ФАКТОРИЗАЦИЯ ЧИСЕЛ РО-МЕТОДОМ ПОЛЛАРДА

Кузнецова Арина 6373

Ро-алгоритм Полларда

В 1975 году Поллард опубликовал статью, в которой он изложил идею алгоритма факторизации чисел, работающего за время, пропорциональное $N^{1/4}$.

С его помощью было разложено на множители число Ферма $F_8 = 2^{256} + 1$.

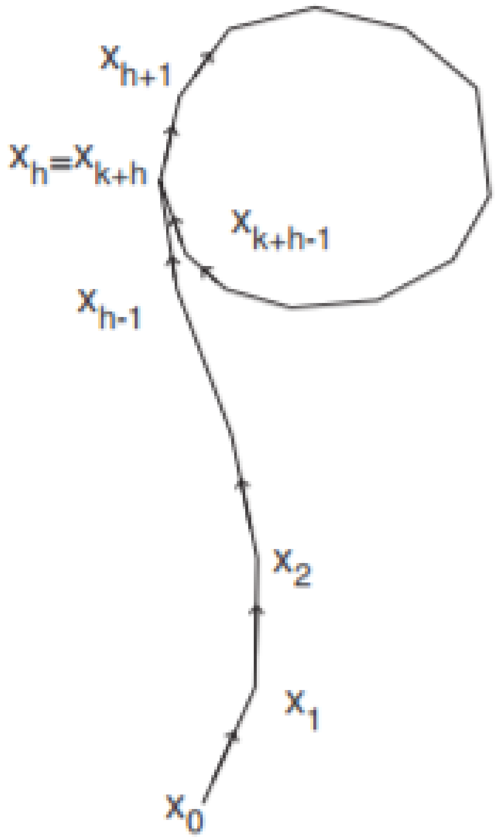
Оригинальная версия алгоритма

Возьмем некоторое случайное отображение $f\colon Z_n \to Z_n$, которое сгенерирует некоторую случайную последовательность $x_0, x_1, x_2...$ где $x_i = f(x_{i-1})(mod\ n)$. Обычно берётся многочлен $f(x) = x^2$+1.

Так как последовательность генерируется в Z_n, то в определённый момент она зациклится.

$$x_{k+h} = x_h$$

h называется индексом вхождения, k - длиной цикла.



Оригинальная версия алгоритма

Рассмотрим теперь алгоритм подробнее.

При выбранном многочлене $f(x) = x^2 + 1 \ (mod\ n)$ рассматриваем $x_i = f(x_{i-1})$. Причём i - индекс элемента перед зацикливанием последовательности.

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

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 $F = x^2 + 1$ и $x_0 = 3$.

Вычисляем последовательность по модулю 54, пока она не зациклится:

$$x_0 = 3, x_1 = 3 * 3 + 1 \ (mod\ 54) = 10,$$
$$x_2 = 47, x_3 = 50, x_4 = 17, x_5 = 20, x_6 = 23, x_7 = 44, x_8 = 47 \ldots$$

$x_8 = x_3$, значит, последовательность зациклилась.

Начинаем попарно высчитывать НОД = $gcf\,(|x_i - x_j|, 54)$ чисел в

последовательности, пока он не будет больше 1, но меньше 54.

При $|x_3 - x_2| = 3$: НОД(3, 54) = 3. Следовательно, 3 — простой делитель 54.

54/3 = 18.

Число 18 — не простое, поэтому применяем к нему алгоритм аналогично. И так до тех пор, пока у нас не получится разложение из одних простых чисел.

Обоснование алгоритма

Оценка основывается на «парадоксе дня рождения».

Пусть $\lambda > 0$. Для случайной выборки из $l + 1$ элементов, каждый из которых меньше q , где $l = \sqrt{2\lambda q}$, вероятность того, что два элемента окажутся равными $p > 1 - e^{-\lambda}$.

- Утверждение: Пусть S – фиксированное множество из r элементов, f – какое-либо отображение $f\colon S \to S, x_0 \in S$, последовательность $x_0, x_1, x_2 \dots$ определяется соотношением $x_i = f(x_{i-1})$. Пусть λ>0, $l = 1 + \left[\sqrt{2\lambda r}\right] < r$. Тогда доля тех пар (f, x_0) (где f пробегает все отображения из S в S и x_0 пробегает всё множество S), у которых $x_0, x_1, x_2 \dots x_l$ попарно различны, среди всех пар (f, x_0) не превосходит $e^{-\lambda}$.
- Доказательство:

Всего имеется $r^r * r = r^{r+1}$ различных пар (f, x_0). Пар (f, x_0) , у которых $x_0, x_1, x_2 \dots x_l$ попарно различны будет

$$r(r-1)\dots(r-l) * r^{r-l}$$

Доля таких пар составляет величину

$$t = r^{-r-1} * r^{r-l+1} \prod_{j=1}^{l}(r-j) = \prod_{j=1}^{l}(1 - {}^{j}/_{r})$$

Поскольку при 0< x< 1 выполнено неравенство ln(1 – x) < –x, то

$$\ln(t) = \sum_{j=1}^{l}(1 - {}^{j}/_{r}) < -\sum_{j=1}^{l}({}^{j}/_{r}) = -\frac{l(l+1)}{2r} < -\frac{l^2}{2r} < -\frac{2\lambda r}{2t} = -\lambda$$

Обоснование алгоритма

Отметим, что вероятность $p = 0,5$ в парадоксе дня рождения достигается при $\lambda \approx 0,69$.

Пусть последовательность $\{u_n\}$ состоит из разностей $|x_i - x_j|$, проверяемых в ходе работы алгоритма. Определим новую последовательность $\{z_n\}$, где $z_n = u_n \ (mod\ q)$, q – меньший из делителей n.

Все члены последовательности $\{z_n\}$ меньше $\sqrt{n}$. Если рассматривать $\{z_n\}$ как случайную последовательность чисел, меньших q , то, согласно парадоксу дней рождений, вероятность того, что среди первых $l + 1$ ее членов попадутся два одинаковых, превысит $^1/_2$ при λ ≈ 0,69, тогда l должно быть не меньше $\sqrt{2\lambda q} \approx \sqrt{1.4q} \approx 1,18\sqrt{q}$.

Обоснование алгоритма

Если $z_i = z_j$, тогда $x_i - x_j = 0 \pmod p \rightarrow x_i - x_j = kq$ для некоторого k ∈ Z.

Если $x_i \neq x_j$, что выполняется с большой вероятностью, то искомый делитель q числа n будет найден как НОД $= gcf(|x_i - x_j|, n)$.

Поскольку $\sqrt{q} \leq n^{1/4}$, то с вероятностью, превышающей 0,5, делитель n может быть найден за $1,18 \cdot n^{1/4}$ итераций. Таким образом, ρ-метод Полларда является вероятностным методом, позволяющим найти нетривиальный делитель q числа n за O($q^{1/2}$) ≤ O($n^{1/4}$) итераций.

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 $({}^{1}/_{4})^{k}$. В своей программе я проверяла 3 случайных числа, так в этом случае достигается достаточно удовлетворительная вероятность 0.015625.

В моей программе функция носит название isPrimeNum().

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 Z_n $(a < n)$ выполняются:

1) $a^t = 1\ (mod\ n)$

2) ∃ r, что $0 \le r < s:\ a^{2^s t} = -1\ (mod\ n)$

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.

Пример использования теста на простоту чисел Миллера-Рабина

- Возьмём заведомо известное простое число, например, 23. Проверим его на простоту согласно тесту Миллера-Рабина.
- Представим его в виде 22 = 2^1*11

Проверяем первое условие:

- Возьмём любое натуральное число меньше 23, например, 5
- $5^{11} \pmod{23} = 22$ – не удовлетворяет

Проверяем второе условие (так как первое условие выполняется, то второе проверять необязательно, но для примера проверяем и второе):

- $5^{11} \pmod{23} = 22$ – удовлетворяет

Следовательно, число 23 – простое.

Возведение в степень по модулю

Данный алгоритм основывается на том факте, что для заданных a и b следующие 2 уравнения эквивалентны:

$$\text{c} = a * b \ (mod\ m)$$

$$c = a * \big(\ b\ (mod\ m)\big)(mod\ m)$$

Алгоритм следующий:

- Пусть c = 1, e’ = 0
- Увеличим e’ на 1
- Установим $c = b * c\ (mod\ m)$
- Если $e' < e$, возвращаемся к шагу 2. В противном случае, c содержит правильный вариант ответ $\text{c} = b^{e}\ (mod\ m)$.

В моей программе данный алгоритм носит название modular_pow().

Пример использования алгоритма возведения в степень по модулю

Допустим, стоит задача возвести в степень по модулю следующее выражение: $3^6\ (mod\ 32)$. Тогда $\mathrm{b} = 3, \mathrm{e} = 6, \mathrm{m} = 32$.

1. e’ = 1, c = (1*3) (mod 32) = 3;
2. e’ = 2, c = (3*3) (mod 32) = 9;
3. e’ = 3, c = (9*3) (mod 32) = 27;
4. e’ = 4, c = (27*3) (mod 32) = 17;
5. e’ = 5, c = (17*3) (mod 32) = 19;
6. e’ = 6, c = (19*3) (mod 32) = 25.

Следовательно, 3^6 (mod 32) = 25.

Результаты работы программы