Факторизация методом квадратичноего решета

Карл Померанц

17 декабря 2021 г.

0.1 Аннотация

Факторизация методом квадратичноего решета в настоящее время является популярным решением задачи факторизации больших чисел без маленьких делителей. Именно этим методом была выполнена факторизация самого большого числа с 1983 года командой Национальных лабораторий Сандия, состоящей из Джеймса Девиса и Дианы Холдридж. На момент написания статьи, самое большое разложенное число $(10^71 - 1)/9$ длиной 71 десятичный знак. На его факторизацию потребовалось 9.5 часов работы компьютера Cray XMP в Лос Аламосе, Нью Мексико. В этой статье я планирую привести частичную историю алгоритма а также описать некоторые улучшения, которые могут быть предложены.

0.2 Схема Краичика

Существует класс алгоритмов факторизации, которые применяют похожую стратегию. Если необходимо факторизовать некоторое число N, то основная идея - умножать вычеты (congruences) $U \equiv V \mod N$, где $U \neq V$, и полные или частичные разложения (в зависимости от алгоритма) были получены для U и V для того, чтобы получить специальные (congruences) вида $X^2 \equiv Y^2 \mod N$. Тогда существует довольно большой шанс, что наибольший общий делитель $(X-Y, N)$, найденный при помощи алгоритма Эвклида, будет нетривиальным делителем N. Если это не так - то можно попробовать другую комбинацию (congruences). Таким образом, эти алгоритмы состоят из следующих частей:

1. Генерация CONGRUENCE $U \equiv V \mod N$
2. Определения частичного или полного разложения U и V
3. Определение подмножества разложенных (congruences) которые могут быть перемножены для получения специального (congruence) $X^2 \equiv Y^2 \mod N$.
4. Вычисление НОД $(X - Y, N)$

Например, мы хотим факторизовать $N = 91$, и стало видно, что

$$81 \equiv -10, 90 \equiv -1, 75 \equiv -16, 64 \equiv -27$$

Факторизуя эти числа полностью, получаем:

$$3^4 \equiv -2^5, 2 * 3^2 * 5 \equiv -1, 3 * 5^2 \equiv -2^4, 2^6 \equiv -3^3$$

Перемножив два последних соотношения имеем:

$$2^6 * 3 * 5^2 \equiv 2^4 * 3^3$$

Упрощаем:

$$2^2 * 5^2 \equiv 3^2$$

Это дает нам $10^2 \equiv 1^2 \mod 91$ и $7 = (27 - 1, 91)$. Или, если бы мы перемножили первые два соотношения, то получили бы:

$$2 * 3^6 \equiv 2 * 5 -> 3^6 \equiv 1$$

Откуда $27^2 \equiv 1^2 \mod 91$ и $13 = (27 - 1, 91)$.

Эта обобщенная схема для факторизации была опубликована Краичиком [1] в 1926г. Числа U, V раскладываются на простые множители, исключая квадраты. Так как большинство полученных соотношений с большой долей вероятности не будут давать разложения в шаге 2 алгоритма выше, шансы на успех повышаются, если выбрать U, V таким образом, чтобы один из них был полным квадратом, а другой имел большой множитель, который сам является квадратом. В [1], стр 26-27б Краичик объясняет, как следует это делать. Он принимает $U = x^2$, где x выбирается таким образом, чтобы $V = -N + x^2$ имел большой множитель y^2. Возможно гарантировать наличие y^2, если выбрать x равным решению квадратичноего (congruence) $x^2 \equiv N \mod y^2$. Однако, $\frac{V}{y^2}$ не должен быть маленьким и легко факторизуемым. У этого метода есть некоторые проблемы.

Краичик удачно использовал другие (congruences) $U \equiv V \mod N$, которые были предложены особой формой факторизуемого числа. Эти соотношения могут быть недоступны при попытке разложить "случайное" число N. В более поздней работе [2], соотношения $U \equiv V \mod N$ были использованы для помощи в нахождении X и Y, такх что $X^2 - Y^2 = N$. Это - старая стратегия факторизации, отсылающая еще к Ферма. Я думаю, Краичик предпочитал этот метод по двум причинам. Во-первых, меньше соотношений $U \equiv V \mod N$ с (multiplicative) информацией используется в процессе. Во-вторых, когда X, Y обнаружены с $X^2 - Y^2 = N$, можно быть уверенным в нетривиальной факторизации числа N, тогда как в случае другого метода, где шаг 4 алгоритма может дать в результате тривиальное разложение. Краичик, однако, и не представлял, что его заброшенный метод, использующий "циклы" (комбинации соотношений в шаге 3) станет базисом большинства современных алгоритмов факторизации.

0.3 Алгоритм непрерывных дробей

Вместо нахождения $U \equiv V \mod N$с одним из U,V - квадратом, а другим - имеющим большой множитель, являющийся квадратом, другой стратегией может быть выбор одного члена - квадрата, а другого - малого по модулю. Таким образом, повысится вероятность факторизации на шаге 2. В 1932 году Лемер и Поверс [3] предложили использование представление в виде непрерывной дроби $\sqrt{N}$ для генерации (congruences) $U \equiv V \mod N$в схеме Краичика. Это выполняется при помощи простой рекурсивной процедуры, которая создает пары Q_n, A_n, где

$$Q_n \equiv A_n^2 \mod N \tag{1}$$

и $|Q_n| < 2\sqrt{N}$. Старый метод Лежандра также предлагал использование представления $\sqrt{N}$ в виде непрерывной дроби, но его цель была использовать соотношения (1) для нахождения информации о квадратичноем (character) $\mod Q_n$ с простыми делителями $pofN$. Затем прямой поиск, например, перебор делителей может быть значительно ускорен благодаря тому, что многие потенциальные делители не будут иметь подходящего (character). В противоположность этому, Лемер и Пауэрс предлагали перемножать несколько соотношений формы (1), чтобы производить (congruent) квадраты.

Моррисон и Бриллхарт [**?**] были первыми, кто попробовал алгоритм с применением метода непрерывных дробей на современном компьютере. В их реализации он сделали несколько значительных улучшений и уточнений, которые пришлись бы к месту в любом алгоритме из семейства комбинаций (congruence). Во-первых, они использовали так называемую "базу факторизации", или все простые числа до какого-то числа F, чтобы определить, какие (congruences) (1) могут быть полезными. Когда (congruence) (1) было получено, число Q_n подвергалось перебору делителей простыми числами, такими что $p \leq F$. Если полная факторизация может быть получена, то (congruence) сохранялось для дальнейшего использования, иначе - отбрасывалось.

В шаге 3 алгоритма, собственно, комбинации (congruences) применяется метод Гаусса в очень большой матрице над $Z/2Z$. В частности, если база факторизации состоит из простых

чисел $p_1, \ldots, p_f$ и если

$$Q_n = (-1)^{a_0} \prod_{i=1}^{f} {p_i}^{a_i}$$

где a_i - неотрицательные целые числа, тогда получаем вектор

$$\vec{v}(n) = (a_0, a_1, \ldots, a_f) \mod 2$$

Если мы имеем достаточно векторов $\vec{v}(n)$, то методом гаусса получим линейную зависимость

$$\vec{v}(n) + \ldots + \vec{v}(n_k) = \vec{0}$$

Таким образом получаем, что $Q_{n_1} \ldots Q_{n_k}$ - это квадрат, скажем X^2. Если мы вычислим $X \mod N$ и $Y = A_{n_1} \ldots A_{n_k} \mod N$, тогда $X^2 \equiv Y^2 \mod N$ и мы готовы переходить к шагу 4.

Еще одно улучшение, названное "стратегия раннего выхода" была описана в [4]. Это улучшение расширило возможный числовой диапазон применения на обычных компьютерах на примерно 10 десятичных знаков в сравнении с 40-50 годами. (см. [5], [6]).

Дешевый процессор специального назначения, разработанный Дж. Смитом и С. Вагстаффом был сконструирован в Университете штата Джорджия чтобы реализовать алгоритм факторизации на непрерывных дробях с применением стратегии раннего выхода. Он был спроектирован, чтобы производить перебор делителей Q_n параллельно, а также с повышенной точностью, таким образом арифметические операции на длинных целых числах могут производиться с одиночной точностью. Он должен быть завершен в скором времени и мы ожидаем результатов. Скорее всего, он будет в какой-то степени уступать результатам, полученным командой университета Сандии, однако стоит учитывать, что стоимость проекта Смитта-Вегстаффа на несколько порядков ниже, чем стоимость Cray XMP.

0.4 Алгоритм Миллера-Вестерна

Номер журнала "Математика вычислений", который содержит статью Моррисона-Бриллхарта посвящен Д.Лемеру и включает в себя много интересных статей на вычислительной теории чисел. В этом выпуске есть статья, написанная Дж. Миллером [7] о факторизации, которая также использует соотношения $U \equiv V \mod N$. Он присваивает авторство идеи А.Е. Вестерну. Цель - найти (congruences) с U, V полностью факторизованными. Но вместо того, чтобы комбинировать эти (congruences), чтобы получить (congruent) квадраты, каждое (congruence) читается как линейное соотношение индексов с (respect) к какому-то примитивному корню gp, где p - простой делитель числа N. Когда найдено достаточно (congruences), есть возможность найти p при помощи созданных (congruences) вида $a^t \equiv 1 \mod N$. Если какие-то $q|t$ могут быть найдены когда $a^{t/q} \neq 1 \mod N$, тогда, возможно $(a^{t/q} - 1, N)$ есть нетривиальный делитель N. Я не вижу какого-то конкретного превосходства у этого метода в сравнении с просто комбинированием факторизованных (congruences), чтобы получить (congruent) квадраты в схеме Краичика. Я упоминаю этот алгоритм здесь чтобы показать очень простой способ выбора (congruences) $U \equiv V \mod N$. В частности, он всего лишь представляет N как $A + B$, принимая $U = A, V = -B$. Существует интересная неразрешенная проблема Ердёша, которая утверждает, что для каждого $\epsilon > 0$ существует $N_O(\epsilon)$ такое что для каждого целого $N > N_O(\epsilon)$ существует представление N как $A + B$, где никакие простое число в AB не превосходит N^ϵ. Что нам нужно - это алгоритмическое решение проблемы Ердёша, которое дало бы нам много таких пар A, B. Возможно, эта проблема (как и сама факторизация) не так уж сложна.

0.5 Асимптотический анализ Шреппеля

В конце 1970х, некоторые важные продвижения в области факторизации были сделаны Ричардом Шреппелем. Он никогда не публиковал своих результатов, однако они стали известны через копии его писем и публикации третьих лиц ([8], [4]). В начале, Шреппель начал систематическое изучение асимптотики времени работы факторизующих алгоритмов из семейства Краичика. Затем, он нашел алгоритм в этом семейства, где шаг 2 может быть достигнут без времязатратного перебора делителей.

Асимптотический анализ Шреппеля зависел от оптимального выбора параметра F, верхней границы простых чисел в базе факторизации. Маленький выбор F означает что только небольшое количество разложенных (congruences) необходимо для получения линейной зависимости, но такие (congruences) очень сложно найти. Если взять слишком большое F, то ситуация обратная. Где-то между большим и маленьким значением F находится оптимальное значение. Шреппель понял, что чтобы изучить эту ситуацию асимптотически, ему нужно использовать функцию $\phi(x, y)$ - число целых чисел вплоть до x, которые делятся только на простые числа, меньшие y. В частности, было необходимо, чтобы x был сравним со средним размером остатков от перебора делителей и $y = F$. Таким образом, $\phi(x, y)/x$ представляет "вероятность", что остаток будет полностью разложим на базе факторизации.

Например, предположим, что мы изучаем алгоритм на непрерывных дробях. Тогда типичное Q_n будет примерно равно $\sqrt{N}$. Кроме того, если f - число простых чисел в базе факторизаци, следует брать $f \approx F/2logF$ (только те нечетные простые числа p где $(N/p) = 1$ могут делить Q_n). Необходимо получить примерно f полностью факторизованных значений Q_n. Тогда следует ожидать, что мы сгенерируем

$$f(\phi(\sqrt{N}, F)/\sqrt{N})^{-1} = f\sqrt{N}/\phi(\sqrt{N}, F)$$

Значений Q_n пока достаточно факторизованных значений получено, так полное количество шагов перебора делителей, необходимых для факторизации N алгоритмом непрерывных дробей должен быть примерно

$$f^2\sqrt{N}/\phi(\sqrt{N}, F)$$

Игнорируя другие шаги алгоритма, мы выбираем F таким образом, чтобы минимизировать это количество. Шреппель предположил, что

$$\phi(x, x^{1/u}/x = u^{-(1+o(1))U})(\log x)^{\epsilon} < u < (\log x)^{1-\epsilon}$$

(Результат, который был частично доказан в [9]) и нашел, что оптимальное значение F - это $L(N)^{1/\sqrt{8}+o(1)}$, где

$$L(N) = exp(\sqrt{\log N \log\log N})$$

(натуральные логарифмы) и что предполагаемое время работы - $L(N)^{\sqrt{2}+o(1)}$. Конечно, этот аргумент всего лишь эвристический - для начала, он предполагает, без доказательства, что числа Q_n раскладываются на делители по базе факторизации также часто, как и случайные числа сопоставимого размера.

0.6 Линейное решето Шреппеля

Следующий алгоритм Шреппеля с обходом перебора делителей также в семье алгоритмов Краичика. Примем

$$S(A, B) = (\lfloor\sqrt{N}\rfloor + A)(\lfloor\sqrt{N}\rfloor + B) - NT(A, B) = (\lfloor\sqrt{N}\rfloor + A)(\lfloor\sqrt{N}\rfloor + B) \qquad (2)$$

Если $|A|, |B|$ меньше, чем N^ϵ, тогда $|S(A,B)| \leq 2N^{1/2+\epsilon}$, откуда $S(A,B)$ - относительно мало, не намного больше чем значения Q_n из (1). Более того, мы имеем

$$S(A,B) \equiv T(A,B) \mod N$$

что позволяет нам использовать эти выражения как (congruences) в схеме Краичика. Мы пытаемся полностью факторизовать значения $S(A,B)$ по базе факторизации, но мы не пробуем факторизовать значения $T(A,B)$. Стоит отметить, что (2) уже дает нам частичные факторизации $T(A,B)$. Таким образом, используя A, B четное число раз, мы можем гарантировать, что произведение каких-то $T(A,B)$ будет квадратом. Получается, что мы рассматриваем A, B как если бы они были простыми числами в методе Гаусса.

В итоге, шаг с методом Гаусса усложняется, и остатки $S(A,B)$ несколько больше, чем в методе с непрерывными дробями. Однако, здесь есть и преимущество, заключающееся в том, что числа $S(A,B)$ могут быть факторизованы без перебора делителей. Идея заключается в том, что для фиксированного значения A_O для A мы можем взять обход B по последовательным целым числам. Эти числа составляют арифметическую прогрессию, такую что если $p|S(A_O, B_O)$ тогда $p|S(A_O, B_O + p)$ $p|S(A_O, B_O + 2p)$ и т.д. А именно, мы заранее знаем в точности какие значения B будут иметь $S(A_O, B)$ делимым на p. Теперь нам не нужно тратить время на перебор делителей там, где (trial divisor does not go).

Асимптотический анализ Шреппеля предлагает принять время работы алгоритма как $L(N)^{1+o(1)}$. Однако, этот анализ не учитывает время, затраченное на метод Гаусса. Это не является ошибкой в методе на непрерывных дробях, потому что в этом случае на этот шаг действительно тратится намного меньше времени. ОДнако, в алгоритме Шреппеля на метод Гаусса возложена более трудоемкая задача и можно показать (эвристически), что он занимает $L(N)^{3/2+o(1)}$ шагов, хуже чем время работы алгоритма на непрерывных дробях.

0.7 квадратичное решето

В 1981 я предложил принимать $A = B$ в алгоритме линейного решета Шреппеля, называя полученных метод методом квадратичноего решета. Эта незначительная модификация приводит к значительным изменениям. Пусть

$$Q(A) = S(A,A) = (\lfloor \sqrt{N} \rfloor + A)^2 - N \tag{3}$$

Так, мы снова можем получать квадратичные (вычеты) в алгоритме на непрерывных дробях, так что шаг с методом Гаусса не должен представлять особой сложности. Вдобавок, мы все еще можем просеивать как Шреппель. Если $p|Q(A_O)$, то $p|Q(A_O + p)$, $p|Q(A_O + 2p)$, и т.д. Это свойство функции $Q(A)$ получается из факта, что это - многочлен с целыми коэффициентами. Эвристически, время работы алгоритма будет $L(N)^{\sqrt{9/8}+o(1)}$, включая шаг с матрицей, что является улучшением в сравнении с алгоритма на непрерывных дробях. Этот анализ и описания алгоритма можно найти в [4].

Идея (3) заключается в том, чтобы выбирать A так, чтобы $|A| < N^\epsilon$. Так как для маленьких A мы имеем

$$Q(A) \approx 2A\sqrt{N},$$

Таким образом мы имеем $|Q(A)| \leq 2N^{1/2+\epsilon}$, также как у Шреппеля. С удивлением можно отметить, что метод (3) по выбору квадратичных (вычетов) по модулю N очень похож на использованный Краичиком, что было описано выше. Однако, есть некоторые различия. Краичик аккуратно подготовил значения x так, что $x^2 - N$ имело большой квадратный делитель. В методе (3) мы берем все без разбора значения x рядом с $\sqrt{N}$.

Преимущество очевидно, потому что теперь мы можем просеивать через решето. Для каждого нечетного простого p в базе факторизации (p в факторной базе если (N/p) = 1) мы решаем квадратное (congruence)

$$(\lfloor\sqrt{N}\rfloor + A)^2 \equiv N \mod p$$

помечая решения $A_1^{(p)}, A_2^{(p)}$ (необходимо особая обработка $p = 2$). Мы затем вычисляем очень грубые логарифмы каждого из $Q(A)$ для A в длинном интервале (эти логарифмы близки по значениям). Эти логарифмы затем сохраняем в массив, индексированный значениями A. Мы затем достаем каждый логарифм, индекс которого $A \equiv A_1^{(p)}$ или $A_2^{(p)} \mod p$ и вычитаем $\log p$ из числа, хранящегося в ячейке массива (еще раз обращаем внимание, что эти логарифмы посчитаны весьма неточно). Это выполняется для всех чисел в базе факторизации а также для некоторых больших степеней маленьких p. В конце, мы можем просканировать массив на предмет наличия остаточных логарифмов, значения которых близки к нулю. Эти ячейки массива соответствуют значениям $Q(A)$, которые полностью факторизованы. Число $Q(A)$ теперь может быть посчитано и факторизовано методом перебора делителя. Очень мало чисел $Q(A)$ факторизуются полностью, так что количество переборов делителей в алгоритме достаточно небольшое. Стоит отметить, что не только алгоритм квадратичноего решета имеет асимптотически меньше шагов чем алгоритм на непрерывных дробях, но и каждый шаг проще. В алгоритме квадратичноего решета типичный шаг - это вычитание с одинарной точностью, когда в алгоритме на непрерывных дробях типичный шаг - это деление с остатком одинарно точного целого в длинное целое частное.

Асимптотически, алгоритм Шнорра и Ленстры [10] (который не входит в семейство алгоритмов Краичика) должен быть быстрее чем квадратичное решето: его эвристически посчитанная длительность работы $L(N)^{1+o(1)}$. Однако, этот алгоритм не доказан как практичный для применения на компьютерах и точка перехода может быть довольно большой. Типичный шаг в алгоритме Шнорра-Ленстры - это композиция бинарных квадратичных форм с (mult-precision) вхождениями и (reduced form in the class).

0.8 Вариант Дэвиса

Дэвис и Холдридж [11] написали очень понятную статью о реализации квадратичноего решета, и нет причин дублировать их работу здесь. Но мне бы хотелось упомянуть важное улучшение, сделанное Дэвисом. Выглядит довольно очевидным, что алгоритм квадратичноего решета превосходит алгоритм на непрерывных дробях во всех отношениях, кроме как в размере квадратичных вычетов. В частности, в последнем методе каждое $|Q_n|$ меньше, чем $2\sqrt{N}$ тогда как в первом, числа $|QA()|$ сопоставимы с $N^{1/2+\epsilon}$ (где $\epsilon > 0$ мало и медленно стремится к нулю при $N \to$ inf). Конечно же, чем больше вычет тем менее вероятно его факторизация по факторной базе.

Изменение Дэвиса заключается в том, чтобы всего лишь просеивать о различным арифметическим прогрессиям чисел A, так чтобы числа $Q(A)$ были гарантированно факторизуемы фиксированным делителем. В частности, если p - какое-то большое просто число НЕ в факторной базе, и $p|Q(A_O)$ где $0 < A_O < p$, тогда p делит каждое число $Q(A_O + Ap)$, как и было отмечено ранее. Пусть

$$Q_p(A) = Q(A_O + Ap).$$

Тогда

$$Q_p(A)/p \approx 2A\sqrt{N},$$

Так что после того, как известный делитель p "выделен" из $Q_p(A)$, его другой делитель примерно близок по размеру к $Q(A)$. Следовательно, вместо всего лишь одного многочлена с которым можно работать, мы имеем большое семейство многочлендов - один (на самом деле,

два) для каждого возможного p. Для каждого p которое было использовано мы принимаем p как новое простое число в факторной базе. Таким образом, если найдено k факторизованных значений $Q_p(A)$, после отбрасывания p у нас остается $k-1$ векторов на оригинальной базе факторизации. Однако, Дэвис избегает потери даже одного вектора. Он достигает этого при помощи нахождения факторизованного $Q_p(A)$ "бесплатно". Эта магия достигается следующим образом. Если в оригигнальном многочлене $Q(A)$ после просеивания найдено место A_1, где остаточный логарифм не близок к нулю, но меньше чем $2\log F$, тогда сомножитель после $Q(A_1)$ делится на все простые числа в факторной базе простое p где $F < p < F^2$. Мы тогда можем использовать это p, чтобы сформировать $Q_p(A)$ (и мы можем выбрать $A_0 \equiv A_1 \mod p$). мы начнем с одного факторизованного значения до того как мы начнем просеивать следующий многочлен, так что все факторизованные значения, найденные после этого будут к месту.

0.9 Вариант Монтгомери

Независимо от Дэвиса, Питер Монтгомери [12] придумал еще одну стратегию по борьбе со стремлением к бесконечности квадратичных вычетов $Q(A)$. Его метод индивидуально приспасабливает многочлены чтобы они подходили не только числу N, которое требуется разложить на множители, но и длине интервала просеивания до того, как происходит изменение многочлена.

Допустим, мы просеиваем по интервалам длины $2M$ перед сменой многочленов. Тогда у нас есть многочлены

$$F(x) = ax^2 + 2bc + c, N|b^2 - ac$$

Затем

$$aF(x) = a^2X^2 + 2abx + ac = (ax+b)^2 - (b^2 - ac)$$
$$\equiv (ax+b)^2 \mod N \quad (4)$$

Более того, нам надо чтобы значения $F(x)$ были небольшими по модулю на интервале длины $2M$. Тогда довольно обоснованно центрировать этот интервал на вершине параболы $F(x)$ - таким образом мы определяем интервал как

$$I = (-b/a - M, -b/a + M)$$

и выбираем a, b, c так, чтобы

$$-F(-b/a) \approx F(-b/a - M) = F(-b/a + M).$$

В частности, мы выбираем a, b, c так, чтобы

$$b^2 - ac = N \quad (5)$$

Тогда из (4),

$$-aF(-b/a) = N, aF(-b/a - M) = aF(-b/a + M) = a^2M^2 - N$$

Получается, нам следует выбрать a таким образом, что $N \approx a^2M^2 - N$, например

$$a \approx \sqrt{2N}/M. \quad (6)$$

Монтгомери предлагает затем, что нам следует сначала выбрать длину интервала просеивания $2M$. Затем целое число a выбирается, чтобы удовлетворять (5) и затем целые числа b, c, удовлетворяющие (6). (Например, мы можем выбрать a как простое число, удовлетворяющее

$(N/a) = 1$. Тогда квадратичное (congruence) $b^2 \equiv N \mod a$ решается для b и c выбирается как $(b^2 - N)/a$.

Таким образом, мы получили квадратичный многочлен $F(x)$ такой, что на интервале

$$|F(x)| \leq \frac{1}{\sqrt{2}} M\sqrt{N}$$

Это лучше, чем многочлены $Q(A)$ и $Q_p(A)/p$. Для них, на интервале $(-M, M)$ их абсолютные значения ограничены $2M\sqrt{N}$. Таким образом, самые большие вычеты Монтгомери примерно в $2\sqrt{2}$ раз меньше и в какой-то степени факторизуются с большей вероятностью по факторной базе.

Вот идея, которая может улучшить базовый план Монтгомери. Если $k \geq 1$ значений $F(x)$ найдено, которые факторизуются по факторной базе, у нас остается только $k - 1$ векторов, потому что делитель a должен быть исключен из (congruences) (4). Это может быть серьезно, если ожидаемое значение k будет много меньше 1, ведь тогда в редких случаях когда $k > 0$ скорее всего k будет равно единице, что ничего нам не дает. Для решения этой проблемы мы выбираем $a = g^2$, где g - это простое число с $(N/g) = 1$ и $g \approx \sqrt{\sqrt{2N}/M}$. Тогда все остается как прежде, но нам не нужно исключать a из (4) потому что это квадрат. Все факторизованные значения $F(x)$ теперь будут полезны.

квадратичное (congruence)

$$b^2 \equiv N \mod g^2 \tag{7}$$

Может быть решено очень просто если $g \equiv 3 \mod 4$ и $(N/g) = 1$. Просто взять

$$b = N^{(g^2-g+2)/4} \mod g^2$$

Это включает арифметику по модулю g^2. Вместо этого, сначала решим (7) $\mod g$ принимая $b_1 \equiv N^{(g+1)/4} \mod g$, а затем определяя x так, что $(b_1 + xg)^2 \equiv N \mod g^2$, таким образом все вычисления делаются по модулю g. (Эта идея была предложена Вагстаффом - это элементарное применение леммы Хенселя).

Сверху, мы выбрали удовлетворяющее (6) чтобы минимизировать максимальное значение $|F(X)|$ на I. Вместо этого, более корректным будет минимизировать среднее значение $|F(x)|$. Для этого выберем

$$a \approx (1.51274453)\sqrt{N}/M$$

Однако, скорее всего нет большой разницы выберем мы a данным способом или с помощью (6).

В реализации варианта Монтгомери (которая еще не была сделана на момент написания статьи), стоит вычислить стоимость вычисления новых многочленов $F(x)$. Если это дорогая операция, стоит выбрать большое значение для M. Это при том, что нас следует просеивать по самому допустимо короткому интервалу, где накладные расходы на получение новых многочленов и вычисления начальных точек для каждого простого числа, используемого в просеивании, диктует, что интервал не должен быть слишком маленьким.

0.10 Вариант с большими простыми

В [4] был предложен вариант алгоритма квадратичноего решета с большими простыми числами. Эта вариация обычно используется вместе с алгоритмом на непрерывных дробях. Как упомянуто выше, если остаточный логарифм после просеивания не близок к нулю, но меньше чем $2 \log F$, то мы получили квадратичный вычет, который полностью факторизуется на нашей факторной базе, кроме одного большого простого делителя p, где $F < p < F^2$. Не только мы

получили эту информацию без дополнительных затрат, но также такие вычеты очень просты в обработке. Если большое простое число p никогда больше не встречается в другом факторизованном вычете, но оно бесполезно для нас и его можно отбросить. Если оно встречается k раз, мы можем исключить его, оставшись с $k-1$ векторами над факторной базой. Согласно парадоксу "дней рождения" событие $k \geq 2$ не будет таким уж редким.

Если этот метод используется вместе в модификацией Дэвиса, следует использовать другой метод для генерации многочленов $Q_p(A)$. Мы можем использовать (7). Пусть $g > F$ - простое число и $g \equiv 3 \mod 4$ и $(N/g) = 1$. Если b - это решение (7), то примем $A_0 = b - \lfloor \sqrt{N} \rfloor \mod g^2$. Тогда мы можем использовать многочлен

$$Q_{g^2}(A) = Q(A_0 + g^2 A)$$

в модификации Дэвиса. (Мы также можем использовать $A_0 \equiv -b - \lfloor \sqrt{N} \rfloor \mod g^2$). Каждое значение, факторизованное на факторной базе полезно и мы можем использовать вариант с большими простыми числам на всех $Q_{g^2}(A)$ для различных g^2. Стоит отметить, что при генерации многчленов $Q_{g^2}(A)$ требуется меньше накладных расходов, чем для $F(x)$ в варианте Монтгомери, потому что g может быть взят меньшим, чем у Дэвиса.

0.11 Маленькие модули

В переборе делителей на проверку числа 3 требуется столько же времени, сколько на проверку делителя 101. Но просеивание тройкой требует в $101/3$ раз дольше, чем просеивание числом 101, так как "ячейки решета" встречаются намного чаще. Таким образом, большая часть времени, затраченного на просеивание, тратится на проверку маленьких модулей. Это выглядит как трата времени, так как маленькие модули вносят меньше всего информации. Одно из решений - не выполнять просеивание по ним вообще. Скажем, не просеивать числами меньше 30. Тогда если 3 присутствует в нашей факторной базе, мы не будем выполнять просеивание по модулю 3, по модулю 9, по модулю 27. Однако, мы будем просеивать по модулю 81, вычитая $4 \log 3$ (вместо $\log 3$) в ячейках, соответствующих просеиванию данным числом. Если P - это произведения старших степеней пропущенных модулей и если $P < F$, тогда мы ничего не потеряем при использовании данной стратегии. Действительно, максимально возможная ошибка, появившаяся в результате пропуска маленьких модулей в худшем случае будет $\log P < \log F$. Таким образом, если остаточный логарифм меньше, чем $\log F$, то число факторизовалось полностью и каждое полностью факторизованное число будет иметь остаточный логарифм меньше $\log F$.

Если эта идея хорошо себя показала, то можно попробовать "жить опасно" и принять P на сколько-то больше чем F. На самом деле, если мы примем P примерно равным F^2 и будем использовать модификацию с большими простыми числами, мы потеряем только лишь некоторые из вычетов, факторизованных большими простыми делителями. Конечно же, можно отказаться от этого метода и не терять ничего.

0.12 Использование множителя

База факторизации для числа N в алгоритме квадратичноего решета состоит и тех простых чисел, которые удовлетворяют следующим требованиям: $p \leq F$ и $p = 2(N/p) = 1$. Если мы заменим N на λN, где λ - маленькое положительное число без квадратных делителей (Краичик, см. [1] стр 208 и [2] ч 2) тогда база факторизации изменится. Ожидаемый вклад в $\log 4^2 - \lambda N$ внесенный степенью p в $x^2 - \lambda N$ будет

$$E_p = (2 \log p)/(p-1)$$

если x - случайное целое число и $(\lambda N/p) = 1$. Для $p = 2$ ожидаемый вклад

$$E_2 = \begin{cases} \frac{1}{2}\log 2 & \text{, if } \lambda N \equiv 3 \mod 4 \\ \log 2 & \text{, if } \lambda N \equiv 8 \mod 8 \\ \log 2 & \text{, if } \lambda N \equiv 1 \mod 8 \end{cases}$$

Если $p|\lambda$ то ожидаемый вклад E_p - это $(\log g)/p$. Таким образом, нам желательно выбрать значение λ так, чтобы максимизировать функцию

$$F(\lambda, N) = -\frac{1}{2}\log|\lambda| + \sum p \leq FE_p$$

Где сумма по простым числам $p \leq F$ с $p = 2$, $(\lambda N/p) = 1$, или $p|\lambda$. Эта функция очень похожа на ассоциированную с алгоритмом на непрерывных дробях (см [13] стр. 391, или [6]).

0.13 Процессоры специального назначения

Дж. Смит, С. Вагстафф Мл. и я обсудили достижимость сборки специального процессора, на котором можно было бы реализовать алгоритм квадратичноего решета. Мы вдохновлены его перспективами. В рамках бюджета в 25 тысяч долларов в деталях, мы верим, что можно собрать ”квадратичный просеиватель”, который мог бы бороться с Cray в скорости. За сумму в 10 или 20 раз большую можно построить машину, которая будет факторизовать 100-значные числа за месяц. Возможно эти цифры не соответствуют действительности, сложно сказать пока кто-нибудь не попробует.

Базовая идея ”квадратичноего просеивателя” - собрать последовательность 16x4K узлов, каждый из которых будет просеивать по интервалу длино 4096. Самые большие модули (самые быстрые в решете) будут начинать просеивать один за другим на последовательности узлов. Таким образом, не будет перемешивания модулей, так как самые ”быстрые” начнут первыми.

Литература

[1] М. Краичик. *Theorie des Nombres*, volume 2. Guthier-Villars, Paris, 1926.

[2] М. Краичик. *Recherches sur la Theorie des Nombres*, volume 2 of *Факторизация*. Guthier-Villars, Paris, 1929.

[3] Д. Лемер and Р. Пауэрс. О факторизации больших чисел. *Bulletin of the AMS*, 37:770–776, 1931.

[4] К. Померанц. Анализ и сравнение некоторых целочисленных алгоритмов факторизации. *Math. Centrum Tract*, 154:89–139, 1982.

[5] М. Вундерлих. *Доклад о факторизации 2797 чисел с использованием алгоритма на непрерывных дробях*. неопубликованные манускрипты.

[6] К. Померанц and С. Вагстафф мл. Реализация алгоритма на непрерывных дробях. *Cong. Numerantium*, 37:99–119, 1983.

[7] Дж. Миллер. О факторизации с предложенным новым подходом, 1975.

[8] Л. Мони. Algorithmes de factorisation d'entiers. In *Orsay*, 1980.

[9] Е. Канфилд, П. Ердёш, and К. Померанц. О проблеме Оппенгеймера, касающейся факторизации чисел. *Теория Чисел*, 14:1–28, 1983.

[10] Шнорр С and Х. Ленстра мл. *Алгоритм факторизации Монте Карло с ограниченной памятью*. препринт.

[11] Дж. Дэвис and Д. Холдридж. Факторизация с использованием алгоритма квадратичного решета. *Sandia Report Sand*, 83-1346, 1983.

[12] П. Монтгомери. Личная беседа.

[13] Д. Кнут. *Искусство программирования*, volume 2 of *Получисленные алгоритмы*. Addison Wesley, 2 edition, 1981.

[14] М. Моррисон and Дж. Бриллхарт. Методы факторизации. *Math. Comp.*, 29:183–205, 1975.