Механико-математический факультет
Кафедра диф. уравнений и системного анализа

Факторизация чисел. Экспоненциальные методы.

Чергинец Дмитрий Николаевич

Метод пробных делений

Вход

Составное число n.

Выход

Нетривиальный делитель числа n.

1. $d = 2$.
2. Если $d \mid n$, то
   выдаем результат: d — делитель n,
   заканчиваем работу алгоритма.
3. Увеличиваем d на единицу $d = d + 1$ и переходим к шагу 2.

Сложность алгоритма

Метод пробных делений имеет экспоненциальную сложность, количество арифметических операций равно

$$f(n) = d - 1.$$

d – наименьший нетривиальный делитель n.

$$d \leq n^{1/2} \quad \Rightarrow \quad f(n) = O(n^{1/2}).$$

Сложность алгоритма

Метод пробных делений имеет экспоненциальную сложность, количество арифметических операций равно

$$f(n) = d - 1.$$

d – наименьший нетривиальный делитель n.

$$d \leq n^{1/2} \;\Rightarrow\; f(n) = O(n^{1/2}).$$

$$\langle n \rangle = [\log_2 n] + 1 \;\Rightarrow\; n = O(2^{\langle n \rangle}).$$

Сложность алгоритма

Метод пробных делений имеет экспоненциальную сложность, количество арифметических операций равно

$$f(n) = d - 1.$$

d – наименьший нетривиальный делитель n.

$$d \leq n^{1/2} \quad \Rightarrow \quad f(n) = O(n^{1/2}).$$

$$\langle n \rangle = [\log_2 n] + 1 \quad \Rightarrow \quad n = O(2^{\langle n \rangle}).$$

Временная сложность

$$T(N) := \max_{\langle n \rangle \leq N} f(n) = O(2^{\frac{N}{2}}).$$

Сложность алгоритма

Метод пробных делений имеет экспоненциальную сложность, количество арифметических операций равно

$$f(n) = d - 1.$$

d – наименьший нетривиальный делитель n.

$$d \leq n^{1/2} \quad \Rightarrow \quad f(n) = O(n^{1/2}).$$

$$\langle n \rangle = [\log_2 n] + 1 \quad \Rightarrow \quad n = O(2^{\langle n \rangle}).$$

Временная сложность

$$T(N) := \max_{\langle n \rangle \leq N} f(n) = O(2^{\frac{N}{2}}).$$

Парадокс дней рождения

- Вероятность того, что среди 23 человек есть хотя бы 2 человека с одинаковым днем рождения, больше $\frac{1}{2}$.
- Вероятность того, что Ваш день рождения совпадает с днем рождения хотя бы одного человека из группы, состоящей из 158 человек, меньше $\frac{1}{2}$.
- Вероятность того, что среди 58 человек есть хотя бы 2 человека с одинаковым днем рождения, больше 0.99.
- Парадокс здесь в том, что наше предположение о данной вероятности значительно не совпадает с реальной

ρ-алгоритм Полларда

- Пусть p — делитель числа n, нам даны случайные $l+1$ число
$$x_0, x_1, \ldots, x_l \in \mathbb{Z}_n.$$
- С точки зрения парадокса дней рождения с большой вероятностью найдутся такие j, i, $0 \le j < i \le l$, что
$$x_j \equiv x_i \pmod{p}.$$
- Тогда $d := \gcd(x_j - x_i, n) \ge p$, если к тому же $d < n$, то нам повезло и мы нашли делитель.

Алгоритм Черепаха

Полный перебор и генератор случ. чисел

- **Вход.** Составное число $n \in \mathbb{N}$.
- **Выход.** Нетривиальный делитель числа n.
- 1. Задаем начальные данные: $i := 0$, выбираем случайное x_i из кольца $\mathbb{Z}_n$.
- 2. Присваиваем $i := i + 1$ и выбираем случайное x_i из кольца $\mathbb{Z}_n$.
- 3. Для $j = 0, 1, \ldots, i - 1$ вып. шаги $3.1, 3.2$.

  3.1. Вычисляем $d := \gcd(x_i - x_j, n)$.

  3.2. Если $1 < d < n$, то выдаем результат d, конец алгоритма.
- 4. Переходим к шагу 2.

Алгоритм Черепаха

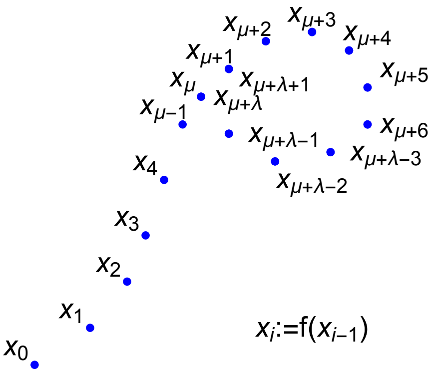
Данный алгоритм можно представить себе как игровое поле, составленное из упорядоченных клеток $x_0, x_1, \dots$. Клетки x_j и x_k называются Одинаковыми, если $\gcd(x_j - x_k, n) > 1$. По данным клеткам начиная с x_0 по-порядку двигается фишка, которую назовем Черепаха. Задача Черепахи найти одинаковые фишки, для которых к тому же $\gcd(x_j - x_k, n) < n$.

Данный алгоритм является вероятностным, но всегда выдает правильный ответ. В данном алгоритме третий шаг является очень трудоемким, так как на нем i раз вычисляется НОД. Попробуем уменьшить количество этих операций.

Генерация случ. послед-ти при помощи функции

- x_0 выбираем случайным.
- Остальные x_i вычисляем при помощи рекуррентной формулы
$$x_i := f(x_{i-1}),$$
где $f : \mathbb{Z}_n \to \mathbb{Z}_n$.
- Как правило в качестве функции f берут многочлен второй степени
$$f(x) := x^2 + 1 \pmod n.$$
- (f, x_0) задает последовательность
$$x_0, x_1, x_2, \ldots$$

Определение μ, λ



Пусть $\mu, \lambda,\ \lambda > 0$, наименьшие числа среди тех, для которых справедливо сравнение

$$x_\mu \equiv x_{\mu+\lambda} \pmod{p} \;\Leftrightarrow\; \text{НОД}(x_\mu - x_{\mu+\lambda}, n) > 1.$$

Цикличность

$$x_\mu \equiv x_{\mu+\lambda} \pmod p \;\Rightarrow$$

$$\Rightarrow\; f(x_\mu) \equiv f(x_{\mu+\lambda}) \pmod p.$$

Тогда для всех $m \in \mathbb{N}$, подействовав на обе части сравнения m раз функцией f, получаем сравнение

$$f^m(x_\mu) \equiv f^m(x_{\mu+\lambda}) \pmod p \Rightarrow$$

$$\Rightarrow x_{\mu+m} \equiv x_{\mu+\lambda+m} \pmod p.$$

Алгоритм Черепаха и Заяц

- **Вход.** Составное $n \in \mathbb{N}$. Посл. (f, x_0).
- **Выход.** Нетривиальный делитель числа n.
- 1. Задаем нач. позиции Черепахи и Зайца
$$Tortoise := x_0, \qquad Hare := x_0.$$
- 2. Черепаха «переползает» на следующий элемент, а Заяц «прыгает» через элемент
$$Tortoise := f(Tortoise), \quad Hare := f(f(Hare)).$$
- 3. Вычисляем $d := \gcd(Tortoise - Hare, n)$.
- 4. Если $1 < d < n$, то выдаем рез. d, конец алг-ма.
- 5. Если $d = n$, то нет решения, конец алгоритма.
- 6. Переходим к шагу 2.

Сложность

количество НОД в алгоритме Черепаха:

$$1+2+3+\cdots+(\mu+\lambda-1)+(\mu+1)=$$

$$=\frac{1}{2}(\mu+\lambda-1)(\mu+\lambda)+(\mu+1)\approx\frac{1}{2}(\mu+\lambda+1)(\mu+\lambda).$$

Количество НОД в алгоритме Черепаха и Заяц. Это количество будет совпадать с индексом i конечной позиции Черепахи x_i. Необходимо найти минимальное $i \in \mathbb{N}$, удовлетворяющее условиям

$$i \geq \mu, \qquad i \equiv 0 \pmod{\lambda}.$$

То есть $i = \lambda\lceil\frac{\mu}{\lambda}\rceil$.

$$\mu \leq \lambda\lceil\tfrac{\mu}{\lambda}\rceil < \mu+\lambda.$$

Черепаха и Ахиллес

Рассмотрим еще один вариант ρ-метода Полларда. Вместо зайца по элементам последовательности

$$x_{2^0}, x_{2^1}, \ldots, x_{2^k}, \ldots$$

будет двигаться Ахиллес. Причем, как утверждал Зенон Элейский, Ахиллес никогда не догонит черепаху.

Алгоритм Черепаха и Ахилес

- **Вход.** Составное $n \in \mathbb{N}$, посл. (f, x_0).
- **Выход.** Нетривиальный делитель p числа n.
- 1. Присваиваем начальные значения
$$Achilles := x_0, \qquad Tortoise := x_0, \qquad k := 0.$$
- 2. Для $i = 1, 2, 3, \ldots, 2^k$ вып. $2.1 - 2.4$:
  2.1. Передвигаем Черепаху $Tortoise := f(Tortoise)$.
  2.2. Выч. $d := \gcd(Tortoise - Achilles, n)$.
  2.3. Если $1 < d < n$, то выдаем рез. d, конец алг.
  2.4. Если $d = n$, то нет решения, конец алгоритма.
- 3. Передвигаем Ахиллеса с $x_{2^{k-1}}$ на элемент x_{2^k}
$$Achilles := Tortoise, \qquad k := k + 1,$$
переходим на шаг 2.

Сложность

Посчитаем количество НОД.
Алгоритм закончит работу, когда
Ахиллес будет находиться на $x_{2^{k-1}}$,
Черепаха на $x_{2^{k-1}+\lambda}$, где k — наименьшее целое, удовлетворяющее неравенствам

$$\begin{cases} \mu \leq 2^{k-1}, \\ \lambda \leq 2^k, \\ k \in \mathbb{N}, \\ k \to \min\,. \end{cases}$$

Сложность

$$\begin{cases} \mu \le 2^{k-1}, \\ \lambda \le 2^k, \\ k \in \mathbb{N}, \\ k \to \min. \end{cases} \Rightarrow \begin{cases} \log_2 \mu + 1 \le k, \\ \log_2 \lambda \le k, \\ k \in \mathbb{N}, \\ k \to \min. \end{cases}$$

$$k := \max\{\lceil \log_2 \mu \rceil + 1, \lceil \log_2 \lambda \rceil\},$$

Количество НОД:

$$2^{k-1} + \lambda = \lambda + 2^{\max\{\lceil \log_2 \mu \rceil, \lceil \log_2 \lambda \rceil - 1\}}.$$

Сравнение алгоритмов по НОД

Количество НОД алг. Черепаха и Ахиллес:

$$\lambda + \max\{\mu, \lambda/2\} \leq \lambda + 2^{\max\{\lceil \log_2 \mu \rceil, \lceil \log_2 \lambda \rceil - 1\}}$$

Количество НОД алг. Черепаха и Заяц:

$$\mu \leq \lambda \lceil \frac{\mu}{\lambda} \rceil < \mu + \lambda.$$

Очевидно, что

$$\mu + \lambda \leq \lambda + \max\{\mu, \lambda/2\}.$$

Поэтому количество вычислений НОД в варианте Черепаха и Заяц меньше, чем в варианте Ахиллес и Черепаха.

Сравнение алгоритмов по $f(x)$

В алгоритме Черепаха и Заяц функция f вычисляется

$$3\mu \leq 3i = 3\lambda\lceil\frac{\mu}{\lambda}\rceil$$

В алгоритме Ахиллес и Черепаха

$$\lambda + 2^{\max\{\lceil\log_2\mu\rceil, \lceil\log_2\lambda\rceil - 1\}} < \lambda + \max\{2\mu, \lambda\}.$$

Тем не менее при вычислении функции f выполняется 3 арифметических операции, а при вычислении НОД: $O(\log n)$.

Парадокс дней рождения

Теорема

Пусть X — множество из n элементов.
$\lambda \in \mathbb{R}$ удовлетворяет неравенствам

$$\lambda > 0, \qquad l := 1 + [\sqrt{2\lambda n}] < n.$$

Тогда вероятность того, что среди $l + 1$ случайным образом взятого элемента

$$x_0, x_1, \ldots, x_l \in X$$

любые два попарно различны, равна

$$(1 - 1/n)(1 - 2/n) \ldots (1 - l/n) < e^{-\lambda}.$$

Парадокс дней рождения

Пусть случайное событие A означает, что $l+1$ случайный элемент попарно различен. Случайным образом выбираем и фиксируем элемент $x_0 \in X$. Вероятность того что следующий случайный элемент $x_1 \in X$ отличен от уже выбранного равна

$$1 - 1/n.$$

Вероятность того, что элемент x_i не совпадает ни с одним из попарно различных элементов $x_0, x_1, \ldots, x_{i-1}$ равна $(1 - \frac{i}{n})$. Таким образом,

$$P(A) = (1 - 1/n)(1 - 2/n) \ldots (1 - l/n).$$

Парадокс дней рождения

$$P(A) = (1 - 1/n)(1 - 2/n)\dots(1 - l/n) =$$

$$= \frac{n!}{n^{l+1}(n-l-1)!}.$$

Воспользуемся $\ln(1 + x) \leq x$

$$\ln P(A) = \sum_{i=1}^{l} \ln(1 - i/n) < -\frac{1}{n} - \dots - \frac{l}{n} =$$

$$= -\frac{l(l+1)}{2n} < -\frac{l^2}{2n} = -\frac{(1 + [\sqrt{2\lambda n}])^2}{2n} < -\lambda.$$

Следовательно, $P(A) < e^{-\lambda}$.

Пусть $n = pq$. p, q – простые. Будем считать, что

$$p \approx q \approx n^{1/2}$$

При каком l среди элементов

$$x_0, x_1, \dots, x_l \in \mathbb{Z}_{|}$$

с вероятностью $1/2$ есть два одинаковых элемента

$$x_\mu \equiv x_{\mu+\lambda} \pmod p.$$

$$e^{-\lambda} = 1/2 \qquad \Rightarrow \qquad \lambda = \ln 2.$$

$$l = 1 + [\sqrt{2\sqrt{n}\ln 2}] = O(n^{1/4}).$$

$O(n^{1/4}\ln n)$ - среднее количество арифметических операций ρ-алгоритма Полларда.

$(p-1)$-алгоритм Полларда

- Пусть p — нетрив. простой делитель n.
- Известно разложение

$$p-1 = q_1^{\alpha_1} q_2^{\alpha_2} \ldots q_s^{\alpha_s}.$$

- Малая теорема Ферма. $a \in \mathbb{N}$, $\gcd(a, p) = 1$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

- Тогда

$$d := \gcd(a^{p-1} - 1, n) \geq p.$$

Если $d < n$, то мы имеем нетривиальный делитель d числа n.

$(p-1)$-алгоритм Полларда

- Но нам неизвестны ни число p, ни его делители q_i.
- Предположим, что нам известны такие числа $M, K \in \mathbb{N}$, что
$$p < M, \qquad q_i < K$$
для всех $i = 1, 2, \ldots, s$.
- Понятно, что, например, при $M = K = n$ неравенства удовлетворяются, но с вычислительной точки зрения нам такие большие числа M и K не подойдут.
- Через B обозначим все простые числа, меньшие K,
$$B := \{2, 3, 5, 7, \ldots, p_i, \ldots, p_m\},$$
p_i — простые, $p_i < K$.

$(p-1)$-алгоритм Полларда

- Так как $q_i < K$, то число $p-1$ имеет разложение
$$p - 1 = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_m^{\alpha_m},$$
где $\alpha_i \geq 0$, но мы не знаем α_i. Так как $p < M$, то
$$\alpha_i < \frac{\ln M}{\ln p_i}.$$
- Обозначим $\beta_i := \left[\frac{\ln M}{\ln p_i}\right]$.
- Получили число
$$P := p_1^{\beta_1} p_2^{\beta_2} \dots p_m^{\beta_m}.$$
- Так как $\beta_i \geq \alpha_i$, то P делится на $p-1$, поэтому
$$a^P \equiv 1 \pmod{p}.$$
Следовательно, $\gcd(a^P - 1, n) \geq p$.

$(p-1)$-алгоритм Полларда

- **Вход.** Составное число n. $M, K \in \mathbb{N}$.
- **Выход.** Нетривиальный делитель p.
- 1. Находим все простые делители, меньшие K, $B := \{2, 3, 5, 7, \ldots, p_i, \ldots, p_m\}$.
- 2. Выбираем случайное $a \in \mathbb{N}$, $1 < a < n$, если $d := \gcd(a, n) > 1$, то d — делитель, конец алгоритма.
- 3. Для $i = 1, 2, \ldots, m$ выполняем операции 3.1, 3.2.

  3.1. Вычисляем $\beta_i := \left[\dfrac{\ln M}{\ln p_i}\right]$.

  3.2. Вычисляем $a := a^{p_i^{\beta_i}} \pmod n$.
- 4. Находим $d := \gcd(a-1, n)$.
- 5. Если $d = 1$, то выдаем результат «делитель не найден, возьмите M, K большими», конец алгоритма.
- 6. Если $1 < d < n$, то выдаем рез. d, конец алг.
- 7. Если $d = n$, то выдаем рез. «делитель не найден, возьмите M, K меньшими или другое a», конец алг.

Китайская теорема об остатках

Theorem

Пусть $n_1, \ldots, n_k \in \mathbb{N}$, $\gcd(n_i, n_j) = 1$ *при* $i \neq j$, $b_1, \ldots, b_k \in \mathbb{Z}$. *Тогда система уравнений*

$$\begin{cases} x \equiv b_1 \pmod{n_1}, \\ x \equiv b_2 \pmod{n_2}, \\ \quad\vdots \\ x \equiv b_k \pmod{n_k}, \end{cases} \tag{1}$$

имеет единственное в кольце $\mathbb{Z}_n$ *решение*

$$x_0 = \sum_{i=1}^{k} b_i N_i C_i,$$

где $n = n_1 \ldots n_k$, $N_i = \frac{n}{n_i}$, C_i *— обратный к* N_i *в* $\mathbb{Z}^*_{n_i}$.

Доказательство

x_0 определено корректно?

$$\gcd(n_i, n_j) = 1 \quad \Rightarrow \quad \gcd(n_i, N_i) = 1$$

$$\gcd(n_i, N_i) = 1 \quad \Rightarrow \quad \exists C_i = N_i^{-1} \pmod{n_i}.$$

x_0 является решением системы?

Для каждого j, $1 \leq j \leq k$, справедливо сравнение

$$x_0 = \sum_{i=1}^{k} b_i N_i C_i \equiv b_j N_j C_j \equiv b_j \pmod{n_j},$$

следовательно, x_0 — решение.

Доказательство

Докажем единственность решения.

Предположим, что существуют два решения системы

$$x_1, x_2 \in \mathbb{Z}_n.$$

Для сравнений справедливо следующее свойство

$$\begin{cases} x_1 \equiv x_2 \pmod{n_1}, \\ x_1 \equiv x_2 \pmod{n_2}, \\ \gcd(n_1, n_2) = 1; \end{cases} \Rightarrow \ x_1 \equiv x_2 \pmod{n_1 n_2}.$$

Применяя его $k-1$ раз, получаем $x_1 \equiv x_2 \pmod{n}$.

Идея алгоритма Гарнера

Формула, указанная в теореме, хороша, но есть более быстрый алгоритм.

Пусть $x_i \in \mathbb{Z}$, $0 \leq x_i < n_1 \ldots n_i$, – решение системы, составленной из первых i уравнений:

$$\begin{cases} x \equiv b_1 \pmod{n_1}, \\ x \equiv b_2 \pmod{n_2}, \\ \quad\vdots \\ x \equiv b_i \pmod{n_i}. \end{cases}$$

Методом математической индукции получим формулы для нахождения x_i.

При $i = 1$ имеем $x_1 := b_1 \pmod{n_1}$.

Идея алгоритма Гарнера

Пусть известно x_{i-1}, найдем x_i. Решение будем искать в виде

$$x_i = x_{i-1} + N_i y_i,$$

где $N_i = n_1 n_2 \dots n_{i-1}$.
За счет данного вида число x_i уже является решением $j = 1, \dots, i-1$ уравнения:

$$x_i \equiv x_{i-1} + N_i y_i \equiv x_{i-1} \equiv b_j \pmod{n_j}.$$

Число $y_i, 0 \le y_i < n_i$, подберем таким образом, чтобы x_i удовлетворяло уравнению

$$x \equiv b_i \pmod{n_i}.$$

Идея алгоритма Гарнера

Подставив x_i в уравнение, получим

$$x_{i-1} + N_i y_i \equiv b_i \pmod{n_i};$$

$$N_i y_i \equiv (b_i - x_{i-1}) \pmod{n_i};$$

$$y_i := C_i(b_i - x_{i-1}) \pmod{n_i},$$

где $C_i := N_i^{-1} \pmod{n_i}$ вычисляется при помощи расширенного алгоритма Евклида.
Отметим, что найденное решение удовлетворяет неравенству

$$x_i = x_{i-1} + N_i y_i < N_i + N_i(n_i - 1) = N_{i+1}.$$

Алгоритм Гарнера

- **Вход:** $b_1, \ldots, b_k \in \mathbb{Z}$,
  $n_1, \ldots, n_k \in \mathbb{N}$ – взаимно простые.
- **Выход:** x, $0 \le x < n_1 \ldots n_k$, – решение (1).
- 1. Задаем начальные значения переменных:
  $N := 1$,
  $x := b_1 \pmod{n_1}$.
- 2. Для $i := 2, \ldots, k$ последовательно вычисляем:
  $N := Nn_{i-1}$,
  $C := N^{-1} \pmod{n_i}$,
  $y := C(b_i - x) \pmod{n_i}$,
  $x := Ny + x$.
- 3. Выдаем результат: x.