SPb HSE, МОАД ВБИБ, осень 2024/25
Практика по алгоритмам #21

Мощь Фурье
6 марта

Собрано 4 марта 2025 г. в 20:59

Содержание

Мощь Фурье

1. **Билетики**

   Количество счастливых билетов из $2n$ цифр по модулю m за $\mathcal{O}(n \log n)$.

2. **Возведение в степень**

   За какое время можно посчитать 2^n в десятичной системе счисления?
   (*) Подумайте про выбор системы счисления и реальное время работы.

3. **Циклические сдвиги**

   Даны A, B, $|A| = |B| = n$. Найти D – такой циклический сдвиг B,
   что скалярное произведение A и D максимально.

4. **Поиск с ошибками**

   Даны текст t и строка s над алфавитом размера k. Для каждого из $|t| - |s| + 1$ наложений s на t узнать количество ошибок. Время $\mathcal{O}(k|t| \log |t|)$.

5. **Поиск с ошибками и шаблоном**

   Апгрейд предыдущей задачи. И в тексте, и в строке допустимы символы «?».

6. **Цепочка умножений**

   Хотим посчитать $P_1(x) \cdot \ldots \cdot P_k(x)$, все многочлены степени n. За сколько можно это сделать?

7. **Уравнение**

   Даны n и m. Найти число троек (x, y, z): $x^n + y^n \equiv z^n \bmod m$. $m \leqslant 10^6$.

8. **Дуэль!**

   В каждой клетке полоски $1 \times n$ или растёт дерево, или нет. За $\mathcal{O}(n \log n)$ найдите количество троек деревьев, подходящих для дуэли (два дуэлянта и секундант). Тройка деревьев на позициях $i < j < k$ подходит, если $j - i = k - j$.

9. **Раскраски**

   С помощью `FFT` за $\mathcal{O}^*(2^n)$ проверьте, можно ли вершины неорграфа покрасить в k цветов.

10. **Поиск подстроки в строке**

    За $\mathcal{O}(1)$ вызовов FFT найти подстроку в строке за $\mathcal{O}(n \log n)$.

11. **`FFT` по простому модулю**

    Мы прошли `FFT` над полем комплексных чисел. Предложите аналог над полем остатков по простому модулю. Примеры простых: $2^{18} \cdot 3 + 1$, $2^{20} \cdot 7 + 1$, $2^{25} \cdot 5 + 1$.

12. **`FFT` и повышение точности**

    Перемножьте с помощью `FFT` $A, B \in K[\mathbb{Z}/10^9\mathbb{Z}]$, степени многочленов до 10^6.
    Есть два решения: используя FFT над $\mathbb{C}$ и FFT над $\mathbb{Z}/p\mathbb{Z}$.

13. **FFT, базовые оптимизации**

    Вспомните код FFT. Модифицируйте так, чтобы было $2n$ тригонометрических операций и

$n \log n$ умножений комплексных чисел. Уменьшите число умножений в 2 раза.

14. **FFT, два в одном**

Научитесь одним FFT от $\mathbb{C}[x]$ получать два FFT от $\mathbb{R}[x]$.

15. **Inplace FFT**

Ниже — код оптимизированного FFT, примерно такой мы разбирали на лекции.

a) Прочитайте код. Поймите.
b) Как изменить FFT, чтобы она стала inplace, то есть изменяла сразу массив `a`?

```
const int K = 20, N = 1 << K;
complex<double> root[N];
int rev[N];
void init():
    for (int j = 0; j < N; j++)
        rev[j] = (rev[j >> 1] >> 1) + ((j & 1) << (K - 1));
    for (int k = 1; k < N; k *= 2)
        num tmp = exp(PI / k);
        root[k] = {1, 0}; // в root[k..2k) хранятся первые k корней степени 2k
        for (int i = 1; i < k; i++)
            root[k+i] = (i & 1) ? root[(k+i) >> 1] * tmp : root[(k+i) >> 1];
void FFT(a, fa): // a → fa
    for (int i = 0; i < N; i++)
        fa[rev[i]] = a[i];
    for (int k = 1; k < N; k *= 2)
        for (int i = 0; i < N; i += 2 * k)
            for (int j = 0; j < k; j++)
                num tmp = root[k + j] * fa[i + j + k];
                fa[i + j + k] = fa[i + j] - tmp;
                fa[i + j] = fa[i + j] + tmp;
```

Разбор задач практики

1. **Билетики**

   ?

2. **Возведение в степень**

   Возведем в степень за $\mathcal{O}(\log n)$, получаем $\texttt{FFT}(n) + \texttt{FFT}(\frac{n}{2}) + \texttt{FFT}(\frac{n}{4}) + \ldots = \mathcal{O}(n \log n)$.

   (*) Длина ответа $\frac{3}{10}n$, а в системе счисления 10^k даже меньше, $\frac{3}{10k}n$. Берём $k = 5$, получаем при $n \leqslant 10^6$ типа `long double` хватает ($10^5 \cdot 10^5 \cdot \frac{3}{50}10^6 < 10^{15}$). 2 вещественных в одном комплексном $\Rightarrow$ умножение чисел $= 2 \cdot \texttt{FFT}$, а время работы алгоритма $\leqslant 4 \cdot \texttt{FFT}$.
   Итого $4 \cdot N \cdot \log N$ умножение комплексных чисел для $N = \frac{3}{50}10^6 < 2^{16}$.
   $4 \cdot 2^{16} \cdot 16 = 4\,000\,000$ умножений комплексных чисел для задачи «посчитать 2^n при $n = 10^6$».

3. **Поиск с ошибками**

   Считаем отдельно для каждого символа s. Фиксируем символ c и строим

   $$S_c = \sum_{s_i = c} x^i,\ T_c = \sum_{t_{|t|-i+1} = c} x^i (\text{то есть } t \text{ разворачиваем})$$

   $P = S_c T_c$, тогда P_i равно числу позиций, где c есть и в s, и в приложении s к i-й позиции t.

   Число вызовов FFT сейчас $3k$. Вспомним, что умеем 2 прямых в одном, получим $2k$.
   Результат обратных мы хотим сложить. $FFT^{-1}(a) + FFT^{-1}(b) = FFT^{-1}(a+b) \Rightarrow$
   все обратные делаются одним.

4. **Поиск с ошибками и шаблоном**

   То же самое, но в T_c учитываем не только c, но и «?».
   В конце нужно вычесть пары «?», «?», все они учтены дважды.

5. **Цепочка умножений**

   Если сделаем k прямых `FFT(nk)`, перемножим, затем обратное `FFT(nk)`, то будет время $\mathcal{O}(k^2 n + FFT(nk)) = \mathcal{O}(nk(k + \log n))$. А можно перемножить пары соседних многочленов за `FFT(2n)` каждую, потом новые пары соседних за `FFT(4n)` каждую и т.д. Суммарно за $\mathcal{O}(\frac{k}{2}FFT(2n) + \frac{k}{4}FFT(4n) + ... + FFT(kn)) = \mathcal{O}(nk \log(nk))$.

6. **Уравнение**

   `a[`$x^n \bmod m$`]++`, для всех $x \in [0, m-1]$. Это делается за $\mathcal{O}(m \log n)$. И даже за $\mathcal{O}(\frac{m}{\log m} \log n)$:
   $(xy)^n = x^n y^n \Rightarrow$ степень считать нужно только для простых.
   Далее $b = a^2, res = \sum b_i a_i$.

7. **Дуэль!**

   Если i-я клетка центр тройки (j, i, k), то $j + k = 2i$.
   Смотрим на многочлен $a = \sum a_i x^i$. $b = a^2$, b_{2i} равно числу нужных пар. Ответ $= \sum b_{2i} - 1$.
   Можно обобщить на случай, когда в клетке растет $a_i \in \mathbb{Z}$ деревьев, тогда ответ $\sum b_{2i} a_i - a_i^2$.

8. **Раскраски**

   ?

9. **Поиск подстроки в строке**

   Посчитаем $errors_i = \sum (s_{i+j} - p_j)^2 = \sum s_{i+j}^2 + \sum p_j^2 - 2\sum s_{i+j} p_j$, получили две частичные

суммы квадратов и скалярное произведение циклических сдвигов. FFT!

10. **FFT по простому модулю**

    Ищем корни из единицы по модулю p. Для этого нужен первообразный корень g по модулю p. Теперь у нас есть циклическая группа $1 = g^0, g^1, g^2, \dots g^{p-1} = 1$. Она аналогична по свойствам группе $w^0, w^1, \dots, w^{p-1} = w^0$ для $w = e^{2\pi i/(p-1)}$. Единственная проблема, что $p - 1 = 2^s t$ не обязательно степень двойки. С этим можно бороться двумя способами: взять группу размера 2^s, порождённую g^t, или объявить вершину рекурсии с многочленом длины t листом рекурсии и обработать его за $\mathcal{O}(t^2)$. Обычно берут берут $p = c2^k + 1$, где c мало. Время работы FFT для группы размера 2^s равно $\mathcal{O}(2^s s)$ Если степень многочлена-результата сильно меньше 2^s, мы можем взять группу любого меньшего размера 2^{s-j}.

11. **FFT и повышение точности**

    Заметим, что коэффициенты ответа при перемножение, как над $K[\mathbb{Z}]$, не более 10^{23}.

    Решение #1. FFT по простому модулю и KTO. 3 раза перемножим многочлены по разным трём простым модулям порядка 10^9 (для этого нам достаточно типа `int64`). Теперь независимо для каждого из коэффициентов ответа воспользуемся KTO: есть остатки по трём модулям...

    Решение #2. Представим коэффициенты многочлена A в виде $a_i = b_i + Mc_i$, где $M = \lceil\sqrt{10^9}\rceil$, и $0 \leqslant b_i, c_i < M$. Теперь многочлен $A(x)$ представлен в виде $B(x) + C(x)M$. У нас есть два многочлена, представим так оба, считаем $A_1(x)A_2(x) = (B_1(x) + C_1(x)M)(B_2(x) + C_2(x)M) = B_1(x)B_2(x) + B_1(x)C_2(x)M + B_2(x)C_1(x)M + B_2(x)C_2(x)M^2$. То есть, сделаем 4 FFT с малыми коэффициентами вместо 1 FFT с большими. Также заметим, что коэффициенты произведений не больше $M^2 10^6 = 10^9 10^6 = 10^{15}$, поэтому типа `long double` нам точно хватит.

12. **FFT, базовые оптимизации**

    Предподсчитаем все нужные нам корни $root_k[j]$ (корни степени 2^k). Вместо $f[i] = f_0[i \bmod \frac{n}{2}] + rook_k[i] \cdot f_1[i \bmod \frac{n}{2}]$, заметим, что при переходе $i \to i + \frac{n}{2}$ поменяется только знак $\Rightarrow$ не нужно второй раз считать то же произведение.

13. **FFT, два в одном**

    $C(x) = A(x) + i \cdot B(x)$. $FFT(C)[k] = A(w^k) + i \cdot B(w^k)$.
    $\overline{FFT(C)[k]} = \overline{A(w^k)} + \overline{i} \cdot \overline{B(w^k)} = \overline{A(w^k)} - i \cdot \overline{B(w^k)} = A(w^{-k}) - i \cdot B(w^{-k}) \Rightarrow$
    $A(w^k) = \frac{1}{2}(FFT(c)[k] + \overline{FFT(C)[n-k]})$, $B(w^k) = \frac{1}{2}(FFT(c)[k] - \overline{FFT(C)[n-k]})$

14. **Inplace FFT**

    `rev[rev[i]] = i` (то есть `rev` — инволюция) $\Rightarrow$ все циклы в перестановке имеют длину $\leqslant 2$.

```
1  def FFT(a):
2      for (int i = 0; i < N; i++)
3          if (i < rev[i])
4              swap(a[i], a[rev[i]]);
5      for (int k = 1; k < N; k *= 2)
6          for (int i = 0; i < N; i += 2 * k)
7              for (int j = 0; j < k; j++)
8                  num tmp = root[k + j] * a[i + j + k];
9                  a[i + j + k] = a[i + j] - tmp;
10                 a[i + j] += tmp;
```