Факторизация чисел Ро-методом Полларда

Кузнецова Арина 6373

Ро-алгоритм Полларда

В 1975 году Полларда опубликовал статью, в которой он, основываясь на алгоритме обнаружения циклов Флойда, изложил идею алгоритма факторизации чисел, работающего за время, пропорциональное N^(¼).

С его помощью было разложено на множители число Ферма F_8 = 2^(256) + 1.

Оригинальная версия алгоритма

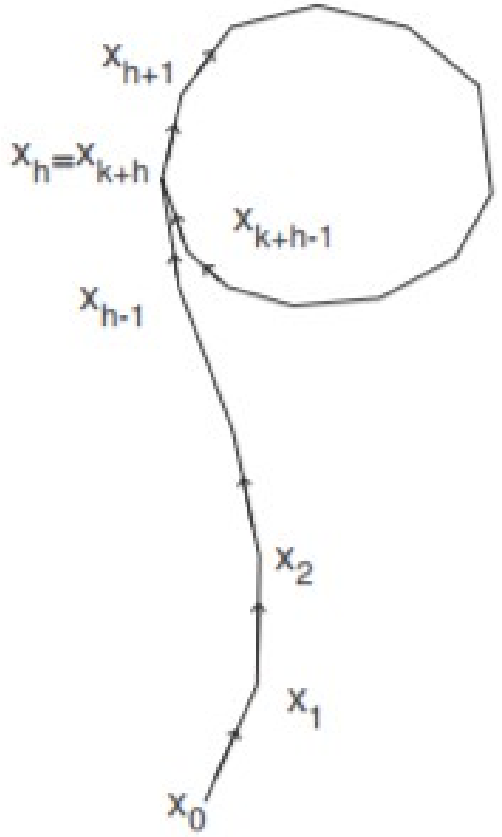
Возьмем некоторое случайное отображение

f : Zn → Zn, которое сгенерирует некоторую случайную последовательность x0, x1, x2, . . . где $x_i = f(x_{i-1})$. Обычно берётся многочлен f(x) = x*x + 1. За x0 берётся случайное число, меньшее того, которое мы факторизуем.

Функция f имеет не более, чем n значений, поэтому последовательность зациклится.

$x_{k+h} = x_h$

h называется индексом вхождения, k - длиной цикла.



Оригинальная версия алгоритма

Рассмотрим теперь алгоритм подробнее.

При выбранном многочлене $F(x) = x*x + 1 \pmod n$ рассматриваем

$x_i = f(x_{i-1})$. Причём i - индекс элемента перед зацикливанием последовательности.

И для всех $j<i$ вычисляем НОД $d = \gcd(|x_i - x_j|, n)$.

Если $n > d > 1$, то d – нетривиальный делитель n.

Если n/d - составное число, то применяем данный алгоритм ещё раз уже к числу n/d. И так продолжаем до тех пор, пока не получим разложение только из простых чисел.

Пример использования алгоритма

Нужно разложить на простые множители число 54.

Возьмём f = x*x+1 и x0=3.

Вычисляем последовательность по модулю 54, пока она не зациклится:

X0 = 3, x1 = 3*3+ 1 (mod 54) = 10, x2 = 47, x3= 49, x4=25, x4=31, x5=43, x6 =13, x7 = 7, x8 = 48...

X8 = x3, значит, последовательность зациклилась.

Начинаем попарно высчитывать НОД = (|xi — xj|, 54) чисел в последовательности, пока он не будет больше 1, но меньше 54.

При |x3-x2| = 2: НОД(2, 54) = 2. Следовательно, 2 — простой делитель 54.

54/2 = 27. Число 27 — не простое, поэтому факторизуем его аналогично. И так до тех пор, пока у нас не получится разложение из одних простых чисел.

Особенности использования алгоритма

Следует отметить, что рассматриваемый алгоритм в значительной степени случаен, его эффективность сильно и непредсказуемо зависит от выбора многочлена и начального элемента в последовательности.

Метод эффективен для нахождения небольших простых делителей числа n. Делители большего размера тоже могут быть обнаружены, однако лишь с некоторой вероятностью.

Тест на простоту чисел Миллера-Рабина

Для реализации факторизации чисел любого метода нужен тест на простоту чисел.

Я выбрала вероятностный тест Миллера-Рабина, так как при проверке k чисел на условия простоты вероятность того, что составное число будет принято за простое, будет меньше (¼)^k. В своей программе я проверяла 3 случайных числа, так в этом случае достигается достаточно удовлетворительная вероятность 0.015625.

Тест на простоту чисел Миллера-Рабина

Для проверки числа n (причём n>2) на простоту, во-первых, оно представляется в виде

n-1 = t * 2^(s), где t -нечётно.

Дальше проверяются следующие утверждения.

Если n - простое, то для любого a из Zn (a < n) выполняются:

1) a^t = 1 (mod n)

2) Существует такое r, что при 0<= r < s : a^(t*(2^s)) = -1 (mod n)

Если хотя бы одно из этих условий соблюдается, то число a является свидетелем простоты числа n.

Делая проверку на 3 случайных числах, мы повышаем вероятность того, что число n не псевдопростое.

Использование модуля факторизации числа Ро-методом Полларда

В связи с использованием достаточно времязатратных алгоритмов производительность программы с четырёхзначных чисел очень падает. Но при проверке небольших чисел выдаётся верный результат. Так как алгоритм вероятностный, то иногда он может выдать неверное разложение, но при перезапуске программы обычно это лечится.

В репозитории с программой лежит отдельный файл factRhoPollard.cpp с примером использования разработанного мной модуля. Также я выделила в него те функции, которые я дополнительного прописывала для реализации поставленного метода. На гитхабе не отображается кириллица, но при копировании репозитория всё должно быть нормально. Но, на всякий случай, размещаю тут, что делают конкретные функции:

Использование модуля факторизации числа Ро-методом Полларда

1)возведение в степень

LNum power(LNum const&, LNum const&);

2)проверка на простоту числа с помощью теста Миллера-Рабина

bool isPrimeNum(LNum const&);

3)обнаружение простого делителя числа или возвращение самого числа, если оно простое

LNum RhoPollard(LNum const&);

4)полное разложение на простые множители числа и занесение этих значений в массив

vector<LNum> factRhoPollard(LNum const& N);

Использование модуля факторизации числа Ро-методом Полларда
(как это всё выглядит)