# Второй курс, осенний семестр 2018/19
# Практика по алгоритмам #7

## Теория чисел

## 7 ноября

Собрано 7 ноября 2018 г. в 21:52

---

## Содержание

# Теория чисел

1. **Решето Эратосфена**
   a) Найти все простые числа на $[n^2, n^2 + n]$ за $\mathcal{O}(n \log \log n)$.
   b) Найти все простые на $[1, n]$ за $\mathcal{O}(n \log \log n)$ с $\mathcal{O}(\sqrt{n})$ памяти.
2. **Применяем решето**
   Для каждого числа от 1 до $n$ найти: число делителей, сумму делителей, функцию Эйлера, функцию Мёбиуса.
3. **Расширенный Евклид**
   a) Придумайте нерекурсивную версию.
   b) Докажите, что в строке $ax_i + by_i = r_i$: $(x_i, y_i) = 1$
   c) Докажите, что $\max |x_i| \leqslant |b|$ и $\max |y_i| \leqslant |a|$
   d) Найдите класс решений диофантового уравнения $ax \equiv b (\mathrm{mod}\, m)$
   e) Найдите $x, y$: $ax + by = c, |x| + |y| \to \min$
4. **RSA**
   a) Пусть $n = pq$, известно $\varphi(n)$, разложите $n$ на множители.
   b) Пусть у нас есть «волшебный» оракул. Для любого открытого ключа $(n, e)$ оракул может взломать 1% из всех возможных зашифрованных сообщений. Придумайте алгоритм, который взламывает любое сообщение. Матожидание времени работы $\mathcal{O}(\mathrm{poly}(\log n))$.
5. **Магия**
   Поймите, что делает код:

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

6. **Первообразный корень**
   Мы обсудили, как найти первообразный корень простого $n$. А если $n$ не простое?
7. **Биномиальные коэффициенты по модулю**
   При данных $n$ и простом $p$ можно представить $n! = p^k r$. Поймем, как быстро найти $k$ и $r \bmod p$.
   Пользуясь этим, найти $\binom{n}{k} \bmod m$.
8. **(*) Башня степеней**
   Посчитайте $a_1^{a_2^{a_3^{\cdot^{\cdot^{\cdot^{a_n}}}}}} \bmod m$.
9. **(*) Квадратный корень по модулю**
   Хотим найти $r$: $r^2 \bmod p = n$. Представим $p = q2^s$.
   Все равенства далее по модулю $p$.
   a) Пусть у нас есть $r$: $r^2 = nt$, где $t^{2^k} = 1$. Основной шаг алгоритма в том, чтобы получить новые $r$ и $t$, чтобы $r^2 = nt$ и $t^{2^{k-1}} = 1$.
   Для этого шага воспользуемся величиной $c$: $c^{2^{s-1}} = -1$.
   b) С чего начать и чем закончить алгоритм?
   c) Где взять нужное $c$?
   d) Оценить время работы.

# Разбор задач практики

1. **Решето Эратосфена**
   a) За $\mathcal{O}(n \log\log n)$ нашли все простые от 1 до $n$. Теперь для каждого простого $p_k$ вычеркнем все числа на отрезке $[n^2..n^2+n]$, кратные ему. Сделаем это за $\mathcal{O}(\frac{n}{p_k})$.
   b) Предподсчитаем все простые от 1 до $\sqrt{n}$. Для каждого отрезка $[i\sqrt{n}..(i+1)\sqrt{n})$ вычеркнем все непростые за $\mathcal{O}(\sqrt{n}+\sqrt{n})$ (количество простые + длина отрезка).

2. **Применяем решето**

```
int d[N]; // d[x] - минимальный простой делитель x
int cnt[N]; // cnt[x] - степень вхождения d[x]
int y[N]; // y[x] = x / d[x]^cnt[x]
// Запустили решето, нашли d[], d[1] = 0
cnt_div[1] = sum_div[1] = mu[1] = 1;
for (int x = 2; x < N; x++) {
  int z = x / d[x];
  if (d[x] == d[z])
    cnt[x] = cnt[z] + 1, y[x] = y[z];
  else
    cnt[x] = 1, y[x] = z;
  cnt_div[x] = cnt_div[y[x]] * (cnt[x] + 1);
  sum_div[x] = sum_div[y[x]] * ((x / y[x] * d[x] - 1) / (d[x] - 1));
  mu[x] = cnt[x] > 1 ? 0 : -mu[y[x]];
}
```

3. **Расширенный Евклид**
   a) Поддерживаем на каждом шаге представление $a_i = x_{a,i}a + y_{a,i}b, b_i = x_{b,i}a + y_{b,i}b$.
   b) Доказываем $(x_i, y_i) = 1$. Пользуемся тем, что алгоритм Евклида корректен.
   Если в какой-то момент $(x_i, y_i) = t$, то $|ax_i + by_i| \geqslant t(a,b) \Rightarrow t = 1$.
   c) Индукция. База: $b = 0 \Rightarrow x = 1, y = 0$. Переход:
   $r = a - kb = a \bmod b \wedge (a-kb)x' + by' = (a,b)$
   $x = x' \wedge y = y' - kx'$, по индукции имеем $|x'| \leqslant b \wedge |y'| \leqslant r \Rightarrow |y' - kx'| \leqslant r + kb = a$ ■
   d) $ax_0 + my = (a,m) \Rightarrow ax_0\frac{b}{(a,m)} \equiv b \bmod m \Rightarrow x \in \{x_0\frac{b}{(a,m)} + k\frac{m}{(a,m)} \quad | \quad \forall k \in \mathbb{Z}\}$.
   e) Нарисуем прямую $ax + by = c$, она пересекает оси координат.
   Проверим две точки рядом с абциссой, две рядом с ординатой.

4. **RSA**
   a) Пусть $n = pq$, известно $\varphi(n) \Rightarrow \varphi(n) = n - p - q + 1 \Rightarrow p + q = -(\varphi(n) - n - 1) \Rightarrow x^2 + (\varphi(n) - n - 1)x + n = 0$ имеет корни $p, q$.
   b) Нам дали $c = m^e \bmod n$. Загадаем случайное число $r$.
   Дадим оракулу $r^e c \bmod n = (rm)^e \bmod n$, с вероятностью 0.01 он расшифрует.

5. **Магия**
   Этот код считает обратные по модулю $p$ ко всем $i = 1..p-1$. Возьмём $0 \equiv p = i \cdot \lfloor \frac{p}{i} \rfloor + (p \bmod i)$. Домножим на $(p \bmod i)^{-1}$, получим $0 \equiv i \cdot \lfloor \frac{p}{i} \rfloor \cdot (p \bmod i)^{-1} + 1 \Rightarrow i \cdot -\left[\lfloor \frac{p}{i} \rfloor \cdot (p \bmod i)^{-1}\right] \equiv 1$.

6. **Первообразный корень**
   Всё то же самое, только вместо $n-1$ испльзуем $\phi(n)$.
   Напомним из алгебры, что первообразный корень есть только у чисел вида $2, 4, p^k, 2p^k$.

7. **Биномиальные коэффициенты по модулю**
   e-maxx.
   Если умеем по модулю $p^k$, то по КТО умеем и по модулю $m$.

8. **(*) Башня степеней**
   Факторизуем один раз $m$. Замечаем $a^x \bmod m = \texttt{КТО}(a^x \bmod p_i^{\alpha_i})$. Решаем задачу по модулю $p_i^{\alpha_i}$.
   Пусть $a = p_i^{\beta_i} g$: $(g, p_i) = 1 \Rightarrow a^x \bmod p_i^{\alpha_i} = \left[g^{x \bmod (p_i-1)p_i^{\alpha_i-1}} p_i^{\min(x\beta_i, \alpha_i)}\right] \bmod p_i^{\alpha_i}$.
   Мы свели задачу к двум: такой же и посчитать $\min(x, \frac{\alpha_i}{\beta_i})$.

9. **(*) Квадратный корень по модулю**
   Алгоритм Tonelli–Shanks.

# Домашнее задание

## 3.1. Обязательная часть

1. **(3) Подсчёт $d^p$ в лоб**
   Мы умеем за $\mathcal{O}(n)$ для всех $x \in [2, n]$ искать $d[x]$ – минимальный простой делитель $x$. Умеем искать и $p[x]$, степень вхождения $d[x]$ в $x$, за $\mathcal{O}(n)$ динамикой.
   Докажите, что наивный поиск $p[x]$ работает тоже за $\mathcal{O}(n)$:
   `for (x = 2..N) for (y = x; d[y] == d[x]; y /= d[x]) p[x]++`
   *Есть короткое док-во совсем без алгебры.*

2. **(3) Я теряю корни**
   На лекции научились по данным $p, a, b$ искать $x$: $x^a = b \pmod p$ ($p$ простое).
   Найти не один, а все такие $x$ (заодно можно заметить, что на лекции был маленький обман).

3. **(3) Долой факторизацию!**
   Для проверки того, что некоторое $g$ – первообразный корень, нам пришлось факторизовать $\phi(p)$. Но часто нам нужен не сам корень, а нужно решить какую-то задачу, используя его. Тогда порой можно обойтись без факторизации.
   Мы применяли первообразный корень для поиска дискретного корня: по данным $p, a, b$ искать $x$: $x^a = b \pmod p$ ($p$ простое).
   Сделайте это не за $\mathcal{O}(T(\text{FindPrimitiveRoot}) + T(\text{DiscreteLog}) + \log n)$, а за $\mathcal{O}((T(\text{DiscreteLog}) + \log n) \log\log n)$. Вероятно, алгоритм выйдет вероятностный.

4. **(3) Взлом RSA при малом $e$**
   У трех людей открытые ключи RSA с $e = 3$, но модуль $n$ у кажого свой.
   Так вышло, что их модули взаимно просты.
   Всем трем отправили одинаковое сообщение $m$, зашифровав их ключами.
   Взломать $m$, если мы перезватили все три шифровки. Детерминировано и гарантированно.

## 3.2. Дополнительная часть

1. **(5)** $\binom{n}{k} \bmod m$.
   Рассмотрим алгоритм подсчёта $\binom{n}{k} \bmod m$ за $\mathcal{O}(n \log m) + \texttt{FACT}(m)$. Разложим $m = \prod_i p_i^{\alpha_i}$. $\binom{n}{k} = \frac{n!}{k!(n-k)!} \equiv \frac{f_p(n)}{f_p(k) f_p(n-k)} p^{cnt_p(n) - cnt_p(k) - cnt_p(n-k)} \bmod p^\alpha$. Далее используем КТО.
   Слабое место этого алгоритма – факторизация. Придумайте аналог за $\mathcal{O}(\text{poly}(n, \log m))$.