопряженных градиентов решения СЛАУ с разреженной матрицей*”***

**Выполнил:**

Студент гр. N3249

Шарифуллин Ильдан Айдарович



**Проверил:**

Доцент, доктор технических наук

Трифонов Петр Владимирович

Санкт-Петербург

2022г.

**Теоретическая основа алгоритма**

Алгоритм квадратичного решета был предложен в 1982 году Карлом Померансом. Около десятилетия он оставался лучшим алгоритмом факторизации натурального числа N. В настоящее время он заслуживает 2 место по своей эффективности (сразу после алгоритма SNFS 1990 года).

В данном алгоритме существует несколько методов применения. В своей работе я буду рассматривать именно метод Померанса.

Прежде, чем перейти к разбору, введу определение функции q(x).

Большая часть методов факторизации числа (и в том числе метод квадратичного решета) основаны на идее, предложенной Ферма. Она в свою очередь основана на поиске пар натуральных чисел A и B, удовлетворяющих условию:

n = A2 – B2

Алгоритм Ферма можно разбить на следующие 3 шага:

1. Вычислим целую часть квадратного корня от n, округленную вверх

m = [sqrt(n)]

1. Для каждого x = 1, 2, 3, … вычислим

**q(x)** = (m + x)2 – n

и будем делать это пока **q(x)** не получится равным полному квадрату любого числа A.

1. **q(x)** = A2. Определим B = m + x, тогда B2 – n = A2. Следовательно n = B2 – A2 = (B + A) \* (B – A). В таком случае множители числа n равны B + A и B – A.

**Цель метода** квадратичного решета: факторизация натурального числа n путем получения для пар чисел A и B (множество которых получено особым образом) соотношения A2 ≡ B2 (mod n).

**Идея Померанца** заключалась в том, что если для простого числа p ∈ FB (факторная база) найден аргумент x такой, что q(x) ≡ 0 (mod p), то на p будут делиться все элементы q(y), где y отличающиеся от x на аргумент, кратный p, т.е. y = x + k·p, k ∈ Z. Поэтому, если для данного p найден корень x уравнения q(x) ≡ 0 (mod p), то для всех y ≡ x (mod p) будет также выполнено условие того же уравнения для q(y).

Прежде, чем перейти к алгоритму введу следующее определение:

Пара целых чисел (A, B) называется гладкой (относительно факторной базы F), если:

1. Выполняется соотношение A2 ≡ B (mod n),

2. B – раскладывается в произведение элементов из F.

**Работа алгоритма Померанца вкратце**:

1. Выбирается некоторый числовой интервал [−L; L], называемый интервалом просеивания,

2. В массив целых чисел W [−L .. L] записываются значения функции q(x) для x ∈ [−L; L],

3. Для каждого числа p из факторной базы FB ищутся числа 0 ≤ x < p такие, что выполняется сравнение

q(x) ≡ 0 (mod p)

Это уравнение может иметь либо 2 решения, либо 0.

1. Для каждого решения x уравнения q(x) ≡ 0 (mod p) в цикле просматриваются числа вида xk = x + kp из интервала [−L; L], k ∈ Z, и выполняется деление элементов W[xk] на p.

Повторяем данную операцию для всех чисел из FB по итогу чего получаем некоторое количество W[xk], равных 1. Такие пары (x, q(x)) являются гладкими. По разным оценкам требуемое количество таких пар должно быть не меньше |FB| + 1 или |FB| + 2. Далее разберем этот момент подробнее. И в таком случае из коэффициентов разложения гладких чисел на элементы факторной базы можно составить систему линейных уравнений, которая будет иметь хотя бы одно необходимое решение.

Обозначим |FB| = k. Для разложения числа n, необходимо отыскать k’ гладких пар. Формируем СЛАУ размерности k × k’ с коэффициентами из поля F2 = {0, 1}. Коэффициенту 0 соответствуют четные степени разложения, единице – нечетные. Для нас принципиально важна именно четность разложения. Пара (A, B), удовлетворяющая уравнению A2 ≡ B2 (mod n), находится как

A = , B =

Теперь делители p исходного n можно найти, вычисляя НОД(n, A ± B). В некоторых случаях оба найденных делителя оказываются тривиальными (равными соответственно 1 или n), тогда нужно искать другое решение системы и новую пару (A, B).

**Разбор этапов алгоритма Померанца:**

**Как построить факторную базу:**

1. Составляем множество из простых чисел, меньших верхней границы B:

FB = {2, 3, 5, ..., pk}.

2. Фильтруем факторную базу. Оставим в FB только такие элементы p, для которых n – сравнимо с k2 по модулю p.

3. Далее для каждого p ∈ FB, нам нужно найти корни r1, r2 сравнения q(x) = 0 (mod p). Когда один корень найден, то второй легко находится с помощью теоремы Виета: r1 + r2 = 2m (mod p). Предположим, теперь, что для всех p ∈ FB r1, r2 найдены.

Для небольших p это можно сделать простой подстановкой чисел, пока решение не будет найдено. Для больших p необходимо использовать алгоритм Шенкса–Тонелли, который позволит найти корни уравнения q(x) ≡ 0 (mod p) за время O(log2p).

В завершение формирования факторной базы нахожденим решения уравнений

q(x) = 0(mod pk), для степеней k > 1, pk < B.

**Просеивание:**

1. Выбираем радиус просеивания L.
2. Формируем массив W значений q(xi) размером 2L.

Просеивание представляет собой двойной цикл в ходе которого перебираются все простые числа из факторной базы, для каждого из которых потом выполняется цикл по элементам массива W. В ходе первого цикла все числа из массива мы стараемся поделить один раз на числа из факторной базы. Таким образом на втором цикле мы уже понимаем, на какие числа мы можем пробовать делить максимальное количество раз (на те числа, на которые элемент из W разделился при первом проходе).

Иначе говоря, после просеивания по первым степеням p факторной базы, следует перейти к степеням p. Если q(x) является гладким числом, тогда после выполнения полного просеивания, соответствующее значение W[x] станет равным 1. Такие значение нам и нужны были.

**Построение множества векторов показателей:**

После просеивания мы получили небольшое (сравнивая с радиусом интервала просеивания) множество B-гладких чисел *Xsm* = {x1, x2, ..., xk}. На первичном просеивании не сохраняются делители чисел. Поэтому нам нужно выполнить то же просеивание второй раз, но теперь только по элементам *Xsm*. Разбиение на 2 этапа рассеивания хоть и требует чуть больше времени, зато позволяет нам сэкономить приличное количество памяти.

Чтобы хранить степени разложения *Xsm* заведем массив V[1..k, 0..sz], sz = |FB| + 1. Элементы V[i, 0] могут принимать два значения:

V[i, 0] = 0, если q(xi) > 1,

V[i, 0] = 1, иначе.

V[i, j] = pj степени, в которой элемент факторной базы FB pj входит в разложение значения q(xi).

**Решение системы линейных уравнений:**

Мы получили систему линейных уравнений с нулевой правой частью и коэффициентами из поля F2, число уравнений m которой строго меньше числа неизвестных k. Коэффициент равен **0**, если степень соответствующего простого числа из W была четная, или же **единице**, в ином случае.

Am×k · X = 0.

Матрица составляется из столбцов, которые являются векторами разложения гладкого числа по FB по модулю 2. Система однозначно имеет решения (как минимум k – m **нетривиальных** решений). Также это система обладает свойством сильной разреженности в нижней ее части (где значения p из факторной базы достаточно велики). Возникает проблема из-за того, что размерность системы очень велика. Поэтому для решения соответствующей системы ЛУ я буду использовать специальный метод сопряженных градиентов решения с разреженной матрицей.

**Оценка сложности метода.**

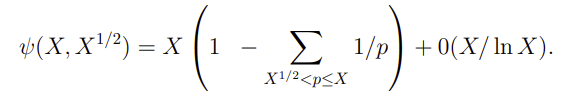
Для оценки сложности метода я буду использовать и цитировать материалы из учебного пособия Ш.Т. Ишмухаметова “ Методы факторизации натуральных чисел”

Пусть ξ – число в интервале от 0 до 1. Скажем, что оно такое, что радиус интервала просеивания можно оценить величиной L + Nξ.

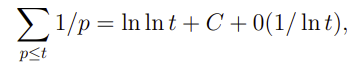
Тогда max q(x) приблизительно равно 2n1/2+ξ (достигается на границах интервала просеивания). Нам нужно ценить вероятность того, что случайное q(x) является гладким в FB. Скажем, что Ψ – искомое количество гладких чисел в множестве X. Предположим, граница B = X1/2:

Ψ = X -

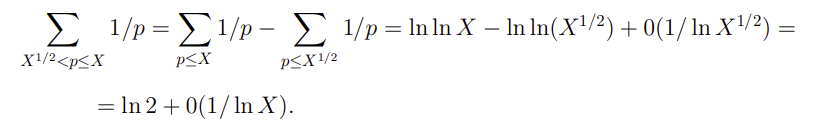
Все числа, меньшие X1/2 по определению X1/2 – гладкие. Число из второй половины интервала [1, X] не будет гладким, если оно кратно какому-нибудь простому числу p, X1/2 < p < X . Число таких кратных для каждого p равно –[X/p] (т.е. целой части от частного X/p). По теореме Чебышева, число простых чисел в интервале [1, X], обозначаемое через π(x), примерно равно X / ln(X). Отсюда



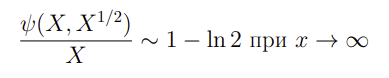
По теореме Мартенса,



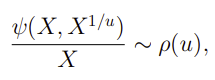
для некоторого C. Используя данную теорему:



Откуда подстановкой несложно получить



Отсюда, около 30 % чисел, меньших X, является sqrt(X) –гладкими. Для произвольной границы B = X1/u выполняется формула К.Дикмана (K. Dickman [1930]):



где ρ(u) для u > 0 функция Дикмана–де Брюина. Эта функция является непрерывным решением дифференциального уравнения uρ’(u) + ρ(u−1) = 0 при u > 1 c начальным условием ρ(u) ≡ 1 на интервале [0, 1].

Заметим, что величина X / ψ(X, X1/u) равна среднему числу попыток для нахождения одного гладкого числа. Каждая попытка может быть оценена в ln(ln(B)) элементарных операций. Нам требуется найти столько гладких чисел, сколько элементов в факторной базе, т.е. около π(B) (в действительности, факторная база содержит примерно половину от этого числа, т.к. только половина p является квадратичными вычетами по модулю n). Отсюда можно подсчитать, что общее число операций T(u) примерно равно



Мы должны минимизировать эту величину. Вычислим ее логарифм:



Производная этой функции равна 0, если u2(ln(u) + 1) = ln(X), откуда



При выборе B равным этому выражению при X = n, число элементарных операций метода квадратичного решета разложения числа n оценивается величиной



Обозначим через L(k, n) функцию



тогда оценка производительности метода квадратичного решета перепишется в вид



**Метод сопряженных градиентов**

Метод сопряженных градиентов является итерационным методом решения СЛАУ.

Пусть дана система линейных уравнений Ax = b. Тогда решение СЛАУ можно представить в виде минимизации такого функционала:

,

где операция (Ax, x) – скалярное произведение.

Так как в методе квадратичного решета мы работаем с СЛАУ в поле GF(2), у нас может возникнуть ситуация, где для ненулевого вектора x скалярное произведение (x ∙ x) = 0. Для того, чтобы минимизировать количество таких ситуаций, нам лучше работать с расширением поля GF(2m). При этом, чем больше m, тем меньше шансов столкнуться с нулевым скалярным произведением для ненулевого вектора, но вместе с тем увеличивается вычислительная сложность и расходуемая программой память. Какое бы расширение мы не брали, есть шанс столкнуться с этим, поэтому при возникновении подобной ситуации нам необходимо выбрать другое начальное приближение и начать итерационный процесс заново.

Перед началом итераций, нам также необходимо привести матрицу системы Ax = b к симметричному виду и погрузить в большое поле. Для этого мы можем домножить левую и правую части на AT\*D2 и решать уравнение ATD2Ax = ATD2b.

**Алгоритм:**

1. Выбираем начальное приближение x0.
2. r0 = b – Ax0, p0 = r0
3. На каждой итерации (j = 1, 2, …):

αj = (rj, rj) / (Apj, pj)

xj+1 = xj + αjpj

rj+1 = rj + αjApj

βj = (rj+1, rj+1) / (rj, rj)

1. Если Axj+1=b, то заканчиваем итерации

Иначе pj+1 = rj+1 + βjpj

**Сложность алгоритма:**

Посчитаем сложность одной итерации:  
Для скалярного произведения нужно n операций (\*) и n−1 операций (+). Итого — 2n−1 операций.  
Обновление вектора требует n операций (\*) и n операций (+). Всего — 2n операций.  
Для умножения матрицы на вектор необходимо выполнить n(2n−1) = 2n2−n операций.  
Итак, перемножив количество итераций на операции, выполняемые в одной итерации, получим:

На одну итерацию 3×(2n − 1) + 3×(2n) + (2n2 − n) = 2n2 + 11n−3 = O(n2) операций.  
Общая сложность алгоритма — O(mn2) , где m≤n — число итераций.

При этом число итераций может сильно больше при выборе неблагоприятных начальных приближений x0.

**Литература:**

1. Методы факторизации натуральных чисел: учебное пособие / **Ш.Т. Ишмухаметов**. – Казань: Казан. ун. 2011.– 190 с.
2. **Сорокин Александр** / Метод сопряженных градиентов (Решение СЛАУ).
3. Modern Computer Algebra Third Edition / **JOACHIM VON ZUR GATHEN** / Bonn–Aachen International Center for Information Technology (B-IT)