Математичка, ты довольна?
Часть 1.

Сапожников Денис

Contents

1 Малая теорема Ферма (МТФ)

Теорема (Малая теорема Ферма)**.** *Пусть* $a \neq 0$ *и* $p \in \mathbb{P}$[1]*, тогда*

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

Proof. Докажем по индукции, что $a^p \equiv a \pmod p$.

База. $a = 1 : 1^p = 1$.

Переход. $(a+1)^p = a^p + C_p^1 a + \dots + 1 = p \cdot (...) + a^p + 1 \Rightarrow$
$\Rightarrow (a+1)^p \equiv a^p + 1 \equiv a + 1 \pmod p$, что и т.д.

Теперь мы можем написать тривиальную цепочку сравнений по модулю:

$$\begin{aligned}
a^p &\equiv a \pmod p \\
a^p - a &\equiv 0 \pmod p \\
a(a^{p-1} - 1) &\equiv 0 \pmod p\ a \neq 0, \textit{значит} \\
a^{p-1} - 1 &\equiv 0 \pmod p \\
a^{p-1} &\equiv 1 \pmod p
\end{aligned}$$

□

[1] $\mathbb{P}$ обозначает множество простых чисел

2 Обратный элемент по модулю

Определение (Обратный элемент по модулю). Обратный элемент для числа a по модулю p — это такое число b, что $a \cdot b \equiv 1 \pmod p$. Такое число b ещё иногда обозначают как a^{-1}.

По МТФ мы можем найти обратный элемент для любого числа по простому модулю:

$$a^{p-1} = 1$$
$$a^{p-2} \cdot a = 1$$
$$b \cdot a = 1, \textit{где } b = a^{p-2}$$

2.1 Бинарное возведение в степень

Заметим, что $a^{2k} = (a^{\frac{k}{2}})^2$ и $a^{2k+1} = (a^{\frac{k}{2}})^2 \cdot a$

Но что мы сделали такой операцией? свели нашу задачу к подсчету задачи в 2 раза меньшей по k. Значит можем сводить так $\log n$ раз и радоваться жизни.

```
int bin_pow(int a, int p, int m) {
  if (p == 0)
    return 1;
  else {
    long long t = bin_pow(a, p / 2, m);
    t = t * t % m;
    if (p % 2 == 1)
      t = t * a % m;
    return t;
  }
}
```

2.2 Обратные ко всем от 1 до m*

Но что если нам понадобиться найти обратные ко всем числам на интервале $[1; m]$ по модулю p за $O(m)$? Оказывается, и такое можно решить!

Пусть r_i - обратный элемент для i. Докажем факт из которого будет следовать очевидный подсчет всех r_i:

$r_i \equiv -\lfloor \frac{p}{i} \rfloor r_{p \bmod i}$

Почему это верно? Совершим цепочку тривиальных преобразований:

$p \bmod i = p - \lfloor \frac{p}{i} \rfloor i$

$p \bmod i \equiv -\lfloor \frac{p}{i} \rfloor i$

Домножим обе части на произведение обратного к i и обратного к $p \bmod i$:

$r_i \equiv -\lfloor \frac{p}{i} \rfloor r_{p \bmod i}$

Итого код такой:

```
  r[1] = 1;
  for (int i = 2; i <= m; i++)
    r[i] = (p - r[p % i] * (p / i)) % p;
```

3 Функция Эйлера и теорема Эйлера

Задача. Найти количество чисел, взаимнопростых с n на отрезке $[1; n]$. Такая функция от n будет иметь название Функция Эйлера и обозначаться $\varphi(n)$.

Например, $\varphi(10) = 4$ (1, 3, 7, 9 взаимнопросты с 10).
О функции Эйлера следует знать следующие факты:

- $\varphi(p) = p - 1, p \in \mathbb{P}$
- $\varphi(p^a) = p^a - p^{a-1}, p \in \mathbb{P}$
- $\varphi(ab) = \varphi(a)\varphi(b)$, если НОД$(a, b) = 1$.

Тогда можно красиво посчитать функцию Эйлера, используя разложение на простые множители: $\varphi(n) = \varphi(p_1^{k_1})\varphi(p_2^{k_2})\dots\varphi(p_t^{k_t}) = \left(p_1^{k_1} - p_1^{k_1-1}\right)\left(p_2^{k_1} - p_2^{k_1-1}\right)\dots\left(p_t^{k_t} - p_t^{k_t-1}\right)$

Благодаря такому разложению код почти не будет отличаться от обычной факторизации:

```
int phi(int n) {
  int result = 1;
  for (int i = 2; i * i <= n; ++i)
    if (n % i == 0) {
      int degp = 1;
      while (n % i == 0)
        n /= i, degp *= i;
      result *= degp - degp / i;
    }
  if (n > 1)
    ans *= n - 1;
  return result;
}
```

Ещё один факт, который стоит знать про функцию Эйлера - это **теорема Эйлера:**

$$a^{\varphi(n)} = 1 \pmod n, \forall a \in \mathbb{N}$$

Proof. Построим граф:
$V = \{1 \le x \le n | (x, n) = 1\}, E = \{(x, ax) | x \in V\}$
Заметим, что граф - набор циклов, докажем, что все циклы одинаковой длины:
Посмотрим на то, куда отображаются цикл при домножении каждого элемента на произвольное b: он перейдет в другой цикл, а значит этот цикл той же длины. А теперь воспользуемся свойством построенного множества, что $\forall a \in V$: здесь должен быть какой-то факт, но я его забыл.
Пусть k - длина цикла, тогда $k = \varphi(n)$
Значит $1 \equiv a^{\frac{|V|}{k}} \pmod n \Leftrightarrow 1 \equiv a^{\varphi(n)} \pmod n$

□

4 Проверка на простоту Миллера-Рабина*

Мы сейчас будем идти к вероятностному алгоритму проверки числа на простоту за $O(\log^3 n)$.

Лемма. *Если p – нечётное простое число и $k \geq 1$, то уравнение*

$$x^2 \equiv 1 \pmod{p^k}$$

имеет лишь 2 решения: $x = 1, x = -1$.

Proof. Это уравнение эквивалентно $(x-1)(x+1) \equiv 0 \pmod{p^k}$. Не более чем одно из чисел $x-1$ или $x+1$ может делиться на p, поскольку если они делятся оба, то и их разность (число 2) также делится на p, что невозможно. Если $x-1$ не делится на p, то $x+1$ должно делиться на p^k, значит $x = -1 \pmod{p^k}$. Второй случай аналогичный. □

Числа 1 и -1 – тривиальные квадратные корни из единицы. Теорема утверждает, что по модулю степени нечётного простого других квадратных корней из единицы не бывает. Если по некоторому модулю n они вдруг найдутся, это значит, что n – составное. Алгоритм Миллера–Рабина для проверки простоты числа, кроме проверки условия малой теоремы Ферма, проверяет ещё и это условие.

Алгоритм, получив на входе число n, сперва записывает число $n-1$ в виде $n-1 = 2^t u$, где u нечётно. Тогда, в частности, $a^{n-1} = a^{2^t u} = (a^u)^{2^t}$, и можно сперва вычислить a^u, а потом дальше возводить в квадрат t раз. Если довозводить до конца, то потом останется только проверить условие Ферма. Но по дороге, после каждого возведения в квадрат, делается ещё одна проверка: если получилась единица, а на прошлом шаге была не 1 и не -1, то найден квадратный корень из единицы.

TO-DO: напиши код.

Теорема. *Если алгоритм выдаёт результат, что число составное, то оно действительно составное, иначе оно выдаёт неверный ответ с вероятностью менее чем $\frac{1}{2^k}$.*

Proof. Легко понять часть про составное число: алгоритм выдаёт такой результат только если не выполнятся МТФ или если нашли нетривиальный корень из единицы, что противоречит условию леммы.

Теперь заметим, что если $\gcd(a,n) \neq 1$, то не выполнится условие МТФ, а значит мы рассматриваем только случай, когда $\gcd(a,n) = 1$ и алгоритм сказал, что n – простое.

Случай 1: $\exists b : \gcd(b,n) = 1, b^{n-1} \not\equiv 1 \pmod{n}$. Иными словами, n – не число Кармайкла. Тогда при каком-то выборе a, в частности при $a = b$, алгоритм найдёт подтверждение тому, что n – составное. Чтобы оценить вероятность этого события, нужно понять, сколько всего будет таких чисел b. Оказывается, что их достаточно много, не меньше половины. Сперва показывается, что множество всех остальных b – образует подгруппу, обозначим её за $\mathbb{Z}_n^\times$.

Действительно, если b и c принадлежат этому множеству, то их произведение тоже принадлежит: $(bc)^{n-1} \equiv b^{n-1}c^{n-1} \equiv 1^2 \equiv 1 \pmod{n}$. Если b принадлежит множеству и $c = b^{-1}$, то, стало быть, $bc \equiv 1 \pmod{n}$, и потому $(bc)^{n-1} \equiv 1 \pmod{n}$. Так как $b^{n-1} \equiv 1 \pmod{n}$,отсюда следует, что $c^{n-1} \equiv 1 \pmod{n}$. Далее, по теореме Лагранжа из теории групп, размер подгруппы – делитель числа элементов в группе, и потому их не больше, чем $\frac{\varphi(n)}{2}$. Стало быть, такое число b будет выбрано с вероятностью, не превышающей $\frac{1}{2}$.

Случай 2: $\forall b : \gcd(b,n) = 1 \Rightarrow b^{n-1} \equiv 1 \pmod{n}$. Или n – это число Кармайкла. Сошлёмся на результат из ТЧ: любое такое число n не может быть степенью простого числа. Пусть $n = pq, \gcd(p,q) = 1, p, q \geq 3$.

Для всякого значения a, алгоритм строит последовательность $a^u, a^{2u}, a^{4u}, \dots, a^{2^t u}$, при этом последний элемент равен 1. Далее, если условие леммы ни на каком шаге не будет нарушено, то последовательность может или полностью состоять из единиц, или же на каком-то шаге в ней впервые встретится -1, после чего пойдут одни единицы. В этом и только в этом случае алгоритм не найдёт подтверждения тому, что n – составное.

Пусть $j \in \{0, ..., t\}$ – наибольшее число, для которого $c^{2^j u} \equiv -1 \pmod{n}$ верно для какого-то $c \in \mathbb{Z}_n^\times$. Заметим, что существует по крайней мере одна такая пара (c, j): для $j = 0$ и $c = -1$ выполняется $(-1)^{2^0 u} = -1$, поскольку u нечётно. Для **наибольшего** j рассматриваем множество:

$$X = \{x \in \mathbb{Z}_n^\times \big| x^{2^j u} \equiv \pm 1 \pmod{n}\}$$

Все значения a, для который алгоритм скажет, что число простое лежат в X. Заметим, что X – это подгруппа в $\mathbb{Z}_n^\times$. Если докажем, что $X \neq \mathbb{Z}_n^\times$ тогда мы опять получим, что размер X не превышает половины группы, а значит вероятность выбрать «плохое» a и в этом случае не превышает $\dfrac{1}{2}$.

Из $c^{2^j u} \equiv -1 \pmod{n} \Rightarrow c^{2^j u} \equiv -1 \pmod{p}$, тогда по КТО найдётся y :

$$\begin{cases} y \equiv -1 \pmod{p} \\ y \equiv 1 \pmod{q} \end{cases} \Rightarrow \begin{cases} y^{2^j u} \equiv -1 \pmod{p} \\ y^{2^j u} \equiv 1 \pmod{q} \end{cases}.$$

Отсюда следует, что $y^{2^j u} \neq \pm 1$, то есть $y^{2^j u} \notin X$. Теперь покажем, что $y \in \mathbb{Z}_n^\times$.

$$\begin{cases} \gcd(c, n) = 1 \Rightarrow \gcd(c, p) = 1 \Rightarrow \gcd(y, p) = 1 \\ y \equiv 1 \pmod{q} \Rightarrow \gcd(y, q) = 1 \end{cases} \Rightarrow \gcd(y, n) = 1 \Rightarrow y \in \mathbb{Z}_n^\times \qquad \square$$