2 курс ПМИ, осень 2024/25
Практика по алгоритмам #10

Теория чисел

26 ноября

Собрано 29 ноября 2024 г. в 10:03

Содержание

Теория чисел

1. **Решето Эратосфена**
   a) $n \leqslant 10^6$. 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: сумму делителей и $\varphi(n)$.

3. **Расширенный Евклид**
   a) Придумаем/вспомним рекурсивного расширенного Евклида: найти x, y: $ax+by=\gcd(a,b)$.
   b) Докажите: на каждом шаге $ax_i + by_i = \gcd(a, b)$ рекурсивного Евклида $\gcd(x_i, y_i) = 1$.
   c) Докажите, что на каждом шаге рекурсивного Евклида $|x_i| \leqslant |b|$ и $|y_i| \leqslant |a|$.
   d) Описать все решения диофантова уравнения $ax \equiv b \pmod m$.
   e) `(*)` Найти x, y: $ax + by = c, |x| + |y| \to \min$.
   f) `(*)` Придумайте нерекурсивную версию.

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 $\mathcal{O}(\text{poly}(\log n))$.

5. **RSA и реальный мир**

   Предложите быстрый способ передачи большого объёма закодированных данных.
   Наш арсенал: Диффи-Хеллман, RSA, xor-кодирование, псевдорандом.

6. **Первообразный корень**

   Мы обсудили, как найти первообразный корень простого n. А если n не простое?

7. **Битые сектора**

   Есть множество $A \subseteq [n]$ и k ячеек, каждая из которых умеет хранить $\mathcal{O}(\log n)$ бит информации. В каждый момент времени максимум t из k ячеек могут оказаться недоступны (мы можем узнать про ячейку, доступна ли она, и, если доступна, прочитать данные). Требуется организовать такой способ хранения информации, чтобы в любой момент времени можно было востановить множество A.
   a) $k = (t + 1) \cdot |A|$.
   b) $k = t + |A|$ при $t = 1$.
   c) $k = t + |A|$.

8. **Биномиальные коэффициенты по модулю**

   При данных $n \leqslant 10^6$ и простом $p \leqslant 10^6$ можно представить $n! = p^k r$. Поймите, как быстро найти k и $r \bmod p$. Пользуясь этим, найдите $\binom{n}{k} \bmod m$. Какой предподсчёт нужно сделать, чтобы, зная заранее m, отвечать на запрос $n, k \leqslant 10^6$, $\binom{n}{k} \bmod m$ за $\mathcal{O}(\log m)$.

9. **Квадратный корень по модулю**

   За сколько уже сейчас вы умеете решать уравнение $x^2 = a \bmod p$?

10. (*) **Башня степеней**

Посчитайте $a_1^{a_2^{\cdot^{\cdot^{a_n}}}} \bmod m$.

Пример: $m = 2^7 3^8$, $a = 2 \cdot 5$. В какую степень возводить?

Кто поможет!

11. (*) **Быстрый квадратный корень**

Хотим найти $r\colon r^2 \bmod p = n$. Представим $p = q2^s$.
Все равенства далее по модулю p.

a) Пусть у нас есть $r\colon r^2 = nt$, 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 $c\colon c^{2^{s-1}} = -1$.

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) См. конспект.

   b) Доказываем $(x_i, y_i) = 1$. Если в какой-то момент $(x_i, y_i) = t$, то на всех следующих шагах $ax_i + by_i$ делится на $t \cdot \gcd(a,b) \Rightarrow$, поскольку Евклид корректно ищет gcd, $t = 1$.

   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) $\forall k \in \mathbb{Z} \;\; (x_0, y_0) + k(\frac{b}{gcd(a,b)}, \frac{-a}{gcd(a,b)})$ — решение. $|x_0 + k\frac{b}{gcd(a,b)}| + |y_0 + k\frac{-a}{gcd(a,b)}|$ — почти линейная функция от k. Если бы мы знали, как раскрывается модуль, функция была бы линейной и минимум достигался бы на одном из краёв $\Rightarrow$

   Решение: нарисуем прямую $ax + by = c$, она пересекает оси координат, проверим два решения ближние к абциссе, два решения ближние к ординате.

   f) См. конспект.

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. **RSA и реальный мир**

   Пример решения – каждый блок длины ≈16m (?) xor-шифровать псевдослучайным ключом. Самый первый ключ получен или DH, или RSA.

6. **Первообразный корень**

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

7. **Битые сектора**

   a) $k = (t+1) \cdot |A|$: дублируем каждую ячейку t+1 раз.
   b) $k = t + |A|$ при $t = 1$: храним XOR или сумму по модулю.
   c) $k = t + |A|$. Решение – сказать, что $a_0, a_1, \ldots, a_{n-1}$ – коэффициенты $A(x)$, храним $A(1), A(2), \ldots, A(k)$. Восстановление = Гаусс. *Альтернативное решение:* видимо, можно хранить сумму i-х степеней $i = 1..t$, зная их и оставшиеся $|A| - t$ ячейку, всё восстанавливается.

8. **Биномиальные коэффициенты по модулю**

   [e-maxx]
   Если умеем по модулю p^k, то по КТО умеем и по модулю m.
   Предподсчёт: для каждого p из разложения m храним все факториалы в форме $p^{k_i} \cdot r_i$.

9. **Квадратный корень по модулю**

   Логарифмируем $a = g^i$, ищем x в виде $x = g^k$, получаем $g^{2k} = g^i \bmod p \Rightarrow 2k = i \bmod (p-1)$.

10. **(*) Башня степеней**

    Факторизуем один раз m. $a^x \bmod m = \texttt{КТО}(a^x \bmod p_i^{\alpha_i})$. 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 $\min(x, \frac{\alpha_i}{\beta_i})$.

11. **(*) Быстрый квадратный корень**

    [Алгоритм Tonelli-Shanks]

Домашнее задание

3.1. Обязательная часть

1. **(2) Не самая наивная факторизация**

   Приходят запросы факторизовать число $a_i \leqslant N$.
   Предпосчет $\mathcal{O}(\sqrt{N})$, ответ на запрос за $\mathcal{O}(\sqrt{N}/\log N)$.
   Ну, мы же вроде знаем, сколько бывает простых чисел от 1 *до* N*?*

2. **(2) Евклид и полезность**

   Лягушонок Вася живёт на прямой. Он умеет целеустремлённо прыгать только вперёд.
   Устройство лапок позволяет прыгать ему только на a или b вперёд.
   Может ли он попасть в точку c?

3. **(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).

4. **(3) Эратосфен и гладкость**

   Обозначим i-е по возрастанию простое число, как p_i.

   Назовём число b-гладким, если все его простые делители не превосходят p_b.

   Дано $n \leqslant 10^6$. Для каждого $b \leqslant n$ найдите количество b-гладких чисел от 1 до n.

5. **(3) Взлом RSA при малом** e

   Аня решила послать приглашение на секретную вечеринку Боре, Ване и Гоше. Аня разослала им одинаковый текст приглашения M закодированный с помощью RSA. У Вани, Бори и Гоши выбраны различные n, но ключ e у всех одинаковый: $e = 3$. Придумайте, как Дима сможет узнать (за полиномиальное от суммы длин всех чисел время), где будет происходить секретная вечеринка, если ему доступны все три шифрованных приглашения и открытые ключи.

3.2. Дополнительная 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. **(4) $\pi(n)$ для больших n**

   $\pi(n)$ – количество простых от 1 до n.
   С помощью решета Эратосфена умеем считать $\pi(n)$ за $\mathcal{O}(n)$. Можно быстрее. Эта задача даёт вам подсказку и предлагает придумать алгоритм.
   $\Phi(n, d)$ – количество чисел от 1 до n, все простые делители которых больше d.
   Можно вывести рекуррентную формулу пересчета $\Phi(n, p_i)$, выразить через эту величину π, а из этого получить алгоритм.
   Баллы будут ставиться за любое решение за $\mathcal{O}(n^{1-\varepsilon})$. Существует решение за $\mathcal{O}(n^{2/3})$.

3. **(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 $\mathcal{O}(\text{poly}(n, \log m))$.

4. **(5) Быстрое дискретное логарифмирование**

   Придумайте любой алгоритм для поиска дискретного логарифма за $\mathcal{O}(n^{1/4} \cdot poly(\log))$.